

The Value Problem for Weighted Timed Games with Two Clocks is Undecidable

Quentin Guilmant^[0009-0004-7097-0595], Joël Ouaknine^{*[0000-0003-0031-9356]},
and Isa Vialard^[0000-0002-7261-9342]

Max Planck Institute for Software Systems, Saarland Informatics Campus,
Saarbrücken, Germany

Abstract. We prove that the Value Problem for weighted timed games (WTGs) with two clocks and non-negative integer weights is undecidable — even under a time bound. The Value Problem for weighted timed games (WTGs) consists in determining, given a two-player weighted timed game with a reachability objective and a rational threshold, whether or not the value of the game exceeds the threshold. This problem was shown to be undecidable some ten years ago for WTGs making use of at least three clocks, and is known to be decidable for single-clock WTGs. Our reduction encodes a deterministic two-counter machine using two clocks and uses “punishment” gadgets that let the opponent detect and penalize any incorrect simulation. This closes one of the last remaining major gaps in our algorithmic understanding of WTGs.

1 Introduction

Real-time systems are not only ubiquitous in modern technological society, they are in fact increasingly pervasive in critical applications — from embedded controllers in automotive and avionics platforms to resource-constrained communication protocols. In such systems, exacting timing constraints and quantitative objectives must often be met simultaneously. Weighted Timed Games (WTGs), introduced over two decades ago [15,2,13,1,4], provide a powerful modelling framework for the automatic synthesis of controllers in such settings: they combine the expressiveness of Alur and Dill’s clock-based timed automata with **Min-Max** gameplay and non-negative integer weights on both locations and transitions, enabling one to reason about quantitative aspects such as energy consumption, response times, or resource utilisation under adversarial conditions.

A central algorithmic task for WTGs is the *Value Problem*: given a two-player, turn-based WTG with a designated start configuration and a rational threshold c , determine whether Player **Min** can guarantee reaching a goal location with cumulative cost at most c , despite best adversarial play by Player

* Joël Ouaknine is also affiliated with Keble College, Oxford as emmy.network Fellow, and is supported by ERC grant DynAMiCs (101167561) and DFG grant 389792660 as part of TRR 248.

Max. This problem lies at the heart of quantitative controller synthesis and performance analysis for real-time systems.

Unfortunately, fundamental algorithmic barriers are well known. In particular, the Value Problem is known to be undecidable for WTGs making use of three or more clocks [5]. In fact, even approximating the value of three-clock WTGs with arbitrary (positive and negative) weights is known to be computationally unsolvable [12]. On the positive side, the Value Problem for WTGs making use of a single clock is decidable, regardless of whether weights range over \mathbb{N} or \mathbb{Z} [6,17]. There is a voluminous literature in this general area; for a comprehensive overview and discussion of the state of the art, we refer the reader to [11]. See also Fig. 1 in which we summarise some of the key existing results.

Clocks	Weights in	Value Problem	Existence Problem
1	\mathbb{N}	decidable [6]	decidable [7]
	\mathbb{Z}	decidable [17]	
2	\mathbb{N}	undecidable	undecidable
	\mathbb{Z}	undecidable	undecidable [10]
3+	\mathbb{N}	undecidable [5]	undecidable [3]
	\mathbb{Z}	inapproximable [12]	undecidable

Fig. 1. State of the art on the Value Problem for weighted timed games. Approximability for 2-clock WTGs, and 3-clock WTGs with weights in \mathbb{N} , remain open. This paper’s main contribution (undecidability for WTGs with two clocks and weights in \mathbb{N}) is highlighted in boldface blue.

The case of WTGs with exactly two clocks (and non-negative weights) has remained stubbornly open. Resolving this question is essential, since two clocks suffice to encode most practical timing constraints (e.g., deadline plus cooldown), and efficient single-clock algorithms cannot in general be lifted to richer timing scenarios. The main contribution of this paper is to close this gap by establishing undecidability:

Theorem 1. *The Value Problem for two-player, turn-based, time-bounded, two-clock, weighted timed games with non-negative integer weights is undecidable. The same holds for weighted timed games over unbounded time otherwise satisfying the same hypotheses.*

Our reduction is from the Halting Problem for deterministic two-counter machines and proceeds via a careful encoding of counter values in clock valuations, combined with “punishment” gadgets that enforce faithful simulation or allow the adversary to drive the accumulated cost upwards. Key technical novelties include:

- Counter-Evolution Control (*CEC*) modules, which enforce precise proportional delays for encoding the incrementation and decrementation of counters.
- Multiplication-Control (*MC*) gadgets, which enable the adversary to verify whether simulated counter updates correspond to exact multiplication factors.

Together, these constructions fit within the two-clock timing structure and utilize only non-negative integer weights, thereby demonstrating that even the two-clock fragment — previously the only remaining decidability candidate — admits no algorithmic solution to the Value Problem. As a complementary result, we also show that the related Existence Problem (i.e., does **Min** have a strategy to achieve a cost at most c ?) is undecidable under the same hypotheses.

Finally, we note that our reduction is implemented via WTGs with bounded duration by construction. This is particularly noteworthy given that many algorithmic problems for real-time and hybrid systems, which are known to be undecidable over unbounded time, become decidable in a time-bounded setting. See, for example, [18,19,14,8,9].

2 Weighted Timed Games

Let \mathcal{X} be a finite set of **clocks**. **Clock constraints** over \mathcal{X} are expressions of the form $x \sim n$ or $x - y \sim n$, where $x, y \in \mathcal{X}$ are clocks, $\sim \in \{<, \leq, =, \geq, >\}$ is a comparison symbol, and $n \in \mathbb{N}$ is a natural number. We write \mathcal{C} to denote the set of all clock constraints over \mathcal{X} . A **valuation** on \mathcal{X} is a function $\nu : \mathcal{X} \rightarrow \mathbb{R}_{\geq 0}$. For $d \in \mathbb{R}_{\geq 0}$ we denote by $\nu + d$ the valuation such that, for all clocks $x \in \mathcal{X}$, $(\nu + d)(x) = \nu(x) + d$. Let $X \subseteq \mathcal{X}$ be a subset of all clocks. We write $\nu[X := 0]$ for the valuation such that, for all clocks $x \in X$, $\nu[X := 0](x) = 0$, and $\nu[X := 0](y) = \nu(y)$ for all other clocks $y \notin X$. For a set $C \subseteq \mathcal{C}$ of clock constraints over \mathcal{X} , we say that the valuation ν **satisfies** C , denoted $\nu \models C$, if and only if all the comparisons in C hold when replacing each clock x by its corresponding value $\nu(x)$.

Definition 1. A (*turn-based*) **weighted timed game** is given by a tuple $\mathcal{G} = (L_{\text{Min}}, L_{\text{Max}}, G, \mathcal{X}, T, w)$, where:

- L_{Min} and L_{Max} are the (disjoint) sets of **locations** belonging to Players **Min** and **Max** respectively; we let $L = L_{\text{Min}} \cup L_{\text{Max}}$ denote the set of all locations. (In drawings, locations belonging to **Min** are depicted by blue circles, and those belonging to **Max** are depicted by red squares.)
- $G \subseteq L_{\text{Min}}$ are the **goal locations**.
- \mathcal{X} is a set of clocks.
- $T \subseteq (L \setminus G) \times 2^{\mathcal{C}} \times 2^{\mathcal{X}} \times L$ is a set of (**discrete**) **transitions**. A transition $\ell \xrightarrow{C, X} \ell'$ enables moving from location ℓ to location ℓ' , provided all clock constraints in C are satisfied, and afterwards resetting all clocks in X to zero.

– $w : (L \setminus G) \cup T \rightarrow \mathbb{Z}$ is a **weight function**.

In the above, we assume that all data (set of locations, set of clocks, set of transitions, set of clock constraints) are finite.

Let $\mathcal{G} = (L_{\text{Min}}, L_{\text{Max}}, G, \mathcal{X}, T, w)$ be a weighted timed game. A **configuration** over \mathcal{G} is a pair (ℓ, ν) , where $\ell \in L$ and ν is a valuation on \mathcal{X} . Let $d \in \mathbb{R}_{\geq 0}$ be a **delay** and $t = \ell \xrightarrow{C, X} \ell' \in T$ be a discrete transition. One then has a valid **delayed transition** (or simply a **transition** if the context is clear) $(\ell, \nu) \xrightarrow{d, t} (\ell', \nu')$ provided that $\nu + d \models C$ and $\nu' = (\nu + d)[X := 0]$. Intuitively, control remains in location ℓ for d time units, after which it transitions to location ℓ' , resetting all the clocks in X to zero in the process. The **weight** of such a delayed transition is $d \cdot w(\ell) + w(t)$, taking account both of the time spent in ℓ as well as the weight of the discrete transition t .

As noted in [11], without loss of generality one can assume that no configuration (other than those associated with goal locations) is deadlocked; in other words, for any location $\ell \in L \setminus G$ and valuation $\nu \in \mathbb{R}_{\geq 0}^{\mathcal{X}}$, there exists $d \in \mathbb{R}_{\geq 0}$ and $t \in T$ such that $(\ell, \nu) \xrightarrow{d, t} (\ell', \nu')$.¹

Let $k \in \mathbb{N}$. A **run** ρ of length k over \mathcal{G} from a given configuration (ℓ_0, ν_0) is a sequence of matching delayed transitions, as follows:

$$\rho = (\ell_0, \nu_0) \xrightarrow{d_0, t_0} (\ell_1, \nu_1) \xrightarrow{d_1, t_1} \dots \xrightarrow{d_{k-1}, t_{k-1}} (\ell_k, \nu_k).$$

The **weight** of ρ is the cumulative weight of the underlying delayed transitions:

$$\text{weight}(\rho) = \sum_{i=0}^{k-1} (d_i \cdot w(\ell_i) + w(t_i)).$$

An infinite run ρ is defined in the obvious way; however, since no goal location is ever reached, its weight is defined to be infinite: $\text{weight}(\rho) = +\infty$.

A run is **maximal** if it is either infinite or cannot be extended further. Thanks to our deadlock-freedom assumption, finite maximal runs must end in a goal location. We refer to maximal runs as **plays**.

We now define the notion of **strategy**. Recall that locations of \mathcal{G} are partitioned into sets L_{Min} and L_{Max} , belonging respectively to Players Min and Max. Let Player $P \in \{\text{Min}, \text{Max}\}$, and write $\mathcal{FR}_{\mathcal{G}}^P$ to denote the collection of all non-maximal finite runs of \mathcal{G} ending in a location belonging to Player P . A **strategy** for Player P is a mapping $\sigma_P : \mathcal{FR}_{\mathcal{G}}^P \rightarrow \mathbb{R}_{\geq 0} \times T$ such that for all finite runs

¹ This can be achieved by adding unguarded transitions to a sink location for all locations controlled by Min and unguarded transitions to a goal location for the ones controlled by Max (noting that in all our constructions, Max-controlled locations always have weight 0). Nevertheless, in the interest of clarity we omit such extraneous transitions and locations in our representation of WTGs; we merely assume instead that neither player allows him- or herself to end up in a deadlocked situation, unless a goal location has been reached.

$\rho \in \mathcal{FR}_G^P$ ending in configuration (ℓ, ν) with $\ell \in L_P$, the delayed transition $(\ell, \nu) \xrightarrow{d,t} (\ell', \nu')$ is valid, where $\sigma_P(\rho) = (d, t)$ and (ℓ', ν') is some configuration (uniquely determined by $\sigma_P(\rho)$ and ν).

Let us fix a starting configuration (ℓ_0, ν_0) , and let σ_{Min} and σ_{Max} be strategies for Players Min and Max respectively (one speaks of a *strategy profile*). We write $\text{play}_G((\ell_0, \nu_0), \sigma_{\text{Min}}, \sigma_{\text{Max}})$ to denote the unique maximal run starting from configuration (ℓ_0, ν_0) and unfolding according to the strategy profile $(\sigma_{\text{Min}}, \sigma_{\text{Max}})$: in other words, for every strict finite prefix ρ of $\text{play}_G((\ell_0, \nu_0), \sigma_{\text{Min}}, \sigma_{\text{Max}})$ in \mathcal{FR}_G^P , the delayed transition immediately following ρ in $\text{play}_G((\ell_0, \nu_0), \sigma_{\text{Min}}, \sigma_{\text{Max}})$ is labelled with $\sigma_P(\rho)$.

Recall that the objective of Player Min is to reach a goal location through a play whose weight is as small possible. Player Max has an opposite objective, trying to avoid goal locations, and, if not possible, to maximise the cumulative weight of any attendant play. This gives rise to the following two symmetrical definitions:

$$\begin{aligned} \overline{\text{Val}}_G(\ell_0, \nu_0) &= \inf_{\sigma_{\text{Min}}} \left\{ \sup_{\sigma_{\text{Max}}} \{ \text{weight}(\text{play}_G((\ell_0, \nu_0), \sigma_{\text{Min}}, \sigma_{\text{Max}})) \} \right\} \text{ and} \\ \underline{\text{Val}}_G(\ell_0, \nu_0) &= \sup_{\sigma_{\text{Max}}} \left\{ \inf_{\sigma_{\text{Min}}} \{ \text{weight}(\text{play}_G((\ell_0, \nu_0), \sigma_{\text{Min}}, \sigma_{\text{Max}})) \} \right\}. \end{aligned}$$

$\overline{\text{Val}}_G(\ell_0, \nu_0)$ represents the smallest possible weight that Player Min can possibly achieve, starting from configuration (ℓ_0, ν_0) , against best play from Player Max, and conversely for $\underline{\text{Val}}_G(\ell_0, \nu_0)$: the latter represents the largest possible weight that Player Max can enforce, against best play from Player Min.² As noted in [11], turn-based weighted timed games are *determined*, and therefore $\overline{\text{Val}}_G(\ell_0, \nu_0) = \underline{\text{Val}}_G(\ell_0, \nu_0)$ for any starting configuration (ℓ_0, ν_0) ; we denote this common value by $\text{Val}_G(\ell_0, \nu_0)$.

We can now state:

Definition 2 (Value Problem). *Given a WTG \mathcal{G} with starting location ℓ_0 and a threshold $c \in \mathbb{Q}$, the **Value Problem** asks whether $\text{Val}_G(\ell_0, \mathbf{0}) \leq c$.*

The Value Problem differs subtly but importantly from the *Existence Problem*:

Definition 3 (Existence Problem). *Given a WTG \mathcal{G} with starting location ℓ_0 and a threshold $c \in \mathbb{Q}$, the **Existence Problem** asks whether **Min** has a strategy σ_{Min} such that*

$$\sup_{\sigma_{\text{Max}}} \{ \text{weight}(\text{play}_G((\ell_0, \mathbf{0}), \sigma_{\text{Min}}, \sigma_{\text{Max}})) \} \leq c.$$

Remark 1. The Existence problem is undecidable for two-clock WTGs when arbitrary integer (positive and negative) weights are allowed [10].

² Technically speaking, these values may not be literally achievable; however given any $\varepsilon > 0$, both players are guaranteed to have strategies that can take them to within ε of the optimal value.

3 Undecidability

To establish undecidability, we reduce the Halting Problem for two-counter machines to the Value Problem. A two-counter machine is a tuple $\mathcal{M} = (Q, q_i, q_h, T)$ where Q is a finite set of states, $q_i, q_h \in Q$ are the initial and final state and $T \subseteq (Q \times \{c, d\} \times Q) \cup (Q \times \{c, d\} \times Q \times Q)$ is a set of transitions. A two-counter machine is deterministic if for any state q there is at most one transition $t \in T$ which has q as first component. As its name suggests, a two-counter machine comes equipped with two counters, c and d , which are variables with values in \mathbb{N} . The semantics is as follows: a transition (q, e, q') increases the value of counter $e \in \{c, d\}$ by 1 and moves to state q' . A transition (q, e, q', q'') moves to q'' if $e = 0$ and to q' otherwise. In the latter case, it also decreases the value of e by 1. The Halting Problem for (deterministic) two-counter machines is known to be undecidable (see [16, Thm. 14-1]).

3.1 Overview of the reduction

Let \mathcal{M} be a two-counter machine with counters c and d . We construct a WTG $\mathcal{G}_{\mathcal{M}}$ using two clocks, x and y . Player **Min** is responsible for simulating the behavior of \mathcal{M} , while **Max** is given the opportunity to punish any incorrect simulation. Each punishment by **Max** leads directly to the goal state, thereby terminating the game.

Our construction satisfies the following proposition:

Proposition 1. *Let $\mathcal{M} = (Q, q_i, q_h, T)$ be a deterministic two-counter machine.*

- *If \mathcal{M} does not halt, then the WTG $\mathcal{G}_{\mathcal{M}}$, starting from the configuration $(q_i, \mathbf{0})$, has value at most 64.*
- *If \mathcal{M} halts in at most N steps, then $\mathcal{G}_{\mathcal{M}}$, starting from the configuration $(q_i, \mathbf{0})$, has value at least $64 + \frac{11}{12 \times 30^{5N}}$.*

Intuitively, if \mathcal{M} does not halt, then **Min** can faithfully simulate its infinite execution for an arbitrary number of steps. The longer she plays, the closer the accumulated weight is to 64 when she eventually exits to a goal location.

On the other hand, if \mathcal{M} halts in N steps, **Min** has only two options: either simulate the execution faithfully for at most N steps, and exit, yielding a value strictly greater than 64, or attempt to cheat in order to make the game longer. The key point is that we show that in order to push the computation further, she will eventually cheat by some quantity bounded from below depending on N . Detecting this, **Max** is given the opportunity to enforce a punishment reaching a weight of at least $64 + \frac{11}{12 \times 30^{5N}}$.

In the encoding of \mathcal{M} into $\mathcal{G}_{\mathcal{M}}$, control states of \mathcal{M} become locations of weight 30 in L_{Min} . When entering one of these locations, a faithful encoding of the counters c and d is represented by a clock valuation

$$x = 1 - \frac{1}{2^c 3^d 5^n}, y = 0,$$

where n denotes the number of simulated steps of \mathcal{M} so far.

Note that when $x = 1 - \frac{1}{2^c 3^d 5^n}$, reaching a configuration where $x = 1 - \frac{1}{2^{c+1} 3^d 5^{n+1}}$ (e.g., when simulating an increment of counter c , which also increments n) requires waiting for $\frac{9}{10}(1-x)$ time units. Similarly, to simulate a decrement of counter c , or an increment/decrement of counter d , or a simple increment of n , one must wait $\alpha \cdot (1-x)$ time units, where α is one of $\frac{3}{5}$, $\frac{14}{15}$, $\frac{2}{5}$ or $\frac{4}{5}$, respectively.

Hence, each increasing transition $(q, e, q') \in T$ is simulated using a module of the following structure: **Min** selects a delay to update x , and then **Max** is given an opportunity to punish her if the delay is incorrect (see Fig. 2³).

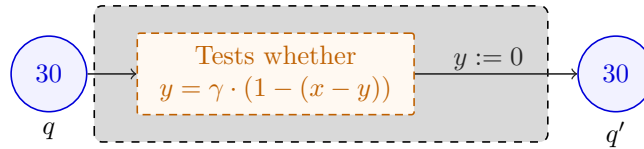


Fig. 2. Transition module for increments. Here $x - y$ is the previous value of x , and y contains the delay **Min** waited in q .

A branching transition $(q, e, q_z, q_{nz}) \in T$, where $q, q_z, q_{nz} \in Q$ and $e \in \{c, d\}$, is simulated in two steps: first, **Min** decides whether to move toward q_z or toward q_{nz} . In both cases, **Max** is given the opportunity to punish if the encoding in x is not valid with respect to **Min**'s choice. Then, **Min** updates x , and **Max** is again given the opportunity to punish her for an invalid update.

³ In this paper, we follow the conventions below to represent WTGs: Blue circles represent locations controlled by **Min**. Red squares represent locations controlled by **Max**. Green circles are goal locations. Grey rectangles represent modules (i.e., subgames): a transition entering a module transfers control to its starting location. Modules may have outgoing edges. Orange rectangles provide the specification of modules. Numbers attached to locations denote their respective weights. Numbers in grey boxes attached to arrows denote the weight of the corresponding transition. Transitions without such boxes have weight 0. Green boxes attached to transitions are comments (or assertions) on the values of the clocks, which hold upon taking the transition. These may be complemented by orange boxes, which represent assertions on the corresponding cost incurred. Some locations are decorated with a numbered flag for ease of reference in proofs.

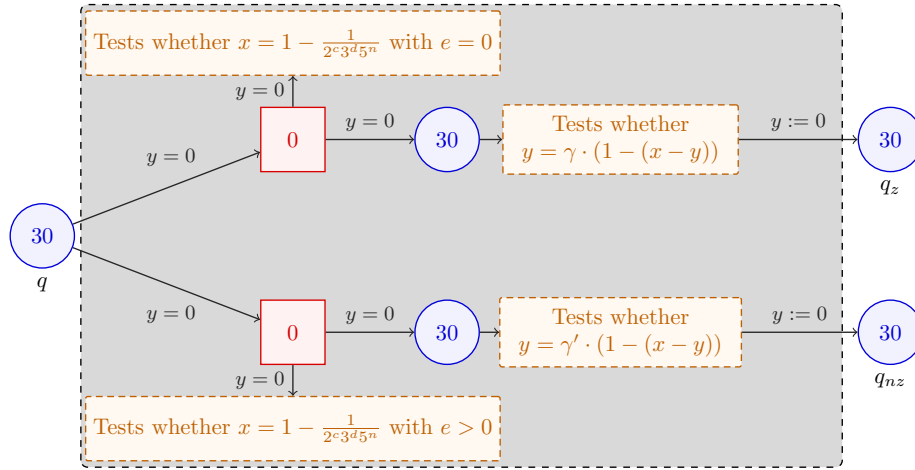


Fig. 3. Transition module for branching decrement of counter $e \in \{c, d\}$.

Finally, for every $q \in Q$ (including q_h), we add the following exit module for **Min**:

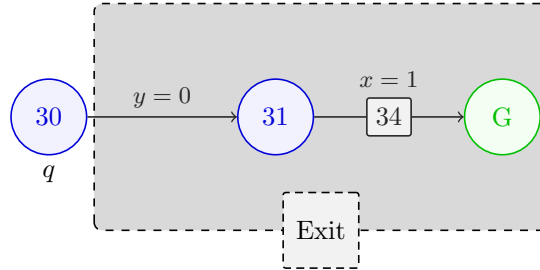


Fig. 4. **Min**'s exit module. Yields a final cost of at most $64 + 1 - x$.

Note that, while faithfully simulating an infinite execution of \mathcal{M} , x will approach 1 arbitrarily closely, since the encoding $\frac{1}{2^e 3^d 5^n}$ tends to 0 as the number of steps n increases. Therefore, when **Min** decides to exit, the accumulated cost will be $30x + 31(1 - x) + 34 = 64 + 1 - x$.

In Section 3.2, we present the punishment module that enforces $y = \alpha(1 - x)$. Section 3.3 then introduces punishment modules for cases where **Min** cheats on branching decisions. Finally, in Section 3.4, we combine these modules to construct $\mathcal{G}_{\mathcal{M}}$ and prove Proposition 1.

3.2 Controlling counter evolution

Let us introduce the gadget behind the punishment module “Punish if $y \neq \alpha(1 - x)$ ”: the *CEC* (Counter Evolution Control) module. The module $CEC_{\alpha,\beta}^M(x, y)$, depicted in Fig. 5, requires that $0 \leq y \leq x < 1$. We will denote the initial values of x and y as $a + b$ and b , respectively.

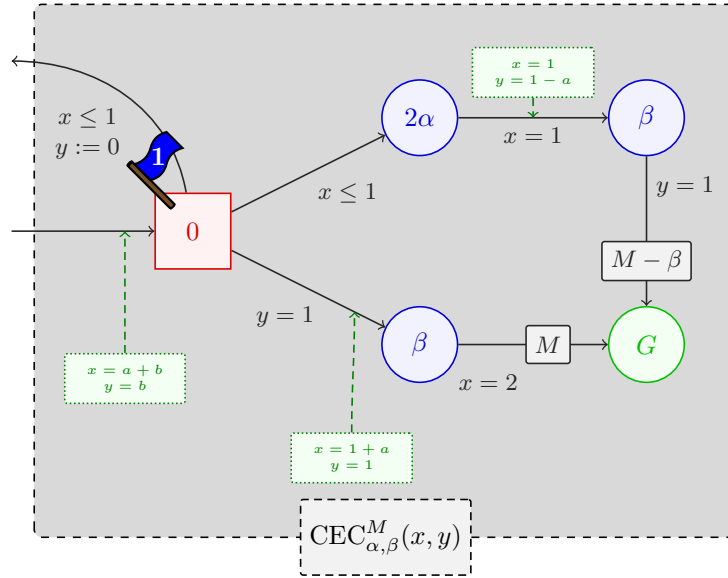




Fig. 5. The CEC (Counter Evolution Control) module. Enforces $b = \gamma(1 - a)$ with $\gamma = 1 - \frac{\beta}{\alpha}$.

Proposition 2. Let $0 \leq \beta < \alpha$. Let t be the time **Max** spends in state  of $CEC_{\alpha,\beta}^M(x, y)$. Provided that, upon entering the state  in $CEC_{\alpha,\beta}^M(x, y)$, $(x, y) = (a + b, b)$ with $0 \leq b \leq a + b \leq 1$, and assuming that the overall cost accumulated so far is $\alpha(a + b) + E$, then **Max** can either end the game with a final cost of

$$\alpha \left(1 + \left| b - \left(1 - \frac{\beta}{\alpha} \right) (1 - a) \right| \right) + E + M$$

or exit the module with $x = a + b + t$, $y = 0$ and accumulated cost $\alpha(a + b + t) + (E - \alpha t)$.

Proof. **Max** can either:

1. exit the module after waiting t time units.
2. take the upper path, which ends the game with cost

$$\begin{aligned} & (\alpha - \beta)(1 - a) - ab - 2\alpha t + \alpha + \beta + M - \beta + E \\ & = (\beta - \alpha)a - \alpha b - 2\alpha t + 2\alpha + M - \beta + E. \end{aligned}$$

3. take the lower path, which ends the game with cost

$$\alpha b - (\alpha - \beta)(1 - a) + \alpha + M + E = \alpha b - (\beta - \alpha)a + \beta + M + E. \quad \square$$

Thus, if we take $\alpha = 30$ and $\beta \in \{18, 12, 6, 3, 2\}$, the value of b minimising the cost of **Max** reaching the goal location is indeed

$$b = \gamma(1 - a) \text{ for } \gamma \in \left\{ \frac{2}{5}, \frac{3}{5}, \frac{4}{5}, \frac{9}{10}, \frac{14}{15} \right\}.$$

For $\alpha = 30$, $E \leq 0$ and $M = 34$, note that when $b = \gamma(1 - a)$ the cost of **Max**'s punishment yields cost at most 64, thus **Max** has no incentive to punish **Min** when she has not cheated.

3.3 Controlling whether a counter is zero

We now address the case in which **Min** has moved to the wrong state when simulating a zero-test. To handle this, depending on the situation, **Max** has access to either a control-if-zero (*ZC*) or control-if-not-zero (*NZC*) module, which checks whether a counter is indeed zero or non-zero by forcing a series of multiplications that can reach 1 only if **Min** chose the right branch. We begin by presenting the module that controls the multiplication (*MC*), depicted in Fig. 6

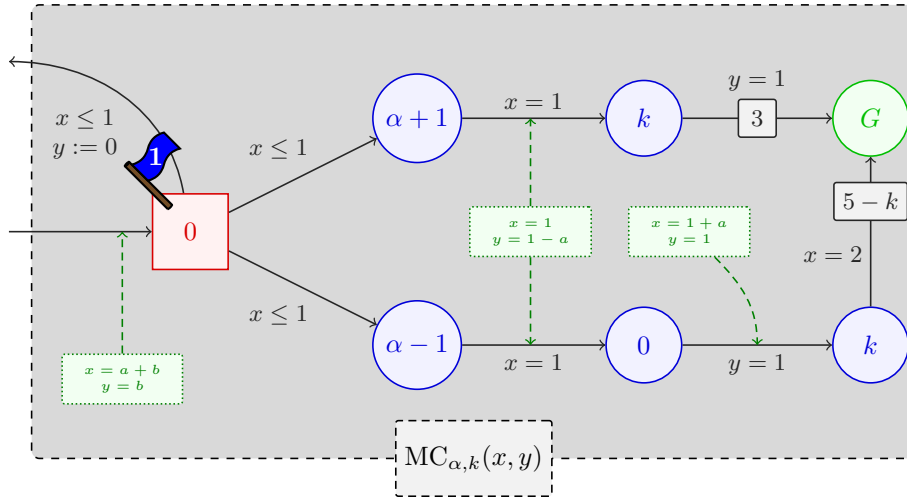




Fig. 6. The *MC* (Multiplication Control) module. Enforces $x = ka$ for $k \in \{2, 3, 5\}$.

$MC_{\alpha,k}(x, y)$ operates similarly to the CEC module:

- It requires that $0 \leq y \leq x < 1$. We will denote the initial values of x and y as $a + b$ and b , respectively.
- Whereas the CEC module enforces that $b = \gamma(1 - a)$ for some γ , $MC_{\alpha,k}(x, y)$ ensures that $a + b = ka$. This is because the MC module is designed to multiply encodings of the form $\frac{1}{2^c 3^d 5^n}$ by $k \in 2, 3, 5$.

Proposition 3. *Let $0 \leq \beta \leq \alpha$. Let t be the time **Max** spends in state  of $MC_{\alpha,k}(x, y)$. Provided that, upon entering the state  in $MC_{\alpha,k}(x, y)$, $(x, y) = (a + b, b)$ with $0 \leq b \leq a + b \leq 1$, and assuming that the overall cost accumulated so far is $\alpha(a + b) + E$, **Max** can either end the game with final cost*

$$\alpha + 4 + E + k + |b - (k - 1)a|$$

or exit the module with $x = a + b + t$, $y = 0$ and an accumulated cost of $\alpha(a + b + t) + (E - \alpha t)$.

Proof. **Max** can either:

1. exit the module after waiting t time units.
2. take the upper path, which ends the game with cost

$$\alpha(a + b) + E + (\alpha + 1)(1 - a - b - t) + ka + 3 = \alpha + 4 + E - (\alpha + 1)t - b + (k - 1)a.$$

3. take the lower path, which ends the game with cost

$$\begin{aligned} & \alpha(a + b) + E + (\alpha - 1)(1 - a - b - t) + k(1 - a) + 5 - k \\ & = \alpha + 4 + E - (\alpha - 1)t + b + k - (k - 1)a. \quad \square \end{aligned}$$

Here we see clearly that the MC module enforces multiplication: the cost of **Max** ending the game is minimised for $b = (k - 1)a$, hence $x = a + b = ka$.

We now introduce modules to control whether a counter is indeed 0 or not.

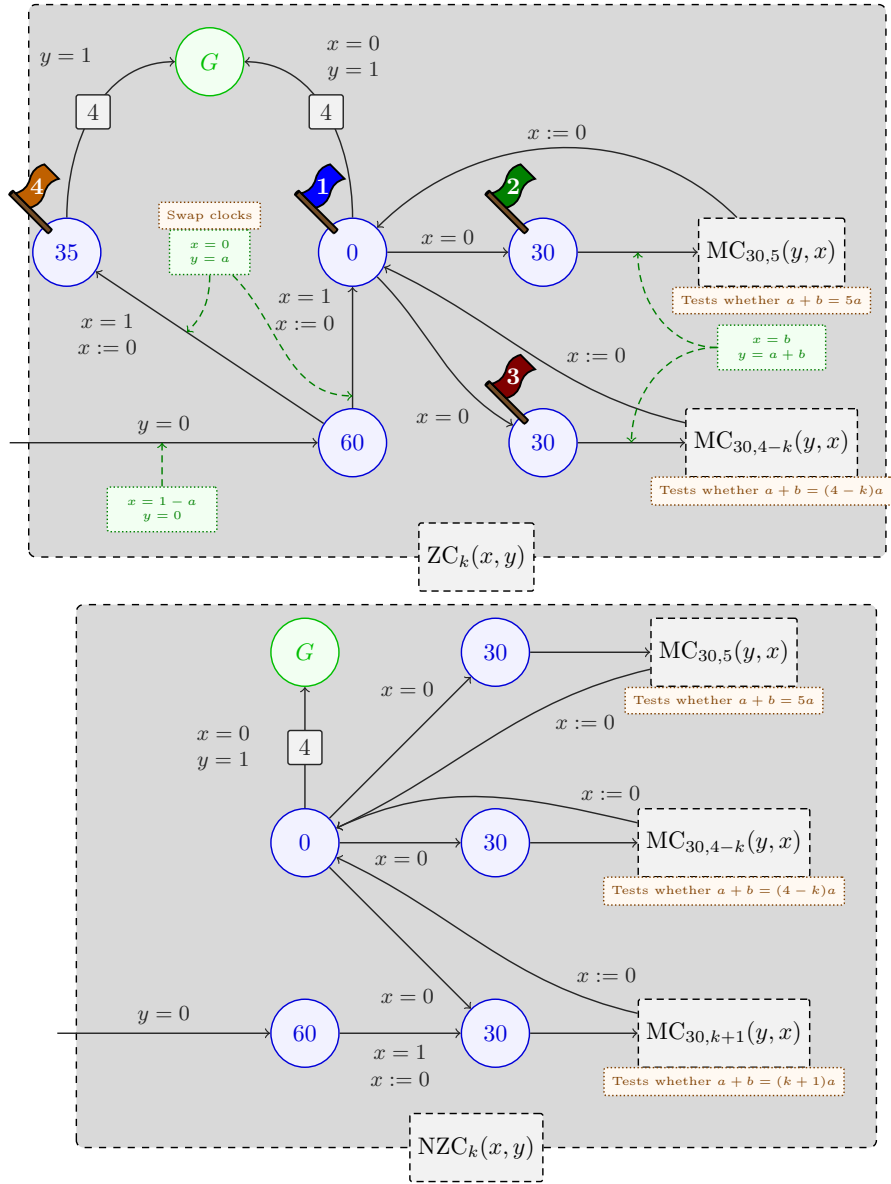


Fig. 7. The Zero-Control and Non-Zero-Control modules, for $k \in \{1, 2\}$.

In these two modules, note that MC is always invoked as $MC(y, x)$, i.e., by swapping the roles of the two clocks. This is because it is easier to translate the

encoding $(1 - a, 0)$ into $(0, a)$ rather than $(a, 0)$. This translation is the first step in both modules.

Proposition 4. *Let $k \in \{1, 2\}$. Let $1 - a \in [0, 1)$ be the initial value of x upon entering the module $ZC_k(x, y)$. Let $30(1 - a) + E$ be the overall cost accumulated to date upon entering the module. Let*

$$\mu = \min \left\{ \left| \frac{1}{(4 - k)^d 5^n} - a \right| : d, n \in \mathbb{N} \right\}.$$








Then

- **Min** has a strategy that ensures a final cost of at most $64 + E + 5\mu$.
- **Max** has a strategy that ensures a final cost of at least $64 + E + \mu$.

Proof sketch. (See the appendix for the detailed proof.)

In the ZC module, a valid encoding is any valuation for y of the form $\frac{1}{(4 - k)^d 5^n}$ with $d, n \in \mathbb{N}$.

If **Min** follows the strategy below, she ensures a final cost of at most $64 + E + 5\mu$:




- First, if a is closer to 1 than to $\frac{1}{4 - k}$ (the largest valid encoding), then **Min** takes the transition to State ; otherwise, she moves to State .
- From State , let $\frac{1}{(4 - k)^d 5^n}$ be the valid encoding closest to a . If $n \geq 1$, move to State  and wait until y reaches $\frac{1}{(4 - k)^d 5^{n-1}}$. Similarly, if $d \geq 1$, move to State  and wait until y reaches $\frac{1}{(4 - k)^{d-1} 5^n}$. Finally, if the closest valid encoding is 1 but $y < 1$, wait in State  until y reaches 1.
- When $y = 1$ in State , exit to the goal location.

Under this strategy, **Min** always updates y to a valid encoding. Hence, any “error” in the MC module arises either from the initial deviation μ or from a delay introduced by **Max** in the previous step. If **Max** punishes the initial deviation, the resulting cost is at most $64 + E + 5\mu$. If **Max** punishes a later deviation, the corresponding punishment cost is offset by the “lost” cost incurred by **Max** while waiting in a zero-weight location in the previous step. Indeed, when only **Min** delays, the accumulated weight is $30 + E + 30x$ for the current value of clock x ; any delay introduced by **Max** subtracts from this $30x$. Therefore, by punishing his own mistakes, **Max** can only obtain a total cost of at most $64 + E$.

Conversely, if **Max** follows the strategy below, he ensures a final cost of at least $64 + E + \mu$:

Upon entering a MC module with clock valuation (x, y) , let

$$\eta = |x - (k' - 1)(y - x)|,$$

where $k' = 5$ or $4 - k$ depending on whether the module is entered from State  or State . Then, if $\eta \geq \mu$, then punish along the path in the MC module that maximises the cost. Otherwise, return to State  without delay.

We claim that, in order to reach valuation $(0, 1)$, **Min** must eventually incur an error of magnitude $\eta \geq \mu$ (see Claim 1 in appendix), at which point **Max** can punish her, resulting in a final cost of at least $64 + E + \mu$. \square

Proposition 5. *Let $k \in \{1, 2\}$. Let $1 - a \in [0, 1)$ be the initial value of x upon entering module $NZC_k(x, y)$. Let $30(1 - a) + E$ be the overall cost accumulated thus far when entering the module. Let*

$$\mu = \min \left\{ \left| \frac{1}{(k+1)^c (4-k)^d 5^n} - a \right| : c, d, n \in \mathbb{N} \quad c > 0 \right\}.$$

Then

- **Min** has a strategy that ensures a final cost of at most $64 + E + 5\mu$.
- **Max** has a strategy that ensures a final cost of at least $64 + E + \mu$.

Proof. The proof is almost identical to that of Prop. 4. The main difference between modules ZC_k and NZC_k is that NZC_k forces **Min** to do a multiplication by $k + 1$ before multiplying by $k + 1$, $4 - k$, and 5 arbitrarily many times. \square

3.4 Combining modules

We can now implement explicitly the modules given in Figures 2 and 3 with the CEC , ZC and NZC modules:

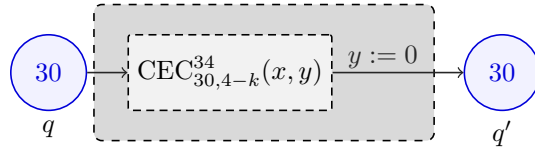


Fig. 8. Transition module simulating an increment transition $(q, e, q') \in T$, with $q, q' \in Q$. Here $k = 1$ or 2 when $e = c$ or d , respectively.

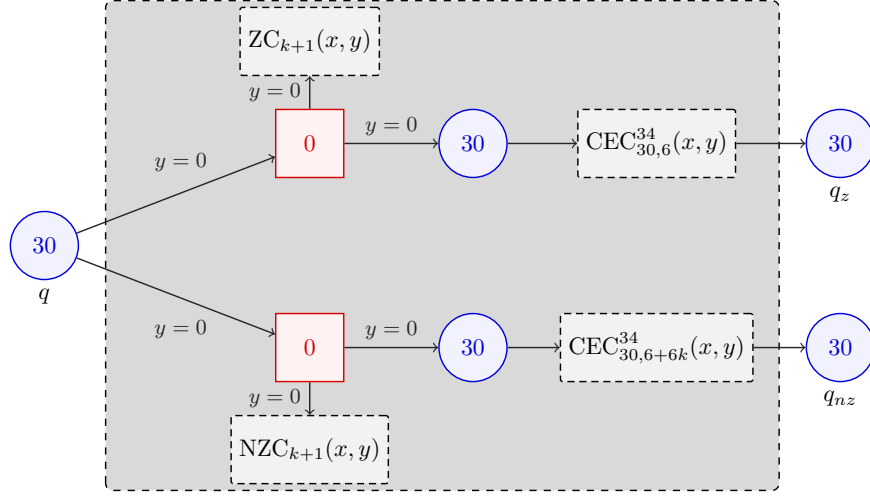


Fig. 9. Transition module simulating a branching transition $(q, e, q_z, q_{nz}) \in T$, with $q, q_z, q_{nz} \in Q$. Here $k = 1$ or 2 when $e = c$ or d , respectively.

Proposition 1. Let $\mathcal{M} = (Q, q_i, q_h, T)$ be a deterministic two-counter machine.

- If \mathcal{M} does not halt, then the WTG $\mathcal{G}_{\mathcal{M}}$, starting from the configuration $(q_i, \mathbf{0})$, has value at most 64.
- If \mathcal{M} halts in at most N steps, then $\mathcal{G}_{\mathcal{M}}$, starting from the configuration $(q_i, \mathbf{0})$, has value at least $64 + \frac{11}{12 \times 30^{5N}}$.

Proof sketch. (See the appendix for the detailed proof.)

If \mathcal{M} does not halt, then **Min** can faithfully simulate the execution of \mathcal{M} for an arbitrary number of steps and then exit via the exit module with x arbitrarily close to 1. The resulting accumulated cost is at most $64 + 1 - x$, which tends to 64 as $x \rightarrow 1$. If **Max** decides to punish **Min** during this run, the punishment module yields a cost of at most 64.

Note that **Max** can perturb the simulation by waiting in the transition modules; however, this is disadvantageous for **Max**. Let a_p and a_{p+1} be the encodings of the counters of \mathcal{M} at steps p and $p+1$. Assume that **Min** has just updated x to a_p , with accumulated cost at most $30a_p$, and that **Max** waits for some $\varepsilon_p > 0$. Then:

- If $a_p + \varepsilon_p \leq a_{p+1}$, **Min** can update x to a_{p+1} with accumulated cost at most $30(a_{p+1} - \varepsilon_p) < 30a_{p+1}$. In effect, **Max** has waited ε_p in a location of weight 0 instead of allowing **Min** to realize the same delay in her location of weight 30, strictly decreasing the total cost. If **Max** then punishes her, the extra cost of the *CEC* module will be offset by the time **Max** wasted waiting ε_p in a weight-0 location.

- If $a_p + \varepsilon_p > a_{p+1}$, **Min** can no longer update x to the next encoding. In that case she can take the exit module, yielding an accumulated cost

$$64 - 30\varepsilon_p + 1 - (a_p + \varepsilon_p) \leq 64.$$

The extra cost of leaving early is offset by the time **Max** wasted waiting ε_p in a weight-0 location.

Conversely, if \mathcal{M} halts in at most N steps, consider the following strategy for **Max**: never wait in the transition modules (as we have just seen, waiting is not to his advantage), and punish **Min** whenever she makes an error of at least $\frac{1}{30^{5N+1}}$ while updating the clocks, or when she takes the wrong transition in a zero-test. Then:

- If **Min** faithfully simulates the execution of \mathcal{M} , she is forced to exit in at most N simulation steps, with accumulated cost $\geq 64 + \frac{11}{12 \times 30^{5N}}$.
- To avoid this outcome, she may attempt to cheat to obtain an arbitrarily long run. However, she has only N simulation steps in which to distribute her cheating. Hence we claim that either she makes an error of at least $\frac{1}{30^{5N+1}}$ while updating the clocks, or she cheats on a zero-test when the value of clock x is within $\frac{1}{12 \times 30^{5N}}$ of a faithful encoding of the simulation (see Claim 2 in appendix). In either case, **Max** can punish her, yielding a final accumulated cost $\geq 64 + \frac{11}{12 \times 30^{5N}}$. \square

Theorem 1. *The Value Problem for two-player, turn-based, time-bounded, two-clock, weighted timed games with non-negative integer weights is undecidable. The same holds for weighted timed games over unbounded time otherwise satisfying the same hypotheses.*

Proof. Let \mathcal{M} be a deterministic two-counter machine. Note that in $\mathcal{G}_{\mathcal{M}}$, outside of control modules, clock x is never reset and always upper-bounded by 1. Therefore any play has duration at most 1 time unit plus the total time spent in the control modules, which is at most 2 time units. In other words, by construction the WTG $\mathcal{G}_{\mathcal{M}}$ requires at most 3 time units for any execution. Prop. 1 asserts that the halting problem for a deterministic two-counter machine reduces to the Value problem for 2-clock weighted timed games, which concludes the proof. \square

Theorem 2. *The Existence Problem for two-player, turn-based, time-bounded, two-clock, weighted timed games with non-negative integer weights is undecidable. The same holds for weighted timed games over unbounded time otherwise satisfying the same hypotheses.*

Proof sketch. Let \mathcal{M} be a deterministic two-counter machine. We define $\mathcal{G}_{\mathcal{M}}^E$ similarly to $\mathcal{G}_{\mathcal{M}}$, except that we add a “soft-exit” module, reachable from the halting state of \mathcal{M} . This soft-exit module is simply the exit module where the

location cost of 31 has been replaced by 30. Then **Min** has a strategy to enforce a cost of at most 64 if and only if \mathcal{M} halts.

Indeed, if \mathcal{M} halts, then **Min** can reach a halting state without cheating (i.e., she faithfully simulates \mathcal{M}), and exits at cost 64 through a soft-exit module. If **Max** decides to exit through a *CEC* or *MC* module, this also yields a cost of at most 64. As seen in Prop. 1, every delay taken by **Max** in a *CEC* or *MC* module is a net negative for **Max**. On the other hand, if \mathcal{M} does not halt, the only way for **Min** to exit the game through a soft-exit module is to cheat in order to reach a halting state. The normal exit module for **Min** yields cost strictly greater than 64. Consider the strategy where **Max** punishes any cheating by exiting the game. Then either **Min** never cheats, and the game never ends, thereby incurring cost $+\infty$, or **Min** cheats and is punished, in which case the game ends with cost strictly above 64. \square

4 Commentary

How does this reduction compare to the reduction to 3-clock WTGs with positive weights in [5]?

- First, in [5], the encoding is of the form $\frac{1}{2^c 3^d}$. Clocks x and y are used to store the previous and new counter encodings at each step (they alternate). A third clock, t , acts as a ticking clock, ensuring that exactly n time units are spent in a module, for some $n \in \mathbb{N}$. This allows, for instance, **Min** to assign a new value to y while keeping x unchanged. Our encoding $1 - \frac{1}{2^c 3^d 5^n}$ has the advantage that it only increases, allowing us to update it without losing the previous value, even without a ticking clock.
- A key property of the 3-clock construction is that it yields an *almost non-Zeno* WTG $\mathcal{G}_{\mathcal{M}}$:

Definition 4. *A WTG \mathcal{G} with non-negative weights is said to be non-Zeno if all its cycles⁴ have weight at least 1. It is said to be almost non-Zeno if all its cycles have weight either exactly 0 or at least 1.*

Indeed, all cycles of $\mathcal{G}_{\mathcal{M}}$ lie in the part of the game simulating the execution of \mathcal{M} ; once we enter a punishment module, no further cycles are possible. In the three-clock reduction, the only positive weights occur in the punishment modules.

However, in our two-clock construction, $\mathcal{G}_{\mathcal{M}}$ is not almost non-Zeno. In fact, it was recently shown in [20] that the Value problem is decidable for almost non-Zeno two-clock WTGs with positive weights.

Let us now give a simple intuition for why the two-clock reduction requires weight > 0 in each location of $Q \subseteq L_{\mathbf{Min}}$: When entering a *CEC* module with clock configuration $(x, y) = (a + b, b)$, we want to offer **Max** a punishment module whose weight function is of the form $|b - \gamma(1 - a)|$ (modulo some multiplicative and additive constants), for some $\gamma \in \mathbb{Q}$.

⁴ Here, we refer to cycles in the *region game* of \mathcal{G} ; see [5] for the definition.

It is straightforward to construct a punishment module with weight functions of the form $k \cdot (1 - a - b)$ or $k \cdot a$ (for $k \in \mathbb{N}$), and to combine them using \max and $+$ operations. However, we cannot obtain a function of the form $k \cdot b$ directly. Therefore, b must be reflected in the accumulated weight *before* entering the punishment module. This can only happen by taking in weight when **Min** waits b time units.

- Our construction is timed-bounded, while the three-clock construction is not (every simulation step takes an integer amount of time). This is due to our $1 - \frac{1}{2 \cdot 3^d \cdot 5^n}$ encoding, but also to the constraint above: since **Min** accumulates weight while updating the clocks, we must ensure that the time spent on clock updates remains bounded.

References

1. Alur, R., Bernadsky, M., Madhusudan, P.: Optimal reachability for weighted timed games. In: Proceedings of the 31st International Colloquium on Automata, Languages and Programming (ICALP'04). pp. 122–133. Springer Berlin Heidelberg, Berlin, Heidelberg (2004)
2. Alur, R., Henzinger, T.A.: Modularity for timed and hybrid systems. In: CONCUR. Lecture Notes in Computer Science, vol. 1243, pp. 74–88. Springer (1997)
3. Bouyer, P., Brihaye, T., Markey, N.: Improved undecidability results on weighted timed automata. Information Processing Letters **98**(5), 188–194 (2006). <https://doi.org/https://doi.org/10.1016/j.ip1.2006.01.012>, <https://www.sciencedirect.com/science/article/pii/S0020019006000652>
4. Bouyer, P., Cassez, F., Fleury, E., Larsen, K.G.: Optimal strategies in priced timed game automata. In: FSTTCS 2004: Foundations of Software Technology and Theoretical Computer Science. pp. 148–160. Springer Berlin Heidelberg, Berlin, Heidelberg (2005)
5. Bouyer, P., Jaziri, S., Markey, N.: On the Value Problem in Weighted Timed Games. In: Aceto, L., de Frutos Escrig, D. (eds.) Proceeding of the 26th International Conference on Concurrency Theory (CONCUR 2015). Leibniz International Proceedings in Informatics (LIPIcs), vol. 42, pp. 311–324. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany (2015). <https://doi.org/10.4230/LIPIcs.CONCUR.2015.311>, <https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.CONCUR.2015.311>
6. Bouyer, P., Larsen, K.G., Markey, N., Rasmussen, J.I.: Almost optimal strategies in one clock priced timed games. In: Arun-Kumar, S., Garg, N. (eds.) FSTTCS 2006: Foundations of Software Technology and Theoretical Computer Science, 26th International Conference, Kolkata, India, December 13-15, 2006, Proceedings. Lecture Notes in Computer Science, vol. 4337, pp. 345–356. Springer (2006). https://doi.org/10.1007/11944836_32, https://doi.org/10.1007/11944836_32
7. Brihaye, T., Bruyère, V., Raskin, J.F.: On optimal timed strategies. In: Formal Modeling and Analysis of Timed Systems. pp. 49–64. Springer Berlin Heidelberg, Berlin, Heidelberg (2005)
8. Brihaye, T., Doyen, L., Geeraerts, G., Ouaknine, J., Raskin, J., Worrell, J.: On reachability for hybrid automata over bounded time. In: Aceto, L., Henzinger, M., Sgall, J. (eds.) Automata, Languages and Programming - 38th International Colloquium, ICALP 2011, Zurich, Switzerland, July 4-8, 2011, Proceedings, Part

- II. Lecture Notes in Computer Science, vol. 6756, pp. 416–427. Springer (2011). https://doi.org/10.1007/978-3-642-22012-8_33, https://doi.org/10.1007/978-3-642-22012-8_33
9. Brihaye, T., Doyen, L., Geeraerts, G., Ouaknine, J., Raskin, J., Worrell, J.: Time-bounded reachability for monotonic hybrid automata: Complexity and fixed points. In: Hung, D.V., Ogawa, M. (eds.) Automated Technology for Verification and Analysis - 11th International Symposium, ATVA 2013, Hanoi, Vietnam, October 15–18, 2013. Proceedings. Lecture Notes in Computer Science, vol. 8172, pp. 55–70. Springer (2013). https://doi.org/10.1007/978-3-319-02444-8_6, https://doi.org/10.1007/978-3-319-02444-8_6
 10. Brihaye, T., Geeraerts, G., Narayanan Krishna, S., Manasa, L., Monmege, B., Trivedi, A.: Adding negative prices to priced timed games. In: Baldan, P., Gorla, D. (eds.) CONCUR 2014 – Concurrency Theory. pp. 560–575. Springer Berlin Heidelberg, Berlin, Heidelberg (2014)
 11. Busatto-Gaston, D., Monmege, B., Reynier, P.: Optimal controller synthesis for timed systems. *Log. Methods Comput. Sci.* **19**(1) (2023)
 12. Guilment, Q., Ouaknine, J.: Inapproximability in Weighted Timed Games. In: Majumdar, R., Silva, A. (eds.) 35th International Conference on Concurrency Theory (CONCUR 2024). Leibniz International Proceedings in Informatics (LIPIcs), vol. 311, pp. 27:1–27:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany (2024). <https://doi.org/10.4230/LIPIcs.CONCUR.2024.27>, <https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.CONCUR.2024.27>
 13. Henzinger, T.A., Horowitz, B., Majumdar, R.: Rectangular hybrid games. In: CONCUR. Lecture Notes in Computer Science, vol. 1664, pp. 320–335. Springer (1999)
 14. Jenkins, M., Ouaknine, J., Rabinovich, A., Worrell, J.: Alternating timed automata over bounded time. In: Proceedings of the 25th Annual IEEE Symposium on Logic in Computer Science, LICS 2010, 11–14 July 2010, Edinburgh, United Kingdom. pp. 60–69. IEEE Computer Society (2010). <https://doi.org/10.1109/LICS.2010.45>, <https://doi.org/10.1109/LICS.2010.45>
 15. Maler, O., Pnueli, A., Sifakis, J.: On the synthesis of discrete controllers for timed systems (an extended abstract). In: STACS. Lecture Notes in Computer Science, vol. 900, pp. 229–242. Springer (1995)
 16. Minsky, M.L.: Computation: Finite and Infinite Machines. Prentice-Hall Series in Automatic Computation, Prentice-Hall (1967)
 17. Monmege, B., Parreaux, J., Reynier, P.A.: Decidability of One-Clock Weighted Timed Games with Arbitrary Weights. In: Klin, B., Lasota, S., Muscholl, A. (eds.) 33rd International Conference on Concurrency Theory (CONCUR 2022). Leibniz International Proceedings in Informatics (LIPIcs), vol. 243, pp. 15:1–15:22. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany (2022). <https://doi.org/10.4230/LIPIcs.CONCUR.2022.15>, <https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.CONCUR.2022.15>
 18. Ouaknine, J., Rabinovich, A., Worrell, J.: Time-bounded verification. In: Bravetti, M., Zavattaro, G. (eds.) CONCUR 2009 - Concurrency Theory, 20th International Conference, CONCUR 2009, Bologna, Italy, September 1–4, 2009. Proceedings. Lecture Notes in Computer Science, vol. 5710, pp. 496–510. Springer (2009). https://doi.org/10.1007/978-3-642-04081-8_33, https://doi.org/10.1007/978-3-642-04081-8_33
 19. Ouaknine, J., Worrell, J.: Towards a theory of time-bounded verification. In: Abramsky, S., Gavioille, C., Kirchner, C., auf der Heide, F.M., Spirakis, P.G. (eds.)

- Automata, Languages and Programming, 37th International Colloquium, ICALP 2010, Bordeaux, France, July 6–10, 2010, Proceedings, Part II. Lecture Notes in Computer Science, vol. 6199, pp. 22–37. Springer (2010). https://doi.org/10.1007/978-3-642-14162-1_3, https://doi.org/10.1007/978-3-642-14162-1_3
20. Vialard, I.: Deciding the value of two-clock almost non-zero weighted timed games (2025), <https://arxiv.org/abs/2508.00014>

A Detailed proofs






Proposition 4. *Let $k \in \{1, 2\}$. Let $1 - a \in [0, 1)$ be the initial value of x upon entering the module $ZC_k(x, y)$. Let $30(1 - a) + E$ be the overall cost accumulated to date upon entering the module. Let*

$$\mu = \min \left\{ \left| \frac{1}{(4 - k)^d 5^n} - a \right| : d, n \in \mathbb{N} \right\}.$$

Then

- **Min** has a strategy that ensures a final cost of at most $64 + E + 5\mu$.
- **Max** has a strategy that ensures a final cost of at least $64 + E + \mu$.

Detailed proof. Let $p \in \mathbb{N} \setminus \{0\}$. We introduce the following quantities:

- a_p is the value of clock y upon entering  for the p -th time (if it exists).
- C_p is the accumulated cost so far upon entering  for the p -th time (if it exists).
- t_p is the time spent by **Min** in one of the states  or  upon leaving  for the p -th time (if it exists).
- ε_p is the time **Max** waits in one of the locations of MC controlled by him for the p -th time (if it exists).
- $E_p = C_p - 30(1 + a_p)$ is the difference between the actual and expected costs.

We have the following initial conditions:

$$a_1 = a \quad C_1 = 30(1 + a) + E_1 \quad \text{and} \quad E_1 = E.$$

When everything is well defined we have the following recurrence relations:

$$a_{p+1} = a_p + t_p + \varepsilon_p \quad C_{p+1} = C_p + 30t_p \quad \text{and} \quad E_{p+1} = E_p - 30\varepsilon_p.$$

Overall we have


$$a_p = a + \sum_{q=0}^{p-1} (t_q + \varepsilon_q) \quad E_p = E - 30 \left(\sum_{q=0}^{p-1} \varepsilon_q \right) \quad \text{and} \quad C_p = 30(1 + a_p) + E_p.$$


⁵ $30(1 + a_p)$ is the accumulated cost in a run where **Max** has never waited a delay > 0 .

– Let us prove the first assertion of the proposition. Let d_p and n_p be integers that minimise $\mu_p \stackrel{\text{def}}{=} |\delta_p|$ for $\delta_p \stackrel{\text{def}}{=} \frac{1}{(4-k)^{d_p} 5^{n_p}} - a_p$. Now consider the following strategy for **Min**:

If $\frac{1}{2} \left(\frac{1}{4-k} + 1 \right) \leq a \leq 1$, i.e. a is closer to 1 than to another valid encoding,

then **Min** takes the transition to State . Otherwise **Min** moves to State .

If **Min** is in  for the p -th time then:



1. If $n_p \geq 1$ then go to state  and wait $t_p = 4a_p + 5\delta_p$. Doing so, y has value $\frac{1}{(4-k)^{d_p} 5^{n_p-1}}$ upon entering the *MC* module. Note that $t_p \geq 0$, since otherwise

$$\frac{1}{(4-k)^{d_p} 5^{n_p}} < \frac{1}{(4-k)^{d_p} 5^{n_p-1}} < a_p$$

and thus

$$\left| \frac{1}{(4-k)^{d_p} 5^{n_p-1}} - a_p \right| < \mu_p$$


which is a contradiction.

2. If $n_p = 0$ and $d_p \geq 1$ then go to State  and wait $t_p = (3-k)a_p + (4-k)\delta_p$. Doing so, y has value $\frac{1}{(4-k)^{d_p-1}}$ upon entering the *MC* module. Note that for the same reason as earlier, $t_p \geq 0$.
3. If $d_p = n_p = 0$ and $a_p < 1$, then go to state  and wait $1 - a_p$.
4. If $a_p = 1$, then go to the goal state.

Note that following this strategy, **Min** always selects t_p in such a way that $a_p + t_p$ is of the form $\frac{1}{(4-k)^d 5^n}$. Hence if $p > 1$ then

$$\varepsilon_{p-1} \in \left\{ \left| \frac{1}{(4-k)^d 5^n} - a_p \right| : d, n \in \mathbb{N} \right\}.$$

Therefore $\mu_p \leq \varepsilon_{p-1}$. We then have the following alternatives:

- **Min** goes directly to state . In this case, we have $a \geq \frac{1}{2} \left(1 + \frac{1}{4-k} \right)$, which entails that $\mu = 1 - a$. Hence the final cost is

$$30(1-a) + E + 60a + 35(1-a) + 4 = 64 + E + 5(1-a) = 64 + E + 5\mu.$$

- If the game ends via Case 4 of the above strategy then the final cost is exactly

$$64 + E_p = 64 + E - 30 \sum_{q=0}^{p-1} \varepsilon_q \leq 64 + E + 5\mu.$$



- **Max** decides to end the game after Case 1 of the above strategy. If $p = 1$, using Cor. 3, with optimal play from **Max**, the final cost is exactly

$$64 + E + 5\mu.$$

If $p > 1$, using Cor. 3, with optimal play from **Max**, the final cost is exactly

$$\begin{aligned} 64 + E_p + 5\mu_p &= 64 + E - 30 \sum_{q=0}^{p-1} \varepsilon_q + 5\mu_p \\ &\leq 64 + E - 30 \sum_{q=0}^{p-2} \varepsilon_q - 25\varepsilon_{p-1} \text{ since } \mu_p \leq \varepsilon_{p-1} \\ &\leq 64 + E \leq 64 + E + 5\mu. \end{aligned}$$

- Similarly, if **Max** decides to end the game after Case 2, then the final cost is at most $64 + E + 5\mu$
- If **Max** decides to end the game after Case 3 of the above strategy, then first, note that in this case $p > 1$. Indeed, in Case 3, $d_p = n_p = 0$, which in turn entails that $\frac{5-k}{8-2k} \leq a_p < 1$, i.e., a_p is closer to 1 than to $\frac{1}{4-k}$,

which (for $p = 1$) mandates moving to state  instead of state .

We can also note that this situation can only happen if **Max** waited at least $\frac{3-k}{8-2k}$ time units in the previous round. Indeed, following **Min**'s

strategy, we know that $a_{p-1} + t_{p-1}$ is of the form $\frac{1}{(4-k)^d 5^n}$ with either

$d > 0$ or $n > 0$. Therefore $a_{p-1} + t_{p-1} \leq \frac{1}{4-k}$ and $a_p \geq \frac{5-k}{8-2k}$, hence



$$\varepsilon_{p-1} \geq \frac{3-k}{8-2k}.$$

This means that $|1 - (4-k)a_p| \leq 3-k \leq 30\varepsilon_{p-1}$. In other words, the weight lost by **Max** by waiting ε_{p-1} more than compensates the punishment he can now apply.

Therefore, using Cor. 3, with optimal play from **Max**, the final cost is


$$\begin{aligned} 64 + E_p + |1 - (4-k)a_p| &= 64 + E - 30 \sum_{q=0}^{p-1} \varepsilon_q + |1 - (4-k)a_p| \\ &\leq 64 + E - 30 \sum_{q=0}^{p-2} \varepsilon_q \leq 64 + E \leq 64 + E + 5\mu. \end{aligned}$$

In all the above cases, **Min** ensures a cost of at most $64 + E + 5\mu$.

- Let us prove the second assertion of the proposition. Assume that **Min** has gone to State  instead of . Let us write:

- $k_p = \begin{cases} 5 & \text{if } \mathbf{Min} \text{ chooses State } \color{green}{\blacktriangleright} \text{ at step } p \\ 4 - k & \text{if } \mathbf{Min} \text{ chooses State } \color{red}{\blacktriangleright} \text{ at step } p. \end{cases}$
- $\eta_p = |t_p - (k_p - 1)a_p|$.



Consider the following strategy for **Max**:

1. Always choose $\varepsilon_p = 0$.
2. If **Min** picks t_p such that $\eta_p \geq \mu$, then punish her in the next *MC* module through the path maximising the cost.
3. Otherwise, immediately accept and go back to state .

Claim 1. Either this game never ends, or there exists $p \in \mathbb{N}$ such that $\eta_p \leq \mu$.

Using Cor. 3, if **Max** ends the game in Case 2, he secures a final cost of

$$64 + E_p + \eta_p = 64 + E + \eta_p \geq 64 + E + \mu. \quad \square$$

Proof of Claim 1. Assume that we always have $\eta_p < \mu$, i.e., **Max** never ends the game by himself. Then we will show that **Min** can never achieve the condition $y = 1$ needed to exit from State . For the sake of contradiction, assume that **Min** can exit the game at step P for some $P \in \mathbb{N}$, hence $a_P = 1$. Since $\varepsilon_p = 0$ for all p , and since **Max** always takes the transition back to State , we have $t_p = a_{p+1} - a_p$ for all p . Therefore $\eta_p = |a_{p+1} - k_p a_p|$.

Let us consider the following property for $p \in \{1, \dots, P-1\}$:

$$(\mathcal{P}_p): \quad \left| \prod_{q=1}^p \frac{1}{k_{P-q}} - a_{P-p} \right| < \mu \sum_{m=1}^p \prod_{q=m}^p \frac{1}{k_{P-q}}.$$

We prove this property by induction on p .

- We have $|1 - k_{P-1}a_{P-1}| = |a_P - k_{P-1}a_{P-1}| = \eta_{P-1} < \mu$. This can be rewritten as

$$\left| \frac{1}{k_{P-1}} - a_{P-1} \right| < \frac{\mu}{k_{P-1}},$$

which is exactly (\mathcal{P}_1) .

- Assume that (\mathcal{P}_p) has been proven for some $p \in \{1, \dots, P-2\}$. We have

$$\begin{aligned} & \left| \prod_{q=1}^p \frac{1}{k_{P-q}} - k_{P-(p+1)}a_{P-(p+1)} \right| \\ & \leq \left| \prod_{q=1}^p \frac{1}{k_{P-q}} - a_{P-p} \right| + |a_{P-p} - k_{P-(p+1)}a_{P-(p+1)}| \\ & \leq \left| \prod_{q=1}^p \frac{1}{k_{P-q}} - a_{P-p} \right| + \eta_{P-p} \\ & < \mu \left(\sum_{m=1}^p \prod_{q=m}^p \frac{1}{k_{P-q}} \right) + \mu. \end{aligned}$$

Thus

$$\begin{aligned} \left| \prod_{q=1}^{p+1} \frac{1}{k_{P-q}} - a_{P-(p+1)} \right| &< \mu \left(\sum_{m=1}^p \prod_{q=m}^{p+1} \frac{1}{k_{P-q}} \right) + \frac{\mu}{k_{P-(p+1)}} \\ &= \mu \sum_{m=1}^{p+1} \prod_{q=m}^{p+1} \frac{1}{k_{P-q}}. \end{aligned}$$

This proves (\mathcal{P}_{p+1}) , which concludes our induction.

We have

$$\begin{aligned} \left| \prod_{q=1}^{P-1} \frac{1}{k_{P-q}} - a_1 \right| &< \mu \sum_{m=1}^{P-1} \prod_{q=m}^{P-1} \frac{1}{k_{P-q}} \leq \mu \sum_{m=1}^{P-1} \frac{1}{(4-k)^{P-m}} \\ &\qquad \qquad \qquad < \frac{1}{4-k} \\ &= \mu \underbrace{\sum_{m=1}^{P-1} \frac{1}{(4-k)^m}}_{< \frac{1}{3-k} < 1} < \mu. \end{aligned}$$

But we notice that

$$\left| \prod_{q=1}^{P-1} \frac{1}{k_{P-q}} - a_1 \right| = \left| \prod_{q=1}^{P-1} \frac{1}{k_{P-q}} - a \right| \in \left\{ \left| \frac{1}{(4-k)^{d5^n}} - a \right| : d, n \in \mathbb{N} \right\}.$$

Thus by definition of μ ,

$$\mu \leq \left| \prod_{q=1}^{P-1} \frac{1}{k_{P-q}} - a_1 \right| < \mu, \quad \square$$

a contradiction.

Proposition 1. *Let $\mathcal{M} = (Q, q_i, q_h, T)$ be a deterministic two-counter machine.*

- *If \mathcal{M} does not halt, then the WTG $\mathcal{G}_{\mathcal{M}}$, starting from the configuration $(q_i, \mathbf{0})$, has value at most 64.*
- *If \mathcal{M} halts in at most N steps, then $\mathcal{G}_{\mathcal{M}}$, starting from the configuration $(q_i, \mathbf{0})$, has value at least $64 + \frac{11}{12 \times 30^{5N}}$.*

Detailed proof. We first introduce some key quantities. Let $p \in \mathbb{N} \setminus \{0\}$. We let $(q_p, c_p, d_p)_p$ be the unique sequence of states and values of the two counters along the (deterministic) execution of \mathcal{M} . Here, for all p , $c_p + d_p \leq p$. Recall that \mathcal{L}_{Min} is a superset of Q . We let (\hat{q}_p) be the sequence of states in Q visited along a given play of $\mathcal{G}_{\mathcal{M}}$. Let us write, upon entering a state \hat{q}_p :

- a_p to denote the value of clock x .

- C_p for the accumulated cost so far.
- $E_p = C_p - 30a_p$ for the difference between the actual and expected costs.
- $\mu_p = \min \left\{ \left| 1 - \frac{1}{2^c 3^d 5^p} - a_p \right| : c, d \in \mathbb{N} \quad c + d \leq p \right\}$ to denote the minimal difference between a_p and a valid counter encoding.
- \widehat{c}_p and \widehat{d}_p to stand for natural numbers such that $\widehat{c}_p + \widehat{d}_p \leq p$ and

$$\left| 1 - \frac{1}{2^{\widehat{c}_p} 3^{\widehat{d}_p} 5^p} - a_p \right| = \mu_p.$$

- $\delta_p = 1 - \frac{1}{2^{\widehat{c}_p} 3^{\widehat{d}_p} 5^p} - a_p$.

We have the following initial conditions:

$$a_1 = 0 \quad \mu_1 = \delta_1 = 0 \quad E_1 = 0 \quad \text{and} \quad C_1 = 0.$$

After entering a state \widehat{q}_p , and until visiting the next state \widehat{q}_{p+1} , **Min** can wait in a state of weight 30^6 , immediately followed by a CEC where **Max** can also wait. We let t_p be the time spent by **Min** before the CEC module is entered, and ε_p be the time spent by **Max** in his state of the CEC module before \widehat{q}_{p+1} .

When everything is well defined we have the following recurrence relations:

$$a_{p+1} = a_p + t_p + \varepsilon_p \quad C_{p+1} = C_p + 30t_p \quad \text{and} \quad E_{p+1} = E_p - 30\varepsilon_p.$$

Overall we have

$$a_p = \sum_{q=0}^{p-1} (t_q + \varepsilon_q) \quad E_p = -30 \sum_{q=0}^{p-1} \varepsilon_q \quad \text{and} \quad C_p = 30a_p + E_p = 30 \sum_{q=0}^{p-1} t_q.$$

- Let us prove the first assertion of the proposition. Assume that \mathcal{M} does not halt. Let $N \in \mathbb{N}$. Consider the following strategy for **Min**: upon entering q_p , if $1 - \frac{1}{30^N} < a_p$ or $1 - \frac{1}{2^{c_{p+1}} 3^{d_{p+1}} 5^{p+1}} < a_p$ then take the exit module. Otherwise, **Min** plays in order to reach q_{p+1} , and thus she waits

$$t_p = 1 - \frac{1}{2^{c_{p+1}} 3^{d_{p+1}} 5^{p+1}} - a_p = \delta_p + \frac{1}{2^{c_p} 3^{d_p} 5^p} - \frac{1}{2^{c_{p+1}} 3^{d_{p+1}} 5^{p+1}}.$$

In other words, **Min**'s strategy is to simulate the execution of \mathcal{M} faithfully, until either a_p is $\frac{1}{30^N}$ -close to 1, in which case the total cost is

$$64 + E_p + 1 - a_p \leq \frac{1}{30^N},$$

⁶ This state is either \widehat{q}_p itself, or the next visited location in \mathcal{L}_{Min} when simulating a zero-test.

or **Max** has waited long enough in ε_{p-1} to prevent **Min** from reaching the next step. In this case, the weight lost by waiting ε_{p-1} is enough to compensate the cost of leaving for **Min**: Indeed

$$\begin{aligned}\varepsilon_{p-1} &= a_p - a_{p-1} - t_{p-1} = a_p - \left(1 - \frac{1}{2^{c_p} 3^{d_p} 5^p}\right) \\ &> \frac{1}{2^{c_p} 3^{d_p} 5^p} - \frac{1}{2^{c_{p+1}} 3^{d_{p+1}} 5^{p+1}},\end{aligned}$$

hence $30\varepsilon_{p-1} > \frac{1}{2^{c_{p+1}} 3^{d_{p+1}} 5^{p+1}} > 1 - a_p$. The total cost of **Min** leaving is then

$$64 + E_p + 1 - a_p < 64 - 30 \sum_{q=0}^{p-2} \varepsilon_q \leq 64.$$

Note that, along the game, **Max** gains nothing by punishing **Min**. Indeed, for any $p > 0$ such that the run is defined up to p steps,

$$a_p = a_{p-1} + t_p + \varepsilon_p \quad \text{and} \quad a_{p-1} + t_{p-1} = 1 - \frac{1}{2^{c_p} 3^{d_p} 5^p}.$$

Therefore

$$\varepsilon_{p-1} \in \left\{ \left| 1 - \frac{1}{2^c 3^d 5^p} - a_p \right| : c, d \in \mathbb{N} \quad c + d \leq p - 1 \right\},$$

hence $\varepsilon_{p-1} \geq \mu_p$. This means that any ‘‘error’’ on the encoding at step p must have been introduced by **Max** at step $p - 1$. Thus, according to Prop. 4, 5 and 2, the final weight is at most $64 < 64 + \frac{1}{30^N}$.

Overall, for any $N \in \mathbb{N}$, **Min** can secure a cost of at most $64 + \frac{1}{30^N}$. Thus the value of the game $\mathcal{G}_{\mathcal{M}}$ is at most 64.

– Let us prove the second assertion of the proposition. Assume that \mathcal{M} halts in N steps. Consider the following strategy for **Max**:

1. Always choose $\varepsilon_p = 0$.
2. Immediately punish a wrong transition going to ZC_k^0 or NZC_k^0 if **Min** unfaithfully simulates a zero-test.
3. At step p , accept the transition if $\left| t_p - \left(1 - \frac{\beta}{30}\right) (1 - a_p) \right| < \frac{1}{30^{5N+1}}$ and punish otherwise.

Note that, in particular, $E_p = 0$ for all p .

Now, if **Min** ends the game at step $p \leq N$, the final cost is then

$$\begin{aligned}64 + 1 - a_p &= 64 + \delta_p + \frac{1}{2^{c_p} 3^{d_p} 5^p} \geq 64 + \delta_p + \frac{1}{5^{2N}} \\ &\geq 64 + \frac{1}{5^{2N}} - \frac{1}{12 \times 30^{5N}} \\ &\geq 64 + \frac{11}{12 \times 30^{5N}}.\end{aligned}$$

On the other hand, if **Min** wants to play more than N steps, then she has to cheat. However, if **Max** ends the game in Case 3 in a $CEC_{30,\beta}^{34}$ module, then Cor. 2 ensures that the final cost is

$$64 + E_p + 30 \left| t_p - \left(1 - \frac{\beta}{30}\right) (1 - a_p) \right| \geq 64 + \frac{1}{30^{5N}} \geq 64 + \frac{11}{12 \times 30^{5N}},$$

since $\left| t_p - \left(1 - \frac{\beta}{30}\right) (1 - a_p) \right| \geq \frac{1}{30^{5N+1}}$.

However, avoiding this punishment limits **Min**'s possibilities:

Claim 2. For any $p \leq N$, if the game runs at least p steps, then the following property holds:

$$(\mathcal{P}_p) : (\widehat{c}_p, \widehat{d}_p) = (c_p, d_p) \wedge \mu_p \leq \frac{1}{12 \times 30^{5N}}.$$

In other words, to avoid triggering **Max**'s punishment, **Min** has to stay $\frac{1}{12 \times 30^{5N}}$ -close to a faithful simulation of \mathcal{M} 's execution. We prove this claim in the appendix.

Staying close to a faithful simulation means that cheating on a zero-test simulation, which triggers Case 3, is costly for **Min**. Indeed, if for instance⁷, **Max** punishes her with a NZC_1^0 module while $c_p = 0$, then using Prop. 5, **Max** can secure a cost of

$$\begin{aligned} & 64 + \min \left\{ \left| \frac{1}{2^c 3^d 5^n} - 1 + a_p \right| : c, d, n \in \mathbb{N}, 0 < c \right\} \\ &= 64 + \min \left\{ \left| \frac{1}{2^c 3^d 5^n} - \frac{1}{3^{d_p} 5^{p}} - \delta_p \right| : c, d, n \in \mathbb{N}, 0 < c \right\} \\ &\geq 64 + \min \left\{ \left| \frac{1}{2^c 3^d 5^n} - \frac{1}{3^{d_p} 5^p} \right| : c, d, n \in \mathbb{N}, 0 < c \right\} - \mu_p. \end{aligned}$$

Since $d_p \leq p$, we have $2^{5p} > 3^{3p} > 5^{2p} > 3^{d_p} 5^p$, and thus

$$\begin{aligned} & \min \left\{ \left| \frac{1}{2^c 3^d 5^n} - \frac{1}{3^{d_p} 5^p} \right| : c, d, n \in \mathbb{N}, 0 < c \right\} \\ &= \min \left\{ \left| \frac{1}{2^d 3^d 5^n} - \frac{1}{3^{d_p} 5^p} \right| : \begin{array}{l} c \in \{1, \dots, 5p\} \\ d \in \{0, \dots, 3p\} \\ n \in \{0, \dots, 2p\} \end{array} \right\}. \end{aligned}$$

Every elements in this last set are multiples of $\frac{1}{30^{5p}}$ and cannot be 0 since $c > 0$. Since $p \leq N$, we can conclude that when **Min** cheats on a zero-test transition, **Max** can secure a cost of at least

$$64 + \frac{1}{30^{5p}} - \frac{1}{12 \times 30^{5N}} \geq 64 + \frac{11}{12 \times 30^{5N}}. \quad \square$$

⁷ All other cases work similarly.

Proof of Claim 2. We prove (\mathcal{P}_p) by induction on p .

- (\mathcal{P}_1) holds by definition.
- Assume that (\mathcal{P}_p) holds and that state q_{p+1} exists. Reaching q_{p+1} entails that **Min** did choose the correct transition in case of zero-tests and that she did not exit via the exit module. The run then went through a $CEC_{30,\beta}^{34}$ module with $\beta \in \{18, 12, 6, 3, 2\}$ such that

$$\frac{\beta}{30} \frac{1}{2^{c_p} 3^{d_p} 5^p} = \frac{1}{2^{c_{p+1}} 3^{d_{p+1}} 5^{p+1}}.$$

According to **Max**'s strategy, we have

$$\left| t_p - \left(1 - \frac{\beta}{30}\right) (1 - a_p) \right| < \frac{1}{30^{5N+1}},$$

since otherwise **Max** would have ended the game. Thus

$$\begin{aligned} \left| 1 - \frac{1}{2^{c_{p+1}} 3^{d_{p+1}} 5^{p+1}} - a_{p+1} \right| &= \left| 1 - \frac{\beta}{30} \frac{1}{2^{c_p} 3^{d_p} 5^p} - a_p - t_p \right| \\ &= \left| 1 - a_p - \frac{\beta}{30} (1 - a_p - \delta_p) - t_p \right| \\ &= \left| \frac{\beta}{30} \delta_p + \left(1 - \frac{\beta}{30}\right) (1 - a_p) - t_p \right| \\ &\leq \frac{\beta}{30} \mu_p + \left| \left(1 - \frac{\beta}{30}\right) (1 - a_p) - t_p \right| \\ &< \frac{3}{5} \mu_p + \frac{1}{30^{5N+1}} \\ &\leq \frac{3}{5} \frac{1}{12 \times 30^{5N}} + \frac{1}{30^{5N+1}} = \frac{1}{12 \times 30^{5N}}. \end{aligned}$$

Now we have

$$\begin{aligned} &\min \left\{ \left| \frac{1}{2^c 3^d 5^{p+1}} - \frac{1}{2^{c_{p+1}} 3^{d_{p+1}} 5^{p+1}} \right| : \begin{array}{l} c, d \in \mathbb{N} \\ c + d \leq p + 1 \\ (c, d) \neq (c_{p+1}, d_{p+1}) \end{array} \right\} \\ &\geq \frac{1}{30^{p+1}} \geq \frac{1}{30^N} \end{aligned}$$

since any element of the above set is a multiple of $\frac{1}{30^{2(p+1)}}$ and cannot be 0.

Therefore, if $(\widehat{c}_{p+1}, \widehat{d}_{p+1}) \neq (c_{p+1}, d_{p+1})$, we would have

$$\begin{aligned} &\left| 1 - \frac{1}{2^{\widehat{c}_{p+1}} 3^{\widehat{d}_{p+1}} 5^{p+1}} - a_p \right| \\ &\geq \left| \frac{1}{2^{\widehat{c}_{p+1}} 3^{\widehat{d}_{p+1}} 5^{p+1}} - \frac{1}{2^{c_{p+1}} 3^{d_{p+1}} 5^{p+1}} \right| - \left| 1 - \frac{1}{2^{c_{p+1}} 3^{d_{p+1}} 5^{p+1}} - a_{p+1} \right| \\ &\geq \frac{1}{30^N} - \frac{1}{12 \times 30^{5N}} > \frac{1}{12 \times 30^{5N}}. \end{aligned}$$

Hence $(\widehat{c}_{p+1}, \widehat{d}_{p+1}) = (c_{p+1}, d_{p+1})$, and

$$\mu_{p+1} = \left| 1 - \frac{1}{2^{c_{p+1}} 3^{d_{p+1}} 5^{p+1}} - a_{p+1} \right| \leq \frac{1}{12 \times 30^{5N}}. \quad \square$$