View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Dagstuhl Research Online Publication Server

On the Value Problem in Weighted Timed Games*

Patricia Bouyer, Samy Jaziri, and Nicolas Markey

LSV – CNRS & ENS Cachan, France

Abstract

A weighted timed game is a timed game with extra quantitative information representing e.g. energy consumption. Optimizing the weight for reaching a target is a natural question, which has already been investigated for ten years. *Existence* of optimal strategies is known to be undecidable in general, and only very restricted classes of games have been identified for which optimal weight and almost-optimal strategies can be computed. In this paper, we show that the *value problem* is undecidable in weighted timed games. We then introduce a large subclass of weighted timed games (for which the undecidability proof above applies), and provide an algorithm to compute arbitrary approximations of the value in such games. To the best of our knowledge, this is the first approximation scheme for an undecidable class of weighted timed games.

1998 ACM Subject Classification D.2.4 Software/Program Verification

Keywords and phrases Timed games, undecidability, approximation

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2015.311

1 Introduction

Timed automata [3] have been introduced in the early 1990's as a powerful model to reason about (the correctness of) real-time computerized systems. Timed automata extend finite-state automata with several clocks, which can be used to enforce timing constraints between various events in the system. They provide a convenient formalism and enjoy reasonably-efficient algorithms (e.g. reachability can be decided using polynomial space), which explains the enormous interest that they provoked in the community of formal methods. *Timed games* [5] extend timed automata with a way of modeling systems interacting with external, uncontrollable components: some transitions of the automaton cannot be forced or prevented to happen. The reachability problem then asks whether there is a *strategy* to reach a given state, whatever the uncontrollable components do. This problem is also decidable, in exponential time.

Hybrid automata [2] are another extension of timed automata, involving *hybrid variables*: those variables can be used to measure other quantities than time (e.g. temperature, energy consumption, ...). Their evolution may follow differential equations, depending on the state of the system. Those variables unfortunately make the reachability problem undecidable [18], even in the restricted case of stopwatches, which are clocks that can be stopped and restarted. Weighted (or priced) timed automata [4, 6] and games [19, 1, 11] have been proposed in the early 2000's as an intermediary model for modelling resource consumption or allocation problems in real-time systems (*e.g.* optimal scheduling [7]). As opposed to (linear) hybrid systems, an execution in a weighted timed model is simply one in the underlying timed model: the extra quantitative information is just an *observer* of the system, and it does not

* This work has been partially supported by the EU under ERC project EQualIS (FP7-308087) and FET project Cassting (FP7-601148).

© Patricia Bouyer, Samy Jaziri, and Nicolas Markey;
licensed under Creative Commons License CC-BY
26th International Conference on Concurrency Theory (CONCUR 2015).
Editors: Luca Aceto and David de Frutos Escrig; pp. 311–324
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

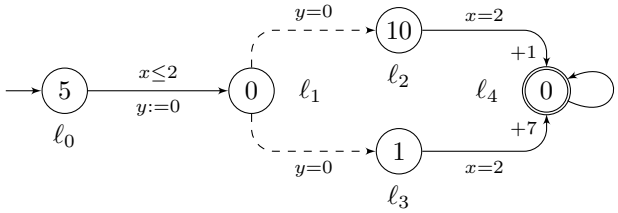

Figure 1 A two-clock weighted timed game.

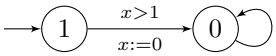

Figure 2 A weighted timed game with value 1 where Player 1 has no optimal strategy.

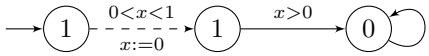

Figure 3 A weighted timed game with value 1 where Player 1 has a strategy to secure weight strictly less than 1.

modify the possible behaviors of the system. Figure 1 displays an example of a weighted timed game: each location carries an integer, which is the rate by which the weight increases when time elapses in that location. Some edges also carry a weight, which indicates how much the weight increases when crossing this edge. Dashed edges are *uncontrollable*, and cannot be forced or prevented to occur. Notice that the constraints on edges never depend on the value of the weight, but only on the values of the clocks.

While optimal weight and almost-optimal schedules can be computed in weighted timed automata [4, 6, 8], the situation is less appealing in the context of weighted timed games: indeed, it is in general undecidable whether a player has a strategy to reach a target with total weight no more than a given value [14], even for timed games with only three clocks [9]. Optimal weight and almost-optimal winning strategies can be computed in restricted classes of weighted timed games, such as games with strong divergence properties on the weight [1, 11], or one-clock turn-based games [13, 22, 17, 15].

We point out a discrepancy in the set of results mentioned above: decidability results concern the *value problem* (is the infimum, over all strategies of Player 1, of the accumulated weight, less than or equal to some constant?), whereas undecidability results deal with the *existence problem* (is there a strategy for Player 1 under which the accumulated weight is less than or equal to some constant?). Both problems are obviously related, but the undecidability of the existence problem does not entail the undecidability of the value problem: indeed, there are obvious examples (see Fig. 2) – and more complex ones (see [12]) – where Player 1 has no optimal strategy for securing the exact value of the game. More surprisingly, there also exist games where Player 1 has *super-optimal* strategies, which when combined with any strategy of Player 2, achieves final weight strictly better than the value of the game. The value of the game of Fig. 3 is 1, but if Player 1 plays a delay $\epsilon/2$ when Player 2 (controlling the dashed edge) played a delay of $1-\epsilon$ in the first location, she gets final weight strictly less than 1 against any Player-2 strategy.

Our contributions in this paper are the following:

- We show that the value problem is undecidable: given a game and a rational c, no algorithm can decide whether the value of the game is less than or equal to c. The proof of this result shares similarities with the undecidability proof for the existence problem, but requires a more careful analysis of the strategies of the players.
- We exhibit a subclass of timed games for which arbitrary precise approximations of the value of a game can be computed. This subclass is large enough to include the

games that are used in the undecidability proof mentioned above. We believe that this approximability result is an important result, since getting the exact value is rarely needed in practice, and optimal strategies need not exist anyway. We notice that in all cases we know of where the optimal weight can be computed (namely [4, 6, 8, 1, 11, 13, 22, 17, 15]), only almost-optimal strategies are actually synthesized; hence it is not really meaningful to know the precise value, and an approximation thereof is sufficient. As a side-result, we get that the optimal weight is co-recursively enumerable in this subclass.

For lack of space, detailed proofs are deferred to the research report [12].

2 Definitions

2.1 Weighted timed games

Let X be a finite set of variables (called *clocks* in our context). A clock valuation for X is a mapping $X \to \mathbb{R}_{\geq 0}$. Given such a clock valuation v, $d \in \mathbb{R}_{\geq 0}$ and $Y \subseteq X$, we define the valuations $v + d$ and $v[Y \leftarrow 0]$ respectively by $(v + d)(x) = v(x) + d$ for every $x \in X$, and by $(v[Y \leftarrow 0])(x) = 0$ if $x \in Y$ and $(v[Y \leftarrow 0])(x) = v(x)$ otherwise.

A clock constraint over a finite set of clocks X is a conjunction of atomic constraints of the form $x \bowtie c$, where $\bowtie \in \{<, \leq, =, \geq, >\}$ and $c \in \mathbb{N}$. We say that a valuation $v \colon X \mapsto \mathbb{R}_{\geq 0}$ satisfies a constraint $x \bowtie c$ whenever $v(x) \bowtie c$; the semantics of conjunction is natural. We also allow $\top$ and $\bot$ as trivial clock constraints (which always evaluate to true and false, respectively).

▶ **Definition 1.** A *weighted timed game* is a tuple $\mathcal{G} = \langle L, \ell_0, L_{\mathsf{f}}, X, E, E_1, E_2, \mathsf{wt} \rangle$ where L is a finite set of locations; $\ell_0 \in L$ is the initial location; $L_{\mathsf{f}} \subseteq L$ is the subset of target locations; X is a finite set of clocks; $E \subseteq L \times \mathcal{C}(X) \times 2^X \times L$ is a finite set of edges, partitioned into the edges E_1 of Player 1, and the edges E_2 of Player 2; and $\mathsf{wt} \colon L \cup E \to \mathbb{Q}_{\geq 0}$ assigns a value to every location and to every edge. We assume that for any $\ell \in L$ and any $v \in \mathbb{R}_{\geq 0}^X$, there is a transition $(\ell, g, r, \ell') \in E$ such that $v \models g$.

The game is *turn-based* whenever L can be partitioned into L_1 and L_2 such that all the edges in E_1 (resp. E_2) have their sources in L_1 (resp. L_2).

The semantics of such a weighted timed game is defined as a game on an infinite graph: a configuration of $\mathcal{G}$ is a pair (ℓ, v) consisting of a location and a clock valuation of X. The configuration $(\ell_0, \mathbf{0})$, where valuation $\mathbf{0}$ assigns 0 to every clock, is the initial configuration. A target configuration is a configuration (ℓ, v) where $\ell \in L_{\mathsf{f}}$ and v is any clock valuation. For any $e = (\ell, g, Y, \ell') \in E$ and $d \geq 0$, there is a transition $(\ell, v) \xrightarrow{d,e} (\ell', v')$ whenever $v + d \models g$ and $v' = v[Y \leftarrow 0]$. In that case, we say that action (d, e) is enabled at (ℓ, v). The weight of such a transition is $\mathsf{wt}_{\mathcal{G}}\big((\ell, v) \xrightarrow{d,e} (\ell', v')\big) = d \cdot \mathsf{wt}(\ell) + \mathsf{wt}(e)$.

A *run* (or *path*) in $\mathcal{G}$ is a finite or infinite sequence of consecutive transitions. A run is initial whenever it starts in $(\ell_0, \mathbf{0})$. If ρ is finite, we write $\mathit{last}(\rho)$ for its last configuration. We write $\mathsf{FPath}^{\mathcal{G}}(\ell, v)$ for the set of finite runs in $\mathcal{G}$ starting in configuration (ℓ, v). We write $\mathsf{FPath}^{\mathcal{G}}$ for the set of all finite runs in $\mathcal{G}$. Given a run ρ, we write $\rho_{|\mathsf{f}}$ for the prefix of ρ ending in the first target configuration it reaches (or $\rho_{|\mathsf{f}} = \rho$ if no target location is visited along ρ). If ρ is a finite run, its weight, written $\mathsf{wt}_{\mathcal{G}}(\rho)$, is defined as the sum of the weights of all its intermediary transitions. If ρ is an infinite run, its weight is $+\infty$ if it does not visit a target configuration, and it is $\mathsf{wt}(\rho_{|\mathsf{f}})$ otherwise.

We now explain how the game is played between Player 1 and Player 2. Let $p \in \{1, 2\}$. A *strategy* for Player p is a mapping $\sigma_p \colon \mathsf{FPath}^{\mathcal{G}} \to (\mathbb{R}_+ \times E_p) \cup \{\bot\}$ such that for every run

$\rho \in \mathsf{FPath}^{\mathcal{G}}$, the action $\sigma_p(\rho)$ is enabled at $last(\rho)$. The special value $\perp$ is used in case no action is enabled for Player p (and only in this case). We write $\mathsf{Strat}_p^{\mathcal{G}}$ for the set of strategies of Player p in $\mathcal{G}$. The (unique) *outcome* of a pair of strategies $(\sigma_1, \sigma_2) \in \mathsf{Strat}_1^{\mathcal{G}} \times \mathsf{Strat}_2^{\mathcal{G}}$ from some configuration s, denoted $\mathsf{Out}_{\mathcal{G}}((\sigma_1, \sigma_2), s)$, is the unique infinite run $\rho = s_0 \xrightarrow{d_1,e_1} s_1 \xrightarrow{d_2,e_2} \dots$ such that $s_0 = s$ and for every $n \in \mathbb{N}$, writing $\rho_{\leq n}$ for $s_0 \xrightarrow{d_1,e_1} s_1 \xrightarrow{d_2,e_2} \dots s_n$, it holds:

- if for some $p \in \{1, 2\}$, $\sigma_p(\rho_{\leq n}) = \perp$, then necessarily $\sigma_{3-p}(\rho_{\leq n}) = (d, e)$ (it cannot be $\perp$, according to our definitions), and $(d_{n+1}, e_{n+1}) = (d, e)$;
- if $\sigma_1(\rho_{\leq n}) = (d^1, e^1)$ and $\sigma_2(\rho_{\leq n}) = (d^2, e^2)$, then $d_{n+1} = \min(d^1, d^2)$. Then the action of the player with the smallest delay is selected: if $d^p < d^{3-p}$ with $p \in \{1, 2\}$, then $e_{n+1} = e^p$, whereas if $d^1 = d^2$ then $e_{n+1} = e^2$. This last condition expresses a priority given to Player 2 (this choice is arbitrary; other variants, e.g. with non-determinism [16], could be handled similarly).

Given two strategies $(\sigma_1, \sigma_2) \in \mathsf{Strat}_1^{\mathcal{G}} \times \mathsf{Strat}_2^{\mathcal{G}}$, their joint weight from configuration s is defined as the weight of their outcome from s. We write $\mathsf{wt}_{\mathcal{G}}((\sigma_1, \sigma_2), s) = \mathsf{wt}(\mathsf{Out}_{\mathcal{G}}((\sigma_1, \sigma_2), s)$. If σ_1 (resp. σ_2) is a Player-1 (resp. Player-2) strategy, we define its weight from s as:

$$\mathsf{wt}_{\mathcal{G}}(\sigma_1, s) = \sup_{\sigma_2' \in \mathsf{Strat}_2^{\mathcal{G}}} \mathsf{wt}_{\mathcal{G}}((\sigma_1, \sigma_2'), s) \qquad \mathsf{wt}_{\mathcal{G}}(\sigma_2, s) = \inf_{\sigma_1' \in \mathsf{Strat}_1^{\mathcal{G}}} \mathsf{wt}_{\mathcal{G}}((\sigma_1', \sigma_2), s)$$

We then define the optimal weight for Player 1 (resp. Player 2) in s as follows:

$$\mathsf{optwt}_{\mathcal{G}}^1(s) = \inf_{\sigma_1 \in \mathsf{Strat}_1^{\mathcal{G}}} \mathsf{wt}_{\mathcal{G}}(\sigma_1, s) \qquad \mathsf{optwt}_{\mathcal{G}}^2(s) = \sup_{\sigma_2 \in \mathsf{Strat}_2^{\mathcal{G}}} \mathsf{wt}_{\mathcal{G}}(\sigma_2, s)$$

One easily notices that $\mathsf{optwt}_{\mathcal{G}}^1(s) \geq \mathsf{optwt}_{\mathcal{G}}^2(s)$ for any s. The converse does not hold in general, but it holds for the class of turn-based games, thanks to Martin's theorem [20]. In the sequel, we call $\mathsf{optwt}_{\mathcal{G}}^1(s)$ the *value* of the game from s (even in the case where $\mathsf{optwt}_{\mathcal{G}}^1(s) \neq \mathsf{optwt}_{\mathcal{G}}^2(s)$), and write it $\mathsf{val}_{\mathcal{G}}(s)$. In the sequel, for all the notations introduced in this paragraph, we may omit to mention the configuration s in case we mean the initial configuration $(\ell_0, \mathbf{0})$. A strategy σ_1 of Player 1 is said *optimal* whenever $\mathsf{wt}_{\mathcal{G}}(\sigma_1) = \mathsf{val}_{\mathcal{G}}$. Given $\epsilon > 0$, σ_1 is said *ϵ-optimal* whenever $\mathsf{wt}_{\mathcal{G}}(\sigma_1) \leq \mathsf{val}_{\mathcal{G}} + \epsilon$. Optimal strategies need not exist, first due to strict clock constraints, or due to more complex convergence phenomena that may happen.

▶ **Example 2.** Consider the weighted timed automaton $\mathcal{G}_{\text{ex}}$ of Fig. 1, and initial configuration $s_0 = (\ell_0, \mathbf{0})$. Locations of Player 1 (resp. Player 2) are depicted with circles (resp. squares), and target location is marked with a double circle. Weight information labels the locations and the transitions (if the weight is 0, then it is omitted on the picture). In this game, Player 1 always reaches the target state. For minimizing the weight, the only choice she has is the time at which she takes the transition leaving ℓ_0. Then Player 2 decides either to switch to location ℓ_2 or to location ℓ_3. We can write the following equation:

$$\mathsf{optwt}_{\mathcal{G}_{\text{ex}}}^1 = \inf_{0 \leq t \leq 2} \max(5t + 10(2 - t) + 1, 5t + (2 - t) + 7) = 14 + \frac{1}{3}.$$

The optimal strategy for Player 1 is to fire the first transition when $x = \frac{4}{3}$.

2.2 Decision problems

In the following, a *threshold* is a pair $(\bowtie, c)$ (which we more often write $\bowtie c$) with $\bowtie \in \{<, \leq, =, \geq, >\}$ and $c \in \mathbb{Q}$.

▶ **Definition 3** (existence problem)**.** Given a weighted timed game $\mathcal{G}$ and a threshold $\bowtie c$, the existence problem asks whether there is a strategy σ_1 for Player 1 s.t. for every strategy σ_2 for Player 2, it holds $\mathsf{wt}_{\mathcal{G}}(\sigma_1, \sigma_2) \bowtie c$.

▶ **Definition 4** (value problem)**.** Given a weighted timed game $\mathcal{G}$ and a threshold $\bowtie c$, the value problem asks whether $\mathsf{optwt}^1_{\mathcal{G}} \bowtie c$.

The existence problem has been shown undecidable (for threshold $\leq c$) 10 years ago [14, 9]. In the present paper, we extend this undecidability result to the value problem. We then introduce a large subclass of weighted timed games (imposing a technical condition on the accumulated weight along cycles), and propose an algorithm for computing an approximation (up to any $\epsilon > 0$) of any game in this subclass, together with almost-optimal strategies. Based on this approximation algorithm, we prove that the value problem for threshold $\leq c$ is co-recursively enumerable on our subclass.

3 Undecidability of the value problem

In this section we show that the value problem in weighted timed games is undecidable. More precisely, we reduce the non-halting problem for a two-counter machine to the value problem of weighted timed games with threshold $\leq c$. Our reduction adapts an earlier reduction of the halting problem for a two-counter machine to the existence problem for weighted timed games [9]. The correctness proof makes use of more refined arguments developed in Section 3.2.

3.1 Reduction

We assume the reader is familiar with the model of counter machines, whose (non-)halting problem is known to be undecidable [21]. We fix a deterministic two-counter machine $\mathcal{M}$, and we define a three-counter machine $\mathcal{M}^\star$, which is obtained by adding to $\mathcal{M}$ a third counter, and by inserting an incrementing instruction for this counter after each transition of the original machine $\mathcal{M}$. In particular, $\mathcal{M}$ halts if and only if $\mathcal{M}^\star$ halts.

Our reduction consists in mimicking the behaviour of the deterministic three-counter machine $\mathcal{M}^\star$ using a (turn-based) weighted timed game $\mathcal{G}_{\mathcal{M}^\star}$: the role of Player 1 is to simulate the execution of $\mathcal{M}^\star$, resetting the clocks at well-chosen times so that the counter values c_i are encoded as clock values $1/2^{c_i}$ when entering selected locations. The role of Player 2 is to check that Player 1 simulates the run of the two-counter machine correctly, and in particular that she accurately updates the values of the clocks. At any time, Player 2 can decide to stop simulating the counter machine and leave the game, resulting in a final weight $3 + \epsilon$, where ϵ is positive if Player 1 did not simulate the run of $\mathcal{M}^\star$ correctly. Player 1 in turn can decide to stop the simulation and to leave the game at any time, securing a final weight $3 + \alpha_N$, where $\alpha_N > 0$ tends to zero when the length N of the execution simulated so far tends to $+\infty$.

If the machine does not halt, Player 1 can accurately simulate the machine for a large number of steps, before leaving the game when $\alpha_N < \epsilon$. Hence for any $\epsilon > 0$, Player 1 can secure final weight at most $3 + \epsilon$, and the value of the game is 3. Conversely, if the machine does halt, its unique computation has finite length N; our construction enforces that Player 1 will not be able to secure final accumulated weight better than $3 + \beta_N$ for some $\beta_N > 0$ (which only depends on N), yielding value 3 for the game.

We now explain this construction in more details. The construction is based on a few simple modules that we then plug together. Those modules explicitly use some clocks,

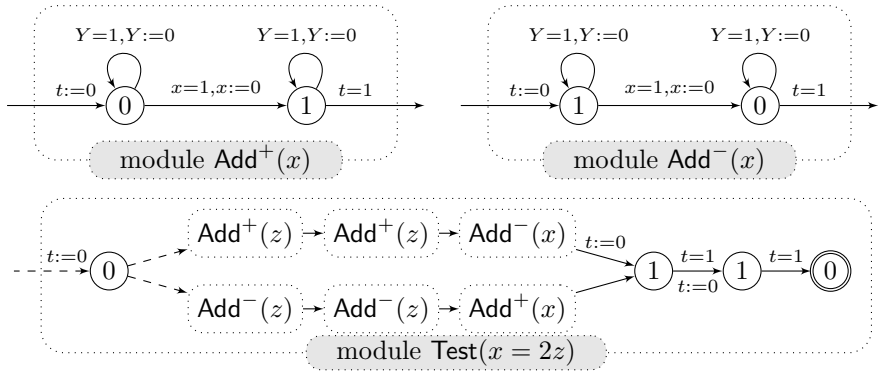

Figure 4 Modules Add^+ and Add^-: the weight is increased by x_0, resp. $1-x_0$ (x_0: initial value of x when entering the module). Module $\mathsf{Test}(x=2z)$: Player 2 can increase the cost by $3+|x_0-2z_0|$.

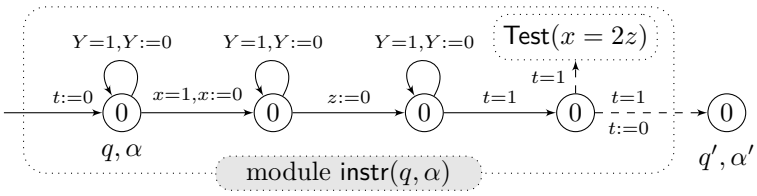

Figure 5 Module $\mathsf{instr}(q,\alpha)$ encoding an incrementing transition (q,i,q').

while some other clocks are only useful for the global reduction; the values of the latter are then preserved by each module, thanks to self-loops on all locations (we symbolically write $Y=1, Y:=0$ to indicate that each other clock is reset when it reaches value 1) and to an (implicit) global invariant requiring that no clock may exceed 1.

3.1.1 Comparing clock values

As sketched above, Player 2 is in charge of checking that Player 1 updates the clocks so as to preserve the encoding $x=1/2^c$ (c is a counter value). When incrementing this counter, this amounts to checking that clock z (after the increment) equals $x/2$. Following the ideas of [9], we build a module $\mathsf{Test}(x=2z)$ in which Player 2 can achieve final accumulated weight $3+|x-2z|$. In other words, the final weight is 3 if $x=2z$ when entering this module, and it is strictly more otherwise. Technically, as shown on Fig. 4, this is achieved by offering two branches to Player 2: one in which the final weight is $3+x-2z$ and one where it is $3+2z-x$. Hence Player 2 can enforce cost $2+|x-2z|$. Accumulating these weights is easily achieved by elapsing delays x, z, $1-z$ and $1-x$ in locations with local weights 0 and 1.

3.1.2 Incrementing and decrementing counters

Incrementing counter c is achieved by asking Player 1 to reset some clock z at a well-chosen time, in such a way that $z=x/2$ when Player 2 is given the opportunity to enter the Test module. The new value of the counter is then encoded by clock z.

Figure 5 displays the module used for encoding increments: clock t serves as a tick ($t=0$ when entering the module, and $t=1$ at the end); clock x encodes the value of the counter initially (that is, $x=1/2^c$ when entering the module), while clock z is used to encode the value of the same counter after the increment (hence $z=x/2$ at the end of the module).

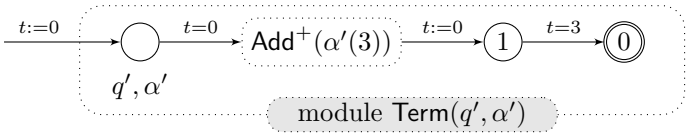

Figure 6 Module $\mathsf{Term}(q', \alpha')$: $\alpha'(3)$ is the clock encoding the third counter, which counts the number of steps simulated so far.

Finally, notice that the module depends on a function α, which is used to keep track of which clock encodes which counter.

Decrementing counter c follows the same idea, but it first performs a zero-test at entrance. Counter $c = 0$ if, and only if, the corresponding clock $x = 1$ when entering the module (i.e., when clock $t = 0$).

3.1.3 Leaving the game

In the game built so far, Player 1 does not have a way to leave for sure to a final location (e.g. when the counter machine does not halt). We give her the opportunity to do so right after incrementing the third counter (which is done every two instructions), by plugging a copy of module Term of Fig. 6 in the corresponding locations.

Notice that this transition is the only possible transition in the locations corresponding to q_{halt}.

3.2 Analysis of the construction

Our construction does not check that the values of the clocks are of the form $1/2^c$. Hence we don't have a correspondence between configurations of $\mathcal{G}_{\mathcal{M}^\star}$ and configurations of $\mathcal{M}^\star$. We define *pseudo-configurations* of $\mathcal{M}^\star$ to tackle this problem: a *pseudo-configuration* of $\mathcal{M}^\star$ is a pair $\gamma = (q, v)$, where q is a discrete state of $\mathcal{M}^\star$ and $v\colon \{1, 2, 3\} \to \mathbb{R}_{\geq 0}$ assigns a non-negative *real* number to every counter. A *pseudo-run* in $\mathcal{M}^\star$ is a sequence of pseudo-configurations. Let $\rho^\star = \gamma_0^\star \to \gamma_1^\star \to \dots$ be the unique maximal (finite or infinite) execution of $\mathcal{M}^\star$. Consider a (finite or infinite) pseudo-run $\rho = \gamma_0 \to \gamma_1 \to \dots$. If there is some $k \geq 0$ such that the discrete states of γ_k and $\gamma_k^\star$ do not coincide, we say that ρ is *strongly-invalid*, which means that compared to the valid run $\rho^\star$, some discrete transition has not been taken appropriately. Writing k_0 for the first position where this occurs, the consecution $\gamma_{k_0-1} \to \gamma_{k_0}$ is said to be *strongly-invalid*. If all discrete states coincide, but ρ is not a prefix of $\rho^\star$, we say that ρ (and its erroneous consecutions) are *weakly-invalid* (in that case, some counter values may not be correctly encoded, but this has no impact on the sequence of visited states, hence on the nature – halting, non-halting – of the path).

Let $\sigma_2^\perp$ be the strategy of Player 2 that consists in never switching to a Test module (*i.e.*, it remains in the *main part* of the game). With any strategy $\sigma_1 \in \mathsf{Strat}_1(\mathcal{G}_{\mathcal{M}^\star})$, we associate the unique maximal outcome in $\mathsf{Out}_{\mathcal{G}_{\mathcal{M}^\star}}(\sigma_1, \sigma_2^\perp)$ (it can either end at a target location of a terminating module, or be infinite). When entering the modules of the instructions, it is clear from the syntax which clock encodes which counter, hence we can extract from that path the sequence of configurations (q_k, α_k, ν_k) when entering (or leaving) an instruction module, where q_k is the discrete state, α_k is the encoding mapping, and ν_k is the clock valuation. The corresponding counter values can be recovered by defining $v_k(i) = -\log_2(\nu_k(\alpha_k(i)))$, and we thus have that $\gamma_k = (q_k, v_k)$ is a pseudo-configuration of $\mathcal{M}^\star$. We then write ρ_{σ_1} for the pseudo-run $\gamma_0 \to \gamma_1 \to \dots$ associated with σ_1.

The following lemma is the crux of our proof: it states that the earlier an occurrence of a strongly-invalid transition, the larger the penalty that can be inflicted to Player 1.

▶ **Lemma 5.** *Let $\sigma_1 \in \mathsf{Strat}_1^{\mathcal{G}_{\mathcal{M}^\star}}$ be a strategy such that ρ_{σ_1} is a strongly-invalid pseudo-run of $\mathcal{M}^\star$. Let $\gamma_k \longrightarrow \gamma_{k+1}$ be the first strongly-invalid consecution of ρ_{σ_1}. Then $k > 0$, and there is a strategy $\sigma_2 \in \mathsf{Strat}_2^{\mathcal{G}_{\mathcal{M}^\star}}$ such that $\mathsf{wt}_{\mathcal{G}_{\mathcal{M}^\star}}(\sigma_1, \sigma_2) \geq 3 + \frac{1}{4k \cdot 2^k}$.*

Using this lemma, we can prove:

▶ **Proposition 1.** *The counter machine $\mathcal{M}^\star$ does not halt if, and only if, the value of game $\mathcal{G}_{\mathcal{M}^\star}$ is (at most) 3.*

Sketch of proof. Assume $\mathcal{M}^\star$ halts, and let N be the length of the halting computation. We argue that the value of the game is lower-bounded by $3 + \epsilon_N$, where ϵ_N is a positive number that depends only on N. Then, either Player 1 decides to leave the game at some point before having simulated the N steps of the counter machine, or Player 1 cheats so that the simulation goes for longer than N steps. In the former case, the weight will be given by gadget Term; it will be $3+\alpha_N$, where α_N is the value of the clock encoding third counter; it is lower-bounded, since the third counter is upper-bounded by N. In the latter case, there must be a strongly-invalid consecution before N steps have been simulated; applying Lemma 5 with $k \leq N$, we again get a lower-bound of the form $3 + \beta_N$ for some positive β_N. In both cases, the weight of the strategy is lower-bounded by a constant that only depends on N.

If $\mathcal{M}^\star$ does not halt, then Player 1 can correctly mimic the counter machine, and leave the game after an arbitrary long simulation, yielding a weight arbitrary close to 3. Hence the value of the game is 3. ◀

This reduction proves undecidability of the value problem for thresholds $= c$ and $\leq c$. By swapping the roles of Players 1 and 2 and slightly modifying the construction, we can prove that the value problem is also undecidable for threshold $\geq c$.

4 Computing an approximation of the value

In this section, we introduce a subclass of weighted timed games, and explain how to approximate the values of games in this class. Our subclass is the set of games $\mathcal{G}$ for which there exists $\kappa > 0$ such that for any finite run in the game that follows a region cycle of the region automaton[1] of $\mathcal{G}$, either the weight is 0, or it is larger than or equal to κ;[2] We call such games *almost strongly non-Zeno weighted timed games*. It is decidable whether a weighted timed game is almost strongly non-Zeno or not (by enumerating all simple cycles of the region abstraction of $\mathcal{G}$). Notice that the undecidability proof for the existence problem [9], as well as our undecidability proof for the value problem in Section 3, is valid for this subclass of weighted timed games. Finally note that if we strengthen the first condition above by assuming that all cyclic runs have weight at least κ (forbidding zero cycles), then we get the class of strongly non-Zeno weighted timed games, for which the value can be computed exactly [1, 11].

In the sequel, we prove the approximability of the optimal weight:

▶ **Theorem 6.** *Given an almost strongly non-Zeno weighted timed game $\mathcal{G}$ and a positive real ϵ, we can compute a rational v, and a strategy σ_1 for Player 1, such that $|v - \mathsf{val}_{\mathcal{G}}| \leq \epsilon$ and $|\mathsf{wt}_{\mathcal{G}}(\sigma_1) - v| \leq \epsilon$.*

[1] We assume familiarity with region equivalence, and refer to [3] for details.
[2] Applying results for weighted timed automata [10], we may assume $\kappa = 1$.

4.1 A basic characterization of the value.

The following fixpoint characterization was given in [11]:

▶ **Proposition 2** ([11]). *Optimal weight for Player* 1 *is the largest fixpoint of the following equation. For every state* s *of* $\mathcal{G}$*:*

$$\mathit{val}_{\mathcal{G}}(s) = \inf_{\substack{d\geq 0, e\in E_1\\ s\xrightarrow{d,e}s'}} \max \left\{ \begin{array}{ll} (1) & d\cdot \mathsf{wt}(\ell) + \mathsf{wt}(e) + \mathit{val}_{\mathcal{G}}(s'), \\ (2) & \sup\limits_{\substack{d'\leq d, e'\in E_2\\ s\xrightarrow{d',e'}s''}} \big(d'\cdot \mathsf{wt}(\ell) + \mathsf{wt}(e') + \mathit{val}_{\mathcal{G}}(s'')\big) \end{array} \right\}$$

However there is no obvious good property of this functional (that we know of) that could be useful for designing an approximation algorithm. Instead, we will partially unfold the game in a careful way in order to preserve the optimal weight, and prove that we can approximate the value of the resulting game.

4.2 The semi-unfolded game

In order to approximate the value of the game, we first build a tree-shaped weighted timed game $\widetilde{\mathcal{G}}$, with the same value as $\mathcal{G}$. Then we explain how to approximate the value of $\widetilde{\mathcal{G}}$.

Let W be an upper bound on the optimal weight from every winning state of $\mathcal{G}$; such a bound is easy to compute, *e.g.* by picking a memoryless and region-uniform winning strategy for Player 1 (in the underlying timed game) [5], and computing a bound on its weight from all configurations.

We write $\mathcal{R}(\mathcal{G})$ for the timed game obtained from $\mathcal{G}$ by splitting its state space into regions (that is, we apply the standard region-automaton construction [3] and interpret it as a timed game). The weighted timed game $\mathcal{R}(\mathcal{G})$ is called the *region game* of $\mathcal{G}$. In $\mathcal{R}(\mathcal{G})$, we additionally assume – for technical reasons – that for every state (ℓ, r) with $\mathsf{wt}(\ell) > 0$, for every $v \in r$, there is no transition $(\ell, v) \xrightarrow{0,e}$. This can be achieved by a simple transformation (at the expense of an extra clock to isolate the case where no time is elapsed in (ℓ, r)), hence it causes no loss of generality.

The values $\mathsf{val}_{\mathcal{G}}$ and $\mathsf{val}_{\mathcal{R}(\mathcal{G})}$ obviously coincide, as there is a tight correspondence between the runs in both games. Now, in game $\mathcal{R}(\mathcal{G})$, we mark in green all locations with weight 0, as well as all edges with discrete weight 0. All other locations and edges are marked in red. Fully green cycles in $\mathcal{R}(\mathcal{G})$ (involving only green locations and edges) then characterize cycles with weight 0. We define the *kernel* $\mathcal{K}$ of $\mathcal{G}$ as the restriction of $\mathcal{R}(\mathcal{G})$ to fully-green strongly connected components. Edges that leave $\mathcal{K}$ are called the *output edges* of $\mathcal{K}$. Notice that any segment of a run from an output edge back to the kernel must visit a red state or a red edge.

The idea is to partially unfold the game $\mathcal{R}(\mathcal{G})$, to obtain a finite tree-like structure (see the left part of Fig. 7). Before formally describing the construction, we begin by informally explaining how it is obtained: we first unfold the game $\mathcal{R}(\mathcal{G})$, and along a branch, as soon as we enter the kernel, we put a copy of $\mathcal{K}$ (removing all states that are not reachable, but without unfolding $\mathcal{K}$); we restart unfolding again from the output edges of that copy of $\mathcal{K}$. We stop this process when along any branch, a red state or edge of $\mathcal{R}(\mathcal{G})$ is visited at least $W + 2$ times.

We now formalize this construction. We first build a tree $\mathcal{T}$, which carries labels both on its nodes and on the edges between nodes. The root n_0 of $\mathcal{T}$ is labelled with the initial location (ℓ_0, r_0) of $\mathcal{R}(\mathcal{G})$. The tree is then built inductively, starting from the root:

- If a node n is labelled with (ℓ, r), then, for every transition $(\ell, r) \xrightarrow{g,Y} (\ell', r')$ in $\mathcal{R}(\mathcal{G})$, we consider two cases:

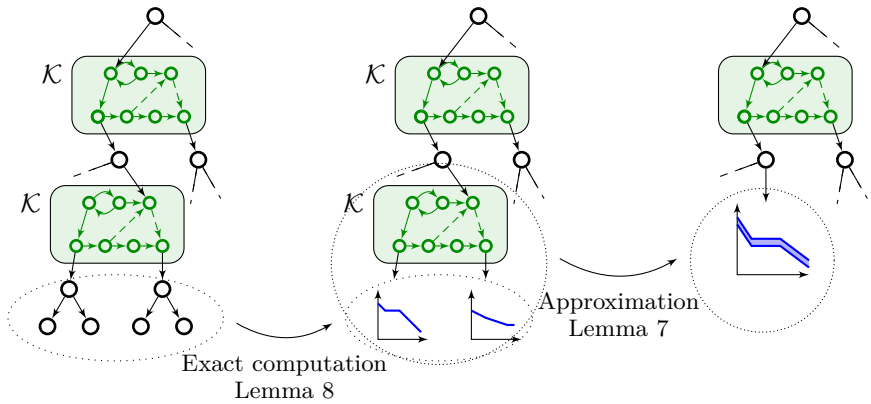

Figure 7 Approximation scheme: in the tree-shaped parts of the semi-unfolding, an exact computation can be performed; in the kernels, we apply under- and over-approximation algorithms.

- if (ℓ', r') belongs to the kernel $\mathcal{K}$, then node n has a son n' labelled with $\mathcal{K}_{(\ell',r')}$;
- Otherwise, n has a son n' labelled with (ℓ', r').

In both cases, the edge between n and n' is labelled with (g, Y).

- If n is labelled with $\mathcal{K}_{(\ell,r)}$, then, for every output edge $e = (\ell'', r'') \xrightarrow{g,Y} (\ell', r')$ of $\mathcal{K}$ that is reachable from (ℓ, r), we add a son n_e labelled with (ℓ', r'). We label the corresponding edge with (g, Y).

We stop this construction as soon as all the branches of this tree contain either $W + 2$ occurrences of the same pair (ℓ, r), or $W + 2$ occurrences of a red transition. We require all the branches to end with some non-kernel node. One quickly realizes that tree $\mathcal{T}$ is finite.

From this tree $\mathcal{T}$, we define a weighted timed game $\widetilde{\mathcal{G}}$, by replacing the kernel nodes with copies of the kernel. More formally, pick a node n labelled with $\mathcal{K}_{(\ell,r)}$, and write n_1 to n_k for its sons. These sons originate from output edges e_1 to e_k of the kernel, which leave some locations s_1 to s_k of the kernel. We then replace node n with a set of locations (n, s), one for each location $s \in \mathcal{K}_{(\ell,r)}$. The edge entering n is now directed to $(\mathsf{n}, (\ell, r))$. Each edge (s, g, Y, s') in the kernel gives rise to an edge $((\mathsf{n}, s), g, Y, (\mathsf{n}, s'))$. And finally, each son n_i of the former node n has an incoming edge from location $(\mathsf{n}, \mathsf{s}_i)$, labelled in the same way as the former edge e_i.

After adding self loops on the leaves with weight 0, and importing the weights on locations and edges and the partition of edges between both players, we end up with a weighted timed game $\widetilde{\mathcal{G}}$, which we can prove satisfies the following:

▶ **Proposition 3.** *The values of games $\mathcal{G}$ and $\widetilde{\mathcal{G}}$ coincide.*

▶ Remark. It is worth noticing that the tree built from a game used in the undecidability reduction (Section 3) is one single kernel with finite trees from the output edges (corresponding to the (acyclic) test or leaving modules). Any outcome visits only once the kernel.

4.3 Approximation algorithm

The approximation algorithm is made of two distinct sub-algorithms (see Fig. 7):

- the **tree-shaped part analysis:** an exact computation of the optimal weight can be achieved in each tree-shaped part [19, 1];
- the **kernel analysis:** this is the difficult part of the algorithm. The idea is to under- and over-approximate the optimal weight computed so far from the output edges by piecewise-constant functions, that are constant over subregions with a known small granularity.

After this transformation, the game played within a kernel is a *(non-weighted) timed game* with an extended reachability condition over output edges, which can be solved with basic techniques. Each kernel analysis induces a bounded imprecision in the computation.

The algorithm then proceeds upwards from the leaves to the root of the built tree, alternatively applying the analysis of the tree-shaped parts and of the kernel.

In order to describe the approximation algorithm, we consider an extension of the model of weighted timed games with *outside weight functions* associated to target locations [13]. A generalized weighted timed game is now a tuple $\langle L, \ell_0, L_{\mathsf{f}}, X, E, E_1, E_2, \mathsf{wt}, \mathsf{outwt}\rangle$, where the first eight components form a weighted timed game as of Def. 1, and $\mathsf{outwt}\colon L_{\mathsf{f}} \to (\mathbb{R}_+^X \to (\mathbb{R}_+ \cup \{+\infty\}))$ assigns a weight function with every target location. Compared to classical weighted timed games, the weight of a run ρ reaching a target state is augmented by $\mathsf{outwt}(\ell)(v)$, assuming $\mathit{last}(\rho) = (\ell, v)$ with $\ell \in L_{\mathsf{f}}$. All notations are extended to this model in a straightforward way. From a given configuration $(\ell_{\mathsf{init}}, v)$, the aim of Player 1 now is to reach the target states while minimizing this modified weight. Following [13], we use this natural extension of the original model to iteratively compute the value in $\widetilde{\mathcal{G}}$.

The main idea of our algorithm is to approximate the value of outside weight functions by piecewise-constant functions. For this, we split regions into smaller sets that we call *regions of granularity* $1/2^N$, or simply $1/2^N$*-regions*, hereafter. Formally, a set R is a $1/2^N$-region for a maximal constant M if, and only if, the set $2^N \cdot R$ (obtained by scaling R by 2^N in all dimensions) is a region for maximal constant $2^N \cdot M$. In order to be able approximate outside weight functions with piecewise-constant functions, we have to restrict them: an outside weight function outwt is said smooth whenever there exists an integer M and a granularity $1/2^N$ such that for every $\ell \in L_{\mathsf{f}}$, the function $\mathsf{outwt}(\ell)$ is uniformly continuous (or constantly equal to $+\infty$) over $1/2^N$-regions for maximal constant M. We additionally require that within any $1/2^N$-region, the function $\mathsf{outwt}(\ell)$ does not depend on the exact values of the clocks whose value is larger than M: for any two clock valuations v and v' in the same $1/2^N$-region such that $v(x) = v'(x)$ whenever $v(x) \leq M$ or $v'(x) \leq M$, it holds $\mathsf{outwt}(\ell)(v) = \mathsf{outwt}(\ell)(v')$. Notice that contrary to [13], we impose no other conditions (affineness, monotonicity) on outside weight functions.

Computing the value in the kernel. We now explain how we compute an approximation of the value in the kernel, assuming that we have smooth outside weight functions in the locations reached by output edges.

▶ **Lemma 7.** *Consider a maximal strongly connected component K of the kernel $\mathcal{K}$ of some region game $\mathcal{R}(\mathcal{G})$, and pick a state (ℓ, r) of K. Let $S_O = \{(\ell_e, r_e) \mid e \text{ output edge of } K\}$ be the set of target locations in $\mathcal{R}(\mathcal{G})$ of the output edges of K. Define a game $\mathcal{H} = \langle K \cup S_O, (\ell, r), S_O, X, E_{|K\cup S_0}, E_1 \cap E_{|K\cup S_0}, E_2 \cap E_{|K\cup S_0}, \mathsf{wt}_{|\mathcal{H}}, \mathsf{outwt}\rangle$, where outwt is a smooth outside weight function with maximal partial derivative P.*

Then, for every $\epsilon > 0$, we can compute ϵ-over-approximations and ϵ-under-approximations of the value of the game $\mathcal{H}$ at any configuration (ℓ, v) with $v \in r$, and (2ϵ)-optimal strategies from all those configurations. Additionally, the computed approximations are constant on each $1/2^N$-subregion of r, as soon as $1/2^N \leq \epsilon/P$. Finally, the computation can be achieved in time $O(|\mathcal{R}(\mathcal{G})| \cdot (P/\epsilon)^{|X|})$.

The key idea behind that lemma is to under- and over-approximate smooth outside weight functions by constant functions over refined ($1/2^N$-grained) regions, which then reduces the game to some extended reachability timed game: each output edge is assigned a single value by the computed constant outside weight function (the constant under- or

over-approximation), and this value is the weight for leaving *via* this edge (there is no weight involved elsewhere in the kernel); hence this defines a preference order over output edges, and the aim of Player 1 now is to enforce the 'less expensive' edge; this game is timed, but with a (qualitative) extended-reachability condition over output edges; it can be solved using standard attractor techniques over timed games.

Computing the value in a finite tree. The computation of the optimal weight in tree-shaped parts of the weighted timed games $\widetilde{\mathcal{G}}$ is known to be computable [19, 1] using an iterative backward algorithm (which can also compute almost-optimal strategies) based on equations given in Prop. 2. In our setting, the techniques developed in [1] entail the following lemma:

▶ **Lemma 8.** *Let $\mathcal{S}$ be a finite unfolding of a region game $\mathcal{R}(\mathcal{G})$ from some region (ℓ_0, r_0). We equip $\mathcal{S}$ with an outside weight function* outwt *associating with every leaf (ℓ, r) of $\mathcal{S}$ a function from r to $\mathbb{R}_{\geq 0}$. We require that there exists an integer N such that for any terminal node (ℓ, r), $\mathsf{outwt}(\ell, r)$ is constant over $1/2^N$-subregions of r.*

Then we can compute the function $v \in r_0 \mapsto \mathsf{val}_{\mathcal{S}}(\ell_0, v)$ in time $2^{O(N\cdot|X|^2+|\mathcal{S}|^2)}$. Moreover, those functions are piecewise-affine, with partial derivatives in $\{0\} \cup \{\mathsf{wt}(\ell) \mid \ell \in L\}$.

Global algorithm. Our general algorithm consists in iteratively applying the two lemmas above, hence computing approximation functions from the leaves to the root of the tree $\mathcal{T}$. More precisely, fix $\epsilon > 0$, and pick a node n of $\mathcal{T}$. This node is labelled either with (ℓ, r), or with $\mathcal{K}_{(\ell,r)}$. Our algorithm computes two functions $f^+_\epsilon(\mathsf{n})\colon r \to \mathbb{R}$ and $f^-_\epsilon(\mathsf{n})\colon r \to \mathbb{R}$ that are smooth and respectively ϵ-over-approximate and ϵ-under-approximate the value of the game $\widetilde{\mathcal{G}}$ from all configurations in the region (ℓ, r) corresponding to node n.

Write $\epsilon' = \epsilon/(|\mathcal{R}(\mathcal{G})| \cdot (W + 2) + 1)$. We begin with initializing the computation at the leaves of $\mathcal{T}$. Let n be a leaf. By construction, it must be labelled with some (ℓ, r). We define $f^+_{\epsilon'}(\mathsf{n})$ and $f^-_{\epsilon'}(\mathsf{n})$ as follows:

- if $\ell \in L_{\mathsf{f}}$, we let $f^+_{\epsilon'}(\mathsf{n})(v) = f^-_{\epsilon'}(\mathsf{n})(v) = 0$ for every $v \in r$;
- otherwise, we let $f^+_{\epsilon'}(\mathsf{n})(v) = f^-_{\epsilon'}(\mathsf{n})(v) = +\infty$ for every $v \in r$.

These functions obviously correspond to the exact value. Moreover, they fulfill the hypothesis of both Lemmas 7 and 8.

Now pick a node n of $\mathcal{T}$, assuming that each son n' of n has been assigned approximation functions $f^+_{k\epsilon'}(\mathsf{n})$ and $f^-_{k\epsilon'}(\mathsf{n})$. We consider two cases:

- if n is labelled with (ℓ, r) and is the son of a kernel node, then we apply Lemma 8 to compute $f^+_{k\epsilon'}(\mathsf{n})$ and $f^-_{k\epsilon'}(\mathsf{n})$;
- if n is labelled with $\mathcal{K}_{(\ell,r)}$, then we apply Lemma 7 with approximation value ϵ' and smooth outside weight function $f^+_{k\epsilon'}(\mathsf{n}')$ (resp. $f^-_{k\epsilon'}(\mathsf{n}')$), and get piecewise-constant weight functions $f^+_{(k+1)\epsilon'}(\mathsf{n})$ (resp. $f^-_{(k+1)\epsilon'}(\mathsf{n})$).

The correctness of the algorithm is stated as follows:

▶ **Lemma 9.** *Pick a node n of $\mathcal{T}$ labelled by (ℓ, r) or $\mathcal{K}_{(\ell,r)}$. Assume node n has been assigned a value by $f^+_{k\epsilon'}$ and $f^-_{k\epsilon'}$. Then for every $v \in r$,*

$$f^-_{k\epsilon'}(\mathsf{n})(v) \leq \mathsf{val}_{\widetilde{\mathcal{G}}}(\ell, v) \leq f^+_{k\epsilon'}(\mathsf{n})(v).$$

Furthermore, for any node n where $f^-_{k\epsilon'}(\mathsf{n})$ and $f^+_{k\epsilon'}(\mathsf{n})$ are defined, it holds

$$\left|f^+_{k\epsilon'}(\mathsf{n})(v) - f^-_{k\epsilon'}(\mathsf{n})(v)\right| \leq k \cdot \epsilon'.$$

Additionally, we can compute $(2k\epsilon')$-optimal winning strategies.

The maximal number of kernels along any branch of $\mathcal{T}$ being bounded by $(W+2)\cdot|\mathcal{R}(\mathcal{G})|+1$, there exists $k \leq (W+2)\cdot|\mathcal{R}(\mathcal{G})|+1$ for which $f^-_{k\epsilon'}$ and $f^+_{k\epsilon'}$ are defined at the root of $\mathcal{T}$. Those functions ϵ-under-approximate and ϵ-overapproximate the value of $\mathcal{G}$, as required.

Complexity of the algorithm. Our algorithm inductively applies Lemma 7 on each copy of the kernel, and Lemma 8 on each intermediary tree between kernels. This may result in $|\mathcal{R}(\mathcal{G})|^{(W+2)\cdot|\mathcal{R}(\mathcal{G})|+1}$ applications of both lemmas. As W can be bounded by $|\mathcal{R}(\mathcal{G})|\cdot P$ where P is the maximal rate appearing in the automaton, the time-complexity is bounded by $\left(\frac{1}{\epsilon}\right)^{|X|^2}\cdot 2^{O(|\mathcal{R}(\mathcal{G})|^4)}$ (hence doubly-exponential in the size of the original timed game).

4.4 The value problem is (co-)recursively enumerable

Using our approximation algorithm, we immediately get the following result:

▶ **Theorem 10.** *Over the class of almost strongly non-Zeno weighted timed games, the value problem is co-recursively enumerable for thresholds $\leq c$, $= c$ and $\geq c$ (for $c \in \mathbb{Q}$). It is recursively enumerable for thresholds $< c$ and $> c$ (with $c \in \mathbb{Q}$).*

Proof. We prove this result using the approximation algorithm above. We consider the case of threshold $\leq c$, but the other cases are similar. Given a game $\mathcal{G}$, we build a Turing machine that halts if, and only if, the value of $\mathcal{G}$ is **not** less than or equal to c.

The machine runs as follows: it first sets the tolerance ϵ to 1, and the approximation v to $-\infty$. Then, as long as $c \geq v-\epsilon$, it divides ϵ by 2, and computes an ϵ-approximation v of the value of $\mathcal{G}$ using the approximation algorithm above.

One easily sees that if the value of $\mathcal{G}$ is indeed larger than c, then eventually ϵ will be less than $\mathsf{val}_{\mathcal{G}} - c$, and the machine will halt. Conversely, if the value of $\mathcal{G}$ is larger than or equal to c, the machine will run forever. ◀

5 Conclusion

We proved in this paper that for weighted timed games, no algorithm can compute the value of the game. On the other hand, we developed an approximation algorithm for a large subclass of weighted timed games, for which the undecidability proof already applies.

We have two natural ways for continuing this work: the first one is to extend the algorithm to handle the full class of weighted timed automata; the second one is to improve the complexity of our approximation algorithm, which currently is doubly-exponential.

References

1. Rajeev Alur, Mikhail Bernadsky, and P. Madhusudan. Optimal reachability in weighted timed games. In *ICALP'04*, volume 3142 of *LNCS*, pages 122–133. Springer, 2004.
2. Rajeev Alur, Costas Courcoubetis, Thomas A. Henzinger, and Pei-Hsin Ho. Hybrid automata: an algorithmic approach to specification and verification of hybrid systems. In *HSCC'91-'92*, volume 736 of *LNCS*, pages 209–229. Springer, 1993.
3. Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.
4. Rajeev Alur, Salvatore La Torre, and George J. Pappas. Optimal paths in weighted timed automata. In *HSCC'01*, volume 2034 of *LNCS*, pages 49–62. Springer, 2001.
5. Eugene Asarin, Oded Maler, Amir Pnueli, and Joseph Sifakis. Controller synthesis for timed automata. In *Proc. IFAC Symp. System Structure & Control*, pages 469–474. Elsevier, 1998.

6 Gerd Behrmann, Ansgar Fehnker, Thomas Hune, Kim G. Larsen, Paul Pettersson, Judi Romijn, and Frits Vaandrager. Minimum-cost reachability for priced timed automata. In *HSCC'01*, volume 2034 of *LNCS*, pages 147–161. Springer, 2001.

7 Gerd Behrmann, Kim G. Larsen, and Jacob I. Rasmussen. Optimal scheduling using priced timed automata. *ACM Sigmetrics Performance Eval. Review*, 32(4):34–40, 2005.

8 Patricia Bouyer, Thomas Brihaye, Véronique Bruyère, and Jean-François Raskin. On the optimal reachability problem. *Formal Methods in System Design*, 31(2):135–175, 2007.

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

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

11 Patricia Bouyer, Franck Cassez, Emmanuel Fleury, and Kim G. Larsen. Optimal strategies in priced timed game automata. In *FSTTCS'04*, volume 3328 of *LNCS*, pages 148–160. Springer, 2004.

12 Patricia Bouyer, Samy Jaziri, and Nicolas Markey. On the value problem in weighted timed games. Research Report LSV-14-12, Laboratoire Spécification et Vérification, ENS Cachan, France, October 2014. 24 pages.

13 Patricia Bouyer, Kim G. Larsen, Nicolas Markey, and Jacob I. Rasmussen. Almost optimal strategies in one-clock priced timed automata. In *FSTTCS'06*, volume 4337 of *LNCS*, pages 345–356. Springer, 2006.

14 Thomas Brihaye, Véronique Bruyère, and Jean-François Raskin. On optimal timed strategies. In *FORMATS'05*, volume 3821 of *LNCS*, pages 49–64. Springer, 2005.

15 Thomas Brihaye, Gilles Geeraerts, Shankara Narayanan Krishna, Lakshmi Manasa, Benjamin Monmege, and Ashutosh Trivedi. Adding negative prices to priced timed games. In *CONCUR'14*, volume 8704 of *LNCS*, pages 560–575. Springer, 2014.

16 Luca de Alfaro, Marco Faella, Thomas A. Henzinger, Rupak Majumdar, and Mariëlle Stoelinga. The element of surprise in timed games. In Roberto Amadio and Denis Lugiez, editors, *CONCUR'03*, volume 2761 of *LNCS*, pages 142–156. Springer, 2003.

17 Thomas Dueholm Hansen, Rasmus Ibsen-Jensen, and Peter Bro Miltersen. A faster algorithm for solving one-clock priced timed games. In *CONCUR'13*, volume 8052 of *LNCS*, pages 531–545. Springer, 2013.

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

19 Salvatore La Torre, Supratik Mukhopadhyay, and Aniello Murano. Optimal-reachability and control for acyclic weighted timed automata. In *TCS'02*, volume 223 of *IFIP Conf. Proc.*, pages 485–497. Kluwer, 2002.

20 Donald A. Martin. Borel determinacy. *Annals of Mathematics*, 102:363–371, 1975.

21 Marvin Minsky. *Computation: Finite and Infinite Machines*. Prentice Hall, 1967.

22 Michał Rutkowski. Two-player reachability-price games on single-clock timed automata. In *QAPL'11*, volume 57 of *ENTCS*, pages 31–46, 2011.