# OPTIMAL CONTROLLER SYNTHESIS FOR TIMED SYSTEMS

DAMIEN BUSATTO-GASTON, BENJAMIN MONMEGE, AND PIERRE-ALAIN REYNIER

Université libre de Bruxelles, Belgique
*e-mail address*: damien.busatto-gaston@ulb.be

Aix-Marseille Univ, CNRS, Université de Toulon, LIS, France
*e-mail address*: benjamin.monmege@univ-amu.fr

Aix-Marseille Univ, CNRS, Université de Toulon, LIS, France
*e-mail address*: pierre-alain.reynier@univ-amu.fr

ABSTRACT. Weighted timed games are zero-sum games played by two players on a timed automaton equipped with weights, where one player wants to minimise the cumulative weight while reaching a target. Used in a reactive synthesis perspective, this quantitative extension of timed games allows one to measure the quality of controllers in real-time systems. Weighted timed games are notoriously difficult and quickly undecidable, even when restricted to non-negative weights. For non-negative weights, the largest class that can be analysed has been introduced by Bouyer, Jaziri and Markey in 2015. Though the value problem is undecidable, the authors show how to approximate the value by considering regions with a refined granularity. In this work, we extend this class to incorporate negative weights, allowing one to model energy for instance, and prove that the value can still be approximated, with the same complexity. A small restriction also allows us to obtain a class of decidable weighted timed games with negative weights and an arbitrary number of clocks. In addition, we show that a symbolic algorithm, relying on the paradigm of value iteration, can be used as an approximation/computation schema over these classes. We also consider the special case of untimed weighted games, where the same fragments are solvable in polynomial time: this contrasts with the pseudo-polynomial complexity, known so far, for weighted games without restrictions.

## 1. INTRODUCTION

We are interested in the design of programs sensitive to real-time, where keeping track of how much time elapses between the decisions taken by the program is required to differentiate the good and bad behaviours of the system. This is a common requirement for embedded systems, as they interact with the real world. The design of such programs is a notoriously difficult problem, because they must take care of delicate timing issues, and are difficult to debug a posteriori. In order to ease the design of real-time software, it appears important to automatise the process by using formal methods. The situation may be modelled into a *timed automaton* [2], namely a transition system equipped with real-valued variables, called clocks,

evolving with a uniform rate. Transitions are equipped with timing constraints expressed over the clocks, and may only be taken when these constraints are met. Finally, clocks may be reset so as to keep track of how much time has elapsed since a particular transition was taken.

In order to check whether the real-time system fulfils a certain specification, one determines whether there exists an accepting execution in the timed automaton: this is the classical *model-checking problem.* A simple, yet realistic specification asks that a target state is reached at some point. It has been proven in early works [2] that the reachability problem on timed automata is PSPACE-complete, by partitioning the state space into *regions.* While optimal from a theoretical complexity point of view, practical tools tend to favour efficient *symbolic* algorithms for solving these problems, that use *zones* instead of regions, as they allow an on-demand partitioning of the state space. This leads to much better performances, as witnessed by successful model-checking tools like UPPAAL [26], KRONOS [11], or TCHECKER [19, 20].

Instead of verifying a system, we can try to synthesise one automatically. One setting consists in using *game theory.* Configurations of the system are then partitioned into two *players*: a controller whose role is to fulfil the specification, and an antagonistic environment. The goal becomes to find automatically a good *strategy* for the controller, which is called the *controller synthesis problems.* In timed systems, both players alternatively choose transitions and delays in a timed automaton: this is called a *timed game.* Strategies of players are recipes dictating how to play (timing delays and transitions to follow). In this ambitious setting, we will focus on reachability objectives, and we are thus looking for a strategy of the controller so that the target is reached no matter how the environment plays. Reachability timed games are decidable [4], and EXPTIME-complete [23].

If the controller has a winning strategy in a given reachability timed game, several such winning strategies could exist. Weighted extensions of these games have been considered in order to measure the quality of the winning strategy for the controller [9]. This means that the game now takes place over a *weighted (or priced) timed automaton* [5, 3], where edges are equipped with weights, and locations with rates of weights (the cost is then proportional to the time spent in this location, with the rate as proportional coefficient). The optimal reachability problem asks what is the lowest cumulative weight that the controller can guarantee for reaching a target from a given initial state, against any decisions made by the antagonistic environment. The controller is therefore the minimiser player, while the environments wants to maximise the weight accumulated along the execution. This lowest weight is called the *value* of the game, and computing it can be seen as a natural generalisation of the classical shortest path problem in a weighted graph to the case of two-player timed games.

**Example 1.1.** As a motivating example, we present a ride-sharing scenario. As a driver, we wish to travel from point $A$ to point $B$, and must choose between several options, as displayed in Figure 1. We can use a direct road, and reach $B$ in two to three hours, or an highway that lets us reach our destination in one hour. Alternatively, we can make a detour: another traveller is waiting at point $C$, and wishes to reach point $D$. For this portion too, a faster highway is available.

While all four possible paths satisfy the objective "reaching $B$", we want to select the one that lets us spend as little money as possible for the trip. The cost of each path depends on several factors. There are fixed entry fees (of $1 \in$) for the highways, and we need to keep track of fuel consumption, as the rate at which fuel is used differs in roads and highways.
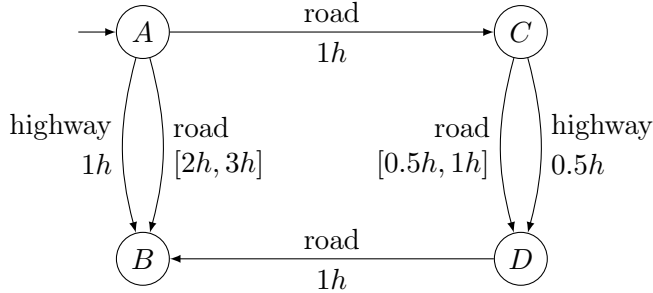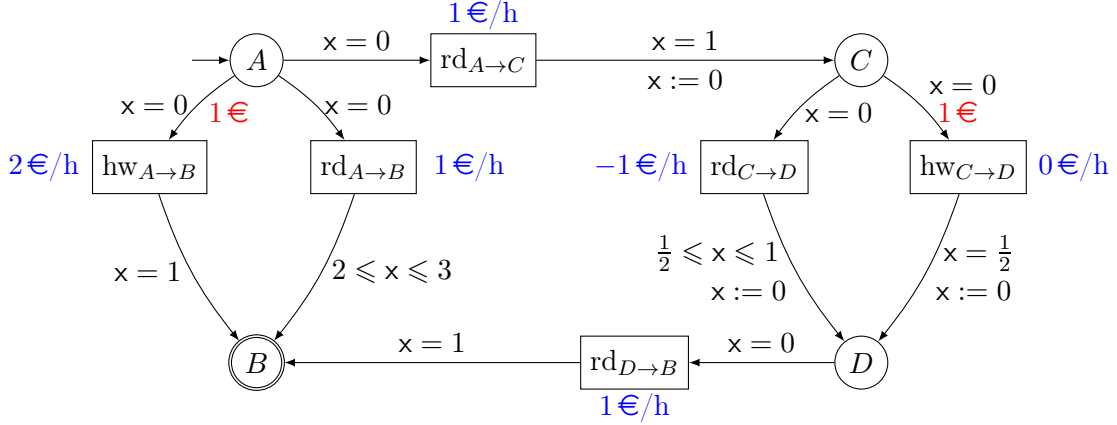
Figure 1: A ride-sharing decision diagram.



Figure 2: A weighted timed game modelling Figure 1. Transitions are labelled by a guard over clock x and by the reset x := 0 when needed. The cost of waiting in a state is displayed in blue, the cost of taking a transition is in red. States and transitions without costs have a weight of $0 €$

Thus, we say that roads cost $1 €$ per hour, while highways cost $2 €$ per hour. Moreover, if we share the portion from $C$ to $D$, the other traveller will pay for his trip (at a rate of $2 €/h$), and that can lower our costs. A shared road therefore costs us $-1 €$ per hour (negative rate means we are making a profit), while a shared highway costs $0 €/h$.

The situation can be modelled as a weighted timed game, displayed in Figure 2. The controller chooses delays and transitions in circle states, while the environment controls the square ones. For example, if we choose to use the direct road from $A$ to $B$, we go (immediately) to state $rd_{A \to B}$, and stay there until going to state $B$. This requires letting between two and three hours elapse in $rd_{A \to B}$, with a cost of $1 €/h$. The delay is chosen by the environment, as it depends on external influences like traffic density.

In this example, the optimal strategy is to share the road from $C$ to $D$. This lets us ensure a total weight of at most $1.5 €$: going from $A$ to $C$ costs $1 €$ in the worst case; going from $D$ to $B$ similarly costs at most $1 €$; and sharing the trip from $C$ to $D$ is guaranteed to bring us at least $0.5 €$.

While solving weighted timed automata has been shown to be PSPACE-complete [6] (i.e. the same complexity as the non-weighted version), weighted timed games are known to

be undecidable [13]. This has led to many restrictions in order to regain decidability, the first and most interesting one being the class of strictly non-Zeno cost with only non-negative weights (in edges and locations) [9]: this hypothesis states that every execution of the timed automaton that follows a cycle of the region abstraction has a weight far from 0 (in interval $[1, +\infty)$, for instance).

Less is known for weighted timed games in the presence of negative weights in edges and locations. In particular, no results existed before this work for a class that does not restrict the number of clocks of the timed automaton to 1. However, several clocks are needed to keep track of how much time elapsed since multiple reference points, and negative weights are particularly interesting from a modelling perspective, for instance in case weights represent the consumption level of a resource (money, energy...) with the possibility to spend and gain some resource. We thus introduce a generalisation of the strictly non-Zeno cost hypothesis in the presence of negative weights, that we call *divergence*. We show the decidability of the class of divergent weighted timed games for the optimal synthesis problem. The technique uses a *value iteration* procedure to solve weighted timed games for a bounded horizon, i.e. when controller has a fixed number of steps to reach his targets. It follows closely the framework of [1], but is more symbolic and allows for negative weights. We then show that optimal strategies in divergent weighted timed games can be restricted to a bounded horizon, that matches the one obtained in the non-negative case from the study of [9].

The techniques providing these decidability results cannot be extended if the conditions are slightly relaxed. For instance, if we add the possibility for an execution of the timed automaton following a cycle of the region automaton to have weight *exactly 0*, the decision problem is known to be undecidable [10], even with non-negative weights only. For this extension, in the presence of non-negative weights only, it has been proposed an approximation schema to compute arbitrarily close estimates of the optimal weight that the controller can guarantee [10]. To this end, the authors consider regions with a refined granularity so as to control the precision of the approximation.

Our contribution on the approximation front is two-fold. We extend the class considered in [10] to the presence of negative weights, and provide an approximation schema for the resulting class of *almost-divergent games*. We then show that the approximation can be obtained using a symbolic computation, that avoids an a priori refinement of regions. Table 1 summarises our results on weighted timed games.

The classes of weighted timed games that we study induce interesting classes of *finite weighted game* when there are no clocks, that can be solved with a lower (polynomial) complexity than arbitrary weighted games, see Table 2.

Other types of payoffs than the cumulative weight we study (i.e. total payoff) have been considered for weighted timed games. For instance, energy and mean-payoff timed games have been introduced in [12]. They are also undecidable in general. Interestingly, a subclass called *robust timed games*, not far from our divergence hypothesis, admits decidability results for other payoffs. A weighted timed game is robust if, to say short, every simple cycle (cycle without repetition of a state) has weight non-negative or less than a constant $-\varepsilon$. Solving robust timed game can be done in EXPSPACE, and is EXPTIME-hard. Moreover, deciding if a weighted timed game is robust has complexity 2-EXPSPACE (and coNEXPTIME-hard). This contrasts with our PSPACE results for the membership problem.[1] It has to be noted

---

[1]While our divergent games have a similar definition, both classes are incomparable.

| Timed | weights in ℕ | | |
|---|---|---|---|
| | divergent | almost-divergent | all WTG |
| Value pb. | 2-EXPTIME [9]+[1] | undecidable [10] | undecidable [13] |
| Approx. pb. | / | 2-EXPTIME [10]+[1] | ? |
| Value +∞ | EXPTIME-complete [9]+[23] | | |

| Timed | weights in ℤ | | |
|---|---|---|---|
| | divergent | almost-divergent | all WTG |
| Value pb. | 3-EXPTIME EXPTIME-hard Thm. 3.3 | undecidable [10] | undecidable [16] |
| Approx. pb. | / | 3-EXPTIME Thm. 3.8 | ? |
| Value −∞ | EXPTIME-complete Prop. 3.10,5.5 | | undecidable Prop. 3.11 |
| Value +∞ | EXPTIME-complete Prop. 5.3 | | |
| Membership | PSPACE-complete Thm. 3.3,3.8 | | / |

Table 1: Solving weighted timed games with arbitrary weights

| Untimed | weights in ℕ | weights in ℤ | | |
|---|---|---|---|---|
| | all games | divergent | almost-div. | all games |
| Value pb. | PTIME [24] | PTIME-complete Thm. 11.4,11.6 | | pseudo-poly. [15] |
| Value −∞ | / | PTIME-complete Section 11.2.1 | | pseudo-poly. [15] |
| Value +∞ | PTIME [24] | PTIME-complete Prop. 5.3, [15] | | |
| Membership | / | NL-complete (unary wt.) PTIME (binary wt.) Thm. 11.4,11.6 | | / |

Table 2: Solving finite weighted games with arbitrary weights

that extending our techniques and results in the case of robust timed games may not be possible: indeed, with weights in ℕ every game is robust, making the value problem for this class undecidable [13], with no known approximation method.

This article is structured as follows. After some preliminary definitions of the models we study in Section 2, we introduce the divergent and almost-divergent classes of weighted timed games and state our results in Section 3. We study structural properties of these classes in order to be able to decide their membership in Section 4. We then start solving these games by studying the qualitative problem of deciding infinite values, using the crucial

and central notion of kernels, in Section 5. The approximation schema for almost-divergent timed games is presented in Section 6 where we introduce and prove the correctness of a semi-unfolding of the game. Section 7 explains how to compute the value of this semi-unfolding in the special case of a tree-shaped game (without kernels) as previously studied in [1]. We then generalise our study in Section 8 to take kernels into account; we also consider the special case of divergent timed games. The first approximation/computation we propose requires to first compute strongly connected components and to build the semi-unfolding of the timed games, which might be highly prohibitive as usual in the timed setting. Therefore, we present a more symbolic algorithm in Section 9.1. In Section 10, we deduce from our study an algorithm to synthesise finite-memory almost-optimal strategies in divergent timed games. Finally, we treat the special case of almost-divergent untimed games in Section 11, where polynomial time algorithms are provided in this case (to be compared with the pseudo-polynomial complexity in general known so far).

## 2. Preliminaries

2.1. **Modelling real-time constraints.** We first introduce notions that let us express timing constraints, useful to then define weighted timed games, and introduce classical tools for their study.

Let $\mathcal{X} = \{x_1, \ldots, x_n\}$ be a finite, non-empty set of variables called clocks. A *valuation* $\nu \colon \mathcal{X} \to \mathbb{R}_{\geqslant 0}$ is a mapping from clocks to non-negative real numbers, such that $\nu(x_1), \ldots, \nu(x_n)$ are called the coordinates of $\nu$. Equivalently, $\nu$ can be seen as a point in space $\mathbb{R}_{\geqslant 0}^{\mathcal{X}}$. We denote $\mathbf{0}$ the valuation such that for all $x \in \mathcal{X}$, $\nu(x) = 0$. Given a real number $d \in \mathbb{R}$, we define $\nu + d$ as the valuation such that $\forall x \in \mathcal{X}, (\nu + d)(x) = \nu(x) + d$ if it exists.[2] If $d$ is non-negative, we say that we performed a time elapse of delay $d$. The time-successors of $\nu$ are the valuations $\nu + d$ with $d \geqslant 0$. Similarly, we refer to all $\nu + d$ in $\mathbb{R}_{\geqslant 0}^{\mathcal{X}}$ with $d \leqslant 0$ as time-predecessors of $\nu$. The set of points that are either time-predecessors or time-successors of a valuation $\nu$ form the unique diagonal line in $\mathbb{R}_{\geqslant 0}^{\mathcal{X}}$ that contains $\nu$. If $\mathcal{Y}$ is a subset of $\mathcal{X}$, we define $\nu[\mathcal{Y} := 0]$ as the valuation such that $\forall x \in \mathcal{Y}, (\nu[\mathcal{Y} := 0])(x) = 0$ and $\forall x \in \mathcal{X} \backslash \mathcal{Y}, (\nu[\mathcal{Y} := 0])(x) = \nu(x)$. This operation is called a reset of clocks $\mathcal{Y}$.

We extend those notions to sets of valuations in a natural way. The set of time-successors of $Z \subseteq \mathbb{R}_{\geqslant 0}^{\mathcal{X}}$, denoted $\mathsf{PostTime}(Z)$, contains the valuations that are time-successors of valuations in $Z$. The reset of $Z \subseteq \mathbb{R}_{\geqslant 0}^{\mathcal{X}}$ by $\mathcal{Y}$, denoted $Z[\mathcal{Y} := 0]$, contains the valuations $\nu[\mathcal{Y} := 0]$ such that $\nu \in Z$.

The term *atomic constraint* will refer to an affine inequality in one of the following forms:

- A strict (resp. non-strict) *non-diagonal* atomic constraint over clock $x \in \mathcal{X}$ and constant $c \in \mathbb{Q}$ is an inequality of the form $x \bowtie c$ with $\bowtie \in \{>, <\}$ (resp. $\bowtie \in \{\geqslant, \leqslant\}$).
- A strict (resp. non-strict) *diagonal* atomic constraint over clocks $x_1$ and $x_2 \in \mathcal{X}$ and constant $c \in \mathbb{Q}$ is an inequality of the form $x_1 - x_2 \bowtie c$ with $\bowtie \in \{>, <\}$ (resp. $\bowtie \in \{\geqslant, \leqslant\}$).

Let $\top$ and $\bot$ denote two special atomic constraints, defined as $x \geqslant 0$ and $x < 0$ for an arbitrary $x \in \mathcal{X}$. A *guard* $g$ over $\mathcal{X}$ is a finite conjunction of atomic constraints over clocks in $\mathcal{X}$. In particular, guards let us define $x = c$ as shorthand for $x \leqslant c \wedge x \geqslant c$, and $c_1 < x < c_2$ as shorthand for $x > c_1 \wedge x < c_2$. A guard is said strict (resp. non-strict, diagonal, non-diagonal)

---

[2]if $d$ is negative, $\nu + d$ may not belong to $\mathbb{R}_{\geqslant 0}^{\mathcal{X}}$

if all of its atomic constraints are strict (resp. non-strict, diagonal, non-diagonal). $\mathsf{Guards}(\mathcal{X})$ denotes the set of all guards over $\mathcal{X}$, and $\mathsf{Guards}^{\mathrm{nd}}(\mathcal{X})$ the subset of non-diagonal guards. For all constants $c \in \mathbb{Q}$ and $\bowtie \in \{\geqslant, \leqslant, >, <\}$, we say that valuation $\nu \in \mathbb{R}_{\geqslant 0}^{\mathcal{X}}$ satisfies the atomic constraint $\mathsf{x} \bowtie c$ (resp. $\mathsf{x}_1 - \mathsf{x}_2 \bowtie c$), and write $\nu \models \mathsf{x} \bowtie c$ (resp. $\nu \models \mathsf{x}_1 - \mathsf{x}_2 \bowtie c$), if $\nu(\mathsf{x}) \bowtie c$ (resp. $\nu(\mathsf{x}_1) - \nu(\mathsf{x}_2) \bowtie c$). We say that valuation $\nu \in \mathbb{R}_{\geqslant 0}^{\mathcal{X}}$ satisfies guard $g$, and write $\nu \models g$, if $\nu$ satisfies all atomic constraints in $g$. For $g \in \mathsf{Guards}(\mathcal{X})$, let $[\![g]\!]$ denote the set of all $\nu \in \mathbb{R}_{\geqslant 0}^{\mathcal{X}}$ such that $\nu \models g$. Such sets are called *zones* and form convex polyhedra of $\mathbb{R}_{\geqslant 0}^{\mathcal{X}}$. A guard $g$ is said satisfiable when the zone $[\![g]\!]$ is non-empty, and a zone is called rectangular when the associated guard is non-diagonal. The universal zone refers to $[\![\top]\!] = \mathbb{R}_{\geqslant 0}^{\mathcal{X}}$ and the empty zone refers to $[\![\bot]\!] = \emptyset$. Guard $\overline{g}$ is the closed version of a satisfiable guard $g$ where every strict constraint of comparison operator $<$ or $>$ is replaced by its non-strict version $\leqslant$ or $\geqslant$. The zone $[\![\overline{g}]\!]$ is the topological closure of $Z = [\![g]\!]$, and is also denoted $\overline{Z}$.

2.2. **Weighted timed games.** We now turn our attention to weighted timed two-player games with a shortest-path objective towards a set of target locations. We will first define weighted timed games, giving their semantics in terms of an *infinite* transition system on which the traditional notions of strategies and values are defined.

**Definition 2.1.** A *weighted timed game* (WTG) is a tuple $\mathcal{G} = \langle L_{\mathsf{Min}}, L_{\mathsf{Max}}, \mathcal{X}, L_{\mathsf{t}}, E, \mathsf{wt}, \mathsf{fwt} \rangle$ with $L = L_{\mathsf{Min}} \uplus L_{\mathsf{Max}}$ a finite set of locations split between players $\mathsf{Min}$ and $\mathsf{Max}$ (in drawings, locations belonging to $\mathsf{Min}$ are depicted by circles and the ones belonging to $\mathsf{Max}$ by squares), $\mathcal{X}$ a finite set of clocks, $E \subseteq L \times \mathsf{Guards}^{\mathrm{nd}}(\mathcal{X}) \times 2^{\mathcal{X}} \times L$ a finite set of edges $\ell \xrightarrow{g, \mathcal{Y}} \ell'$ from location $\ell$ to location $\ell'$, labelled by a non-diagonal guard $g$ and a set $\mathcal{Y}$ of clocks to reset, $\mathsf{wt} \colon E \uplus L \to \mathbb{Z}$ a weight function associating an integer weight with each location and edge, $L_{\mathsf{t}} \subseteq L_{\mathsf{Min}}$ a set of target locations for player $\mathsf{Min}$, and $\mathsf{fwt} \colon L_{\mathsf{t}} \times \mathbb{R}_{\geqslant 0}^{\mathcal{X}} \to \mathbb{R}_\infty$ is a function mapping each target configuration to a final weight of $\mathbb{R}_\infty = \mathbb{R} \cup \{-\infty, +\infty\}$.

The addition of final weights is not standard, but we will use it in the process of solving those games: in any case, it is possible to simply map a given target location to the weight 0, allowing us to recover the standard definitions of the literature.

The semantics of a weighted timed game $\mathcal{G}$ is defined in terms of an infinite labelled transition system $[\![\mathcal{G}]\!]$ whose states are configurations $(\ell, \nu) \in L_{\mathsf{t}} \times \mathbb{R}_{\geqslant 0}^{\mathcal{X}}$. Configurations are split into players according to the location $\ell$. A configuration $(\ell, \nu)$ is a target if $\ell \in L_{\mathsf{t}}$, and its final weight is $\mathsf{fwt}(\ell, \nu)$. The labels of $[\![\mathcal{G}]\!]$ are given by $\mathbb{R}_{\geqslant 0} \times E$ and will encode the delay that a player wants to spend in the current location, before firing a certain edge. For every delay $d \in \mathbb{R}_{\geqslant 0}$, edge $e = \ell \xrightarrow{g, \mathcal{Y}} \ell' \in E$ and valuation $\nu$, there is a transition $(\ell, \nu) \xrightarrow{d, e} (\ell', \nu')$ if $\nu + d \models g$ and $\nu' = (\nu + d)[\mathcal{Y} := 0]$. The weight of such a transition takes into account both discrete and continuous costs, and is given by $d \cdot \mathsf{wt}(\ell) + \mathsf{wt}(e)$.

As usual in related work [1, 9, 10], we assume throughout this article that **WTGs have non-diagonal guards where all constants are integers, that all clocks are** *bounded* **by the greatest constant $M$ to appear in guards, and we restrict** $[\![\mathcal{G}]\!]$ **to configurations in** $L \times [0, M)^{\mathcal{X}}$.[3] Without loss of generality, we suppose the absence of deadlocks in $[\![\mathcal{G}]\!]$ except on target locations, i.e. for each location $\ell \in L \backslash L_{\mathsf{t}}$ and valuation $\nu \in [0, M)^{\mathcal{X}}$, there exist $d \in \mathbb{R}_{\geqslant 0}$ and $e \in E$ such that $(\ell, \nu) \xrightarrow{d, e} (\ell', \nu')$, and no edges

---

[3]Observe that the "bounded" assumption is without loss of generality for (weighted) timed automata [5], but we do not know if it is the case for weighted timed games: in particular in the presence of negative weights, the technique of [5] cannot be directly applied.
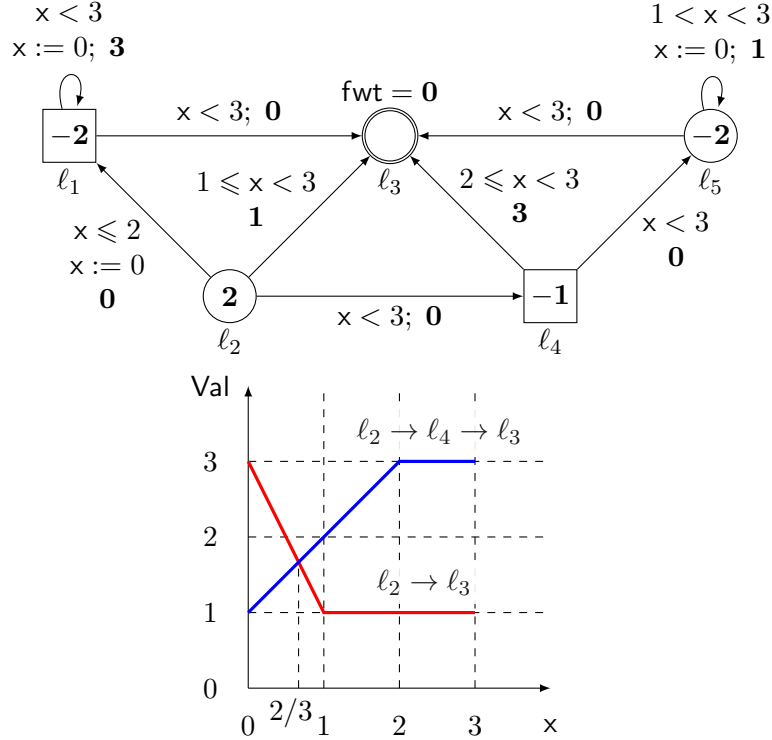
Figure 3: A weighted timed game with a single clock x, and a depiction of its value function. Weights are indicated in bold font on locations and edges. The target location is $\ell_3$, whose final weight function is zero.

start from $L_{\mathsf{t}}$. We also assume that the final weight functions satisfy a sufficient property ensuring that they can be encoded in finite space: they must be piecewise affine with a finite number of pieces and are continuous on each region (see later for a formal definition). In particular, infinite final weights are constant over regions, i.e. if some configuration $(\ell_{\mathsf{t}}, \nu)$ has final weight $+\infty$ or $-\infty$, then for every valuation $\nu'$ in the same region as $\nu$, $\mathsf{fwt}(\ell_{\mathsf{t}}, \nu) = \mathsf{fwt}(\ell_{\mathsf{t}}, \nu')$. The standard final weight function $(\ell, \nu) \mapsto 0$ satisfies this property. Moreover, the computations we will perform in the following maintain this property as an invariant. An example of WTG satisfying those assumptions is depicted on Figure 3.

Plays are maximal paths in the transition system $[\![\mathcal{G}]\!]$: they are either infinite or end in a target location. For a player $\mathsf{P} \in \{\mathsf{Min}, \mathsf{Max}\}$, the set of non-maximal paths $\rho$ (often called *finite plays*) such that $\mathsf{last}(\rho) \in L_{\mathsf{P}}$ is denoted $\mathsf{FPlays}^{\mathsf{P}}$. First and last elements of a finite play $\rho$ are denoted $\mathsf{first}(\rho)$ and $\mathsf{last}(\rho)$, respectively.

A strategy $\sigma_{\mathsf{P}}$ for player $\mathsf{P}$ is a mapping $\mathsf{FPlays}^{\mathsf{P}} \to \mathbb{R}_{\geqslant 0} \times e$, such that for all $\rho \in \mathsf{FPlays}^{\mathsf{P}}$ ending in configuration $(\ell, \nu)$, the transition system $[\![\mathcal{G}]\!]$ contains a transition labelled by $\sigma_{\mathsf{P}}(\rho)$ from $(\ell, \nu)$. A strategy is said *positional* (or *memoryless*) if for all $\rho, \rho' \in \mathsf{FPlays}^{\mathsf{P}}$ ending in the same configuration, $\sigma_{\mathsf{P}}(\rho) = \sigma_{\mathsf{P}}(\rho')$. Let $\mathsf{play}((\ell_0, \nu_0), \sigma_{\mathsf{Min}}, \sigma_{\mathsf{Max}})$ denote the unique maximal play starting from configuration $(\ell_0, \nu_0)$ such that for every prefix $\rho$ of $\mathsf{play}((\ell_0, \nu_0), \sigma_{\mathsf{Min}}, \sigma_{\mathsf{Max}})$ in $\mathsf{FPlays}^{\mathsf{P}}$, the next transition in $\mathsf{play}((\ell_0, \nu_0), \sigma_{\mathsf{Min}}, \sigma_{\mathsf{Max}})$ is labelled by $\sigma_{\mathsf{P}}(\rho)$.

The objective of Min is to reach a target location, while minimising the cumulative weight up to the target. Hence, we associate to every finite play $\rho = (\ell_0, \nu_0) \xrightarrow{d_1, e_1} (\ell_1, \nu_1) \cdots \xrightarrow{d_k, e_k} (\ell_k, \nu_k)$ its cumulative weight

$$\mathsf{wt}_\Sigma(\rho) = \sum_{i=1}^{k} d_i \cdot \mathsf{wt}(\ell_{i-1}) + \mathsf{wt}(e_i)$$

Then, the weight of a maximal play $\rho$, also denoted by $\mathsf{wt}(\rho)$, is defined by $+\infty$ if $\rho$ is infinite and thus does not reach $L_t$, and $\mathsf{wt}_\Sigma(\rho) + \mathsf{fwt}(\ell, \nu)$ if it is finite and ends in $(\ell, \nu)$ with $\ell \in L_t$. Then, the respective values of the strategies are defined by

$$\mathsf{Val}_{\mathcal{G}}((\ell_0, \nu_0), \sigma_{\mathsf{Min}}) = \sup_{\sigma_{\mathsf{Max}}} \mathsf{wt}(\mathsf{play}((\ell_0, \nu_0), \sigma_{\mathsf{Min}}, \sigma_{\mathsf{Max}}))$$

$$\mathsf{Val}_{\mathcal{G}}((\ell_0, \nu_0), \sigma_{\mathsf{Max}}) = \inf_{\sigma_{\mathsf{Min}}} \mathsf{wt}(\mathsf{play}((\ell_0, \nu_0), \sigma_{\mathsf{Min}}, \sigma_{\mathsf{Max}}))$$

Finally, we let

$$\underline{\mathsf{Val}}(\ell_0, \nu_0) = \sup_{\sigma_{\mathsf{Max}}} \mathsf{Val}_{\mathcal{G}}((\ell_0, \nu_0), \sigma_{\mathsf{Max}}) \qquad \overline{\mathsf{Val}}(\ell_0, \nu_0) = \inf_{\sigma_{\mathsf{Min}}} \mathsf{Val}_{\mathcal{G}}((\ell_0, \nu_0), \sigma_{\mathsf{Min}})$$

be the *lower* and *upper values* of configuration $(\ell_0, \nu_0)$, respectively. We may easily show that $\underline{\mathsf{Val}} \leqslant \overline{\mathsf{Val}}$. We say that a strategy $\sigma_{\mathsf{Min}}^\star$ of Min is $\varepsilon$-*optimal* if, for all initial configurations $(\ell_0, \nu_0)$

$$\mathsf{Val}_{\mathcal{G}}((\ell_0, \nu_0), \sigma_{\mathsf{Min}}^\star) \leqslant \overline{\mathsf{Val}}(\ell_0, \nu_0) + \varepsilon \,.$$

It is said *optimal* if this holds for $\varepsilon = 0$. A symmetric definition holds for optimal strategies of Max.

**Example 2.2.** In the WTG of Figure 3, observe that location $\ell_1$ (resp. $\ell_5$) has value $+\infty$ (resp. $-\infty$). Indeed, from any configuration $(\ell_1, \nu)$ with $\nu \in [0, 3)^{\mathcal{X}}$, player Max can play the self-loop on $\ell_1$, ensuring that the target $\ell_3$ is never reached. Moreover, from any configuration $(\ell_5, \nu)$ with $\nu \in [0, 3)^{\mathcal{X}}$, player Min can play the self-loop on $\ell_5$, leading to a reset of clock x. From $(\ell_5, (x = 0))$ Min can pick a delay of 1.5, and loop again while accumulating a weight of $-2(1.5) + 1 = -2$. This loop of $\ell_5$ can be iterated arbitrarily many times before playing the edge to $\ell_3$, ensuring a weight arbitrarily low, hence every configuration on $\ell_5$ has value $-\infty$. As a consequence, the value in $\ell_4$ is determined by the edge to $\ell_3$, and is depicted in blue (player Max prefers using the shortest delay admissible, leading to a weight of $3 - \min(2, 2 - \nu(x))$). In location $\ell_2$, player Min wishes to minimise his choice of delay. The value associated with the edge to $\ell_3$ is depicted in red, the blue curve is the value associated with the edge to $\ell_4$, and the value in $\ell_2$ is obtained as the minimum of these two curves. Observe the intersection in $x = 2/3$ requiring to refine the classical region abstraction.

It is known that (turn-based) weighted timed games are determined[4], i.e. $\underline{\mathsf{Val}}(\ell, \nu) = \overline{\mathsf{Val}}(\ell, \nu)$ for each location $\ell$ and valuation $\nu$, therefore we use the notation $\mathsf{Val}_{\mathcal{G}}$ to refer to both values.

We denote by $w_{\max}^L$ (resp. $w_{\max}^E$) the maximal weight in absolute values of locations (resp. of edges) in $\mathcal{G}$:

$$w_{\max}^L = \max_{\ell \in L} |\mathsf{wt}(\ell)| \qquad \text{and} \qquad w_{\max}^E = \max_{e \in E} |\mathsf{wt}(e)|$$

---

[4]The result is stated in [14] for weighted timed games (called priced timed games) with one clock, but the proof does not use the assumption on the number of clocks.

Moreover, we denote by $w_{\max}$ a bound on the weight of transitions in $[\![\mathcal{G}]\!]$, that exists since clocks are bounded by $M$:
$$w_{\max} = M w_{\max}^L + w_{\max}^E$$
The integer $w_{\max}$ is at most exponential in the size of $\mathcal{G}$, and can thus be stored in polynomial space.

We consider the *value problem*: given a WTG $\mathcal{G}$, a configuration $(\ell, \nu)$ and a threshold $\alpha \in \mathbb{Z}$, we want to know whether $\mathsf{Val}_\mathcal{G}(\ell, \nu) \leqslant \alpha$. We also consider the $+\infty$-*value problem* (resp. $-\infty$-*value problem*) asking if $\mathsf{Val}_\mathcal{G}(\ell, \nu)$ equals $+\infty$ (resp. $-\infty$). In the context of timed games, optimal strategies may not exist, even for finite values.[5] We thus generally focus on the search for $\varepsilon$-optimal strategies, that guarantee the optimal value, up to a small error $\varepsilon \in \mathbb{R}_{>0}$: this is the *synthesis problem*. Moreover, when the value problem is undecidable, we also consider the *value approximation problem* that consists, given a precision $\varepsilon \in \mathbb{Q}_{>0}$, in computing an $\varepsilon$-approximation of $\mathsf{Val}_\mathcal{G}(\ell, \mathbf{0})$. More generally, we will try to compute an $\varepsilon$-approximation of the whole value function (and not only for an initial configuration with all clocks being 0): this means that we want to compute (in a format that we detail in the end of this preliminary section) a function $\boldsymbol{V} \colon L \times [0, M)^\mathcal{X} \to \mathbb{R}_\infty$ such that $\|\mathsf{Val}_\mathcal{G} - \boldsymbol{V}\|_\infty \leqslant \varepsilon$, where $\|\cdot\|_\infty$ denotes the classical $\infty$-norm of mappings, so that $\|f\|_\infty = \sup_x |f(x)|$. In particular, we ask that $\mathsf{Val}_\mathcal{G} = \boldsymbol{V}$ on all configurations where either function has infinite value, and as such solving the infinite value problems will be a requirement.

2.3. **Related work.** In the one-player case, computing the optimal value and an $\varepsilon$-optimal strategy for WTGs (called weighted or priced timed automata) is known to be PSPACE-complete [6]. In the two-player case, the value problem of WTGs (also called priced timed games in the literature) is undecidable with 3 clocks [13, 10], or even 2 clocks in the presence of negative weights [16] (for the existence problem, asking if a strategy of player Min can guarantee a given threshold).

To obtain decidability, one possibility has been to limit the number of clocks to 1: then, there is an exponential-time algorithm to compute the value as well as $\varepsilon$-optimal strategies in the presence of non-negative weights only [7, 29, 18], whereas the problem is only known to be PTIME-hard. A similar result can be lifted to arbitrary weights, under restrictions on the resets of the clock in cycles [14].

The other possibility to obtain a decidability result [9, 1] with non-negative weights only is to enforce a semantical property of divergence, originally called strictly non-Zeno cost: it asks that every play following a cycle of the region abstraction (see later for a formal definition) has weight at least 1. In [10], the authors slightly extend the strictly non-Zeno cost property, to allow for cycles of weight exactly 0 while still preventing those of weight arbitrarily close to 0. Unfortunately, this leads to an undecidable value problem, but they propose a solution to the value approximation problem. **This article aims at extending these two restrictions in the presence of both non-negative and negative weights calling it the divergence property, in order to obtain either decidability of the value problem or approximations of the value for a large class of multi-clocks weighted timed games in the presence of arbitrary weights.**

---

[5]For example, a player may want to let time elapse as much as possible, but with delay $d < 1$ because of a strict guard.
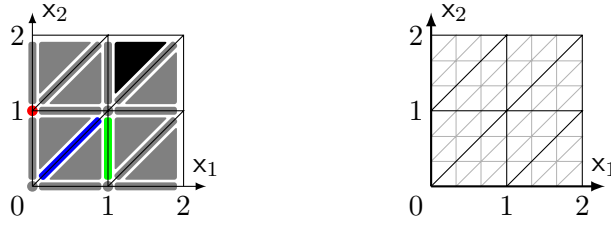
Figure 4: All 1/1-regions in $\mathsf{Reg}_1(\{x_1, x_2\}, 2)$ on the left, their refinement of granularity $1/3$ in $\mathsf{Reg}_3(\{x_1, x_2\}, 2)$ on the right.

Other objectives, not directly related to optimal reachability, have been considered in [12] for WTG, like mean-payoff and parity objectives. In this work, the authors manage to solve these problems for the so-called class of $\delta$-robust WTGs that they introduce.[6]

2.4. **Region abstraction.** We will rely on the crucial notion of regions, as introduced in the seminal work on timed automata [2]. Let $\mathbb{Q}_N = \{a/N \mid a \in \mathbb{Z}\}$ be the set of rational numbers of granularity $1/N$ for a fixed $N \in \mathbb{N}_{>0}$. Given a finite set of rational numbers $\mathcal{S} \subseteq \mathbb{Q}$, $\mathcal{S}$ is said to be of granularity $1/N$ if $\mathcal{S} \subseteq \mathbb{Q}_N$. Such $N$ always exists, and one can find the smallest one by decomposing elements of $\mathcal{S}$ as irreducible fractions $c/c'$ with $c \in \mathbb{Z}$, $c' \in \mathbb{N}_{>0}$ and use the least common multiple of all $c'$ as $N$. A guard $g$ is said to be of granularity $1/N$ if all constants in the atomic constraints of $g$ form a set of granularity $1/N$. A zone is of granularity $1/N$ if it can be described by a guard of granularity $1/N$. Let $\mathsf{Guards}_N(\mathcal{X}, M)$ denote the set of guards over $\mathcal{X}$ bounded by $M$ and of granularity $1/N$, and let $\mathsf{Guards}_N^{\mathrm{nd}}(\mathcal{X}, M)$ denote the non-diagonal ones. Given a finite set of guards $G \subseteq \mathsf{Guards}^{\mathrm{nd}}(\mathcal{X}, M)$, we can find $N$ such that $G \subseteq \mathsf{Guards}_N^{\mathrm{nd}}(\mathcal{X}, M)$, by denoting $\mathcal{S} \subseteq \mathbb{Q}$ the set of constants used in atomic constraints of $G$ and using $N$ the smallest integer such that $\mathcal{S}$ is of granularity $1/N$. For all $a \in \mathbb{R}_{\geqslant 0}$, $\lfloor a \rfloor \in \mathbb{N}$ denotes the integral part of $a$, and $\mathsf{fract}(a) \in [0, 1)$ its fractional part, such that $a = \lfloor a \rfloor + \mathsf{fract}(a)$.

**Definition 2.3.** With respect to the set $\mathcal{X}$ of clocks, a granularity $N \in \mathbb{N}_{>0}$ and an upper bound $M \in \mathbb{N}_{>0}$, we define $1/N$-*regions* as subsets of valuations $r$ characterised by a valuation $\iota \in [0, M)^{\mathcal{X}}$ called the integral part of $r$ such that $\iota(x) \in \mathbb{Q}_N$ for every $x \in \mathcal{X}$, and an ordered partition $R_0 \uplus R_1 \uplus \cdots \uplus R_m$ splitting $\mathcal{X}$ into $m + 1$ subsets. The ordered partition is denoted $0 = R_0 < R_1 < \cdots < R_m$, where $R_0$ can be empty but $R_i \neq \emptyset$ for $1 \leqslant i \leqslant m$. We denote by $\mathsf{Reg}_N(\mathcal{X}, M)$ the set of $1/N$-regions bounded by $M$.

A valuation $\nu$ in $[0, M)^{\mathcal{X}}$ belongs to $r$ if
- for all $x \in \mathcal{X}$, $\iota(x)N = \lfloor \nu(x)N \rfloor$;
- for all $x \in R_0$, $\mathsf{fract}(\nu(x)N) = 0$;
- for all $0 \leqslant i \leqslant m$, for all $x_1, x_2 \in R_i$, $\mathsf{fract}(\nu(x_1)N) = \mathsf{fract}(\nu(x_2)N)$.
- for all $i, j$ such that $0 \leqslant i < j \leqslant m$, for all $x_1 \in R_i$ and all $x_2 \in R_j$, $\mathsf{fract}(\nu(x_1)N) < \mathsf{fract}(\nu(x_2)N)$.

With granularity $N = 1$, we recover the classical notion of regions from [2], and we omit $N$ from previous notations about regions, such that $1/N$-regions are simply called regions,

---

[6]As mentioned in introduction, while our divergent games have a similar definition, both classes are incomparable.

and $\mathsf{Reg}_N(\mathcal{X}, M)$ is denoted $\mathsf{Reg}(\mathcal{X}, M)$. The set of valuations contained by a $1/N$-region $r$ characterised by $\iota$ and $0 = \{\mathsf{x}_1^0, \ldots, \mathsf{x}_{m_0}^0\} < \{\mathsf{x}_1^1, \ldots, \mathsf{x}_{m_1}^1\} < \cdots < \{\mathsf{x}_1^m, \ldots, \mathsf{x}_{m_m}^m\}$ can be described by a formula, constructed as a conjunction of inequalities over $\mathcal{X}$: If we let $f(\mathsf{x})$ denote the term $(\mathsf{x} - \iota(\mathsf{x}))N$, and $E^i$ denote the formula $f(\mathsf{x}_1^i) = \cdots = f(\mathsf{x}_{m_i}^i)$ for every $0 \leqslant i \leqslant m$, then $\nu \in r$ if and only if its coordinates satisfy

$$E^0 \wedge E^1 \wedge \cdots \wedge E^m \wedge (0 = f(\mathsf{x}_1^0) < f(\mathsf{x}_1^1) < \cdots < f(\mathsf{x}_1^m) < 1).$$

Therefore, every $1/N$-region is a zone of granularity $1/N$. $\mathsf{Reg}_N(\mathcal{X}, M)$ forms a finite partition of $[0, M)^{\mathcal{X}}$, and the number of $1/N$-regions is polynomial in $MN$ and exponential in $|\mathcal{X}|$. If $\nu$ is a valuation in $[0, M)^{\mathcal{X}}$, $[\nu]$ denotes the unique region that contains $\nu$. Valuations in the same $1/N$-region satisfy the same guards in $\mathsf{Guards}_N(\mathcal{X}, M)$. In fact, zones associated to guards in $\mathsf{Guards}_N(\mathcal{X}, M)$ can be described as a finite union of regions in $\mathsf{Reg}_N(\mathcal{X}, M)$. If $r$ is a $1/N$-region in $\mathsf{Reg}_N(\mathcal{X}, M)$, then the time-successor valuations in $\mathsf{PostTime}(r) \cap [0, M)^{\mathcal{X}}$ form a finite union of regions in $\mathsf{Reg}_N(\mathcal{X}, M)$, and the reset $r[\mathcal{Y} := 0]$ of $\mathcal{Y} \subseteq \mathcal{X}$ is a region in $\mathsf{Reg}_N(\mathcal{X}, M)$. A $1/N$-region $r'$ is said to be a time successor of the $1/N$-region $r$ if there exists $\nu \in r$, $\nu' \in r'$, and $d > 0$ such that $\nu' = \nu + d$.

   If $r$ is an $1/N$-region, let $\overline{r}$ denote its topological closure, i.e. the smallest zone that contains $r$ associated to a non-strict guard. The *corners* of $r$ are valuations in $\overline{r}$ that belong to $\mathbb{Q}_N^{\mathcal{X}}$. If $r$ is characterised by an integral part $\iota$ and a clock ordering $0 = R_0 < R_1 < \cdots < R_m$, then $\iota$ is a corner of $r$. If $m = 0$ then $r = \{\iota\}$, otherwise $r$ does not include its corners but contains valuations arbitrarily close to them. The corners of $r$ are the vertices of the polytope $\overline{r}$, such that $\overline{r}$ is their convex hull. There are at most $|\mathcal{X}| + 1$ corners in each $1/N$-region.

**Example 2.4.** Figure 4 represents the 24 regions of granularity $N = 1$ with upper bound $M = 2$ over two clocks. The green region is characterised by $\iota = (1, 0)$ and $0 = \{\mathsf{x}_1\} < \{\mathsf{x}_2\}$, corresponds to the formula $0 = \mathsf{x}_1 - 1 < \mathsf{x}_2 < 1$ and thus is equal to the zone $[\![\mathsf{x}_1 = 1 \wedge 0 < \mathsf{x}_2 < 1]\!]$. Its corners are the valuations $(1, 0)$ and $(1, 1)$. The red region is characterised by $\iota = (0, 1)$ and $0 = \{\mathsf{x}_1, \mathsf{x}_2\}$. The black region is characterised by $\iota = (1, 1)$ and $0 < \{\mathsf{x}_1\} < \{\mathsf{x}_2\}$. The blue region is characterised by $\iota = (0, 0)$ and $0 < \{\mathsf{x}_1, \mathsf{x}_2\}$.

   The partition into regions can be maintained throughout the play in a WTG. By forgetting about the precise valuation of clocks, we obtain a *region abstraction* (that is usually called the *region automaton* in the literature).

**Definition 2.5.** Given a WTG $\mathcal{G} = \langle L_{\mathsf{Min}}, L_{\mathsf{Max}}, \mathcal{X}, L_{\mathsf{t}}, E, \mathsf{wt}, \mathsf{fwt} \rangle$ such that all clocks are bounded by $M$ and all guards belong to $\mathsf{Guards}_N^{\mathrm{nd}}(\mathcal{X}, M)$ for some granularity $1/N$, we define the *region abstraction* of $\mathcal{G}$ as the finite labelled transition system $\langle L \times \mathsf{Reg}_N(\mathcal{X}, M), T \rangle$ labelled over $\mathsf{Reg}_N(\mathcal{X}, M) \times E$, where $T$ contains all transitions $(\ell, r) \xrightarrow{r'', e} (\ell', r')$ such that $e = \ell \xrightarrow{g, \mathcal{Y}} \ell'$ is an edge of $\mathcal{G}$, $r''$ is a time-successor of $r$, $r'' \models g$ and $r''[\mathcal{Y} := 0] = r'$.

   The states of the region abstraction are called *region states*, and its paths are called *region paths*. As there are finitely many regions, the region abstraction of $\mathcal{G}$ is a finite transition system, where paths $\mathsf{p}$ represent sequences of regions alternating between letting time elapse and taking edges, following some path $\pi$ in $\mathcal{G}$. We say that $\pi$ contains the region path $\mathsf{p}$. From an execution in $\mathcal{G}$ described by $\rho = (\ell_0, \nu_0) \xrightarrow{d_1, e_1} (\ell_1, \nu_1) \xrightarrow{d_2, e_2} \ldots$, we construct a region path $\mathsf{p} = (\ell_0, [\nu_0]) \xrightarrow{[\nu_0 + d_1], e_1} (\ell_1, [\nu_1]) \xrightarrow{[\nu_1 + d_2], e_2} \ldots$, and say that $\rho$ follows $\mathsf{p}$.

From the region abstraction, we can construct a *region game* that can be seen as a product of the original WTG with the region abstraction.

**Definition 2.6.** Given a WTG $\mathcal{G} = \langle L_{\mathsf{Min}}, L_{\mathsf{Max}}, \mathcal{X}, L_{\mathsf{t}}, E, \mathsf{wt}, \mathsf{fwt} \rangle$ such that all clocks are bounded by $M$ and all guards belong to $\mathsf{Guards}_N^{\mathrm{nd}}(\mathcal{X}, M)$ for some granularity $1/N$, we define the *region game* of $\mathcal{G}$ as the WTG $\mathcal{R}_N(\mathcal{G}) = \langle L_{\mathsf{Min}} \times \mathsf{Reg}_N(\mathcal{X}, M), L_{\mathsf{Max}} \times \mathsf{Reg}_N(\mathcal{X}, M), \mathcal{X}, L_{\mathsf{t}} \times \mathsf{Reg}_N(\mathcal{X}, M), E', \mathsf{wt}', \mathsf{fwt}' \rangle$ whose locations are region states, $\mathsf{wt}'$ and $\mathsf{fwt}'$ are obtained trivially from $\mathsf{wt}$ and $\mathsf{fwt}$ by forgetting about regions, and $E'$ is defined by transforming every transition $(\ell, r) \xrightarrow{r'', e} (\ell', r')$ in $T$, where $e$ is labelled by $(g, \mathcal{Y})$, into an edge $(\ell, r) \xrightarrow{g'', \mathcal{Y}} (\ell', r')$, with $[\![g'']\!] = r'' \subseteq [\![g]\!]$.

Every play in $\mathcal{G}$ exists in $\mathcal{R}_N(\mathcal{G})$ as a play following a region path $\mathsf{p}$, and conversely every play in $\mathcal{R}_N(\mathcal{G})$ following some region path $\mathsf{p}$ is contained in the path $\pi$ of $\mathcal{G}$ followed by $\mathsf{p}$. By projecting away the region information of $\mathcal{R}_N(\mathcal{G})$, we simply obtain:

**Lemma 2.7.** *Games $\mathcal{G}$ and $\mathcal{R}_N(\mathcal{G})$ contain the same plays. For all $\ell \in L$, $1/N$-regions $r$, and $\nu \in r$, $\mathsf{Val}_{\mathcal{G}}(\ell, \nu) = \mathsf{Val}_{\mathcal{R}_N(\mathcal{G})}((\ell, r), \nu)$.*

As we assume that guards have integer constants, we can use the granularity $1/N = 1$, and we will write $\mathcal{R}(\mathcal{G})$ instead of $\mathcal{R}_N(\mathcal{G})$ in that case.

2.5. **Corner-point abstraction.** Despite all the interest and success of regions to study timed systems, they are not sufficient to handle weighted timed automata/games. Indeed, for a single-clock case, and in a location $\ell$ of weight 1, spending time in $\ell$ from region $(0, 1)$ to region $\{2\}$ can cost any possible weight in the interval $(2, 3)$. We therefore must also keep a more precise information of *where we are inside each region*. This is the goal of the *corner-point abstraction* introduced in [5, 25] to study one-player WTGs with non-negative weights, generalised in [6] to handle negative weights, and in [8] for the multi-cost setting.

Recall that a valuation $\mathsf{v}$ is said to be a corner of a $1/N$-region $r$, if it belongs to the topological closure $\overline{r}$ and has coordinates multiple of $1/N$ ($\mathsf{v} \in (1/N)\mathbb{N}^{\mathcal{X}}$). We call corner state a triple $(\ell, r, \mathsf{v})$ that contains information about a region state $(\ell, r)$ of $\mathcal{R}_N(\mathcal{G})$, and a corner $\mathsf{v}$ of the $1/N$-region $r$. Every region has at most $|\mathcal{X}| + 1$ corners.

**Definition 2.8.** The corner-point abstraction $\Gamma_N(\mathcal{G})$ of a WTG $\mathcal{G}$ is the WTG obtained as a refinement of $\mathcal{R}_N(\mathcal{G})$ where guards on edges are enforced to stay on one of the corners of the current $1/N$-region: the locations of $\Gamma_N(\mathcal{G})$ are all corner states of $\mathcal{R}_N(\mathcal{G})$, associated to each player accordingly, and edges are all $(\ell, r, \mathsf{v}) \xrightarrow{g'', \mathcal{Y}} (\ell', r', \mathsf{v}')$ such that there exists $t = (\ell, r) \xrightarrow{g, \mathcal{Y}} (\ell', r')$ an edge of $\mathcal{R}_N(\mathcal{G})$ such that the model of guard $g''$ is a corner $\mathsf{v}''$ satisfying the guard $\overline{g}$ (recall that $\overline{g}$ is the closed version of $g$), $\mathsf{v}'' \in \mathsf{PostTime}(\mathsf{v})$, $\mathsf{v}' = \mathsf{v}''[\mathcal{Y} := 0]$, and there exist two valuations $\nu \in r$, $\nu' \in r'$ such that $((\ell, r), \nu) \xrightarrow{d', t} ((\ell', r'), \nu')$ for some $d' \in \mathbb{R}_{\geqslant 0}$ (the latter condition ensures that the edge between corners is not spurious, i.e. created by the closure of guards). Weights of locations and edges are trivially recovered from $\Gamma_N(\mathcal{G})$. We define the final weight function of $\Gamma_N(\mathcal{G})$ over the only valuation $\mathsf{v}$ reachable in location $(\ell, r, \mathsf{v})$ (with $\ell \in L_{\mathsf{t}}$) by $\mathsf{fwt}((\ell, r, \mathsf{v}), \mathsf{v}) = \lim_{\nu \to \mathsf{v}, \nu \in r} \mathsf{fwt}(\ell, \nu)$ (the limit is well defined since $\mathsf{fwt}$ is piecewise affine with a finite number of pieces on region $r$).

The WTG $\Gamma_N(\mathcal{G})$ can be seen as a finite weighted game, i.e. a WTG without clocks, by removing guards, resets and rates of locations, and replacing the weights of edges by the

actual weight of jumping from one corner to another: an edge $((\ell, r), \mathsf{v}) \xrightarrow{g'', \mathcal{Y}} ((\ell', r'), \mathsf{v}')$ becomes a transition from $((\ell, r), \mathsf{v})$ to $((\ell', r'), \mathsf{v}')$ with weight $d \cdot \mathsf{wt}(\ell) + \mathsf{wt}(t)$, with $d \in \mathbb{R}_{\geqslant 0}$ the only delay such that $[\![g'']\!] = \{\mathsf{v} + d\}$.[7] Note that delay $d$ is necessarily a rational of the form $\alpha/N$ with $\alpha \in \mathbb{N}$, since it must relate corners of $1/N$-regions. In particular, this proves that the cumulative weight $\mathsf{wt}_\Sigma(\overline{\rho})$ of a finite play $\overline{\rho}$ in $\Gamma_N(\mathcal{G})$ is indeed a rational number with denominator $N$.

We will call *corner play* every play $\overline{\rho}$ in the corner-point abstraction $\Gamma_N(\mathcal{G})$: it can also be interpreted as an execution in $\mathcal{G}$ where all guards are closed (as explained in the definition above). It straightforwardly projects on a finite path $\mathsf{p}$ in the region game $\mathcal{R}_N(\mathcal{G})$: in this case, we say again that $\overline{\rho}$ follows $\mathsf{p}$. Figure 5 depicts a play, its projected path in the region game and one of its associated corner plays.
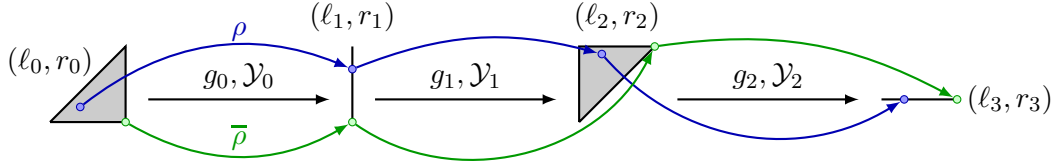


Figure 5: A play $\rho$ (in blue), its projected path $\mathsf{p}$ in the region game (in black), and one of its associated corner plays $\overline{\rho}$ (in green).

Let $\overline{\rho}$ be a corner play following a region path $\mathsf{p}$. The weight of $\overline{\rho}$ refers to its weight in $\Gamma_N(\mathcal{G})$. It is possible to find a play $\rho$ following $\mathsf{p}$ close to $\overline{\rho}$, in the sense that we control the difference between their respective cumulative weights:

**Lemma 2.9** ([8], Prop. 5)**.** *For all $\varepsilon > 0$, all finite region paths $\mathsf{p}$, and all corner plays $\overline{\rho}$ following $\mathsf{p}$, there exists a play $\rho$ in $\mathcal{G}$ following $\mathsf{p}$ such that $|\mathsf{wt}_\Sigma(\rho) - \mathsf{wt}_\Sigma(\overline{\rho})| \leqslant \varepsilon$.*

Thus, corner plays allow one to obtain faithful information on the plays that follow the same path.

**Lemma 2.10.** *If $\mathsf{p}$ is a finite region path in $\mathcal{R}_N(\mathcal{G})$, the set of cumulative weights $\{\mathsf{wt}_\Sigma(\rho) \mid \rho \text{ play of } \mathcal{G} \text{ following } \mathsf{p}\}$ is an interval bounded by the minimum and the maximum values of the set $\{\mathsf{wt}_\Sigma(\overline{\rho}) \mid \overline{\rho} \text{ corner play of } \Gamma_N(\mathcal{G}) \text{ following } \mathsf{p}\}$.*

*Proof.* The set $\{\mathsf{wt}_\Sigma(\rho) \mid \rho \text{ finite play following } \mathsf{p}\}$ is an interval as the image of a convex set by an affine function (see [6, Sec. 3.2] for an explanation).

The good properties of the corner-point abstraction allow us to conclude, since for every play $\rho$ following $\mathsf{p}$, one can find a corner play following $\mathsf{p}$ of smaller weight and one of larger weight [5], and for every corner play $\rho$ following $\mathsf{p}$ and every $\varepsilon > 0$, one can find a play following $\mathsf{p}$ whose weight is at most $\varepsilon$ away from $\mathsf{wt}_\Sigma(\rho)$ by Lemma 2.9. $\qquad\square$

An important property of the corner-point abstraction, derived from the assumption on the absence of deadlocks in the game, is that corner plays cannot get stuck as long as they follow a region path:

---

[7] In case several edges lead to the same transition, for instance when two transitions with different guards reset all clocks, we either allow for multi-transitions or choose the best weight according to the player owning the current location.
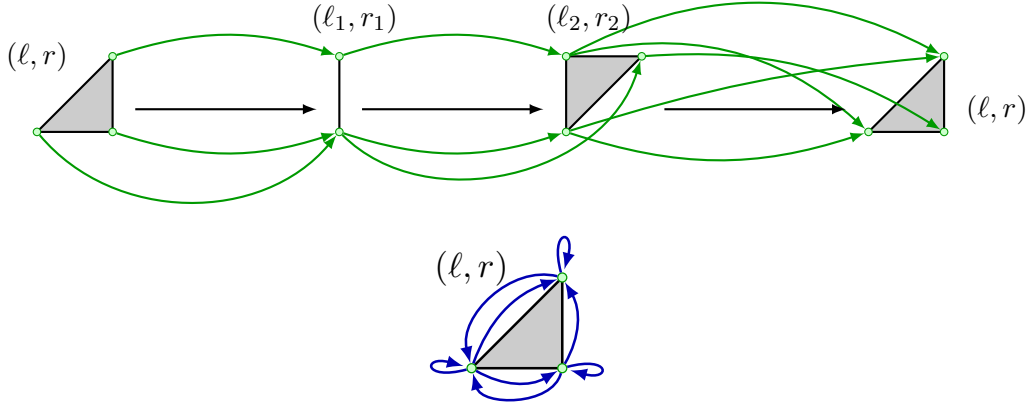
Figure 6: A region cycle $\mathsf{p}$ in the region game (in black), its associated corner plays (in green), and its folded orbit graph (in blue). Note that there is no edge between the top right corner and the bottom right corner, as no corner play goes from the former to the latter.

**Lemma 2.11** ([28], Lem. 8). *Let $\mathsf{p}$ be a region path starting from $(\ell, r)$ and ending in $(\ell', r')$. For all corners $\mathsf{v}$ of $r$, there exists a corner play following $\mathsf{p}$ that starts in $(\ell, r, \mathsf{v})$. For all corners $\mathsf{v}'$ of $r'$ there exists a corner play following $\mathsf{p}$ that ends in $(\ell', r', \mathsf{v}')$.*

Useful theoretical tools stem from the corner-point abstraction. Notably, let us focus on a cycle of the region abstraction. In order to study some properties of the corner plays following this cycle, we only need to consider the aggregation of all the behaviours following it. Inspired by the *folded orbit graphs* (FOG) introduced in [28], we define the folded orbit graph $\mathsf{FOG}(\mathsf{p})$ of a region cycle $\mathsf{p} = (\ell_1, r = r_1) \xrightarrow{e_1} (\ell_2, r_2) \xrightarrow{e_2} \cdots \xrightarrow{e_n} (\ell_1, r)$ in $\mathcal{R}_N(\mathcal{G})$ as a graph whose vertices are the corners states of region $r$, and that contains an edge from corner $\mathsf{v}$ to corner $\mathsf{v}'$ if there exists a corner play $\overline{\rho}$ from $(\ell_1, r, \mathsf{v})$ to $(\ell_1, r, \mathsf{v}')$ following $\mathsf{p}$. We fix $\overline{\rho}$ arbitrarily and label the edge between $\mathsf{v}$ and $\mathsf{v}'$ in $\mathsf{FOG}(\mathsf{p})$ by this corner play: it is then denoted by $\mathsf{v} \xrightarrow{\overline{\rho}} \mathsf{v}'$. An example is depicted in Figure 6.

The folded orbit graph inherits interesting topological properties from the corner-point abstraction. Notably, by Lemma 2.11, for all vertices $\mathsf{v}$, there exists at least one outgoing edge $\mathsf{v} \xrightarrow{\overline{\rho}'} \mathsf{v}'$, and at least one incoming edge $\mathsf{v}'' \xrightarrow{\overline{\rho}''} \mathsf{v}$ in $\mathsf{FOG}(\mathsf{p})$.

2.6. **Value functions and value iteration algorithm.** The value of a game has been defined as a mapping of each configuration $(\ell, \nu)$ to a value in $\mathbb{R}_\infty$. We call *value functions* such mappings from $L \times \mathbb{R}_{\geqslant 0}^{\mathcal{X}}$ to $\mathbb{R}_\infty$. If $\boldsymbol{V}$ represents a value function, we denote by $\boldsymbol{V}_\ell$ the mapping $\nu \mapsto \boldsymbol{V}(\ell, \nu)$. As observed in [9, 1], one step of the game is summarised in the following operator $\mathcal{F}$ mapping each value function $\boldsymbol{V}$ to a value function $\boldsymbol{V}' = \mathcal{F}(\boldsymbol{V})$ defined by $\boldsymbol{V}'_\ell(\nu) = \mathsf{fwt}(\ell, \nu)$ if $\ell \in L_\mathsf{t}$, and otherwise

$$\boldsymbol{V}'_\ell(\nu) = \begin{cases} \sup_{(\ell, \nu) \xrightarrow{d, e} (\ell', \nu')} \left[ d \cdot \mathsf{wt}(\ell) + \mathsf{wt}(e) + \boldsymbol{V}_{\ell'}(\nu') \right] & \text{if } \ell \in L_\mathsf{Max} \\ \inf_{(\ell, \nu) \xrightarrow{d, e} (\ell', \nu')} \left[ d \cdot \mathsf{wt}(\ell) + \mathsf{wt}(e) + \boldsymbol{V}_{\ell'}(\nu') \right] & \text{if } \ell \in L_\mathsf{Min} \end{cases} \tag{2.1}$$

where $(\ell, \nu) \xrightarrow{d, e} (\ell', \nu')$ ranges over valid transitions in $\mathcal{G}$.

Then, starting from $\boldsymbol{V}^0$ mapping every configuration $(\ell, \nu)$ to $+\infty$, except for the targets mapped to $\mathsf{fwt}(\ell, \nu)$, we let $\boldsymbol{V}^i = \mathcal{F}(\boldsymbol{V}^{i-1})$ for all $i > 0$. The value function $\boldsymbol{V}^i$ contains the value $\mathsf{Val}_{\mathcal{G}}^i$, which is intuitively what $\mathsf{Min}$ can guarantee when forced to reach the target in at most $i$ steps. More formally, we define $\mathsf{wt}^i(\rho)$ the weight of a maximal play $\rho$ at horizon $i$, as $\mathsf{wt}(\rho)$ if $\rho$ reaches a target in at most $i$ steps, and $+\infty$ otherwise. Then, $\mathsf{Val}_{\mathcal{G}}^i(\ell, \nu) = \inf_{\sigma_{\mathsf{Min}}} \sup_{\sigma_{\mathsf{Max}}} \mathsf{wt}^i(\mathsf{play}((\ell, \nu), \sigma_{\mathsf{Min}}, \sigma_{\mathsf{Max}}))$ refers to the value at horizon $i$.

We compare value functions componentwise: if $\boldsymbol{V}, \boldsymbol{V}'$ are two value function, we let $\boldsymbol{V} \leqslant \boldsymbol{V}'$ if $\boldsymbol{V}(\ell, \nu) \leqslant \boldsymbol{V}'(\ell, \nu)$ for all configurations $(\ell, \nu)$. Notice that $\mathcal{F}$ is a monotonic operator, i.e. if $\boldsymbol{V} \leqslant \boldsymbol{V}'$, then $\mathcal{F}(\boldsymbol{V}) \leqslant \mathcal{F}(\boldsymbol{V}')$. Moreover, $\mathcal{F}(\boldsymbol{V}^0) \leqslant \boldsymbol{V}^0$ since $\boldsymbol{V}^0$ maps every non-target state to $+\infty$, and target state keep the same value. It follows that the sequence $(\boldsymbol{V}^i)_{i \in \mathbb{N}}$ is non-increasing, as $\boldsymbol{V}^i = \mathcal{F}^i(\boldsymbol{V}^0) \geqslant \mathcal{F}^i(\mathcal{F}(\boldsymbol{V}^0)) = \boldsymbol{V}^{i+1}$.

Let us now present known results for the special case of games with no clocks. In this case, the definition of the operator $\mathcal{F}$ of (2.1) is simplified into

$$\boldsymbol{V}_\ell' = \begin{cases} \min_{\ell \to \ell'} \left[ \mathsf{wt}(\ell, \ell') + \boldsymbol{V}_{\ell'} \right] & \text{if } \ell \in L_{\mathsf{Min}} \\ \max_{\ell \to \ell'} \left[ \mathsf{wt}(\ell, \ell') + \boldsymbol{V}_{\ell'} \right] & \text{if } \ell \in L_{\mathsf{Max}} \end{cases} \tag{2.2}$$

Notice that we remove the valuation part of the notation: configurations are thus simply locations of the game. The value iteration algorithm proposed in [15] consists in finding the greatest fixpoint of operator $\mathcal{F}$, i.e. the limit of the sequence $(\boldsymbol{V}^i)_{i \in \mathbb{N}}$. Indeed, this greatest fixpoint is known to be the vector of values of the game (see, e.g., [15, Corollary 11]). For the special case of acyclic games of depth $d$, the fixpoint is reached after $d$ steps, and $\mathsf{Val} = \boldsymbol{V}^d$. In this case, the infinite values in $\mathcal{G}$ (i.e. configurations $\ell$ with $\mathsf{Val}_{\mathcal{G}}(\ell) \in \{-\infty, +\infty\}$) are derived from reaching targets with infinite final weights. If the game contains cycles, infinite values can also come from arbitrarily long plays: a state can have value $+\infty$ if $\mathsf{Max}$ can force an infinite play, never reaching any target, and it can have value $-\infty$ if $\mathsf{Min}$ can enforce an arbitrarily low weight, e.g. by staying in a cycle of negative cumulative weight. These $+\infty$ states correspond to a safety objective for player $\mathsf{Max}$, and can be computed in polynomial time: it is shown in [15] that for all locations $\ell$, $\mathsf{Val}_{\mathcal{G}}(\ell) = +\infty$ if and only if $\boldsymbol{V}_\ell^{|L|} = +\infty$. In contrast, deciding if a location has value $-\infty$ has no known polynomial solution (it is as hard as solving mean-payoff games). In [15], it is shown that in the presence of negative weights the sequence $(\boldsymbol{V}^i)_{i \in \mathbb{N}}$ stabilises after a number of iterations pseudo-polynomial on states with value in $\mathbb{R} \cup \{+\infty\}$, and that states with value $-\infty$ can be detected in this computation (they are those where the computed value goes under a given threshold).

2.7. **Piecewise affine value functions.** Such a computation of the greatest fixed point in finite time might not be possible in the timed setting, since the value problem is undecidable in general. We will come back later on this technique, also used by [9] for special classes of weighted timed games.

We will heavily rely on the class of *piecewise affine value functions*, and a way to efficiently encode them, as developed in [1]. This class is closed by the application of operator $\mathcal{F}$, as we will recall precisely in Section 7. Let $n$ denote the number of clocks, such that $\mathcal{X} = \{\mathsf{x}_1, \ldots, \mathsf{x}_n\}$, and $\ell \in L$ a location of $\mathcal{G}$. An *affine value function* is a mapping $\boldsymbol{V}_\ell : [0, M)^{\mathcal{X}} \to \mathbb{R}_\infty$ such that for all $\nu \in [0, M)^{\mathcal{X}}$,

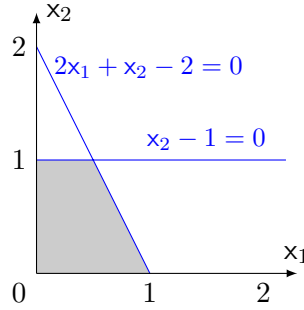$$\boldsymbol{V}_\ell(\nu) = a_1 \cdot \nu(\mathsf{x}_1) + \cdots + a_n \cdot \nu(\mathsf{x}_n) + b$$

Figure 7: The cell $2\mathsf{x}_1 + \mathsf{x}_2 - 2 < 0 \wedge \mathsf{x}_2 - 1 < 0$ in gray, and its borders in blue.

with partial derivatives $a_i \in \mathbb{Q}$ for $1 \leqslant i \leqslant n$, and additive constant $b \in \mathbb{Q}$. In this case, we say that $\boldsymbol{V}_\ell$ is defined by the equation $\mathsf{y} = a_1\mathsf{x}_1 + \cdots + a_n\mathsf{x}_n + b$ where the variable $\mathsf{y} \notin \mathcal{X}$ refers to $\boldsymbol{V}_\ell(\mathsf{x}_1, \ldots, \mathsf{x}_n)$. We also consider infinite mappings $\nu \mapsto +\infty$ and $\nu \mapsto -\infty$ to be affine value functions, defined with null partial derivatives and an infinite constant $b$.

Intuitively, we define a piecewise affine value function as a partition of $[0, M)^{\mathcal{X}}$ into finitely many polyhedra, called cells, each equipped by an affine value function. Formally, an affine inequality is an equation $I$ of the form

$$a_1\mathsf{x}_1 + \cdots + a_n\mathsf{x}_n + b \prec 0$$

where $b \in \mathbb{Q}$ is the additive constant of $I$, $\prec \in \{<, \leqslant\}$ is its comparison operator, and for every $1 \leqslant i \leqslant n$, $a_i \in \mathbb{Q}$ is the $i$-th partial derivative of $I$. Similarly, an affine equality is an equation $E$ of the form

$$a_1\mathsf{x}_1 + \cdots + a_n\mathsf{x}_n + b = 0.$$

We say that $\nu \in \mathbb{R}_{\geqslant 0}^{\mathcal{X}}$ satisfies $I$ (resp. $E$), and write $\nu \models I$ (resp. $\nu \models E$), if $a_1 \cdot \nu(\mathsf{x}_1) + \cdots + a_n \cdot \nu(\mathsf{x}_n) + b \prec 0$ (resp. $= 0$) holds. In this case, $[\![I]\!]$ (resp. $[\![E]\!]$) refers to the set of valuations that satisfy $I$ (resp. $E$). Equalities (resp. inequalities) are equivalent when they are satisfied by the same valuations. In particular, multiplying the additive constant $b$ and all partial derivatives $a_i$ by the same factor $N \in \mathbb{N}_{>0}$ gives an equivalent equality (resp. inequality), and we will therefore assume that they are always integers.

**Definition 2.12.** A *cell* is a set $c \subseteq \mathbb{R}_{\geqslant 0}^{\mathcal{X}}$, defined by a conjunction of affine inequalities $I_1 \wedge \cdots \wedge I_m$, such that $\nu \in c$ if and only if for all $1 \leqslant i \leqslant m$, $\nu \models I_i$. We write $c = [\![I_1 \wedge \cdots \wedge I_m]\!]$ in this case.

Cells are convex polyhedra, and the intersection of finitely many cells is a cell. From every affine inequality $I$, we can extract an affine equality $E(I)$, of identical partial derivatives and additive constant. Then, we call *borders* of a cell $c = [\![I_1 \wedge \cdots \wedge I_m]\!]$ the affine equalities $E(I_1), \ldots, E(I_m)$. The closure $\overline{c}$ of a cell $c$ is obtained by replacing every comparison operator $<$ by $\leqslant$ in its affine inequalities. Note that regions and zones are particular cases of cells, where borders are of the form $\mathsf{x} + b = 0$ or $\mathsf{x} - \mathsf{x}' + b = 0$. An example of cell and its borders is given in Figure 7.

Let $E$ be an affine equality of equation $a_1\mathsf{x}_1 + \cdots + a_n\mathsf{x}_n + b = 0$. We say that $\mathbb{R}_{\geqslant 0}^{\mathcal{X}}$ is partitioned by $E$ into three cells:

- $c_<$, defined by $a_1\mathsf{x}_1 + \cdots + a_n\mathsf{x}_n + b < 0$;
- $c_>$, defined by $a_1\mathsf{x}_1 + \cdots + a_n\mathsf{x}_n + b > 0$, i.e. $-a_1\mathsf{x}_1 - \cdots - a_n\mathsf{x}_n - b < 0$;

- $c_=$, defined by $a_1 x_1 + \cdots + a_n x_n + b = 0$, i.e. the conjunction of $a_1 x_1 + \cdots + a_n x_n + b \leqslant 0$ and $-a_1 x_1 - \cdots - a_n x_n - b \leqslant 0$.

Then, given a set $\mathcal{E} = \{E_1, \ldots, E_m\}$ of affine equalities, we denote $c_<^j$, $c_>^j$ and $c_=^j$ the three cells obtained from $E_j \in \mathcal{E}$. For every mapping $\phi : \mathcal{E} \to \{<, >, =\}$, we define $c_\phi$ as the cell $c_{\phi(E_1)}^1 \cap \cdots \cap c_{\phi(E_m)}^m$. Every valuation of $\mathbb{R}_{\geqslant 0}^{\mathcal{X}}$ belongs to some $c_\phi$, and if $\phi \neq \phi'$ then $c_\phi \cap c_{\phi'} = \emptyset$: hence, the set of mappings $\{<, >, =\}^{\mathcal{E}}$ provides a partition of $\mathbb{R}_{\geqslant 0}^{\mathcal{X}}$ into $3^m$ cells. We say that $\mathbb{R}_{\geqslant 0}^{\mathcal{X}}$ is partitioned by $\mathcal{E}$ into $m' \in \mathbb{N}$ cells if $m'$ of those $3^m$ cells are non-empty. We denote $\mathsf{Splits}(m, n)$ the greatest $m'$ over any partition of $\mathbb{R}_{\geqslant 0}^{\mathcal{X}}$ by $m$ affine equalities. Similarly, a cell $c \subseteq \mathbb{R}_{\geqslant 0}^{\mathcal{X}}$ is partitioned by $\mathcal{E}$ into at most $\mathsf{Splits}(m, n)$ sub-cells that have non-empty intersection with $c$. In particular, under the bounded clocks assumption we will partition $[0, M)^{\mathcal{X}}$ instead of $\mathbb{R}_{\geqslant 0}^{\mathcal{X}}$.
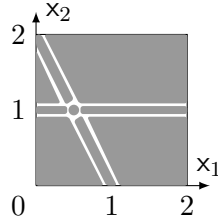


Figure 8: A partition of $[0, 2)^{\mathcal{X}}$ according to the two affine equalities of Figure 7.

**Example 2.13.** The $\mathsf{Splits}(2, 2) = 9$ cells that partition $[0, 2)^{\mathcal{X}}$ according to $\mathcal{E} = \{2x_1 + x_2 - 2 = 0, x_2 - 1 = 0\}$ are represented in Figure 8.

In fact, the number of cells $\mathsf{Splits}(m, n)$ is bounded by $\mathcal{O}((2m)^n)$, which we will use abundantly to compute the complexity of our algorithms.

**Lemma 2.14.** *For all $m, n \geqslant 0$, $\mathsf{Splits}(m, n) \leqslant 2^n (m + 1)^n$.*

*Proof.* The number $\mathsf{Splits}(m, n)$ is known in discrete geometry as the number of faces obtained when partitioning $\mathbb{R}^n$ by $m$ hyperplanes. We refer to [27, Chap. 6.1] for more details on this problem, and make use of their vocabulary in this proof, so that cells are called faces and affine equalities are called hyperplanes. The faces are relatively open convex sets, with dimensions ranging from $0$ to $n$. We call $k$-faces the faces of dimension $k$. For example, in Figure 8 the 0-face is the point at the intersection of the two hyperplanes, the four 1-faces are half-lines, and the four 2-faces partition the remaining points of $\mathbb{R}^2$.

According to [27, Prop. 6.1.1], the number of $n$-faces in a space of dimension $n$ is $\sum_{\ell=0}^{\min(n,m)} \binom{m}{\ell}$. These faces are known as faces of full dimension. If $n$ or $m$ equals $0$, $\mathsf{Splits}(m, n) = 1$. If $n, m \geqslant 1$, the $(n-1)$-faces are obtained in each of our hyperplanes (spaces of dimension $n - 1$), by considering their partition by the $m - 1$ other hyperplanes. In these partitions, the $(n-1)$-faces are of full dimension, so that each hyperplane contains $\sum_{\ell=0}^{\min(n-1,m-1)} \binom{m-1}{\ell}$ such faces. The total number of $(n-1)$-faces is therefore $m \sum_{\ell=0}^{\min(n-1,m-1)} \binom{m-1}{\ell}$. Let us now count the number of $k$-faces for $k \in [0, n-2]$.

As explained in [27, Chap. 6.1], the number of faces is maximised when the hyperplanes are in general position, i.e. for any $k \in [2, \min(n+1, m)]$, the intersection of any set of $k$ hyperplanes is $(n-k)$-dimensional.[8] Then, if $n \geqslant 2$ the faces of dimension $k \in [\max(0, n -$

---

[8] for $n - k < 0$ this means that the intersection is empty.

$m), n - 2]$ lie at the intersection of $n - k$ hyperplanes, and for each such intersection $L$ the $k$-faces are obtained by partitioning $L$ by the $m - n + k$ other hyperplanes. They are faces of full dimension in $L$, so that $L$ contains $\sum_{\ell=0}^{\min(k,m-n+k)} \binom{m-n+k}{\ell}$ such faces. There are $\binom{m}{n-k}$ intersections $L$, so that the total number of $k$-faces is $\sum_{\ell=0}^{\min(k,m-n+k)} \binom{m}{n-k}\binom{m-n+k}{\ell}$. Note that this formula matches the number of $(n-1)$-faces and the number of $n$-faces previously given, and that if $m \leqslant n$ there are no faces of dimension $k < n - m$. Therefore,

$$\mathsf{Splits}(m,n) = \sum_{k=\max(0,n-m)}^{n} \sum_{\ell=0}^{\min(k,m-n+k)} \binom{m}{n-k}\binom{m-n+k}{\ell}.$$

Let us show $\mathsf{Splits}(m,n) \leqslant 2^n(m+1)^n$. First, note that for all $0 \leqslant \ell \leqslant k \leqslant n$ with $n - k \leqslant m$ and $\ell \leqslant m - n + k$, it holds that $\binom{m}{n-k}\binom{m-n+k}{\ell} = \binom{m}{n-k+\ell}\binom{n-k+\ell}{n-k} \leqslant \binom{m}{n-k+\ell}\binom{n}{k}$. Then, $\mathsf{Splits}(m,n) \leqslant \sum_{k=\max(0,n-m)}^{n} \left[ \binom{n}{k} \sum_{\ell=0}^{\min(k,m-n+k)} \binom{m}{n-k+\ell} \right]$. Moreover, $\sum_{\ell=0}^{\min(k,m-n+k)} \binom{m}{n-k+\ell} \leqslant \sum_{\ell=0}^{\min(m,n)} \binom{m}{\ell}$.

Then, note that $\sum_{\ell=0}^{n} \binom{m}{\ell}$ is the number of selections of size at most $n$ out of a set of $m$ elements, without repetitions. It is smaller than the number of selections of $n$ elements, with repetitions, in a set of size $m+1$ (the extra element can be used as padding). Therefore, $\sum_{\ell=0}^{\min(n,m)} \binom{m}{\ell} \leqslant (m+1)^n$. On the other hand, $\sum_{k=\max(0,n-m)}^{n} \binom{n}{k} \leqslant \sum_{k=0}^{n} \binom{n}{k} = 2^n$. It follows that $\mathsf{Splits}(m,n) \leqslant 2^n(m+1)^n$. $\qquad\square$

**Definition 2.15.** A *partition* $P$ is defined by a cell $c_P$ and a set $\mathcal{E}_P$ of affine equalities, such that $P$ encodes the set of cells that partition $c_P$ according to $\mathcal{E}_P$, called *base cells*.

The cell $c_P$ is called the domain of $P$. We denote $[\nu]_P$ the base cell that contains valuation $\nu \in c_P$.

**Definition 2.16.** A *partition value function* $F$ defined over a partition $P$ is a mapping from the base cells of $P$ to affine value functions. It encodes a mapping from $c_P$ to $\mathbb{R}_\infty$, denoted $[\![F]\!]$: if $\nu \in c_P$ and $F([\nu]_P)$ is defined by $\mathsf{y} = a_1\mathsf{x}_1 + \cdots + a_n\mathsf{x}_n + b$, then $[\![F]\!](\nu)$ equals $a_1 \cdot \nu(\mathsf{x}_1) + \cdots + a_n \cdot \nu(\mathsf{x}_n) + b$.

A partition value function $F$ of domain $c_P$ is *continuous* if for all $\nu \in c_P$, for every base cell $c_b$ such that $\nu \in \overline{c_b}$, if $F(c_b)$ is defined by $\mathsf{y} = a_1\mathsf{x}_1 + \cdots + a_n\mathsf{x}_n + b$ then $[\![F]\!](\nu) = a_1 \cdot \nu(\mathsf{x}_1) + \cdots + a_n \cdot \nu(\mathsf{x}_n) + b$. In other words, the affine equations provided by $F$ to neighbouring cells should match on the borders that separate them.

Finally, the *piecewise affine value function* $\boldsymbol{V}_\ell : [0, M)^{\mathcal{X}} \to \mathbb{R}_\infty$ of a location $\ell$ is encoded as a pair $(P, F)$ where $P$ is a partition of domain $[0, M)^{\mathcal{X}}$, and $F$ is a partition value function defined over $P$, such that $[\![F]\!] = \boldsymbol{V}_\ell$.

A piecewise affine value function $(P, F)$ is said *continuous on regions* if for every region $r \in \mathsf{Reg}(\mathcal{X}, M)$, the restriction of $F$ to domain $r$ is continuous. There could be discontinuities in $F$, but only at borders separating different regions. In particular, if a partition value function is continuous over regions, and $[\![F]\!](\nu) = +\infty$ (resp. $-\infty$) for some $\nu$, then for all $\nu'$ in the same region as $\nu$, $[\![F]\!](\nu') = [\![F]\!](\nu)$.

*Remark.* In [1], the domain of partitions is always a single region, and one value function is associated to each region. We define value functions over $[0, M)^{\mathcal{X}}$ instead, in order to obtain a symbolic algorithm, independant of regions. This induces slight differences in the way value functions are defined, because the mappings of [1] are continuous everywhere while

ours can have discontinuities at borders between regions. They define their partitions with overlaps over borders, such that $\mathbb{R}^{\mathcal{X}}_{\geqslant 0}$ is partitioned by an affine equality into two cells, $c_{\leqslant}$ and $c_{\geqslant}$, instead of the three $c_<$, $c_>$ and $c_=$. This changes the number of cells $\mathsf{Splits}(m, n)$ to $\mathcal{O}(m^n)$ instead of $\mathcal{O}((2m)^n)$.

Cells are bounded, convex polyhedra. Elementary operations over cells (emptiness, intersection and inclusion tests) can be seen as instances of linear programming, and can thus be performed in polynomial time. As we will formally explain in Section 7, this data structure allows one to effectively compute the iterates $(\boldsymbol{V}^i)_{i \in \mathbb{N}}$. Contrary to the case without clocks, recalled before, this sequence does not stabilise in general. The decidability results obtained before, as well as the ones we obtain in this article, are all based on reasons to either make the sequence stabilise or stop its computation after a sufficient number $i$ of turns to obtain a good approximation of the value.

## 3. Divergent and almost-divergent WTGs

In this section, we introduce several classes of weighted timed games for which we state the results that this article shows. Let us start with the class of weighted timed games studied in [9], to our knowledge the greatest class of WTG where the value problem is known to be decidable.

**Definition 3.1.** A weighted timed game $\mathcal{G}$ with non-negative weights satisfies the *strictly non-Zeno cost* property when every finite play $\rho$ in $\mathcal{G}$ following a cycle in the region automaton $\mathcal{R}(\mathcal{G})$ satisfies $\mathsf{wt}_\Sigma(\rho) \geqslant 1$.

The intuition behind this class is that the weight of any long enough execution in $\mathcal{G}$ will ultimately grow above any fixed bound, and diverge towards $+\infty$ for an infinite execution. Therefore, the value of $\mathcal{G}$ is equal to the value $\mathsf{Val}^i_\mathcal{G}$ at some horizon $i$ large enough (as defined in Section 2.6), making the value problem decidable. It is shown in [9] that $i$ can be bounded exponentially in the size of $\mathcal{G}$.

We introduce divergent weighted timed games, as a natural generalisation of the strictly non-Zeno cost property to weights in $\mathbb{Z}$.

**Definition 3.2.** A WTG $\mathcal{G}$ is *divergent* when every finite play $\rho$ in $\mathcal{G}$ following a cycle in the region automaton $\mathcal{R}(\mathcal{G})$ satisfies $\mathsf{wt}_\Sigma(\rho) \notin (-1, 1)$.

If $\mathcal{G}$ has only non-negative weights on locations and edges, this definition matches with the strictly non-Zeno cost property of [9], we will therefore refer to their class as the class of divergent WTG with non-negative weights.

*Remark.* As in [9], we could replace $(-1, 1)$ by $(-\kappa, \kappa)$ to define a notion of $\kappa$-divergence. However, since weights and guard constraints in weighted timed games are integers, for $\kappa \in (0, 1)$, a weighted timed game $\mathcal{G}$ is $\kappa$-divergent if and only if it is divergent. This will be formally implied by Proposition 4.3 and Lemma 2.10.

Our contributions on divergent WTGs summarise as follows:

**Theorem 3.3.** *The value problem over divergent WTGs is decidable in* 3-EXPTIME, *and is* EXPTIME-*hard. Moreover, deciding if a given WTG is divergent is a* PSPACE-*complete problem.*

| weights in $\mathbb{Z}$ | weights in $\mathbb{N}$ |
|---|---|

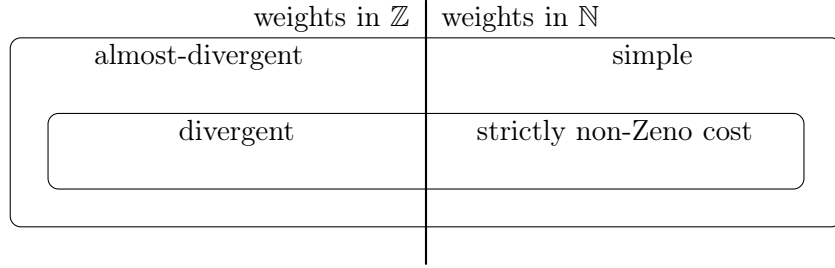| almost-divergent | simple |
|---|---|
| divergent | strictly non-Zeno cost |

Figure 9: Classes of weighted timed games, and their respective restrictions to non-negative weights.

In [10], the authors slightly extend the strictly non-Zeno cost property, to allow for cycles of weight exactly 0 while still preventing those of weight arbitrarily close to 0:

**Definition 3.4.** A WTG $\mathcal{G}$ with non-negative weights is called *simple* when every finite play $\rho$ in $\mathcal{G}$ following a cycle in the region automaton $\mathcal{R}(\mathcal{G})$ satisfies $\mathsf{wt}_\Sigma(\rho) \in \{0\} \cup [1, +\infty)$.

Unfortunately, it is shown in [10] that the value problem is undecidable for simple WTGs. They propose a solution to the value approximation problem, as a procedure computing an approximation of the value of every configuration. The intuition is that cycles of weight exactly 0 are only possible when every (non-negative) weight encountered along the cycle equals 0, allowing one to define a subgame where every cyclic execution has weight 0. One can then analyse this subgame separately, by applying a semi-unfolding procedure on $\mathcal{R}(\mathcal{G})$.

We introduce a class of WTGs that will extend the notion of simple WTGs and allow negative weights, so that cyclic executions of weight exactly 0 are allowed, but not those close to 0. The first attempt would lead to the requirement that every finite play following a cycle in the region automaton $\mathcal{R}(\mathcal{G})$ has a weight in $(-\infty, -1] \cup \{0\} \cup [1, +\infty)$. However, we did not obtain positive results for the value approximation problem on this class of WTGs since cycles of weight exactly 0 do not have the good property presented above for simple WTGs. Instead, we require a stability by decomposition for cycles of weight 0.

If $\mathsf{p} = (\ell_0, r_0) \xrightarrow{r_1, e_0} (\ell_1, r_1) \xrightarrow{r_2, e_1} \cdots (\ell_{k-1}, r_{k-1}) \xrightarrow{r_0, e_{k-1}} (\ell_0, r_0)$ is a region cycle in $\mathcal{R}(\mathcal{G})$, it is either simple (i.e. for all $i, j$ such that $0 \leqslant i < j < k$, $(\ell_i, r_i) \neq (\ell_j, r_j)$) or we can extract smaller cycles from it. Indeed, if $\mathsf{p}$ is not simple, there exists a pair $(i, j)$ such that $0 \leqslant i < j < k$ and $(\ell_i, r_i) = (\ell_j, r_j)$. Then, for such a pair, we can write $\mathsf{p} = \mathsf{p}_1\mathsf{p}_2\mathsf{p}_3$ such that $|\mathsf{p}_1| = i$, $|\mathsf{p}_3| = k - j$. It follows that $\mathsf{p}_2$ and $\mathsf{p}_3\mathsf{p}_1$ are both region cycles around $(\ell_i, r_i)$. This process is called a decomposition of $\mathsf{p}$ into smaller cycles $\mathsf{p}' = \mathsf{p}_3\mathsf{p}_1$ and $\mathsf{p}'' = \mathsf{p}_2$.

**Definition 3.5.** A WTG $\mathcal{G}$ is *almost-divergent* if every play $\rho$ following a cycle $\mathsf{p}$ of $\mathcal{R}(\mathcal{G})$ satisfies $\mathsf{wt}_\Sigma(\rho) \in (-\infty, -1] \cup \{0\} \cup [1, +\infty)$, and if $\mathsf{wt}_\Sigma(\rho) = 0$ then for every decomposition of $\mathsf{p}$ into smaller cycles $\mathsf{p}'$ and $\mathsf{p}''$, and plays $\rho'$ and $\rho''$ following $\mathsf{p}'$ and $\mathsf{p}''$, respectively, it holds that $\mathsf{wt}_\Sigma(\rho') = \mathsf{wt}_\Sigma(\rho'') = 0$.[9]

Clearly, every divergent WTG is almost-divergent. Moreover, as we will see in Proposition 4.4, when weights are non-negative, this class matches the simple WTGs of [10] therefore inheriting their undecidability result. We will thus refer to simple WTGs as almost-divergent

---

[9]Once again, we could replace $-1, 1$ by $-\kappa, \kappa$ with $0 < \kappa < 1$ to define an equivalent notion of $\kappa$-almost-divergence.
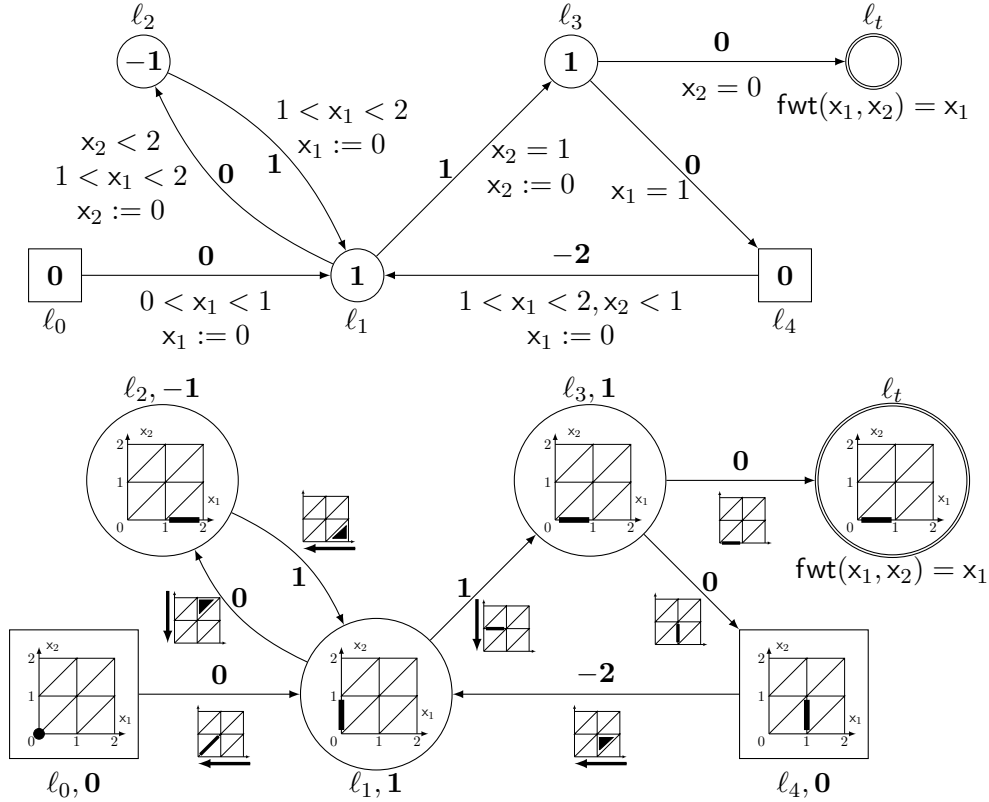
Figure 10: A weighted timed game $\mathcal{G}$ with two clocks $x_1$ and $x_2$, and the portion of its region game $\mathcal{R}(\mathcal{G})$ accessible from configuration $(\ell_0, (0,0))$. The states of $\mathcal{R}(\mathcal{G})$ are labelled by their associated region, location and weight, and edges are labelled by a representation of their guards and resets. For example, the edge from $(\ell_0, r_0)$ to $(\ell_1, r_1)$ in $\mathcal{R}(\mathcal{G})$ highlights the time successors of the region $r_0$ that satisfy the guard $0 < x_1 < 1$, and the arrow represents the direction in which this set of points is projected by the clock reset $x_1 := 0$, so that we end up in the region $r_1$.

WTGs with non-negative weights. Figure 9 represents the hierarchy of the classes of WTG that we introduced.

**Example 3.6.** Consider the WTG $\mathcal{G}$ in Figure 3. The self-loop on $\ell_1$ contains a cycle of $\mathcal{R}(\mathcal{G})$ around $(\ell_1, \mathbf{0})$ that jumps to the region $1 < x < 2$. For every $d \in (1,2)$, there exists a play $\rho$ following this cycle that uses delay $d$, so that $\mathsf{wt}_\Sigma(\rho) = 3 - 2d \in (-1, 1)$. It follows that $\mathcal{G}$ is neither divergent nor almost-divergent. Changing the guard on this self-loop to $2 \leqslant x < 3$ makes $\mathcal{G}$ divergent, as every region cycle left in $\mathcal{G}$ iterates the self-loops around $(\ell_1, \mathbf{0})$ and $(\ell_5, \mathbf{0})$ of cumulative weights in $(-3, -1]$ and $(-5, -1)$, respectively.

**Example 3.7.** Consider the WTG $\mathcal{G}$ in Figure 10, and its region game $\mathcal{R}(\mathcal{G})$. We chose an example where $\mathcal{R}(\mathcal{G})$ is isomorphic to $\mathcal{G}$ for readability reasons. $\mathcal{R}(\mathcal{G})$ contains one SCC $\{\ell_1, \ell_2, \ell_3, \ell_4\}$, made of two simple cycles, $\mathsf{p}_1 = \ell_1 \to \ell_2 \to \ell_1$ and $\mathsf{p}_2 = \ell_1 \to \ell_3 \to \ell_4 \to \ell_1$, so that:

- all plays following $\mathsf{p}_1$ have cumulative weight in the interval $(1, 3)$,

- and all plays following $p_2$ have cumulative weight 0. This can be checked by Lemma 2.10. As every cycle $p$ of $\mathcal{R}(\mathcal{G})$ either iterates $p_2$ only, or contains $p_1$, it holds that a play $\rho$ following $p$ satisfies $\mathsf{wt}_\Sigma(\rho) \in \{0\} \cup [1, +\infty)$, and if $\mathsf{wt}_\Sigma(\rho) = 0$ then any decomposition of $p$ into smaller cycles $p'$ and $p''$ implies that they all follow iterates of $p_2$, so that plays along them must have cumulative weight 0. Therefore, $\mathcal{G}$ is almost-divergent. If one removes $\ell_4$, $\mathcal{G}$ becomes divergent.

Our first result on almost-divergent WTGs is the following extension of the approximation procedure for non-negative weights:

**Theorem 3.8.** *Given an almost-divergent WTG $\mathcal{G}$, a location $\ell$ and $\varepsilon \in \mathbb{Q}_{>0}$, we can compute an $\varepsilon$-approximation of $\mathsf{Val}_\mathcal{G}(\ell, \mathbf{0})$ in time triply-exponential in the size of $\mathcal{G}$ and polynomial in $1/\varepsilon$. Moreover, deciding if a WTG is almost-divergent is* PSPACE-*complete.*

To obtain these results on divergent and almost-divergent WTGs, we follow an computation schema that we now outline. First, we will always reason on the region game $\mathcal{R}(\mathcal{G})$ of the almost-divergent WTG $\mathcal{G}$. The goal is to compute an $\varepsilon$-approximation of $\mathsf{Val}_{\mathcal{R}(\mathcal{G})}((\ell_0, [\mathbf{0}]), \mathbf{0})$ for some initial location $\ell_0$. Techniques of [1] (that we will recall in Section 7) allow one to compute the (exact) values of a WTG played on a finite tree, using operator $\mathcal{F}$ of Section 2.6. The idea is thus to decompose as much as possible the game $\mathcal{R}(\mathcal{G})$ as a WTG over a tree. First, we decompose the region game into strongly connected components (SCCs, left of Figure 11): we must think about the final weight functions as the previously computed approximations of the values of SCCs coming after the current one in the topological order. We will keep as an invariant that final weight functions are piecewise affine with a finite number of pieces, and are continuous on each region.

For an SCC of $\mathcal{R}(\mathcal{G})$ and an initial state $(\ell_0, [\mathbf{0}])$ of $\mathcal{R}(\mathcal{G})$ provided by the SCC decomposition, we show that the game on the SCC is equivalent to a game on a tree built from a semi-unfolding (see middle of Figure 11) of $\mathcal{R}(\mathcal{G})$ from $(\ell_0, [\mathbf{0}])$ of finite depth, with certain nodes of the tree being *kernels* (parts of $\mathcal{R}(\mathcal{G})$ that contain all cycles of weight 0). The semi-unfolding is stopped either when reaching a final location, or when some location (or kernel) has been visited for a certain fixed number of times. Notice that, for divergent WTGs, there are no kernels, which simplifies the computation.

Then, we compute an approximation of $\mathsf{Val}_\mathcal{G}(\ell_0, \mathbf{0})$ with a bottom-up computation on the semi-unfolding. This computation is exact on nodes labelled by a single region state $s$, but approximated on kernel nodes $\mathsf{K}_s$. For the latter, we use the corner-point abstraction (right of Figure 11) over $1/N$-regions to compute values, and prove that, with an appropriately chosen $N$, this provides an $\varepsilon$-approximation of values.

This resolution of the value problem for divergent WTGs and approximation problem for almost-divergent WTGs heavily relies on the region abstraction, and requires one to construct $\mathcal{R}(\mathcal{G})$ entirely and compute its SCCs, before unfolding it partially in a tree-shaped structure. Our second result is a more symbolic approximation schema based on the value iteration algorithm only: the computations are not performed on the region abstraction, but instead use the cell partitions introduced in Section 2.7 that can cover several regions.

**Theorem 3.9.** *Let $\mathcal{G}$ be an almost-divergent WTG such that $\mathsf{Val}_\mathcal{G}(\ell, \nu) > -\infty$ for every configuration $(\ell, \nu)$. Then the sequence $(\mathsf{Val}_\mathcal{G}^k)_{k \geqslant 0}$ converges towards $\mathsf{Val}_\mathcal{G}$ and for every $\varepsilon \in \mathbb{Q}_{>0}$, we can compute an integer $P$ such that $\mathsf{Val}_\mathcal{G}^P$ is an $\varepsilon$-approximation of $\mathsf{Val}_\mathcal{G}$ for all configurations.*
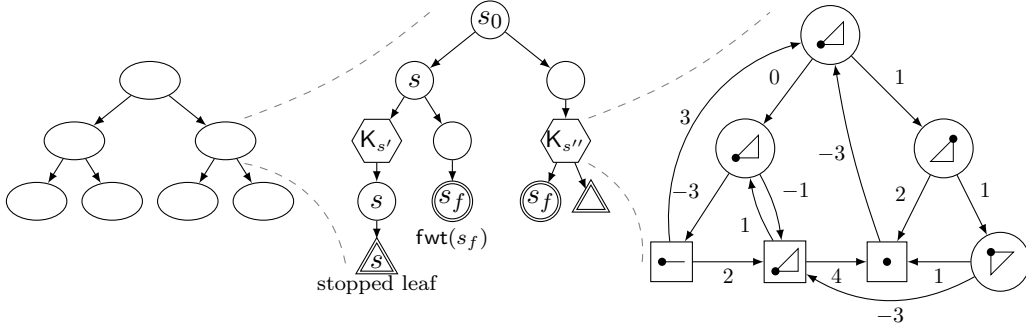
Figure 11: Static approximation schema: SCC decomposition of $\mathcal{R}(\mathcal{G})$, semi-unfolding of an SCC, corner-point abstraction for the kernels

Note that we have to control for configurations $(\ell, \nu)$ of value $-\infty$, where the non-increasing sequence $(\mathsf{Val}_{\mathcal{G}}^k(\ell, \nu))_{k \in \mathbb{N}}$ (that starts at $+\infty$) will diverge towards $-\infty$, but has no hope of approximating it. However, we will show that the configurations with value $-\infty$ can be computed pre-emptively:

**Proposition 3.10.** *The $-\infty$-value problem is* EXPTIME*-complete for almost-divergent WTGs.*

The exponential upper bound is obtained in Section 5. This contrasts with the general case (not necessarily almost-divergent), where the $-\infty$-value problem is undecidable, as detailed in Proposition 3.11.

3.1. **Hardness of value problems.** The EXPTIME-hardness results of Theorem 3.3 and Proposition 3.10 are reductions from the problem of solving timed games with reachability objectives [23].

To a reachability timed game $\mathcal{G}$, we simply add weights 1 on every edge and 0 on every location, making it a divergent WTG. We set the final weight of every target configuration at 0 (resp. $-\infty$). Then, Min wins the reachability timed game if and only if the value in the WTG is lower than threshold $\alpha = |\mathcal{R}(\mathcal{G})|$ (resp. equals $-\infty$). One direction of this statement is immediate by definition of having a value smaller than $+\infty$, and the other comes from the fact that reachability in the timed game implies reachability in the region game in less than $\alpha$ transitions, in turn implying that Min can ensure target reachability in the WTG with cumulative weight below $\alpha$, i.e. $\mathsf{Val}_{\mathcal{G}}(s, \nu) \leqslant \alpha + \mathsf{fwt}$, with $\mathsf{fwt} = 0$ (resp. $-\infty$).

**Proposition 3.11.** *Given a WTG $\mathcal{G}$ and an initial location $\ell_0$, the decision problem asking whether $\mathsf{Val}_{\mathcal{G}}(\ell_0, \mathbf{0}) = -\infty$ is undecidable.*

*Proof.* The proof goes via a reduction to the existence problem on turn-based WTG: given a WTG $\mathcal{G}$ (without final weight function), an integer threshold $\alpha$ and a starting location $\ell_0$, does there exist a strategy for Min that can guarantee reaching the unique target location $\ell_t$ from $\ell_0$ with weight $< \alpha$. In the non-negative setting, it is proven in [7] that the problem is undecidable for the comparison $\leqslant \alpha$. In the setting with arbitrary weights, formal proofs are given for all comparison signs in [16].

Consider $\mathcal{G}'$ the WTG built from $\mathcal{G}$ by adding a transition from $\ell_t$ to $\ell_0$, without guards and resetting all the clocks, of discrete weight $-\alpha$. We add a new target location $\ell_t'$, and add transitions of weight 0 from $\ell_t$ to $\ell_t'$. Locations $\ell_t$ and $\ell_t'$ belong to Min. Let us prove that

$\mathsf{Val}_{\mathcal{G}'}(\ell_0, \mathbf{0}) = -\infty$ if and only if $\mathsf{Min}$ has a strategy to guarantee a weight $< \alpha$ in $\mathcal{G}$. Assume first $\mathsf{Val}_{\mathcal{G}'}(\ell_0, \mathbf{0}) = -\infty$. If $\mathsf{Val}_{\mathcal{G}}(\ell_0, \mathbf{0}) = -\infty$, we are done. Otherwise, $\mathsf{Min}$ must follow in $\mathcal{G}'$ the new transition from $\ell_{\mathsf{t}}$ to $\ell_0$ to enforce a cycle of negative value, and thus enforce a play from $(\ell_0, \mathbf{0})$ to $\ell_{\mathsf{t}}$ with weight less than $\alpha$. Therefore, there exists a strategy for $\mathsf{Min}$ in $\mathcal{G}$ that can guarantee a weight $< \alpha$. Reciprocally, if there exists a strategy for $\mathsf{Min}$ that can guarantee a weight $< \alpha$, then $\mathsf{Min}$ can force a negative cycle play and $\mathsf{Val}_{\mathcal{G}'}(\ell_0, \mathbf{0}) = -\infty$. $\qquad\square$

## 4. Deciding divergence and almost-divergence

In this section, we will study properties that region cycles must satisfy in divergent or almost-divergent WTGs. This will give us a better understanding of the modelling power these classes confer, as well as enable us to provide procedures of optimal complexity to decide if a WTG fulfils the divergence or almost-divergence conditions.

4.1. **Cycles in an almost-divergent WTG.** Let us start with properties that hold for all almost-divergent weighted timed games $\mathcal{G}$. A region cycle $\mathsf{p}$ of $\mathcal{R}(\mathcal{G})$ is said to be a positive cycle (resp. a negative cycle, a 0-cycle) if every finite play $\rho$ following $\mathsf{p}$ satisfies $\mathsf{wt}(\rho) \geqslant 1$ (resp. $\mathsf{wt}(\rho) \leqslant -1$, $\mathsf{wt}(\rho) = 0$).

We start by showing that, in an almost-divergent game, all cycles $\mathsf{p} = t_1 \cdots t_n$ of $\mathcal{R}(\mathcal{G})$ (with $t_1, \ldots, t_n$ edges of $\mathcal{R}(\mathcal{G})$) are either 0-cycles, positive cycles or negative cycles[10], and we can classify a cycle by looking only at one of the corner plays following it:

**Lemma 4.1.** *Let $\mathcal{G}$ be an almost-divergent WTG. A region cycle $\mathsf{p}$ is a positive cycle (resp. a negative cycle, a 0-cycle) if and only if there exists a corner play $\overline{\rho}$ following $\mathsf{p}$ with $\mathsf{wt}_\Sigma(\overline{\rho}) > 0$ (resp. $\mathsf{wt}_\Sigma(\overline{\rho}) < 0$, $\mathsf{wt}_\Sigma(\overline{\rho}) = 0$). Moreover, every region cycle in $\mathcal{G}$ is either positive, negative, or a 0-cycle.*

*Proof.* If $\mathsf{p}$ is a positive cycle (resp. a negative cycle, a 0-cycle), every such corner play $\overline{\rho}$ will have weight above 0 (resp. under 0, equal to 0), by Lemma 2.10. Reciprocally, if such a corner play exists, all corner plays following $\mathsf{p}$ have weight above 0 (resp. under 0, equal to 0): otherwise the set $\{\mathsf{wt}_\Sigma(\rho) \mid \rho \text{ play following } \mathsf{p}\}$ would have non-empty intersection with the set $(-1, 0) \cup (0, 1)$ by Lemma 2.10, which would contradict that the game is almost-divergent.

Let $\mathsf{p}$ be a region cycle of $\mathcal{G}$. All the plays following $\mathsf{p}$ have a weight in $(-\infty, -1] \cup \{0\} \cup [1, +\infty)$. If it has a play in two different subintervals $(-\infty, -1]$, $\{0\}$, and $[1, +\infty)$, then Lemma 2.10 implies also that a play following $\mathsf{p}$ will have a weight in $(-1, 0) \cup (0, 1)$, which is forbidden by Definition 3.5. $\qquad\square$

An important result is that the sign of cycles is stable by rotation. This is not trivial because plays following a cycle can start and end in different valuations, therefore changing the starting region state of the cycle could *a priori* change the plays that follow it and the sign of their weights.

**Lemma 4.2.** *Let $\mathsf{p}$ and $\mathsf{p}'$ be region paths of an almost-divergent WTG. If $\mathsf{p}\mathsf{p}'$ is a positive cycle (resp. a negative cycle, a 0-cycle), then $\mathsf{p}'\mathsf{p}$ is a positive cycle (resp. a negative cycle, a 0-cycle).*

---

[10]In contrast, Definition 3.5 only requires that each play following a region cycle has weight in $(-\infty, -1] \cup \{0\} \cup [1, +\infty)$, without disallowing a region cycle to contain plays of different types.
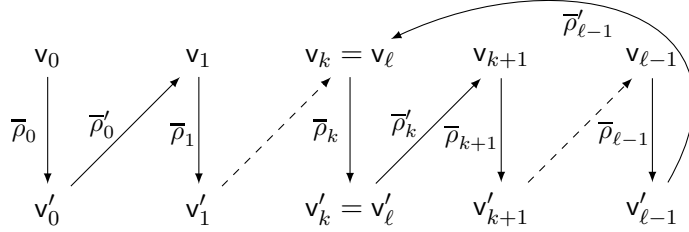
Figure 12: Proof scheme of Lemma 4.2. The top labels are corners of $\mathsf{first}(\mathsf{p})$, the bottom ones are corners of $\mathsf{first}(\mathsf{p}')$, and edges represent corner plays.

*Proof.* Since $\mathsf{p}_1 = \mathsf{p}\mathsf{p}'$ is a cycle, $\mathsf{first}(\mathsf{p}) = \mathsf{last}(\mathsf{p}')$ and $\mathsf{first}(\mathsf{p}') = \mathsf{last}(\mathsf{p})$, so $\mathsf{p}_2 = \mathsf{p}'\mathsf{p}$ is a cycle as well. First, since there are finitely many corners, by constructing a long enough play following an iterate of $\mathsf{p}'\mathsf{p}$, we can obtain a corner play that starts and ends in the same corner. Formally, we define two sequences of region corners $(\mathsf{v}_i \in \mathsf{first}(\mathsf{p}))_i$ and $(\mathsf{v}'_i \in \mathsf{first}(\mathsf{p}'))_i$. We start by choosing any $\mathsf{v}_0 \in \mathsf{first}(\mathsf{p})$. Let $\mathsf{v}'_0$ be a corner of $\mathsf{first}(\mathsf{p}')$ such that $\mathsf{v}'_0$ is accessible from $\mathsf{v}_0$ by following $\mathsf{p}$ with a corner play $\overline{\rho}_0$. For every $i > 0$, let $\mathsf{v}_i$ be a corner of $\mathsf{first}(\mathsf{p})$ such that $\mathsf{v}_i$ is accessible from $\mathsf{v}'_{i-1}$ by following $\mathsf{p}'$ with a corner play $\overline{\rho}'_i$, and let $\mathsf{v}'_i$ be a corner of $\mathsf{first}(\mathsf{p}')$ such that $\mathsf{v}'_i$ is accessible from $\mathsf{v}_i$ by following $\mathsf{p}$ with a corner play $\overline{\rho}_i$. We stop the construction at the first index $\ell$ such that there exists $k < \ell$ with $\mathsf{v}_\ell = \mathsf{v}_k$. Additionally, we let $\overline{\rho}_\ell = \overline{\rho}_k$. We know that this process never gets stuck—i.e. we can always find such corner plays iteratively—by Lemma 2.11, and it is bounded since $\mathsf{first}(\mathsf{p})$ has at most $|\mathcal{X}|+1$ corners.

For every $0 \leqslant i \leqslant \ell$, let $w_i$ be the weight of the corner play $\overline{\rho}_i$ from $\mathsf{v}_i$ to $\mathsf{v}'_i$ along $\mathsf{p}$, and let $w'_i$ be the weight of the corner play $\overline{\rho}'_i$ from $\mathsf{v}'_i$ to $\mathsf{v}_{i+1}$ along $\mathsf{p}'$. The concatenation of the two plays has weight $w_i + w'_i > 0$ (resp. $w_i + w'_i < 0$, $w_i + w'_i = 0$), since it follows the positive cycle (resp. negative cycle, 0-cycle) $\mathsf{p}_1$. For every $0 \leqslant i < \ell$, the concatenation of the corner play $\overline{\rho}'_i$ from $\mathsf{v}'_i$ to $\mathsf{v}_{i+1}$ with the corner play $\overline{\rho}_{i+1}$ from $\mathsf{v}_{i+1}$ to $\mathsf{v}'_{i+1}$ is a play from $\mathsf{v}'_i$ to $\mathsf{v}'_{i+1}$, of weight $w'_i + w_{i+1}$, following $\mathsf{p}_2$. Since $\mathsf{p}_2$ is a cycle, and the game is almost-divergent, all possible values of $w'_i + w_{i+1}$ have the same sign by Lemma 4.1.

Finally, we can construct a corner play from $\mathsf{v}'_k$ to $\mathsf{v}'_\ell$ by concatenating the plays $\overline{\rho}'_k, \overline{\rho}_{k+1}, \ldots, \overline{\rho}_{\ell-1}, \overline{\rho}'_{\ell-1}, \overline{\rho}_\ell$. We denote the weight of that play $W$, and

$$W = \sum_{i=k}^{\ell-1}(w'_i + w_{i+1}) = \sum_{i=k}^{\ell-1}(w_i + w'_i)$$

since $w_k = w_\ell$. As $w_i + w'_i > 0$ (resp. $w_i + w'_i < 0$, $w_i + w'_i = 0$) holds for every $i$, we obtain $W > 0$ (resp. $W < 0$, $W = 0$).

This implies that the terms $w'_i + w_{i+1}$, of constant sign, are all above 0 (resp. under 0, equal to 0). As a consequence, the concatenation of $\overline{\rho}'_k$ and $\overline{\rho}_{k+1}$ is a corner play following $\mathsf{p}_2$ of weight above 0 (resp. under 0, equal to 0). By Lemma 4.1, we conclude that $\mathsf{p}_2$ must be a positive cycle (resp. a negative cycle, a 0-cycle). $\qquad\square$

Therefore, region cycles in almost-divergent games are well-behaved: we can compose and rotate them while preserving their sign in the expected way.
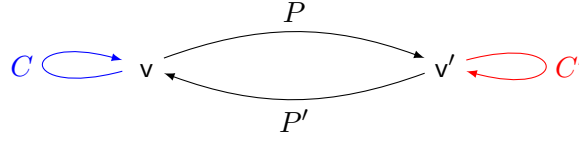
Figure 13: Proof scheme of Lemma 4.5, with paths and cycles around corners of the graph $\mathsf{FOG}(\mathsf{p}, \mathsf{p}')$.

4.2. **SCC-based characterisations.** After studying the properties of region cycles, we now focus on strongly connected components (SCCs) of the region abstraction $\mathcal{R}(\mathcal{G})$. An SCC $S$ of $\mathcal{R}(\mathcal{G})$ is said to be positive (resp. negative) if every cycle in $S$ is positive (resp. negative), i.e. if every play $\rho$ following a region cycle in $S$ satisfies $\mathsf{wt}_\Sigma(\rho) \geqslant 1$ (resp. $\mathsf{wt}_\Sigma(\rho) \leqslant -1$).

**Proposition 4.3.** *A weighted timed game $\mathcal{G}$ is divergent if and only if, each SCC of $\mathcal{R}(\mathcal{G})$ is either positive or negative.*

Likewise, an SCC $S$ of $\mathcal{R}(\mathcal{G})$ is said to be non-negative (resp. non-positive) if every region cycle in $S$ is either a positive cycle or a 0-cycle (resp. either a negative cycle or a 0-cycle), i.e. every play $\rho$ following a region cycle in $S$ satisfies either $\mathsf{wt}_\Sigma(\rho) \geqslant 1$ or $\mathsf{wt}_\Sigma(\rho) = 0$ (resp. either $\mathsf{wt}_\Sigma(\rho) \leqslant -1$ or $\mathsf{wt}_\Sigma(\rho) = 0$). We obtain:

**Proposition 4.4.** *A WTG $\mathcal{G}$ is almost-divergent if and only if each SCC of $\mathcal{R}(\mathcal{G})$ is either non-negative or non-positive.*

We now prove these two results. First, note that if $\mathcal{G}$ is divergent it has no 0-cycle, and Proposition 4.4 implies that each SCC of $\mathcal{R}(\mathcal{G})$ is either positive or negative. Conversely, if each SCC of $\mathcal{R}(\mathcal{G})$ is either positive or negative, Proposition 4.4 implies that $\mathcal{G}$ is divergent. Therefore, Proposition 4.3 is a corollary of Proposition 4.4. The rest of this section now proves Proposition 4.4.

To prove the reciprocal implication of Proposition 4.4, we only need to show that non-negative (resp. non-positive) SCCs of $\mathcal{R}(\mathcal{G})$ satisfy the definition of almost-divergent WTGs. By definition, they only contain executions $\rho$ following region cycles $\mathsf{p}$ such that $\mathsf{wt}_\Sigma(\rho) \in \{0\} \cup [1, +\infty)$ (resp. $\mathsf{wt}_\Sigma(\rho) \in (-\infty, -1] \cup \{0\}$). Then, assume $\mathsf{wt}_\Sigma(\rho) = 0$ and that $\mathsf{p}$ can be decomposed into smaller cycles $\mathsf{p}'$ and $\mathsf{p}''$. Definition 3.5 requires us to show that all plays $\rho'$ and $\rho''$ following $\mathsf{p}'$ and $\mathsf{p}''$, respectively, are such that $\mathsf{wt}_\Sigma(\rho') = \mathsf{wt}_\Sigma(\rho'') = 0$, i.e. $\mathsf{p}'$ and $\mathsf{p}''$ are 0-cycles. Note that by Lemma 4.2, $\mathsf{p}'\mathsf{p}''$ is a 0-cycle. As $\mathsf{p}'$ and $\mathsf{p}''$ are contained in the same SCC, they are either both non-negative cycles or both non-positive cycles. Let $\rho'\rho''$ be a play following $\mathsf{p}'\mathsf{p}''$, so that $\rho'$ follows $\mathsf{p}'$ and $\rho''$ follows $\mathsf{p}''$. Then, $\mathsf{wt}_\Sigma(\rho'\rho'') = \mathsf{wt}_\Sigma(\rho') + \mathsf{wt}_\Sigma(\rho'') = 0$, it follows that $\mathsf{wt}_\Sigma(\rho') = \mathsf{wt}_\Sigma(\rho'') = 0$, i.e. $\mathsf{p}'$ and $\mathsf{p}''$ are 0-cycles.

For the direct implication, the situation is more complex: we need to be careful while composing cycles with each other. To help us, we rely on the folded orbit graphs of region cycles. Suppose that $\mathcal{G}$ is almost-divergent, and consider two cycles $\mathsf{p}$ and $\mathsf{p}'$ in the same SCC of $\mathcal{R}(\mathcal{G})$. We need to show that they are both either non-positive or non-negative. Lemma 4.5 will first take care of the case where $\mathsf{p}$ and $\mathsf{p}'$ share a region state $(\ell, r)$.

**Lemma 4.5.** *If $\mathcal{G}$ is almost-divergent and two cycles $\mathsf{p}$ and $\mathsf{p}'$ of $\mathcal{R}(\mathcal{G})$ share a region state $(\ell, r)$, they are either both non-negative or both non-positive.*

*Proof.* Suppose by contradiction that $\mathsf{p}$ is negative and $\mathsf{p}'$ is positive. We assume that $(\ell, r)$ is the first region state of both $\mathsf{p}$ and $\mathsf{p}'$, possibly performing a rotation of the cycles if necessary (in particular this preserves their sign by Lemma 4.2). We construct a graph $\mathsf{FOG}(\mathsf{p}, \mathsf{p}')$ as the union of $\mathsf{FOG}(\mathsf{p})$ and $\mathsf{FOG}(\mathsf{p}')$ (that share the same set of vertices), colouring in blue the edges of $\mathsf{FOG}(\mathsf{p})$ and in red the edges of $\mathsf{FOG}(\mathsf{p}')$. A path in $\mathsf{FOG}(\mathsf{p}, \mathsf{p}')$ is said blue (resp. red) when all of its edges are blue (resp. red).

Since $\mathsf{FOG}(\mathsf{p})$ and $\mathsf{FOG}(\mathsf{p}')$ are finite graphs with no deadlocks (every corner has an outgoing edge by Lemma 2.11), from every corner of $\mathsf{FOG}(\mathsf{p}, \mathsf{p}')$, we can reach a blue simple cycle, as well as a red simple cycle. Since there are only a finite number of simple cycles in $\mathsf{FOG}(\mathsf{p}, \mathsf{p}')$, there exists a blue cycle $C$ and a red cycle $C'$ that can reach each other in $\mathsf{FOG}(\mathsf{p}, \mathsf{p}')$. Denote by $\mathsf{v}$ and $\mathsf{v}'$ the first corners of cycles $C$ and $C'$, respectively.

We assume first $\mathsf{v} = \mathsf{v}'$. Let $k$ and $k'$ be the respective lengths of $C$ and $C'$, so that $C$ can be decomposed as $\mathsf{v} \xrightarrow{\overline{\rho}_1} \cdots \xrightarrow{\overline{\rho}_k} \mathsf{v}$ and $C'$ as $\mathsf{v} \xrightarrow{\overline{\rho}'_1} \cdots \xrightarrow{\overline{\rho}'_{k'}} \mathsf{v}$, where $\overline{\rho}_i$ are corner plays following $\mathsf{p}$ and $\overline{\rho}'_i$ are corner plays following $\mathsf{p}'$. Let $\overline{\rho}$ be the concatenation of $\overline{\rho}_1, \ldots, \overline{\rho}_k$, and $\overline{\rho}'$ be the concatenation of $\overline{\rho}'_1, \ldots, \overline{\rho}'_{k'}$. Recall that $w = |\mathsf{wt}_\Sigma(\overline{\rho})|$ and $w' = |\mathsf{wt}_\Sigma(\overline{\rho}')|$ are integers. Since $\mathsf{p}$ is negative, so is $\mathsf{p}^k$, the concatenation of $k$ copies of $\mathsf{p}$ (the weight of a play following it is a sum of weights all below $-1$). Therefore, $\overline{\rho}$, that follows $\mathsf{p}^k$, has a weight $\mathsf{wt}_\Sigma(\overline{\rho}) \leqslant -1$ by Lemma 2.10. Similarly, $\mathsf{wt}_\Sigma(\overline{\rho}') \geqslant 1$. Let $\overline{\rho}''$ be the play obtained by concatenating $w'$ copies of $\overline{\rho}$ and $w$ copies of $\overline{\rho}'$. Then, $\mathsf{wt}_\Sigma(\overline{\rho}'') = \mathsf{wt}_\Sigma(\overline{\rho})w' + \mathsf{wt}_\Sigma(\overline{\rho}')w = 0$, and therefore the region cycle $\mathsf{p}''$ composed of $w'$ copies of $\mathsf{p}^k$ and $w$ copies of $\mathsf{p}'^{k'}$ is a 0-cycle. This contradicts the almost-divergence of $\mathcal{G}$, since $\mathsf{p}''$ can be decomposed into smaller cycles that are not 0-cycles.

Therefore, $\mathsf{v}$ and $\mathsf{v}'$ are different, but can reach each other in $\mathsf{FOG}(\mathsf{p}, \mathsf{p}')$. Let $P$ be a path from $\mathsf{v}$ to $\mathsf{v}'$, and $P'$ be a path from $\mathsf{v}'$ to $\mathsf{v}$. The situation is depicted in Figure 13. Consider the cycle $C''$ obtained by concatenating $P$ and $P'$. As a cycle of $\mathsf{FOG}(\mathsf{p}, \mathsf{p}')$, we can map it to a cycle $\mathsf{p}''$ of $\mathcal{R}(\mathcal{G})$ (alternating $\mathsf{p}$ and $\mathsf{p}'$ depending on the colours of the traversed edges), so that $C''$ is a cycle (of length 1) of $\mathsf{FOG}(\mathsf{p}'')$. As $\mathcal{G}$ is almost-divergent, $\mathsf{p}''$ is either positive, negative or a 0-cycle. Moreover, $\mathsf{p}''$ cannot be a 0-cycle as it can be decomposed into smaller cycles that are not 0-cycles. Suppose for instance that it is positive. Since $(\ell, r)$ is the first region state of both $\mathsf{p}$ and $\mathsf{p}''$, we can construct the $\mathsf{FOG}(\mathsf{p}, \mathsf{p}'')$, in which $C$ is a blue cycle and $C''$ is a red cycle, both sharing the same first vertex. We then conclude with the previous case. A similar reasoning with $\mathsf{p}'$ applies to the case that $\mathsf{p}''$ is negative. Therefore, in all cases, we reach a contradiction. $\square$

To finish the proof of the direct implication of Proposition 4.4, we suppose that the two cycles $\mathsf{p}$ and $\mathsf{p}'$, one positive and the other negative, in the same SCC of $\mathcal{R}(\mathcal{G})$, do not share any region states. By strong connectivity, in $\mathcal{R}(\mathcal{G})$, there exists a path $\mathsf{p}_1$ from the first state of $\mathsf{p}$ to the first state of $\mathsf{p}'$, and a path $\mathsf{p}_2$ from the first state of $\mathsf{p}'$ to the first state of $\mathsf{p}$. Consider the cycle $\mathsf{p}''$ of $\mathcal{R}(\mathcal{G})$ defined by $\mathsf{p}\mathsf{p}_1\mathsf{p}'\mathsf{p}_2$. By the almost-divergence of $\mathcal{G}$, $\mathsf{p}''$ must be either positive, negative or a 0-cycle. Since it shares a state with both $\mathsf{p}$ and $\mathsf{p}'$, Lemma 4.5 allows us to prove a contradiction in both positive and negative cases, and therefore $\mathsf{p}''$ must be a 0-cycle. This contradicts the hypothesis as one of the decompositions of $\mathsf{p}''$ into smaller cycles produces $\mathsf{p}$ and $\mathsf{p}_1\mathsf{p}'\mathsf{p}_2$, with $\mathsf{p}$ a non-0-cycle. This concludes the proof of Proposition 4.4.

*Remark.* These characterisations of divergent or almost-divergent WTGs in term of SCCs provide an intuitive understanding of the modelling power these classes hold. For divergence,

the model should have a global structure (the SCC decomposition) linking modules in an acyclic fashion. For each module, we have to choose between a positive dynamic, where weights always eventually increase, and a negative dynamic, where weights always eventually decrease. For almost-divergence, the modules may also have portions that are (eventually) neutral with regard to weight accumulation. In both classes, arbitrarily small weights should not be allowed to accumulate.

4.3. **Deciding membership.** We study the *membership problem* for divergent (resp. almost-divergent) WTGs, i.e. the decision problem that asks if a given WTG is divergent (resp. almost-divergent). As mentioned in Theorems 3.3 and 3.8, we show that it is PSPACE-complete for both of these classes.

Relying on the previous characterisation of Propositions 4.3 and 4.4, the algorithms will consist in only considering region cycles of length bounded by the number of corners in the corner-point abstraction $\Gamma(\mathcal{G})$. For divergent WTGs, this will be correct by using the following result:

**Lemma 4.6.** *Let $\mathcal{G}$ be a WTG. An SCC $S$ of $\mathcal{R}(\mathcal{G})$ is positive (resp. negative) if and only if every region cycle in $S$, of length at most $|\Gamma(\mathcal{G})|$, is positive (resp. negative).*

*Proof.* The direct implication holds by definition. Reciprocally, let us assume that every cycle in $S$ of length at most $|\Gamma(\mathcal{G})|$ is positive (resp. negative), and prove that every cycle $\mathsf{p}$ in $S$ is positive (resp. negative), by induction on the length of $\mathsf{p}$. If $\mathsf{p}$ has length above $|\Gamma(\mathcal{G})|$, every corner play $\overline{\rho}$ following $\mathsf{p}$ can be split as $\overline{\rho} = \overline{\rho}_1\overline{\rho}_2\overline{\rho}_3$, with $\overline{\rho}_2$ a corner play that starts and ends in the same corner. Then we can write $\mathsf{p} = \mathsf{p}_1\mathsf{p}_2\mathsf{p}_3$, with $\overline{\rho}_1$ (resp. $\overline{\rho}_2$, $\overline{\rho}_3$) following $\mathsf{p}_1$ (resp. $\mathsf{p}_2$, $\mathsf{p}_3$). Observe that $\mathsf{p}_2$ and $\mathsf{p}_1\mathsf{p}_3$ are region cycles of $S$, both positive (resp. negative) by induction. It follows that $\mathsf{wt}_\Sigma(\overline{\rho}_2) \geqslant 1$ (resp. $\mathsf{wt}_\Sigma(\overline{\rho}_2) \leqslant -1$), and $\mathsf{wt}_\Sigma(\overline{\rho}_1\overline{\rho}_3) \geqslant 1$ (resp. $\mathsf{wt}_\Sigma(\overline{\rho}_1\overline{\rho}_3) \leqslant -1$), as $\overline{\rho}_1\overline{\rho}_3$ is a valid corner play following $\mathsf{p}_1\mathsf{p}_3$. We can therefore conclude that $\mathsf{wt}_\Sigma(\overline{\rho}) \geqslant 1$ (resp. $\mathsf{wt}_\Sigma(\overline{\rho}) \leqslant -1$). This holds for all corner plays following $\mathsf{p}$, and by Lemma 2.10 $\mathsf{p}$ is positive (resp. negative). $\qquad\square$

Then, to decide if a game is *not divergent*, using Proposition 4.3, it suffices to search for an SCC of the region automaton containing a cycle such that there exists a corner play following it of non-negative weight, and a cycle such that there exists a corner play following it of non-positive weight, both of length bounded by $B = |\Gamma(\mathcal{G})| \leqslant |L| \times |\mathsf{Reg}(\mathcal{X}, M)| \times (|\mathcal{X}| + 1)$. We can test this condition in NPSPACE: we guess a starting region for each cycle, use standard reachability analysis [2] to check that they are in the same SCC of $\mathcal{R}(\mathcal{G})$ (in PSPACE), and use the following result with comparison $\geqslant 0$ and $\leqslant 0$, respectively, to check the sign of each cycle.

**Lemma 4.7.** *Consider a WTG $\mathcal{G}$, a region state $(\ell, r)$ of $\mathcal{R}(\mathcal{G})$, a bound $B \in \mathbb{N}$, and a comparison operator $\bowtie \in \{<, >, \leqslant, \geqslant, =, \neq\}$. Deciding if there exists a corner play $\overline{\rho}$ following a cycle $\mathsf{p}$ of $\mathcal{R}(\mathcal{G})$ starting from $(\ell, r)$, such that $|\mathsf{p}| \leqslant B$ and $\mathsf{wt}_\Sigma(\overline{\rho}) \bowtie 0$, is in PSPACE (i.e. it can be done using space polynomial in $|\mathcal{G}|$ and $\log(B)$).*

*Proof.* We guess a starting corner $\mathsf{v}$ of $r$ for $\overline{\rho}$, and we guess on-the-fly the transitions of $\mathsf{p}$ and $\overline{\rho}$, i.e. a sequences of regions with one of their corners, keeping in memory the cumulative weight of $\overline{\rho}$ and the length $|\mathsf{p}|$. At every step, we check that $|\mathsf{p}| \leqslant B$ in space polynomial in

$\log(B)$ and $\log(|\mathsf{p}|) \leqslant \log(|\mathcal{R}(\mathcal{G})|)$, with $\log(|\mathcal{R}(\mathcal{G})|)$ polynomial in $|\mathcal{G}|$.[11] Similarly, we can check that $\overline{\rho}$ is following $\mathsf{p}$ in polynomial space. At some point, we guess that the cycle is complete, and we check that the current region state equals $(\ell, r)$. Finally, we check that $\mathsf{wt}_\Sigma(\overline{\rho}) \bowtie 0$ in space polynomial in $\log(\mathsf{wt}_\Sigma(\overline{\rho}))$. Note that $\mathsf{wt}_\Sigma(\overline{\rho})$ is an integer bounded (in absolute value) by $B \times w_{\max}$, and can thus be stored in polynomial space. This shows that the problem is in NPSPACE, and thus in PSPACE using Savitch's theorem [30].      $\square$

Since the bound $B$ is at most exponential in $|\mathcal{G}|$, this check can be performed in PSPACE. This shows that the membership problem for divergent weighted timed games is in coNPSPACE = coPSPACE = PSPACE by the theorems of Immerman-Szelepcsényi [22, 31] and Savitch [30].

Let us now show the PSPACE-hardness (indeed the coPSPACE, which is identical) by a reduction from the reachability problem in a timed automaton. Consider a timed automaton with a starting location and a different target location without outgoing edges. We construct from it a weighted timed game by distributing all locations to Min, and equipping all edges with weight 1, and all locations with weight 0. We also add a loop with weight $-1$ on the target, and an edge from the target location to the initial location with weight 0, both with guard $\top$ and resetting all clocks. Then, the WTG is not divergent if and only if the target can be reached from the initial location in the timed automaton.

For almost-divergent WTGs, the length of the required region cycles is bigger, because of the possible presence of 0-cycles.

**Lemma 4.8.** *Let $\mathcal{G}$ be a WTG. An SCC $S$ of $\mathcal{R}(\mathcal{G})$ is non-negative (resp. non-positive) if and only if every region cycle in $S$, of length at most $|\Gamma(\mathcal{G})|^2$, is either a positive cycle or a 0-cycle (resp. either a negative cycle or a 0-cycle).*

*Proof.* The direct implication holds by definition. Reciprocally, suppose that every cycle in $S$ of length at most $|\Gamma(\mathcal{G})|^2$ is either a positive cycle or a 0-cycle (resp. either a negative cycle or a 0-cycle). Let us prove that every cycle $\mathsf{p}$ in $S$ is either a positive cycle or a 0-cycle (resp. either a negative cycle or a 0-cycle), by induction on the length of $\mathsf{p}$. Consider a region cycle $\mathsf{p}$ with length above $|\Gamma(\mathcal{G})|^2$. Let us show that for all corner plays $\overline{\rho}$, $\overline{\rho}'$ following $\mathsf{p}$, either $\mathsf{wt}_\Sigma(\overline{\rho}) = \mathsf{wt}_\Sigma(\overline{\rho}') = 0$, or both $\mathsf{wt}_\Sigma(\overline{\rho}) \geqslant 1$ and $\mathsf{wt}_\Sigma(\overline{\rho}') \geqslant 1$ (resp. both $\mathsf{wt}_\Sigma(\overline{\rho}) \leqslant -1$ and $\mathsf{wt}_\Sigma(\overline{\rho}') \leqslant -1$) hold. This will allow us to conclude by Lemma 2.10.

From a pair of corner plays $\overline{\rho}$ and $\overline{\rho}'$ following $\mathsf{p}$, we can extract a sequence of pairs of corners states $((\ell_i, r_i, \mathsf{v}_i), (\ell_i, r_i, \mathsf{v}'_i))$, such that $(\ell_i, r_i)$ is the $i$-th region state of $\mathsf{p}$, and $\mathsf{v}_i$ (resp. $\mathsf{v}'_i$) is the $i$-th corner of $\overline{\rho}$ (resp. $\overline{\rho}'$). Since $|\mathsf{p}| > |\Gamma(\mathcal{G})|^2$, there must exist two indexes, $j$ and $k$, such that $j < k$, $(\ell_j, r_j) = (\ell_k, r_k)$ and $(\mathsf{v}_j, \mathsf{v}'_j) = (\mathsf{v}_k, \mathsf{v}'_k)$. In other words, we can write $\mathsf{p} = \mathsf{p}_1\mathsf{p}_2\mathsf{p}_3$, with $\mathsf{p}_2$ and $\mathsf{p}_1\mathsf{p}_3$ region cycles of $S$, and $\overline{\rho}$, $\overline{\rho}'$ can be split as $\overline{\rho} = \overline{\rho}_1\overline{\rho}_2\overline{\rho}_3$, $\overline{\rho}' = \overline{\rho}'_1\overline{\rho}'_2\overline{\rho}'_3$, with $\overline{\rho}_l$ (resp. $\overline{\rho}'_l$) following $\mathsf{p}_l$ (resp. $\mathsf{p}'_l$) for $l \in \{1, 2, 3\}$, such that $\overline{\rho}_2$ and $\overline{\rho}'_2$ are corner cycles, i.e. $\mathsf{first}(\overline{\rho}_2) = \mathsf{last}(\overline{\rho}_2)$ and $\mathsf{first}(\overline{\rho}'_2) = \mathsf{last}(\overline{\rho}'_2)$. Then, by induction either $\mathsf{wt}_\Sigma(\overline{\rho}_2) = \mathsf{wt}_\Sigma(\overline{\rho}'_2) = 0$, or both $\mathsf{wt}_\Sigma(\overline{\rho}_2) \geqslant 1$ and $\mathsf{wt}_\Sigma(\overline{\rho}'_2) \geqslant 1$ (resp. both $\mathsf{wt}_\Sigma(\overline{\rho}_2) \leqslant -1$ and $\mathsf{wt}_\Sigma(\overline{\rho}'_2) \leqslant -1$) hold. The same property holds for $\overline{\rho}_1\overline{\rho}_3$ and $\overline{\rho}'_1\overline{\rho}'_3$, both valid corner plays following $\mathsf{p}_1\mathsf{p}_3$. It follows that either $\mathsf{wt}_\Sigma(\overline{\rho}) = \mathsf{wt}_\Sigma(\overline{\rho}') = 0$, or both $\mathsf{wt}_\Sigma(\overline{\rho}) \geqslant 1$ and $\mathsf{wt}_\Sigma(\overline{\rho}') \geqslant 1$ (resp. both $\mathsf{wt}_\Sigma(\overline{\rho}) \leqslant -1$ and $\mathsf{wt}_\Sigma(\overline{\rho}') \leqslant -1$) hold.      $\square$

Then, to decide if a game is *not almost-divergent*, we distinguish two cases:

_____

[11]The global clock bound $M$ is at most exponential in the size of $\mathcal{G}$, and $|\mathcal{R}(\mathcal{G})|$ is at most exponential in $|\mathcal{X}|$ but polynomial in $M$, therefore $|\mathcal{R}(\mathcal{G})|$ is at most exponential in $|\mathcal{G}|$.

- There exists a region cycle, of length at most $B = |\Gamma(\mathcal{G})|^2$, and two corner plays $\overline{\rho}$ and $\overline{\rho}'$, both following p, such that $\mathsf{wt}_\Sigma(\overline{\rho}) = 0$ and $\mathsf{wt}_\Sigma(\overline{\rho}') \neq 0$.
- An SCC of the region automaton contains a cycle such that there exists a corner play following it of negative weight, and a cycle such that there exists a corner play following it of positive weight, both of length bounded by $B = |\Gamma(\mathcal{G})|^2$.

We can test both conditions in NPSPACE, by guessing the starting regions of these cycles and using respectively Lemma 4.7 (for the second condition) and the following result (for the first condition):

**Lemma 4.9.** *Consider a weighted timed game $\mathcal{G}$, a region state $(\ell, r)$ of $\mathcal{R}(\mathcal{G})$, a bound $B \in \mathbb{N}$, and comparison operators $\bowtie, \bowtie' \in \{<, >, \leqslant, \geqslant, =, \neq\}$. Deciding if there exists a cycle p of $\mathcal{R}(\mathcal{G})$ starting from $(\ell, r)$, and two corner plays $\overline{\rho}$ and $\overline{\rho}'$, both following p, such that $|\mathsf{p}| \leqslant B$, $\mathsf{wt}_\Sigma(\overline{\rho}) \bowtie 0$ and $\mathsf{wt}_\Sigma(\overline{\rho}') \bowtie' 0$, is in PSPACE.*

*Proof.* We follow the same non-deterministic procedure as Lemma 4.7, but this time we guess two corner plays on-the-fly instead of one. $\qquad\square$

This shows that the membership problem for divergent weighted timed games is in coNPSPACE = coPSPACE = PSPACE [22, 31, 30].

Let us now show the PSPACE-hardness (indeed the coPSPACE-hardness) by a reduction from the reachability problem in a timed automaton, similar to the one we used for the PSPACE-hardness of deciding divergence. We consider a timed automaton with a starting location and a different target location without outgoing edges. We construct from it a weighted timed game by distributing all locations to Min, and equipping all edges with weight 0, and all locations with weight 0. We also add a loop with weight 1 on the initial location, one with weight $-1$ on the target location, and an edge from the target location to the initial location with weight 0, all three with guard $\top$ and resetting all clocks. Then, the weighted timed game is not almost-divergent if and only if the target can be reached from the initial location in the timed automaton.

## 5. DECIDING INFINITE VALUES

There are three reasons for an infinite value to appear in a WTG:

- an infinite value ($+\infty$ or $-\infty$) could appear in a final weight function and be propagated along a play;
- a value $+\infty$ could be obtained if Min is not able to reach a target location;
- a value $-\infty$ could be obtained if Min is able to reach a negative cycle, loop arbitrarily many times inside, and then reach a target location.

This section explains how to detect the three situations in the region game $\mathcal{R}(\mathcal{G})$ of an almost-divergent WTG $\mathcal{G}$. Before that, let us start by formalising a way to safely remove region states of $\mathcal{R}(\mathcal{G})$ for which the value of all their associated configurations is known to be infinite. Let $S^{+\infty}$ be a subset of $S = L \times \mathsf{Reg}(\mathcal{X}, M)$, such that $\mathsf{Val}_{\mathcal{R}(\mathcal{G})}((\ell, r), \nu) = +\infty$ for all $(\ell, r) \in S^{+\infty}$ and $\nu \in r$. If a configuration of $\mathcal{G}$ is in the attractor of Max towards $S^{+\infty}$, then Max has a strategy giving it value $+\infty$. Moreover, attractor being computable over regions all the configurations in the same region have a value $+\infty$: therefore we could add this region to $S^{+\infty}$. We will thus assume that $S^{+\infty}$ is *closed by attractor* for Max, i.e. the attractor of Max towards $S^{+\infty}$ equals $S^{+\infty}$. We can define the same notion for a set $S^{-\infty}$ of states with value $-\infty$, that can be assumed closed by attractor of Min.

**Lemma 5.1.** *In $\mathcal{R}(\mathcal{G})$, let $S^{+\infty}$ be a set of region states of value $+\infty$ closed by attractor of Max, and let $S^{-\infty}$ be a set of region states of value $-\infty$ closed by attractor of Min. Removing the region states $S^{+\infty} \cup S^{-\infty}$ from $\mathcal{R}(\mathcal{G})$ will not change any other value.*

*Proof.* We only prove the result for the removal of region states from $S^{-\infty}$, since the proof for $S^{+\infty}$ is entirely symmetrical. Let $S' = S'_{\mathsf{Min}} \uplus S'_{\mathsf{Max}}$ denote $S \backslash S^{-\infty}$, and let $\mathcal{R}(\mathcal{G})'$ denote the restriction of $\mathcal{R}(\mathcal{G})$ to $S'$. Let us show that for all region states $(\ell, r) \in S'$ and $\nu \in r$, $\mathsf{Val}_{\mathcal{R}(\mathcal{G})}((\ell, r), \nu) = \mathsf{Val}_{\mathcal{R}(\mathcal{G})'}((\ell, r), \nu))$.

If $\sigma_{\mathsf{Min}}$ is a strategy of Min in $\mathcal{R}(\mathcal{G})$, and $\sigma'_{\mathsf{Min}}$ is a strategy of Min in $\mathcal{R}(\mathcal{G})'$, such that for all plays $\rho$ in $\mathsf{FPlays}^{\mathsf{Min}}_{\mathcal{R}(\mathcal{G})'}$, $\sigma_{\mathsf{Min}}(\rho) = \sigma'_{\mathsf{Min}}(\rho)$, we say that $\sigma_{\mathsf{Min}}$ coincides with $\sigma'_{\mathsf{Min}}$. We define the same notion for strategies of Max. All strategies of Min and Max in $\mathcal{R}(\mathcal{G})'$ can be extended arbitrarily to become strategies in $\mathcal{R}(\mathcal{G})$ that coincide, by making the same choices on plays that stay in $\mathcal{R}(\mathcal{G})'$, and making arbitrary choices otherwise. It follows that if $\sigma_{\mathsf{Min}}$ and $\sigma'_{\mathsf{Min}}$ are strategies of Min that coincide, then for all strategies $\sigma'_{\mathsf{Max}}$ of Max in $\mathcal{R}(\mathcal{G})'$, there exists a corresponding strategy $\sigma_{\mathsf{Max}}$ in $\mathcal{R}(\mathcal{G})$, such that $\mathsf{play}_{\mathcal{R}(\mathcal{G})'}(((\ell, r), \nu), \sigma'_{\mathsf{Min}}, \sigma'_{\mathsf{Max}}) = \mathsf{play}_{\mathcal{R}(\mathcal{G})}(((\ell, r), \nu), \sigma_{\mathsf{Min}}, \sigma_{\mathsf{Max}})$, and therefore

$$\mathsf{Val}_{\mathcal{R}(\mathcal{G})'}(((\ell, r), \nu), \sigma'_{\mathsf{Min}}) \leqslant \mathsf{Val}_{\mathcal{R}(\mathcal{G})}(((\ell, r), \nu), \sigma_{\mathsf{Min}})$$

as the supremum over strategies of Max in $\mathcal{R}(\mathcal{G})$ is at least equal to the one in $\mathcal{R}(\mathcal{G})'$. Similarly, if $\sigma_{\mathsf{Max}}$ and $\sigma'_{\mathsf{Max}}$ are strategies of Max that coincide, then

$$\mathsf{Val}_{\mathcal{R}(\mathcal{G})'}(((\ell, r), \nu), \sigma'_{\mathsf{Max}}) \geqslant \mathsf{Val}_{\mathcal{R}(\mathcal{G})}(((\ell, r), \nu), \sigma_{\mathsf{Max}})$$

By closure by attractor of Min, all transitions starting in a region state of $S'_{\mathsf{Min}}$ must end in a region state of $S'$ (otherwise the first state would belong to the attractor of Min towards $S^{-\infty}$). Therefore, every strategy of Min in $\mathcal{R}(\mathcal{G})$ has a corresponding strategy in $\mathcal{R}(\mathcal{G})'$ that makes the same choices. In particular, if we consider an $\varepsilon$-optimal strategy $\sigma_{\mathsf{Min}}$ of Min in $\mathcal{R}(\mathcal{G})$, and its corresponding strategy $\sigma'_{\mathsf{Min}}$ in $\mathcal{R}(\mathcal{G})'$, it holds that $\mathsf{Val}_{\mathcal{R}(\mathcal{G})'}((\ell, r), \nu) \leqslant \mathsf{Val}_{\mathcal{R}(\mathcal{G})'}(((\ell, r), \nu), \sigma'_{\mathsf{Min}}) \leqslant \mathsf{Val}_{\mathcal{R}(\mathcal{G})}(((\ell, r), \nu), \sigma_{\mathsf{Min}})$ by definition of Val as an infimum over strategies. Finally, $\mathsf{Val}_{\mathcal{R}(\mathcal{G})}(((\ell, r), \nu), \sigma_{\mathsf{Min}}) \leqslant \mathsf{Val}_{\mathcal{R}(\mathcal{G})}((\ell, r), \nu) + \varepsilon$ by $\varepsilon$-optimality of $\sigma_{\mathsf{Min}}$, and thus $\mathsf{Val}_{\mathcal{R}(\mathcal{G})'}((\ell, r), \nu) \leqslant \mathsf{Val}_{\mathcal{R}(\mathcal{G})}((\ell, r), \nu) + \varepsilon$. Since this holds for all $\varepsilon > 0$, we get that $\mathsf{Val}_{\mathcal{R}(\mathcal{G})'}((\ell, r), \nu) \leqslant \mathsf{Val}_{\mathcal{R}(\mathcal{G})}((\ell, r), \nu)$.

From every state $s$ of $S'_{\mathsf{Max}}$, a positional strategy that chooses a transition jumping into $S^{-\infty}$ must have value $-\infty$. Any other choice would thus be equal or better, and by closure by attractor of Min, there must exist a transition starting in $(\ell, r)$ that ends in $S'$. Therefore, there exists an $\varepsilon$-optimal strategy $\sigma_{\mathsf{Max}}$ of Max in $\mathcal{R}(\mathcal{G})$ with a corresponding strategy $\sigma'_{\mathsf{Max}}$ in $\mathcal{R}(\mathcal{G})'$. Then, it holds that $\mathsf{Val}_{\mathcal{R}(\mathcal{G})'}((\ell, r), \nu) \geqslant \mathsf{Val}_{\mathcal{R}(\mathcal{G})'}(((\ell, r), \nu), \sigma'_{\mathsf{Max}}) \geqslant \mathsf{Val}_{\mathcal{R}(\mathcal{G})}(((\ell, r), \nu), \sigma_{\mathsf{Max}}) \geqslant \mathsf{Val}_{\mathcal{R}(\mathcal{G})}((\ell, r), \nu) - \varepsilon$. We conclude since this sequence holds for all $\varepsilon > 0$.  $\square$

5.1. **Infinite final values.** As a first step, we explain how to compute and remove all region states with value $+\infty$, and region states with value $-\infty$ because of final weights. The only states with infinite value that will remain are some states that derive a value of $-\infty$ from arbitrary accumulation of negative weight, and we will deal with them later.

Recall that the final weight function $\mathsf{fwt}$ has been supposed piecewise affine with a finite number of pieces and is continuous on each region. In particular, final weights $+\infty$ or $-\infty$ are given to entire regions. Then, let $S^{-\infty}_{\mathsf{t}} \subseteq L_{\mathsf{t}} \times \mathsf{Reg}(\mathcal{X}, M)$ (resp. $S^{+\infty}_{\mathsf{t}}$) denote the set of target region states that $\mathsf{fwt}$ maps to $-\infty$ (resp. $+\infty$).

**Proposition 5.2.** *If a region state of $\mathcal{G}$ is in the attractor of* Min *towards $S_t^{-\infty}$ (resp. in the attractor of* Max *towards $S_t^{+\infty}$), then it has value $-\infty$ (resp. $+\infty$). Moreover, if we remove those states from $\mathcal{R}(\mathcal{G})$, the value of the other configurations does not change.*

*Proof.* If a region state $(\ell, r)$ is in the attractor of Min towards $S_t^{-\infty}$, then clearly Min has a (positional) strategy giving value $-\infty$ from $(\ell, r)$, and $\mathsf{Val}_{\mathcal{R}(\mathcal{G})}((\ell, r), \nu) = -\infty$ for all $\nu \in r$. An attractor of Min is always closed by attractor of Min, so we can apply Lemma 5.1 and conclude. Once again, a symmetrical proof lets us deal with the attractor of Max towards $S_t^{+\infty}$. $\qquad\square$

Then, assuming that all final weights are finite, configurations with value $+\infty$ are those from which Min cannot reach the target region states: thus, they can also be computed and removed using the attractor algorithm.

**Proposition 5.3.** *If* fwt *maps all configurations to values in $\mathbb{R}$, then a configuration $((\ell, r), \nu)$ has value $+\infty$ if and only if $(\ell, r)$ is not in the attractor of* Min *towards region states $L_t \times \mathsf{Reg}(\mathcal{X}, M)$. Moreover, if we remove those region states from $\mathcal{R}(\mathcal{G})$, the value of the other configuration does not change.*

*Proof.* If a region state $(\ell, r)$ is not in the attractor of Min towards $L_t \times \mathsf{Reg}(\mathcal{X}, M)$, then Max has a (safety) strategy giving value $+\infty$ from all configurations $((\ell, r), \nu)$ with $\nu \in r$, and $\mathsf{Val}_{\mathcal{R}(\mathcal{G})}((\ell, r), \nu) = +\infty$. Conversely, if $(\ell, r)$ is in the attractor of Min towards $L_t \times \mathsf{Reg}(\mathcal{X}, M)$, then Min has a strategy giving finite value from $((\ell, r), \nu)$, and $\mathsf{Val}_{\mathcal{R}(\mathcal{G})}((\ell, r), \nu) < +\infty$. We conclude by Lemma 5.1. $\qquad\square$

We can now assume that all configurations have value in $\mathbb{R} \cup \{-\infty\}$ and that all target region states have final weight in $\mathbb{R}$: the precomputation needed so far consists only of attractor computations, that can be performed in complexity linear in $|\mathcal{R}(\mathcal{G})|$.

As previously explained, finding all states of value $-\infty$ is harder (in particular undecidable in general), but whenever we do manage to find them we can safely remove them by Lemma 5.1. The solution is quite simple for divergent WTGs, and slightly more involved in almost-divergent WTGs, because of the apparition of 0-cycles. In order to only present one uniform solution, we directly deal with the more general almost-divergent case. To do so, we introduce a new tool, the *kernels*, that will also be very useful in the rest of the study.

5.2. **Kernel of an almost-divergent WTG.** Kernels of an almost-divergent WTG are a way to group all 0-cycles of the game. We study those kernels and give a characterisation allowing computability. In [10], kernels have also been studied, and contain exactly all transitions and locations of weight 0. Contrary to their non-negative case, the situation is more complex in our case with arbitrary weights since 0-cycles could go through locations or transitions that have weight different from 0. Moreover, it is not trivial (and may not be true in a non almost-divergent WTG) to know whether it is sufficient to consider only simple 0-cycles, i.e. cycles without repetitions.

We will now construct the kernel $\mathsf{K}$ as the subgraph of $\mathcal{R}(\mathcal{G})$ containing all 0-cycles. Formally, let $T_\mathsf{K}$ be the set of edges of $\mathcal{R}(\mathcal{G})$ belonging to a *simple* 0-cycle, and $S_\mathsf{K}$ be the set of states covered by $T_\mathsf{K}$. We define the kernel $\mathsf{K}$ of $\mathcal{R}(\mathcal{G})$ as the subgraph of $\mathcal{R}(\mathcal{G})$ defined by $S_\mathsf{K}$ and $T_\mathsf{K}$. Edges in $T \backslash T_\mathsf{K}$ with starting state in $S_\mathsf{K}$ are called the output edges of $\mathsf{K}$. We define it using only simple 0-cycles in order to ensure its computability. However, we now show that this is of no harm, since the kernel contains exactly all the 0-cycles, which will be crucial in our approximation schema.
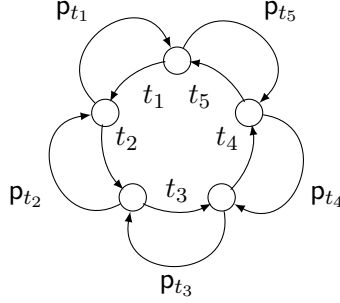
Figure 14: Application of the claim in the proof of Proposition 5.4

**Proposition 5.4.** *A cycle of $\mathcal{R}(\mathcal{G})$ is entirely in $\mathsf{K}$ if and only if it is a 0-cycle.*

*Proof.* We prove that every 0-cycle is in $\mathsf{K}$ by induction on the length of the cycles. The initialisation contains only cycles of length 1, that are in $\mathsf{K}$ by construction. If we consider a cycle $\mathsf{p}$ of length above 1, it is either simple or it can be rotated and decomposed into $\mathsf{p}'\mathsf{p}''$, $\mathsf{p}'$ and $\mathsf{p}''$ being smaller cycles. Let $\rho$ be a corner play following $\mathsf{p}'\mathsf{p}''$. We denote by $\rho'$ the prefix of $\rho$ following $\mathsf{p}'$ and $\rho''$ the suffix following $\mathsf{p}''$. It holds that $\mathsf{wt}_\Sigma(\rho') = -\mathsf{wt}_\Sigma(\rho'')$, and in an almost-divergent SCC this implies $\mathsf{wt}_\Sigma(\rho') = \mathsf{wt}_\Sigma(\rho'') = 0$. Therefore, by Lemma 4.1 both $\mathsf{p}'$ and $\mathsf{p}''$ are 0-cycles, and they must be in $\mathsf{K}$ by induction hypothesis.

We now prove that every cycle in $\mathsf{K}$ is a 0-cycle. By construction, every edge $t \in T_\mathsf{K}$ is part of a simple 0-cycle. Thus, to every edge $t \in T_\mathsf{K}$, we can associate a path $\mathsf{p}_t$ such that $t\mathsf{p}_t$ is a simple 0-cycle (rotate the simple cycle if necessary). The situation is exemplified in Figure 14. We can prove the following claim by relying on another pumping argument on corners:

**Claim.** *If $t_1 \cdots t_n$ is a path in $\mathsf{K}$, then $t_1 t_2 \cdots t_n \mathsf{p}_{t_n} \cdots \mathsf{p}_{t_2} \mathsf{p}_{t_1}$ is a 0-cycle of $\mathcal{R}(\mathcal{G})$.*

*Proof of Claim.* We prove the property by induction on $n$. For $n = 1$, the property is immediate since $t_1 \mathsf{p}_{t_1}$ is a 0-cycle. Consider then $n$ such that the property holds for $n$, and let us prove that it holds for $n+1$. We will exhibit two corner plays following $t_1 \cdots t_{n+1} \mathsf{p}_{t_{n+1}} \cdots \mathsf{p}_{t_1}$ of opposite weight and conclude with Lemma 4.1.

Let $\mathsf{v}_0$ be a corner of $\mathsf{last}(t_{n+1})$. Since $t_{n+1} \mathsf{p}_{t_{n+1}}$ is a 0-cycle, there exists $w \in \mathbb{Z}$, a corner play $\rho_0$ following $t_{n+1}$ ending in $\mathsf{v}_0$ with weight $w$ and a corner play $\rho'_0$ following $\mathsf{p}_{t_{n+1}}$ beginning in $\mathsf{v}_0$ with weight $-w$. We name $\mathsf{v}'_0$ the corner of $\mathsf{last}(t_n)$ where $\rho'_0$ ends. We consider any corner play $\rho_1$ following $t_{n+1}$ from corner $\mathsf{v}'_0$. The corner play $\rho'_0 \rho_1$ follows the path $\mathsf{p}_{t_{n+1}} t_{n+1}$ that is also a 0-cycle by Lemma 4.2, therefore $\rho_1$ has weight $w$. We denote by $\mathsf{v}_1$ the corner where $\rho_1$ ends. By iterating this construction, we obtain some corner plays $\rho_0, \rho_1, \rho_2, \ldots$ following $t_{n+1}$ and $\rho'_0, \rho'_1, \rho'_2, \ldots$ following $\mathsf{p}_{t_{n+1}}$ such that $\rho'_i$ goes from corner $\mathsf{v}_i$ to $\mathsf{v}'_i$, and $\rho_{i+1}$ from corner $\mathsf{v}'_i$ to $\mathsf{v}_{i+1}$, for all $i \geqslant 0$. Moreover, all corner plays $\rho_i$ have weight $w$ and all corner plays $\rho'_i$ have weight $-w$. Consider the first index $l$ such that $\mathsf{v}_l = \mathsf{v}_k$ for some $k < l$, which exists because the number of corners is finite.

We apply the induction to find a corner play following $t_1 \cdots t_n \mathsf{p}_{t_n} \cdots \mathsf{p}_{t_1}$, going through the corner $\mathsf{v}'_k$ in the middle: more formally, there exists $w_\alpha$, a corner play $\rho_\alpha$ following $t_1 \cdots t_n$ ending in $\mathsf{v}'_k$ with weight $w_\alpha$ and a corner play $\rho'_\alpha$ following $\mathsf{p}_{t_n} \cdots \mathsf{p}_{t_1}$ beginning in $\mathsf{v}'_k$ with weight $-w_\alpha$. We apply the induction a second time with corner $\mathsf{v}'_{l-1}$: there exists

$w_\beta$, a corner play $\rho_\beta$ following $t_1 \cdots t_n$ ending in $\mathsf{v}'_{l-1}$ with weight $w_\beta$ and a corner play $\rho'_\beta$ following $\mathsf{p}_{t_n} \cdots \mathsf{p}_{t_1}$ beginning in $\mathsf{v}'_{l-1}$ with weight $-w_\beta$.

The corner play $\rho_\alpha \rho_{k+1} \rho'_{k+1} \rho_{k+2} \rho'_{k+2} \cdots \rho'_{l-1} \rho'_\beta$, of weight $w_\alpha + (w - w)(l - k) - w_\beta = w_\alpha - w_\beta$, follows the cycle $t_1 \cdots t_n (t_{n+1} \mathsf{p}_{t_{n+1}})^{l-k} \mathsf{p}_{t_n} \cdots \mathsf{p}_{t_1}$. The corner play $\rho_\beta \rho_l \rho'_k \rho'_\alpha$, of weight $w_\beta + w - w - w_\alpha = w_\beta - w_\alpha$, follows the cycle $t_1 \cdots t_n t_{n+1} \mathsf{p}_{t_{n+1}} \mathsf{p}_{t_n} \cdots \mathsf{p}_{t_1}$. Since the game is almost-divergent, and those two corner plays are of opposite sign and in the same SCC, both have weight 0. The second corner play of weight 0 ensures that the cycle $t_1 \cdots t_{n+1} \mathsf{p}_{t_{n+1}} \cdots \mathsf{p}_{t_1}$ is a 0-cycle, by Lemma 4.1. $\triangle$

Now, if $\mathsf{p}$ is a cycle of $\mathcal{R}(\mathcal{G})$ in $\mathsf{K}$, there exists a cycle $\mathsf{p}'$ such that $\mathsf{pp}'$ is a 0-cycle. Since $\mathcal{R}(\mathcal{G})$ is an almost-divergent WTG, $\mathsf{p}$ is a 0-cycle. $\square$

5.3. **Values $-\infty$ coming from negative cycles.** Equipped with the kernels, we are now ready to remove the only remaining configurations having a value $-\infty$ in $\mathcal{R}(\mathcal{G})$.

**Proposition 5.5.** *In an SCC of $\mathcal{R}(\mathcal{G})$, the set of configurations with value $-\infty$ is a union of regions computable in time linear in the size of $\mathcal{R}(\mathcal{G})$. Moreover, if we remove those states from $\mathcal{R}(\mathcal{G})$, the value of the other configurations does not change.*

*Proof.* If the SCC is non-negative, the cumulative weight cannot decrease along a cycle, thus, there can be no configurations with value $-\infty$.

If the SCC is non-positive, let $T_\mathsf{t}$ be the set of edges of $\mathcal{R}(\mathcal{G})$ whose end state has location in $L_\mathsf{t}$. We prove that a configuration has value $-\infty$ if and only if it belongs to a region state where player $\mathsf{Min}$ can ensure the LTL formula on edges $\phi = (\mathrm{G} \neg T_\mathsf{t}) \wedge \neg \mathrm{FG}\, T_\mathsf{K}$.

If $(\ell, r)$ is a region state where $\mathsf{Min}$ can ensure $\phi$, $\mathsf{Min}$ can ensure a value of $-\infty$ from all configurations in $(\ell, r)$ by avoiding $S_\mathsf{t}$ for as long as desired, while not getting stuck in $\mathsf{K}$, and thus going through an infinite number of negative cycles by Proposition 5.4. Conversely, if $(\ell, r)$ is a region state from which $\mathsf{Min}$ cannot ensure $\phi$, then, by determinacy of $\omega$-regular timed games, $\mathsf{Max}$ can ensure $\neg\phi = (\mathrm{F}T_\mathsf{t}) \vee \mathrm{FG}T_\mathsf{K}$. Then, from $(\ell, r)$, $\mathsf{Max}$ must be able to reach $S_\mathsf{t}$ or stay in $\mathsf{K}$ forever. In both cases, $\mathsf{Max}$ can ensure a value above $-\infty$.

The procedure to detect $-\infty$ states thus consists in checking LTL formula $(\mathrm{G} \neg T_\mathsf{t}) \wedge \neg\mathrm{FG}\, T_\mathsf{K}$, and thus can be performed via three attractor computations in time linear in $|\mathcal{R}(\mathcal{G})|$. The set of region states with value $-\infty$ can then be removed safely from $\mathcal{R}(\mathcal{G})$, by Lemma 5.1 (since it is closed by attractor of $\mathsf{Min}$). $\square$

From this point on, **we assume that no configurations of $\mathcal{R}(\mathcal{G})$ have value $+\infty$ or $-\infty$, and that the final weight function maps all configurations to $\mathbb{R}$.** Since $\mathsf{fwt}$ is piecewise affine with finitely many pieces, $\mathsf{fwt}$ is bounded. Let $w_{\max}^\mathsf{t}$ denote the supremum of $|\mathsf{fwt}|$, ranging over all target configurations.

## 6. Semi-unfolding of WTGs

Given an almost-divergent WTG $\mathcal{G}$, we describe the construction of its *semi-unfolding* $\mathcal{T}(\mathcal{G})$ (as depicted in Figure 11). This crucially relies on the absence of states with value $-\infty$ which has been explained in Section 5.

We only build the semi-unfolding $\mathcal{T}(\mathcal{G})$ of an SCC of $\mathcal{G}$ starting from some state $(\ell_0, r_0) \in S$ of the region game, since it is then easy to glue all the semi-unfoldings together to get the one of the full game. **We thus assume in this section that $\mathcal{R}(\mathcal{G})$ is an SCC.** Since

every configuration has finite value, we will prove that values of the game are bounded by $|\mathcal{R}(\mathcal{G})|w_{\max} + w_{\max}^{\mathsf{t}}$. As a consequence, we can find a bound $\gamma$ linear in $|\mathcal{R}(\mathcal{G})|$, $w_{\max}$ and $w_{\max}^{\mathsf{t}}$ such that a play that visits some state outside the kernel more than $\gamma$ times has weight strictly above $|\mathcal{R}(\mathcal{G})|w_{\max} + w_{\max}^{\mathsf{t}}$, hence is useless for the value computation. This leads to considering the semi-unfolding $\mathcal{T}(\mathcal{G})$ of $\mathcal{R}(\mathcal{G})$ (nodes in the kernel are not unfolded, see Figure 11) such that each node not in the kernel is encountered at most $\gamma$ times along a branch: the end of each branch is called a *stopped leaf* of the semi-unfolding. In particular, the depth of $\mathcal{T}(\mathcal{G})$ is bounded by $|\mathcal{R}(\mathcal{G})|\gamma$, and thus is polynomial in $|\mathcal{R}(\mathcal{G})|$, $w_{\max}$ and $w_{\max}^{\mathsf{t}}$. Leaves of the semi-unfolding are thus of two types: target leaves that are copies of target locations of $\mathcal{R}(\mathcal{G})$ for which we set the target weight as in $\mathcal{R}(\mathcal{G})$, and stop leaves for which we set their target weight as being constant to $+\infty$ if the SCC $\mathcal{R}(\mathcal{G})$ is non-negative, and $-\infty$ if the SCC is non-positive.

More formally, if $(\ell, r)$ is in $\mathsf{K}$, we let $\mathsf{K}_{\ell,r}$ be the part of $\mathsf{K}$ accessible from $(\ell, r)$ (note that $\mathsf{K}_{\ell,r}$ is an SCC as $\mathsf{K}$ is a disjoint set of SCCs). We define the output edges of $\mathsf{K}_{\ell,r}$ as being the output edges of $\mathsf{K}$ accessible from $(\ell, r)$. If $(\ell, r)$ is not in $\mathsf{K}$, the output edges of $(\ell, r)$ are the edges of $\mathcal{R}(\mathcal{G})$ starting in $(\ell, r)$.

We define a tree $T$ whose nodes are either labelled by region states $(\ell, r) \in S \backslash S_{\mathsf{K}}$ or by kernels $\mathsf{K}_{\ell,r}$, and whose edges are labelled by output edges in $\mathcal{R}(\mathcal{G})$. The root of the tree $T$ is labelled with an initial region state $(\ell_0, r_0)$, or $\mathsf{K}_{\ell_0,r_0}$ (if $(\ell_0, r_0)$ belongs to the kernel), and the successors of a node of $T$ are then recursively defined by its output edges. When a state $(\ell, r)$ is reached by an output edge, the child is labelled by $\mathsf{K}_{\ell,r}$ if $(\ell, r) \in \mathsf{K}$, otherwise it is labelled by $(\ell, r)$. Edges in $T$ are labelled by the edges used to create them. Along every branch, we stop the construction when either a final state is reached (i.e. a state not inside the current SCC) or the branch contains $3|\mathcal{R}(\mathcal{G})|w_{\max} + 2w_{\max}^{\mathsf{t}} + 2$ nodes labelled by the same state ($(\ell, r)$ or $\mathsf{K}_{\ell,r}$). Leaves of $T$ with a location belonging to $L_{\mathsf{t}}$ are called *target leaves*, others are called *stopped leaves*.

We now transform $T$ into a WTG $\mathcal{T}(\mathcal{G})$, by replacing every node labelled by a state $(\ell, r)$ by a different copy $(\tilde{\ell}, r)$ of $(\ell, r)$. Those states are said to inherit from $(\ell, r)$. Edges of $T$ are replaced by the edges labelling them, and have a similar notion of inheritance. Every non-leaf node labelled by a kernel $\mathsf{K}_{\ell,r}$ is replaced by a copy of the WTG $\mathsf{K}_{\ell,r}$, output edges being plugged in the expected way. We deal with stopped leaves labelled by a kernel $\mathsf{K}_{\ell,r}$ by replacing them with a single node copy of $(\ell, r)$, like we dealt with node labelled by a state $(\ell, r)$. State partition between players and weights are inherited from the copied states of $\mathcal{R}(\mathcal{G})$. The only initial state of $\mathcal{T}(\mathcal{G})$ is the state denoted by $(\tilde{\ell}_0, r_0)$ inherited from $(\ell_0, r_0)$ in the root of $T$ (either $(\ell_0, r_0)$ or $\mathsf{K}_{\ell_0,r_0}$). The final states of $\mathcal{T}(\mathcal{G})$ are the states derived from leaves of $T$. If $\mathcal{R}(\mathcal{G})$ is a non-negative (resp. non-positive) SCC, the final weight function $\mathsf{fwt}$ is inherited from $\mathcal{R}(\mathcal{G})$ on target leaves and set to $+\infty$ (resp. $-\infty$) on stopped leaves.

The semi-unfolding of the WTG from Figure 10 can be found in Figure 18 of Appendix A.

**Proposition 6.1.** *Let $\mathcal{G}$ be an almost-divergent WTG, and let $(\ell_0, r_0)$ be some region state of $\mathcal{R}(\mathcal{G})$. The semi-unfolding $\mathcal{T}(\mathcal{G})$ with initial state $(\tilde{\ell}_0, r_0)$ (a copy of a region state $(\ell_0, r_0)$) is equivalent to $\mathcal{G}$, i.e. for all $\nu_0 \in r_0$, $\mathsf{Val}_{\mathcal{G}}(\ell_0, \nu_0) = \mathsf{Val}_{\mathcal{T}(\mathcal{G})}((\tilde{\ell}_0, r_0), \nu_0)$.*

The rest of this section aims at proving Proposition 6.1, only in the case where $\mathcal{R}(\mathcal{G})$ is an SCC, since the general result then follows easily. First, we need information on the weight of finite plays in the region game.

**Lemma 6.2.** *All finite plays in $\mathcal{R}(\mathcal{G})$ have cumulative weight (ignoring final weights) at least $-|\mathcal{R}(\mathcal{G})|w_{\max}$ in the non-negative case, and at most $|\mathcal{R}(\mathcal{G})|w_{\max}$ in the non-positive case. Moreover, values of the game are bounded by $|\mathcal{R}(\mathcal{G})|w_{\max} + w_{\max}^{\mathsf{t}}$.*

*Proof.* Suppose first that $\mathcal{R}(\mathcal{G})$ is a non-negative SCC. Consider a play $\rho$ following a path $\mathsf{p}$. This path $\mathsf{p}$ can be decomposed into $\mathsf{p} = \mathsf{p}_1 \mathsf{p}_1^c \cdots \mathsf{p}_n \mathsf{p}_k^c$ such that every $\mathsf{p}_i^c$ is a cycle, and $\mathsf{p}_1 \ldots \mathsf{p}_k$ is a simple path in $\mathcal{R}(\mathcal{G})$ (thus $\sum_{i=1}^{k} |\mathsf{p}_i| \leqslant |\mathcal{R}(\mathcal{G})|$). Let us define all plays $\rho_i$ and $\rho_i^c$ as the restrictions of $\rho$ on $\mathsf{p}_i$ and $\mathsf{p}_i^c$. Now, since all plays following cycles have cumulative weight at least 0,

$$\mathsf{wt}_\Sigma(\rho) = \sum_{i=1}^{k} \mathsf{wt}_\Sigma(\rho_i) + \mathsf{wt}_\Sigma(\rho_i^c) \geqslant \sum_{i=1}^{k} -w_{\max}|\rho_i| + 0 \geqslant -|\mathcal{R}(\mathcal{G})|w_{\max}$$

Similarly, we can show that every play in a non-positive SCC has cumulative weight at most $|\mathcal{R}(\mathcal{G})|w_{\max}$.

For the bound on the values, consider again two cases. If $\mathcal{R}(\mathcal{G})$ is a non-negative SCC, consider a positional attractor strategy $\sigma_{\mathsf{Min}}$ for $\mathsf{Min}$ toward $S_{\mathsf{t}}$. Since all states have values below $+\infty$, all plays obtained from strategies of $\mathsf{Max}$ will follow simple paths of $\mathcal{R}(\mathcal{G})$, that have cumulative weight at most $|\mathcal{R}(\mathcal{G})|w_{\max}$ in absolute value. Similarly, if $\mathcal{R}(\mathcal{G})$ is a non-positive SCC, following the proof of Proposition 5.5, since all values are above $-\infty$, $\mathsf{Max}$ can ensure $\neg\phi$, i.e. $(\mathsf{F}T_{\mathsf{t}}) \vee \mathsf{FG}T_{\mathsf{K}}$ on all states. Then we can construct a strategy $\sigma_{\mathsf{Max}}$ for $\mathsf{Max}$ combining an attractor strategy toward $S_{\mathsf{t}}$ on states satisfying $\mathsf{F}T_{\mathsf{t}}$, a safety strategy on states satisfying $GT_{\mathsf{K}}$, and an attractor strategy toward the latter on all other states. Then, all plays obtained from strategies of $\mathsf{Min}$ will either not be winning ($GT_{\mathsf{K}}$) or follow simple paths of $\mathcal{R}(\mathcal{G})$. Both cases imply that the values of the game are bounded by $|\mathcal{R}(\mathcal{G})|w_{\max} + w_{\max}^{\mathsf{t}}$. □

We can use this to obtain similar results in the semi-unfolding $\mathcal{T}(\mathcal{G})$.

**Lemma 6.3.** *All plays in $\mathcal{T}(\mathcal{G})$ from the initial state $(\tilde{\ell}_0, r_0)$ to a stopped leaf have cumulative weight at least $2|\mathcal{R}(\mathcal{G})|w_{\max} + 2w_{\max}^{\mathsf{t}} + 1$ if the SCC $\mathcal{R}(\mathcal{G})$ is non-negative, and at most $-2|\mathcal{R}(\mathcal{G})|w_{\max} - 2w_{\max}^{\mathsf{t}} - 1$ if it is non-positive.*

*Proof.* Note that by construction, all finite paths in $\mathcal{T}(\mathcal{G})$ from the initial state to a stopped leaf can be decomposed as $\mathsf{p}'\mathsf{p}_1 \cdots \mathsf{p}_{3|\mathcal{R}(\mathcal{G})|w_{\max}+2w_{\max}^{\mathsf{t}}+1}$ with all $\mathsf{p}_i$ being cycles around the same state. Additionally, those cycles cannot be 0-cycles by Proposition 5.4, since they take at least one edge outside of $\mathsf{K}$. Therefore the restriction of $\rho$ to $\mathsf{p}_1 \cdots \mathsf{p}_{3|\mathcal{R}(\mathcal{G})|w_{\max}+2w_{\max}^{\mathsf{t}}+1}$ has weight at least $3|\mathcal{R}(\mathcal{G})|w_{\max} + 2w_{\max}^{\mathsf{t}} + 1$ (in the non-negative case) and at most $-3|\mathcal{R}(\mathcal{G})|w_{\max} - 2w_{\max}^{\mathsf{t}} - 1$ (in the non-positive case). The beginning of the play, following $\mathsf{p}'$, has cumulative weight at least $-|\mathcal{R}(\mathcal{G})|w_{\max}$ (in the non-negative case) and at most $|\mathcal{R}(\mathcal{G})|w_{\max}$ (in the non-positive case), by Lemma 6.2. □

Consider two plays of the same length $k$ in $\mathcal{R}(\mathcal{G})$ and $\mathcal{T}(\mathcal{G})$, respectively:

$$\rho = ((\ell_1, r_1), \nu_1) \xrightarrow{d_1, t_1} \cdots \xrightarrow{d_{k-1}, t_{k-1}} ((\ell_k, r_k), \nu_k)$$

$$\tilde{\rho} = ((\tilde{\ell}_1, r_1), \nu_1) \xrightarrow{d_1, \tilde{t}_1} \cdots \xrightarrow{d_{k-1}, \tilde{t}_{k-1}} ((\tilde{\ell}_k, r_k), \nu_k).$$

They are said to *mimic* each other if every $(\tilde{\ell}_i, r_i)$ is inherited from $(\ell_i, r_i)$ and every edge $\tilde{t}_i$ is inherited from the edge $t_i$ of $\mathcal{R}(\mathcal{G})$. Combining Lemmas 6.2 and 6.3, we obtain:

**Lemma 6.4.** *If $\mathcal{R}(\mathcal{G})$ is a non-negative (resp. non-positive) SCC, every play from the initial state and with cumulative weight less than $|\mathcal{R}(\mathcal{G})|w_{\max} + 2w_{\max}^{\mathsf{t}} + 1$ (resp. greater than $-|\mathcal{R}(\mathcal{G})|w_{\max} - 2w_{\max}^{\mathsf{t}} - 1$) can be mimicked in $\mathcal{T}(\mathcal{G})$ without reaching a stopped leaf. Conversely, every play in $\mathcal{T}(\mathcal{G})$ reaching a target leaf can be mimicked in $\mathcal{R}(\mathcal{G})$.*

*Proof.* We prove only the non-negative case, since the other case is symmetrical. Let $\rho$ be a play of $\mathcal{R}(\mathcal{G})$ with cumulative weight less than $|\mathcal{R}(\mathcal{G})|w_{\max} + 2w_{\max}^{\mathsf{t}} + 1$. Consider the branch of the unfolded game it follows. If $\rho$ cannot be mimicked in $\mathcal{T}(\mathcal{G})$, then a prefix of $\rho$ reaches the stopped leaf of that branch when mimicked in $\mathcal{T}(\mathcal{G})$. In this situation, $\rho$ starts by a prefix of weight at least $2|\mathcal{R}(\mathcal{G})|w_{\max} + 2w_{\max}^{\mathsf{t}} + 1$ by Lemma 6.3 and then ends with a suffix play of weight at least $-|\mathcal{R}(\mathcal{G})|w_{\max}$ by Lemma 6.2, and that contradicts the initial assumption. The converse is true by construction. $\qquad\square$

Then, intuitively, the plays of $\mathcal{R}(\mathcal{G})$ starting in an initial configuration that cannot be mimicked in $\mathcal{T}(\mathcal{G})$ are not useful for value computation. To obtain Proposition 6.1, we now prove that for all valuations $\nu_0 \in r_0$, $\mathsf{Val}_{\mathcal{G}}(\ell_0, \nu_0) = \mathsf{Val}_{\mathcal{T}(\mathcal{G})}((\tilde{\ell}_0, r_0), \nu_0)$. By Lemma 2.7, we already know that $\mathsf{Val}_{\mathcal{G}}(\ell_0, \nu_0) = \mathsf{Val}_{\mathcal{R}(\mathcal{G})}((\ell_0, r_0), \nu_0)$. Recall that we only left finite values in $\mathcal{R}(\mathcal{G})$ (in the final weight functions, in particular), and more precisely $|\mathsf{Val}_{\mathcal{R}(\mathcal{G})}((\ell_0, r_0), \nu_0)| \leqslant |\mathcal{R}(\mathcal{G})|w_{\max} + w_{\max}^{\mathsf{t}}$ by Lemma 6.2. We first show that the value is also finite in $\mathcal{T}(\mathcal{G})$. Indeed, if $\mathsf{Val}_{\mathcal{T}(\mathcal{G})}((\tilde{\ell}_0, r_0), \nu_0) = +\infty$, since we assumed all final weights of $\mathcal{R}(\mathcal{G})$ bounded, we are necessarily in the non-negative case, and $\mathsf{Max}$ is able to ensure stopped leaves reachability.

**Claim.** *If $\mathsf{Val}_{\mathcal{T}(\mathcal{G})}((\tilde{\ell}_0, r_0), \nu_0) = +\infty$, then there are no strategies in $\mathcal{R}(\mathcal{G})$ for $\mathsf{Min}$ ensuring weight less than $|\mathcal{R}(\mathcal{G})|w_{\max} + w_{\max}^{\mathsf{t}} + 1$ from $(\ell_0, r_0)$.*

Thus, we can obtain the contradiction $\mathsf{Val}_{\mathcal{R}(\mathcal{G})}((\ell_0, r_0), \nu_0) > |\mathcal{R}(\mathcal{G})|w_{\max} + w_{\max}^{\mathsf{t}}$.

*Proof of Claim.* By contradiction, consider a strategy $\sigma_{\mathsf{Min}}$ of $\mathsf{Min}$ ensuring weight $A \leqslant |\mathcal{R}(\mathcal{G})|w_{\max} + w_{\max}^{\mathsf{t}} + 1$ in $\mathcal{R}(\mathcal{G})$. Then, for all $\sigma_{\mathsf{Max}}$, the cumulative weight of the play $\mathsf{play}_{\mathcal{R}(\mathcal{G})}(((\tilde{\ell}_0, r_0), \nu_0), \sigma_{\mathsf{Min}}, \sigma_{\mathsf{Max}})$ (reaching target configuration $(\ell, \nu)$) is at most equal to $A - \mathsf{fwt}(\ell, \nu) \leqslant |\mathcal{R}(\mathcal{G})|w_{\max} + 2w_{\max}^{\mathsf{t}} + 1$, and by Lemma 6.4 this play does not reach a stopped leaf when mimicked in $\mathcal{T}(\mathcal{G})$, which is absurd. $\qquad\triangle$

If $\mathsf{Val}_{\mathcal{T}(\mathcal{G})}((\tilde{\ell}_0, r_0), \nu_0) = -\infty$, we are necessarily in the non-positive case, and, by construction, this implies that $\mathsf{Min}$ ensures stopped leaves reachability in $\mathcal{T}(\mathcal{G})$.

**Claim.** *If $\mathsf{Val}_{\mathcal{T}(\mathcal{G})}((\tilde{\ell}_0, r_0), \nu_0) = -\infty$, then there are no strategies in $\mathcal{R}(\mathcal{G})$ for $\mathsf{Max}$ ensuring weight above $-|\mathcal{R}(\mathcal{G})|w_{\max} - w_{\max}^{\mathsf{t}} - 1$ from $(\ell_0, r_0)$.*

Thus, we can obtain the contradiction $\mathsf{Val}_{\mathcal{R}(\mathcal{G})}((\ell_0, r_0), \nu_0) < -|\mathcal{R}(\mathcal{G})|w_{\max} - w_{\max}^{\mathsf{t}}$.

*Proof of Claim.* By contradiction, consider a strategy $\sigma_{\mathsf{Max}}$ of $\mathsf{Max}$ ensuring weight $A \geqslant -|\mathcal{R}(\mathcal{G})|w_{\max} - w_{\max}^{\mathsf{t}} - 1$ in $\mathcal{R}(\mathcal{G})$. Then, for all $\sigma_{\mathsf{Min}}$, the cumulative weight of the play $\mathsf{play}_{\mathcal{R}(\mathcal{G})}(((\tilde{\ell}_0, r_0), \nu_0), \sigma_{\mathsf{Min}}, \sigma_{\mathsf{Max}})$ (reaching target configuration $(\ell, \nu)$) is at least equal to $A - \mathsf{fwt}(\ell, \nu) \geqslant -|\mathcal{R}(\mathcal{G})|w_{\max} - 2w_{\max}^{\mathsf{t}} - 1$, and by Lemma 6.4 this play does not reach a stopped leaf when mimicked in $\mathcal{T}(\mathcal{G})$, which is absurd. $\qquad\triangle$

If $\mathcal{R}(\mathcal{G})$ is non-negative, for all $\varepsilon > 0$ we can fix an $\varepsilon$-optimal strategy $\sigma_{\mathsf{Min}}$ for $\mathsf{Min}$ in $\mathcal{T}(\mathcal{G})$. Every play derived from $\sigma_{\mathsf{Min}}$ in $\mathcal{T}(\mathcal{G})$ reaches a target leaf, and can thus be mimicked in $\mathcal{R}(\mathcal{G})$ by Lemma 6.4. Therefore, $\sigma_{\mathsf{Min}}$ can be mimicked in $\mathcal{R}(\mathcal{G})$, where it keeps the same value. From this we deduce $\mathsf{Val}_{\mathcal{R}(\mathcal{G})}((\ell_0, r_0), \nu_0) \leqslant \mathsf{Val}_{\mathcal{T}(\mathcal{G})}((\tilde{\ell}_0, r_0), \nu_0)$. If $\mathcal{R}(\mathcal{G})$

is non-positive, the same reasoning applies by considering an $\varepsilon$-optimal strategy for $\mathsf{Max}$ in $\mathcal{T}(\mathcal{G})$.

Let us now show the reverse inequality. If $\mathcal{R}(\mathcal{G})$ is non-negative, let us fix $0 < \varepsilon < 1$, an $\varepsilon$-optimal strategy $\sigma_{\mathsf{Min}}$ for $\mathsf{Min}$ in $\mathcal{R}(\mathcal{G})$, and a strategy $\sigma_{\mathsf{Max}}$ of $\mathsf{Max}$ in $\mathcal{R}(\mathcal{G})$. Let $\rho$ be their outcome $\mathsf{play}_{\mathcal{R}(\mathcal{G})}(((\ell_0, r_0), \nu_0), \sigma_{\mathsf{Min}}, \sigma_{\mathsf{Max}})$, $\rho_k$ be the finite prefix of $\rho$ defining its cumulative weight and $(\ell_k, \nu_k)$ be the configuration defining its final weight, such that $\mathsf{wt}_{\mathcal{R}(\mathcal{G})}(\rho) = \mathsf{wt}_\Sigma(\rho_k) + \mathsf{fwt}(\ell_k, \nu_k)$. Then,

$$\mathsf{wt}_{\mathcal{R}(\mathcal{G})}(\rho) \leqslant \mathsf{Val}_{\mathcal{R}(\mathcal{G})}((\ell_0, r_0), \nu_0) + \varepsilon < |\mathcal{R}(\mathcal{G})| w_{\max} + w_{\max}^{\mathsf{t}} + 1$$

therefore

$$\mathsf{wt}_\Sigma(\rho_k) < |\mathcal{R}(\mathcal{G})| w_{\max} + w_{\max}^{\mathsf{t}} + 1 - \mathsf{fwt}(\ell_k, \nu_k) \leqslant |\mathcal{R}(\mathcal{G})| w_{\max} + 2 w_{\max}^{\mathsf{t}} + 1$$

and by Lemma 6.4 all such plays $\rho$ can be mimicked in $\mathcal{T}(\mathcal{G})$, so that

$$\mathsf{Val}_{\mathcal{T}(\mathcal{G})}((\tilde{\ell}_0, r_0), \nu_0) \leqslant \mathsf{Val}_{\mathcal{R}(\mathcal{G})}((\ell_0, r_0), \nu_0)$$

Once again, if $\mathcal{R}(\mathcal{G})$ is non-positive, the same reasoning applies by considering an $\varepsilon$-optimal strategy for $\mathsf{Max}$ in $\mathcal{R}(\mathcal{G})$. This ends the proof of Proposition 6.1.

*Remark.* Note that the semi-unfolding procedure of an SCC depends on $w_{\max}^{\mathsf{t}}$, where $\mathsf{fwt}$ can be the value function of an SCCs under the current one. Assuming all configurations have finite value, we can extend the reasoning of Lemma 6.2 and bound all values in the full game by $|\mathcal{R}(\mathcal{G})| w_{\max} + w_{\max}^{\mathsf{t}}$, which lets us bound uniformly the unfolding depth of each SCC and gives us a bound on the depth of the complete semi-unfolding tree:

$$|\mathcal{R}(\mathcal{G})|(3|\mathcal{R}(\mathcal{G})| w_{\max} + 2 w_{\max}^{\mathsf{t}} + 2) + 1 \tag{6.1}$$

## 7. Computing values for acyclic WTGs

In this section, we focus on the class of acyclic WTGs where the value problem is decidable, as shown by [1]. In our context, this will give us a way to compute the value of the semi-unfolding $\mathcal{T}(\mathcal{G})$ of an almost-divergent WTG that contains no kernels: this is equivalent for the WTG to be divergent indeed. This section is particularly technical, and can thus be skipped by non-interested readers. We have chosen to give such detailed explanations of the techniques used in [1] (with independent proofs) for several reasons. On the one hand, our setting is more general, in the sense that we allow for negative weights and for final weights, where they do not do so explicitly. On the other hand, their result is stated for concurrent games, a generalisation of the turn-based games we consider. This leads to simplifications in the proofs, and makes easier some parts of the complexity analysis. We will need, in Section 8, to bound the partial derivatives of the functions we compute. This cannot be deduced from their result directly. We present their techniques in a new, more symbolic light, by performing computations on the entire state-space at once instead of region by region. For reasons detailed in Section 7.4 and in [17], we are not able to replicate their (incomplete) complexity analysis. We will therefore rely on a doubly-exponential upper bound instead of the exponential one claimed in [1]. Last but not least, this detailed analysis allows us to solve the synthesis of $\varepsilon$-optimal strategies for both players, as will be detailed in Section 10.

The main result of this section is a symbolic algorithm for computing the value $\boldsymbol{V}^i = \mathsf{Val}_{\mathcal{G}}^i$ defined in Section 2.6.

**Theorem 7.1.** *Given $i \geqslant 0$, computing $\mathsf{Val}^i_{\mathcal{G}}$ can be done in time doubly-exponential in $i$ and exponential in the size of $\mathcal{G}$.*

The intuition behind the result is the observation that the mappings $\boldsymbol{V}^0_\ell$ are piecewise affine for all $\ell$, and a proof that $\mathcal{F}$ preserves piecewise affinity, so that all iterates $\boldsymbol{V}^i_\ell$ can be computed using piecewise affine functions. In order to bound the size of $\boldsymbol{V}^i_\ell$ (in particular, its number of pieces), we need the fine encoding via cells and partition value functions defined in Section 2.7.

7.1. **About complexity bounds.** In this section we will assume without loss of generality that the number of clocks $|\mathcal{X}| = n \geqslant 1$. The case $n = 0$ (finite weighted games) will be detailed in Section 11. The piecewise affine value function $\boldsymbol{V}$ is encoded as pairs $(P_\ell, F_\ell)_{\ell \in L}$ such that $\llbracket F_\ell \rrbracket = \boldsymbol{V}_\ell$ for all $\ell \in L$. We detail now how we measure the complexity of a pair $(P, F)$. As mentioned before, we assume without loss of generality that (in)equalities involved in the definition of cells only use integers. This is not the case for value functions which are described by equations $\mathsf{y} = a_1 \mathsf{x}_1 + \cdots + a_n \mathsf{x}_n + b$ with $a_i$ and $b$ in $\mathbb{Q}$. In order to track their size, we instead write $a_\mathsf{y} \mathsf{y} = a_1 \mathsf{x}_1 + \cdots + a_n \mathsf{x}_n + b$, with all $a_i$ and $b$ integers of $\mathbb{Z}$, and $a_\mathsf{y} \in \mathbb{N}_{>0}$.

**Definition 7.2.** Let $(P, F)$ be a pair composed of a partition $P$ and a value function $F$ defined on that partition. The *complexity of $(P, F)$* is the pair $\langle m, \beta \rangle$ where the *size complexity $m$* and the *constant complexity $\beta$* are defined as follows. First, we define the size complexity as $m = |\mathcal{E}_P| + m_0$, where $m_0$ denotes the number of inequalities in the encoding of the domain $c_P$.[12] Second, the constant complexity $\beta$ is the smallest natural number that upper bounds (in absolute value) every finite constant (partial derivatives and additive constants) in the affine (in)equalities of $P$ and $F$. Given a piecewise affine value function $(P, F)$ of complexity $\langle m, \beta \rangle$, we say that $(P, F)$ has complexity at most $\langle m', \beta' \rangle$ if $m \leqslant m'$ and $\beta \leqslant \beta'$.

From this definition, we can bound the memory needed to store a partition value function:

**Lemma 7.3.** *Let $\boldsymbol{V}$ be represented by a pair $(P, F)$ of complexity at most $\langle m, \beta \rangle$. Then it can be represented using a space in $\mathcal{O}(2^n(n + 2)(m + 1)^{n+1} \lceil \log \beta \rceil)$.*

*Proof.* Storing $P$ requires storing $m$ affine expressions, each stored in space $(n + 1) \lceil \log \beta \rceil$. Moreover, each affine expression of the value function $F$, one for each base cell of $P$, is stored in space $(n + 2) \lceil \log \beta \rceil$. $P$ has at most $2^n(m + 1)^n$ base cells by Lemma 2.14, and storing each base cell explicitly requires storing at most $m$ borders per base cell. In total, $\boldsymbol{V}$ can be represented in space $2^{n+1}(n + 2)(m + 1)^{n+1} \lceil \log \beta \rceil$. $\qquad \square$

7.2. **Operations over value functions.** Our goal is now to compute the value function $\mathcal{F}(\boldsymbol{V})$ encoded by $(P'_\ell, F'_\ell)_{\ell \in L}$, when $\boldsymbol{V}$ is encoded by $(P_\ell, F_\ell)_{\ell \in L}$. We decompose the computation into smaller operations that we first introduce and compute over partitions, while explaining how they affect the complexity parameter $\langle m, \beta \rangle$.

If $P_1$ and $P_2$ are partitions over the same domain $c_P$, let $P_1 \oplus P_2$ denote the coarsest partition consistent with both $P_1$ and $P_2$: each base cell $c_b$ of $P_1 \oplus P_2$ corresponds to an

---

[12] If $c_P = [0, M)^{\mathcal{X}}$, then $m_0 = n$.

intersection $c_1 \cap c_2$, with $c_1$ a base cell of $P_1$ and $c_2$ a base cell of $P_2$. It is obtained from $\mathcal{E}_{P_1 \oplus P_2} = \mathcal{E}_{P_1} \cup \mathcal{E}_{P_2}$. Note that if $P_1 \ldots P_q$ are partitions of complexity at most $\langle m, \beta \rangle$, $P_1 \oplus \cdots \oplus P_q$ is a partition of size complexity at most $\langle qm, \beta \rangle$. Now, the minimum (resp. maximum) of a finite set of piecewise affine value functions can be computed with partitions.

**Lemma 7.4** (Thm. 1, [1]). *Let $(P_i, F_i)_{1 \leqslant i \leqslant q}$ be piecewise affine value functions, defined over the same domain $c_P$, where each $(P_i, F_i)$ has complexity at most $\langle m, \beta \rangle$. There exists a piecewise affine value function $(P', F')$ of domain $c_P$ and complexity at most $\langle m', \beta' \rangle$, where $m' = qm + q^2$, and $\beta' = 2\beta^2$, such that $[\![F']\!] = \min_{1 \leqslant i \leqslant q}[\![F_i]\!]$ (resp. $[\![F']\!] = \max_{1 \leqslant i \leqslant q}[\![F_i]\!]$).*

*Proof.* Let $P'$ be $P_1 \oplus \cdots \oplus P_q$. Let $c$ denote a base cell in $P'$, corresponding to an intersection $c_1 \cap \cdots \cap c_q$ of base cells of $P_1, \ldots, P_q$ respectively. Consider the affine value functions $F_1(c_1), \ldots, F_q(c_q)$. Each of these is defined by an equation of the form $a_y y = a_1 x_1 + \cdots + a_n x_n + b$, that can be seen as affine equalities over variables $\mathcal{X} \uplus \{y\}$, denoted $E_1, \ldots, E_q$, or equivalently as sets of points in $\mathbb{R}^{\mathcal{X} \uplus \{y\}}$, denoted $[\![E_1]\!], \ldots, [\![E_q]\!]$. If $E$ and $E'$ are such equalities, of equations $a_y y = a_1 x_1 + \cdots + a_n x_n + b$ and $a'_y y = a'_1 x_1 + \cdots + a'_n x_n + b'$, the intersection $[\![E]\!] \cap [\![E']\!]$ is either empty or, by elimination of $y$, it satisfies the equation

$$(a_y a'_1 - a'_y a_1)x_1 + \cdots + (a_y a'_n - a'_y a_n)x_n + (a_y b' - a'_y b) = 0 \,.$$

This describes an affine equality over $\mathcal{X}$, that we denote $E \cap_y E'$, with constant complexity at most $2\beta^2$. Now, let us partition $P_1 \oplus \cdots \oplus P_q$ by the set of all such intersections

$$\{E_i \cap_y E_j \mid 1 \leqslant i, j \leqslant q \land [\![E_i]\!] \cap [\![E_j]\!] \neq \emptyset\} \,.$$

On every sub-cell $c'$ of this partition, there exists $j \in \{1, \ldots, n\}$ such that for every $\nu \in c'$, $[\![F_j]\!](\nu) = \min_{1 \leqslant i \leqslant q}[\![F_i]\!](\nu)$. Therefore, we define $F'$ on $c'$ as equal to $F_j(c_j)$. The partition $P_1 \oplus \cdots \oplus P_q$ has size complexity at most $qm$, and we added at most $q^2$ intersections $E \cap_y E'$, resulting in a partition of the expected complexity. $\qquad \square$

For all $\nu \in [0, M)^{\mathcal{X}}$, let $d_\nu = \sup\{d \mid \nu + d \in [0, M)^{\mathcal{X}}\} \in \mathbb{R}_{\geqslant 0}$ denote the greatest valid delay from $\nu$ (in fact, $d_\nu = M - \|\nu\|_\infty$). Consider the following operations:

- If $\mathcal{Y} \subseteq \mathcal{X}$ is a set of clocks and $\ell \in L$, let $\mathsf{Unreset}_{\mathcal{Y}}(\boldsymbol{V}_\ell) : [0, M)^{\mathcal{X}} \to \mathbb{R}_\infty$ denote the value function such that for all $\nu$,

$$\mathsf{Unreset}_{\mathcal{Y}}(\boldsymbol{V}_\ell)(\nu) = \boldsymbol{V}_\ell(\nu[\mathcal{Y} := 0]) \,.$$

- If $g$ is a guard over $\mathcal{X}$ and $\ell \in L$, let $\mathsf{Guard}_g(\boldsymbol{V}_\ell) : [0, M)^{\mathcal{X}} \to \mathbb{R}_\infty$ denote the value function such that for all $\nu$,

$$\mathsf{Guard}_g(\boldsymbol{V}_\ell)(\nu) = \begin{cases} \boldsymbol{V}_\ell(\nu) & \text{if } \nu \models g \\ -\infty & \text{if } \nu \not\models g \land \ell \in L_{\mathsf{Max}} \\ +\infty & \text{if } \nu \not\models g \land \ell \in L_{\mathsf{Min}} \,. \end{cases}$$

The values $+\infty$ and $-\infty$ in $\mathsf{Guards}_g$ ensure that players cannot choose invalid delays: By the no-deadlocks assumption, from every configuration there exists a transition in $[\![\mathcal{G}]\!]$, whose value will win against $+\infty$ or $-\infty$ in the subsequent equality (7.2).

- If $e \in E$ is an edge from $\ell$ to $\ell'$, let $\mathsf{Pre}_e(\boldsymbol{V}_{\ell'}) : [0, M)^{\mathcal{X}} \to \mathbb{R}_\infty$ denote the value function such that for all $\nu$,

$$\mathsf{Pre}_e(\boldsymbol{V}_{\ell'})(\nu) = \begin{cases} \sup_{d \in [0, d_\nu]} \left[d \cdot \mathsf{wt}(\ell) + \mathsf{wt}(e) + \boldsymbol{V}_{\ell'}(\nu + d)\right] & \text{if } \ell \in L_{\mathsf{Max}} \\ \inf_{d \in [0, d_\nu]} \left[d \cdot \mathsf{wt}(\ell) + \mathsf{wt}(e) + \boldsymbol{V}_{\ell'}(\nu + d)\right] & \text{if } \ell \in L_{\mathsf{Min}} \,. \end{cases} \tag{7.1}$$

Then, if $\boldsymbol{V'} = \mathcal{F}(\boldsymbol{V})$, it holds that

$$
\boldsymbol{V'_\ell} = \begin{cases} \boldsymbol{V_\ell} & \text{if } \ell \in L_\mathsf{t} \\ \max_{e=(\ell,g,\mathcal{Y},\ell')} \mathsf{Pre}_e(\mathsf{Guard}_g(\mathsf{Unreset}_\mathcal{Y}(\boldsymbol{V_{\ell'}}))) & \text{if } \ell \in L_\mathsf{Max} \\ \min_{e=(\ell,g,\mathcal{Y},\ell')} \mathsf{Pre}_e(\mathsf{Guard}_g(\mathsf{Unreset}_\mathcal{Y}(\boldsymbol{V_{\ell'}}))) & \text{if } \ell \in L_\mathsf{Min}\backslash L_\mathsf{t} \end{cases} \tag{7.2}
$$

where $e$ ranges over the edges in $\mathcal{G}$ that start from $\ell$. We have detailed in Lemma 7.4 how one can perform the min and max operations over partitions. Let us now focus on the $\mathsf{Guard}_g$ and $\mathsf{Unreset}_\mathcal{Y}$ operations.

**Lemma 7.5.** *Let $(P, F)$ be a piecewise affine value function of complexity at most $\langle m, \beta \rangle$. Let $g$ be a non-diagonal guard in $\mathcal{G}$. Then there exists a piecewise affine value function $(P', F')$ of complexity at most $\langle m', \beta' \rangle$, where $m' = m + 2n$ and $\beta' = \max(\beta, M)$, such that $[\![F']\!] = \mathsf{Guard}_g([\![F]\!])$.*

*Proof.* The non-diagonal guard $g$ can be encoded as a cell $I_1 \wedge \cdots \wedge I_{2n}$, with one upper and one lower inequality for each clock. We define $P'$ from $P$ with the set of affine equalities $\mathcal{E}_P \cup \{E(I_1), \ldots, E(I_{2n})\}$. It follows that each base cell of $P'$ is either entirely included in $g$ or entirely outside of it. We can thus define $F'$ appropriately, such that $[\![F']\!] = \mathsf{Guard}_e([\![F]\!])$. $\qquad\square$
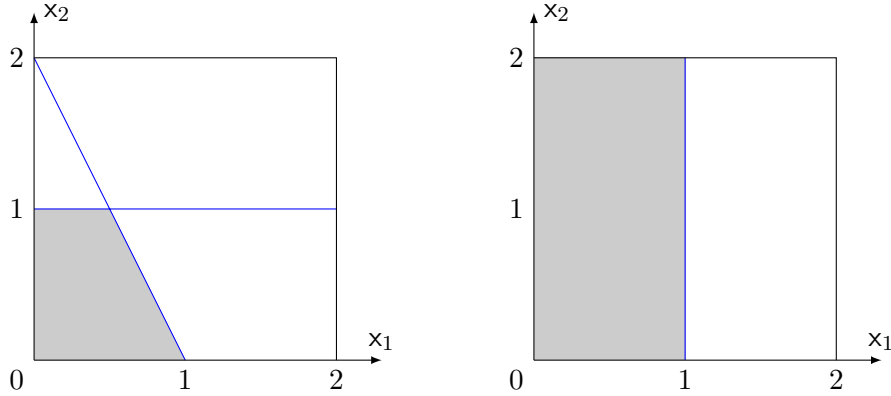


Figure 15: On the left, the partition of Figure 8. On the right, the corresponding partition obtained by applying Lemma 7.6 for reset $\mathcal{Y} = \{x_2\}$. The affine value function of the grey cell on the right is obtained from the grey cell on the left, by setting the partial derivative of $x_2$ to 0.

We continue our study with the reset of clocks, that we illustrate in Figure 15.

**Lemma 7.6.** *Let $(P, F)$ be a piecewise affine value function of complexity at most $\langle m, \beta \rangle$. Let $\mathcal{Y}$ be a set of clocks. Then there exists a piecewise affine value function $(P', F')$ of complexity at most $\langle m, \beta \rangle$ such that $[\![F']\!] = \mathsf{Unreset}_\mathcal{Y}([\![F]\!])$.*

*Proof.* If $E : a_1 x_1 + \cdots + a_n x_n + b = 0$ is an affine equality, let $\mathsf{Unreset}_\mathcal{Y}(E)$ denote the affine equality $a'_1 x_1 + \cdots + a'_n x_n + b = 0$, with $a'_i = 0$ if $x_i \in \mathcal{Y}$ and $a'_i = a_i$ otherwise, for $i \in [1, n]$. We extend this operator to affine inequalities $I$ in the same way. For each valuation $\nu \in \mathbb{R}^\mathcal{X}_{\geqslant 0}$, and $E$ an affine equality (resp. inequality) $\nu \models \mathsf{Unreset}_\mathcal{Y}(E)$ if and only if $\nu[\mathcal{Y} := 0] \models E$. Then, if $c = I_1 \wedge \cdots \wedge I_p$ is a cell, let $\mathsf{Unreset}_\mathcal{Y}(c)$ denote the cell $\mathsf{Unreset}_\mathcal{Y}(I_1) \wedge \cdots \wedge \mathsf{Unreset}_\mathcal{Y}(I_p)$. It follows that $\nu \in \mathsf{Unreset}_\mathcal{Y}(c)$ if and only if $\nu[\mathcal{Y} := 0] \in c$.

In particular, if $c$ does not intersect the sub-space where every clock in $\mathcal{Y}$ equals 0, then $\mathsf{Unreset}_{\mathcal{Y}}(c) = \emptyset$.

Similarly, if $c$ is a base cell of $P$ and $F$ maps $c$ to the affine value function $\mathsf{y} = a_1\mathsf{x}_1 + \cdots + a_n\mathsf{x}_n + b$, let $\mathsf{Unreset}_{\mathcal{Y}}(F(c))$ denote the affine function $\mathsf{y} = a'_1\mathsf{x}_1 + \cdots + a'_n\mathsf{x}_n + b$ with for $i \in [1, n]$, $a'_i = 0$ if $\mathsf{x}_i \in \mathcal{Y}$ and $a'_i = a_i$ otherwise. Then, for every $\nu \in \mathsf{Unreset}_{\mathcal{Y}}(c)$, it holds that $\boldsymbol{V}(\nu[\mathcal{Y} := 0]) = \mathsf{Unreset}_{\mathcal{Y}}(F(c))(\nu)$.

We construct $P'$ from $P$ by replacing $c_P$ by $\mathsf{Unreset}_{\mathcal{Y}}(c_P)$, and by replacing $\mathcal{E}_P = \{E_1, E_2, \ldots\}$ by $\{\mathsf{Unreset}_{\mathcal{Y}}(E_1), \mathsf{Unreset}_{\mathcal{Y}}(E_2), \ldots\}$. If $c$ is a base cell of $P$, and $\mathsf{Unreset}_{\mathcal{Y}}(c)$ is non-empty, we let $F'(c) = \mathsf{Unreset}_{\mathcal{Y}}(F(c))$. The result is a partition $P'$ with the desired complexity, and a partition value function $F'$ such that $[\![F']\!] = \mathsf{Unreset}_{\mathcal{Y}}([\![F]\!])$. $\qquad\square$

All that is left is the $\mathsf{Pre}_e$ operation. It is more challenging, and requires extra machinery related to *diagonal* behaviours that naturally arise when dealing with time-elapses.

### 7.3. Tubes and diagonals.
An affine inequality (resp. equality) is *diagonal* if the sum of its partial derivatives is null, i.e. $a_1 + \cdots + a_n = 0$. It follows that if $\nu$ satisfies a diagonal $I$ then $\nu + d \models I$ for all $d \in \mathbb{R}$. A cell is called a *tube* when all of its inequalities are diagonal. When the cell is a sub-cell of some domain $c_P$ in a partition $P$, we relax this definition slightly, to allow for non-diagonal borders inherited from $c_P$.

A *tube partition* is a partition $P$ where the set $\mathcal{E}_P$ of affine equalities is split between the set $\mathcal{E}_P^d$ of diagonal affine equalities and the set $\mathcal{E}_P^{nd}$ of non-diagonal ones. A tube partition induces a partition of $c_P$ by $\mathcal{E}_P^d$ into base cells that are tubes, called the *base tubes* of $P$.

Given two affine equalities $E : a_1\mathsf{x}_1 + \cdots + a_n\mathsf{x}_n + b = 0$ and $E' : a'_1\mathsf{x}_1 + \cdots + a'_n\mathsf{x}_n + b' = 0$, let $A = a_1 + \cdots + a_n$ and $A' = a'_1 + \cdots + a'_n$ denote the sums of their respective partial derivatives. We define their diagonal intersection as

$$E \cap_d E' : (Aa'_1 - A'a_1)\mathsf{x}_1 + \cdots + (Aa'_n - A'a_n)\mathsf{x}_n + (Ab' - A'b) = 0$$

Observe that $E \cap_d E'$ is a (possibly empty) diagonal equality that contains $E \cap E'$. Moreover, if $E$ (resp. $E'$) is diagonal then $E \cap_d E'$ is equivalent to $E$ (resp. $E'$). Now, given a cell $c = I_1 \wedge \cdots \wedge I_m$ and a set $\mathcal{E}$ of affine equalities, let $\overline{\mathcal{E}}$ denote[13] $\mathcal{E} \cup \{E(I_1), \ldots E(I_m)\}$, and let $\mathsf{Tube}(c, \mathcal{E})$ denote $\{E \cap_d E' \mid E, E' \in \overline{\mathcal{E}}\}$. The pair $(c, \mathcal{E})$ is said *atomic* if $c$ is partitioned by $\mathsf{Tube}(c, \mathcal{E})$ in only one cell (equal to $c$). Intuitively, $(c, \mathcal{E})$ is atomic if the affine equalities in $\mathcal{E}$ and in the borders of $c$ do not intersect within the smallest tube that contains $c$.

A tube partition is *atomic* if for every base tube $c$ in the partition of $c_P$ by $\mathcal{E}_P^d$, the pair $(c, \mathcal{E}_P^{nd})$ is atomic. Intuitively, this means that in the non-diagonal part of $P$, the equalities that split cells into sub-cells do not intersect within their base tube. Tube partitions can be made atomic, by introducing a bounded amount of diagonal affine equalities.

**Lemma 7.7** (Lem. 3, [1]). *Let $(P, F)$ be a piecewise affine value function of complexity at most $\langle m, \beta \rangle$. Then there exists a piecewise affine value function $(P', F')$ where $P'$ is an atomic tube partition, of complexity at most $\langle m', \beta' \rangle$, where[14] $m' = m + m^2\mathsf{Splits}(m, n)$ and $\beta' = 2n\beta^2$, such that $[\![F']\!] = [\![F]\!]$.*

*Proof.* We add a set of new diagonal equalities, that contain all equalities $E \cap_d E'$ derived from the base cells included in base tubes of $P$. Then, there are at most $m^2$ new diagonals for each base cell, and the resulting tube partition must be atomic, as any $E \cap_d E'$ belongs

---

[13]Given an affine inequality $I$, $E(I)$ denotes the associated affine equality, see page 17.

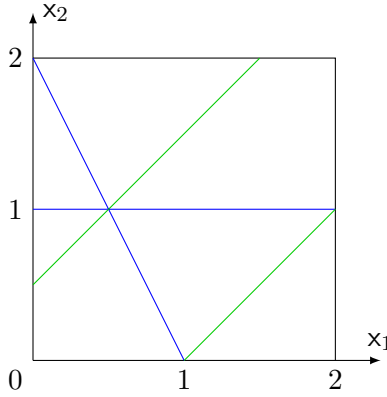[14]$\mathsf{Splits}(m, n)$ has been defined on page 18.

Figure 16: The atomic tube partition derived from the partition of Figure 8 by Lemma 7.7.

to $\overline{\mathcal{E}}$, and base cells of $P$ cannot be partitioned by $\overline{\mathcal{E}}$. As $|A|, |A'| \leqslant n\beta$, we can bound by $2n\beta^2$ the constants in inequalities $E \cap_d E'$. $\qquad\square$

Figure 16 represents the atomic tube partition $P'$ associated to the partition $P$ displayed in Figure 8. The tube partition $P'$ has 2 diagonal equalities in green, resulting in 5 base tubes. The result $P'$ is therefore an atomic tube partition of complexity $\langle 6, 2 \rangle$.[15]

We can now compute $\mathsf{Pre}_e(\boldsymbol{V}_{\ell'})$, with $\boldsymbol{V}_{\ell'}$ a value function encoded as a tube partition $(P, F)$, and $e \in E$ an edge from $\ell$ to $\ell'$, using (7.1). We will assume in the following that $\ell$ is a location of Max, but the case of Min is symmetrical. Let us fix a valuation $\nu \in [0, M)^{\mathcal{X}}$. For every delay $d \in [0, d_\nu)$, consider the term $\boldsymbol{V}_{\ell'}(\nu + d)$ of (7.1). The valuations $\nu + d$ belong to a diagonal line of $\mathbb{R}_{\geqslant 0}^{\mathcal{X}}$, and range from $\nu$ to $\nu + d_\nu$. The segment $[\nu, \nu + d_\nu]$ intersects a finite set $\mathcal{C}_\nu$ of base cells in $P$. For each such cell $c$, we isolate from the segment $[\nu, \nu + d_\nu]$ two delays:

$$d_1 = \inf\{d \in [0, d_\nu] \mid \nu + d \in c\} \quad \text{and} \quad d_2 = \sup\{d \in [0, d_\nu] \mid \nu + d \in c\}$$

As $d \mapsto \llbracket F \rrbracket(\nu + d)$ is affine over $c$, so is $d \mapsto d \cdot \mathsf{wt}(\ell) + \mathsf{wt}(e) + \llbracket F \rrbracket(\nu + d)$, thus the supremum of $d \cdot \mathsf{wt}(\ell) + \mathsf{wt}(e) + \llbracket F \rrbracket(\nu + d)$ for $\nu + d \in c$ must either be reached at (or arbitrarily close to) $d_1$, or at (or arbitrarily close to) $d_2$. Note that $\nu + d_2$ must belong to a non-diagonal border of $c$, while $\nu + d_1$ either belongs to a non-diagonal border of $c$ or equals $\nu$ (whenever $\nu \in c$). Thus, the optimal value of $d$ for evaluating the supremum must correspond to either delay 0 or to a delay leading $\nu$ to a non-diagonal border (this is also proven in [1]).

If $B$ is a non-diagonal border of $c$, and $\nu$ is a valuation of $\mathbb{R}_{\geqslant 0}^{\mathcal{X}}$, there exists a unique $d \in \mathbb{R}$ such that $\nu + d \in \llbracket B \rrbracket$. In fact, if $B$ is described by $a_1 \mathsf{x}_1 + \cdots + a_n \mathsf{x}_n + b = 0$ and $A = a_1 + \cdots + a_n$, then

$$d = -\frac{a_1 \cdot \nu(\mathsf{x}_1) + \cdots + a_n \cdot \nu(\mathsf{x}_n) + b}{A}$$

We denote this delay $d_{\nu, B}$. Observe that it must belong to $[0, d_\nu]$ as $\llbracket B \rrbracket$ is reachable from $\nu$ by time-elapse.[16]

---

[15] the size complexity includes the two borders of the domain

[16] Note that it can be equal to $d_\nu$, as $\mathsf{x} - M = 0$ is a border of the cell $[0, M)^{\mathcal{X}}$.

If $c$ is a cell of $\mathcal{C}_\nu$, let $\mathcal{B}_\nu(c)$ denote the non-diagonal borders of $c$ reachable from $\nu$ by time-elapse. The supremum $\mathsf{Pre}_e(\llbracket F \rrbracket)(\nu)$ is then equal to

$$\max\left( \llbracket F \rrbracket(\nu), \max_{c \in \mathcal{C}_\nu} \max_{B \in \mathcal{B}_\nu(c)} [d_{\nu,B} \cdot \mathsf{wt}(\ell) + \mathsf{wt}(e) + \llbracket F(c) \rrbracket(\nu + d_{\nu,B})] \right)$$

where $\llbracket F \rrbracket(\nu)$ corresponds to the delay 0, and $\llbracket F(c) \rrbracket(\nu + d_{\nu,B})$ corresponds to a jump in cell $c$ arbitrarily close to $B$.[17]

If the tube partition $(P, F)$ is atomic, it follows that every other valuation in the same cell $\nu' \in [\nu]_P$ can reach the same set of borders by time elapse, i.e. $\mathcal{C}_\nu = \mathcal{C}_{\nu'}$ and $\mathcal{B}_\nu(c) = \mathcal{B}_{\nu'}(c)$ for all $c \in \mathcal{C}_\nu$. As a result, we rename those sets $\mathcal{C}_{c'}$ and $\mathcal{B}_{c'}(c)$ if $c' = [\nu]_P$ and $c \in \mathcal{C}_{c'}$. We introduce an operator $\mathsf{Pre}_{e,c,B}$, indexed by an edge, a cell and a non-diagonal border of the cell, that maps a partition value function $F$ to the value function $\nu \mapsto d_{\nu,B} \cdot \mathsf{wt}(\ell) + \mathsf{wt}(e) + \llbracket F(c) \rrbracket(\nu + d_{\nu,B})$. As a consequence, for each base cell $c_b$ of $P$, $\mathsf{Pre}_e(\llbracket F \rrbracket)$ restricted to domain $c_b$ equals

$$\max(\llbracket F \rrbracket, \max_{c \in \mathcal{C}_{c_b}} \max_{B \in \mathcal{B}_{c_b}(c)} \mathsf{Pre}_{e,c,B}(F)) \tag{7.3}$$

**Lemma 7.8.** *Let $(P, F)$ be a piecewise affine value function of complexity at most $\langle m, \beta \rangle$, where $P$ is an atomic tube partition. Let $c_b$ be a base cell of $P$, $e$ be an edge from $\ell$ to $\ell'$, $c$ be a cell in $\mathcal{C}_{c_b}$ and $B$ be a border in $\mathcal{B}_{c_b}(c)$. Then there exists an affine value function $f$ with constants bounded by $\beta' = 4n\beta^2 \max(w_{\max}^L, w_{\max}^E)$, such that $f = \mathsf{Pre}_{e,c,B}(F)$ on $c_b$.*

*Proof.* Let $\nu$ be a valuation in $c_b$. As $B \in \mathcal{B}_{c_b}(c)$ it holds that $d_{\nu,B} \in [0, d_\nu]$, such that

$$\mathsf{Pre}_{e,c,B}(F)(\nu) = d_{\nu,B} \cdot \mathsf{wt}(\ell) + \mathsf{wt}(e) + \llbracket F(c) \rrbracket(\nu + d_{\nu,B}).$$

Let $a_\mathsf{y} \mathsf{y} = a_1 \mathsf{x}_1 + \cdots + a_n \mathsf{x}_n + b$ be the equation of $F(c)$, and let $A = a_1 + \cdots + a_n$. Let $a_1' \mathsf{x}_1 + \cdots + a_n' \mathsf{x}_n + b' = 0$ be the equation of $B$, with $A' = a_1' + \cdots + a_n' \neq 0$ the sum of its partial derivatives. We obtain the following equalities:

$$d_{\nu,B} = -\frac{a_1' \cdot \nu(\mathsf{x}_1) + \cdots + a_n' \cdot \nu(\mathsf{x}_n) + b'}{A'}$$

$$\llbracket F(c) \rrbracket(\nu + d_{\nu,B}) = \frac{a_1 \cdot \nu(\mathsf{x}_1) + \cdots + a_n \cdot \nu(\mathsf{x}_n) + A d_{\nu,B} + b}{a_\mathsf{y}}$$

$$\llbracket F(c) \rrbracket(\nu + d_{\nu,B}) \cdot A' a_\mathsf{y} = (A' a_1 - A a_1') \cdot \nu(\mathsf{x}_1) + \cdots + (A' a_n - A a_n') \cdot \nu(\mathsf{x}_n) + (A' b - A b')$$

$$\mathsf{Pre}_{e,c,B}(F)(\nu) \cdot A' a_\mathsf{y} = \sum_{i=1}^n (A' a_i - A a_i' - a_\mathsf{y} a_i' \cdot \mathsf{wt}(\ell)) \cdot \nu(\mathsf{x}_i)$$
$$+ (A' b - A b' - a_\mathsf{y} b' \cdot \mathsf{wt}(\ell) + A' a_\mathsf{y} \cdot \mathsf{wt}(e))$$

Thus $\mathsf{Pre}_{e,c,B}(F)(\nu)$ is described by an equation $a_\mathsf{y}^f \mathsf{y} = a_1^f \mathsf{x}_1 + \cdots + a_n^f \mathsf{x}_n + b^f$. The announced bound on constants holds for $n \geqslant 1$. □

Let us now bring everything together for the value iteration operator $\mathcal{F}$.

**Proposition 7.9.** *Let $\boldsymbol{V} = (P_\ell, F_\ell)_{\ell \in L}$ be a piecewise affine value function, where every $(P_\ell, F_\ell)$ has complexity at most $\langle m, \beta \rangle$, Let $q = |E|$ be the number of edges in $\mathcal{G}$. Then there exists a piecewise affine value function $\boldsymbol{V}' = (P_\ell', F_\ell')_{\ell \in L}$ such that $\boldsymbol{V}' = \mathcal{F}(\boldsymbol{V})$. In addition, every $(P_\ell', F_\ell')$ has complexity at most $\langle m', \beta' \rangle$ where $m' = \max(m, 36q^2(4m + 8n + 6)^{3n(n+2)})$ and $\beta'$ is polynomial in $n$, $\max(w_{\max}^L, w_{\max}^E)$ and $\beta$.*

---

[17]In particular, if $\nu + d_{\nu,B} \notin c$ then $F(c)$ evaluated on $\nu + d_{\nu,B}$ may not equal $\llbracket F \rrbracket(\nu + d_{\nu,B})$.

*Proof.* Fix a location $\ell \in L_{\mathsf{Min}}$. The case $L_{\mathsf{Max}}$ is symmetrical and will not be detailed. If $\ell \in L_{\mathsf{t}}$, let $(P'_\ell, F'_\ell) = (P_\ell, F_\ell)$ of complexity at most $\langle m, \beta \rangle$. Otherwise,

$$\boldsymbol{V}'_\ell = \min_{e=(\ell,g,\mathcal{Y},\ell')} \mathsf{Pre}_e(\mathsf{Guard}_g(\mathsf{Unreset}_{\mathcal{Y}}(\boldsymbol{V}_{\ell'}))).$$

For every edge $e = (\ell, g, \mathcal{Y}, \ell')$, we construct a piecewise affine value function $(P_e, F_e)$ encoding $\mathsf{Pre}_e(\mathsf{Guard}_g(\mathsf{Unreset}_{\mathcal{Y}}(\boldsymbol{V}_{\ell'})))$.

We construct from $(P_\ell, F_\ell)$, by Lemmas 7.5, 7.6 and 7.7, an atomic tube partition $(P_1, F_1)$ encoding $\mathsf{Guard}_g(\mathsf{Unreset}_{\mathcal{Y}}(\boldsymbol{V}_{\ell'}))$. Following complexity results given in these lemmas, we end up with a complexity at most $\langle m_1, \beta_1 \rangle$, such that:

$$\begin{cases} m_1 & = & (m + 2n)(1 + (m + 2n)\mathsf{Splits}(m + 2n, n)) \\ \beta_1 & = & 2n \max(\beta, M)^2 \end{cases}$$

As explained before (see equation 7.3), $\mathsf{Pre}_e(\llbracket F_1 \rrbracket)$ is obtained using computations on base cells of $P_1$ followed by a minimum with $\llbracket F_1 \rrbracket$. Given a base cell $c_b$ of $P_1$, a cell $c$ in $\mathcal{C}_{c_b}$, and a border $B$ in $\mathcal{B}_{c_b}(c)$, we can apply Lemma 7.8 to deduce the existence of an affine value function $f_{e,c,B}$ such that $f_{e,c,B} = \mathsf{Pre}_{e,c,B}(F_1)$ on $c_b$. We then have, for $\nu$ in base cell $c_b$:

$$\begin{aligned} \mathsf{Pre}_e(\llbracket F_1 \rrbracket)(\nu) &= \min(\llbracket F_1 \rrbracket(\nu), \min_{c \in \mathcal{C}_{c_b}} \min_{B \in \mathcal{B}_{c_b}(c)} \mathsf{Pre}_{e,c,B}(F_1)(\nu)) \\ &= \min(\llbracket F_1 \rrbracket(\nu), \min_{c \in \mathcal{C}_{c_b}} \min_{B \in \mathcal{B}_{c_b}(c)} f_{e,c,B}(\nu)) \end{aligned}$$

More precisely, for each base cell $c_b$ of $P_1$, we can compute the affine value function $f_{e,c,B}$ over $c_b$, with Lemma 7.8, with coefficients bounded by $\beta_2 = 4n\beta_1^2 \max(w_{\max}^L, w_{\max}^E)$. There are at most $\mathsf{Splits}(m, n)$ cells in $\mathcal{C}_{c_b}$ and at most 2 borders in $\mathcal{B}_{c_b}(c)$, thus the above formula for $\mathsf{Pre}_e(\llbracket F_1 \rrbracket)$ involves a maximum amongst $q_1 = 2\mathsf{Splits}(m_1, n) + 1$ elements. Applying Lemma 7.4, we can thus compute a piecewise affine value function $(P_{c_b}, F_{c_b})$ representing $\mathsf{Pre}_e(\llbracket F_1 \rrbracket)$ over $c_b$. Using complexity results given in Lemma 7.4, we can deduce that constant complexity is bounded by $\beta_3 = 2\beta_2^2$, and that size complexity is bounded by $m_3 = q_1 m_1 + q_1^2$.

Then, we can use all $(P_{c_b}, F_{c_b})$ to define a unique $(P_e, F_e)$, by merging all $P_{c_b}$ using the operator $\oplus$ and by setting $\llbracket F_e \rrbracket(\nu) = \llbracket F_{c_b} \rrbracket(\nu)$ with $c_b = [\nu]_{P_1}$. The resulting size complexity $m_e$ thus satisfies $m_e \leq m_3 \mathsf{Splits}(m_1, n)$, while the constant complexity is left unchanged with $\beta_e = \beta_3$.

Finally, we can apply Lemma 7.4 to compute $\min_{e=(\ell,g,\mathcal{Y},\ell')} \llbracket F_e \rrbracket$. The result is a partition value function $(P'_\ell, F'_\ell)$ encoding $\boldsymbol{V}'_\ell$. Regarding complexity, Lemma 7.4 ensure that the value function $\boldsymbol{V}'_\ell$ has complexity at most $\langle m', \beta' \rangle$, with:

$$m' = qm_e + q^2 \quad \text{and} \quad \beta' = 2\beta_e^2$$

Putting everything together, we obtain :

$$\begin{aligned} \mathsf{Splits}(m, n) &\leqslant (2m + 2)^n & \text{(Lemma 2.14)} \\ m_1 &\leqslant (2m + 4n + 2)^{n+2} & (1 \leqslant \mathsf{Splits}(m + 2n, n)) \\ \mathsf{Splits}(m_1, n) &\leqslant (4m + 8n + 6)^{n(n+2)} & ((2x^{n+2} + 2)^n \leqslant (2x + 2)^{n(n+2)}) \\ q_1 &\leqslant 3(4m + 8n + 6)^{n(n+2)} \\ m_3 &\leqslant 18(4m + 8n + 6)^{2n(n+2)} & (q_1 m_1 + q_1^2 \leqslant 2q_1^2) \\ m_e &\leqslant 18(4m + 8n + 6)^{3n(n+2)} \\ m' &\leqslant 36q^2(4m + 8n + 6)^{3n(n+2)} & (qm_e + q^2 \leqslant 2q^2 m_e) \end{aligned}$$

Regarding constant complexity, we have:

$$\begin{aligned}
\beta' = 2(\beta_e)^2 &= 2(\beta_3)^2 = 2^3\beta_2^4 \\
&= 2^3(4n\beta_1^2 \max(w_{\max}^L, w_{\max}^E))^4 && (\beta_2 = 4n\beta_1^2 \max(w_{\max}^L, w_{\max}^E)) \\
&= 2^{11}\ \max(w_{\max}^L, w_{\max}^E)^4\ n^4\ (2n\max(\beta, M)^2)^8 && (\beta_1 = 2n\max(\beta, M)^2) \\
&= 2^{19}\ \max(w_{\max}^L, w_{\max}^E)^4\ n^{12}\ \max(\beta, M)^{16}
\end{aligned}$$

$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\square$

Now that we know how to perform one step of computation of $\mathcal{F}$, we can estimate the complexity of iterated computations of this operator.

**Corollary 7.10.** *Let $\boldsymbol{V} = (P_\ell, F_\ell)_{\ell \in L}$ be a piecewise affine value function, where every $(P_\ell, F_\ell)$ has complexity at most $\langle m, \beta \rangle$. Let $q = |E|$ be the number of edges in $\mathcal{G}$. For every $i \geq 1$, we can compute a piecewise affine value function $\boldsymbol{V}^{(i)} = (P_\ell^{(i)}, F_\ell^{(i)})_{\ell \in L}$ such that $\boldsymbol{V}^{(i)} = \mathcal{F}^i(\boldsymbol{V})$. In addition, every $(P_\ell^{(i)}, F_\ell^{(i)})$ has complexity at most $\langle m^{(i)}, \beta^{(i)} \rangle$ where $m^{(i)} = \max(m, 8n + 6, 10q)^{(6n(n+2))^i}$ and $\beta^{(i)}$ is polynomial in $n$, $\max(w_{\max}^L, w_{\max}^E)$ and $\beta$.*

*Proof.* The result can be obtained by induction on $i$. The initialisation is $m \leqslant m^{(0)}$. For the induction step, by Proposition 7.9 we must show $36q^2(4m^{(i)} + 8n + 6)^{3n(n+2)} \leqslant m^{(i+1)}$. Observe that $36q^2 \leqslant (2q)^{3n(n+2)}$, and $8n+6, 10q \leqslant m^{(i)}$, so that $36q^2(4m^{(i)}+8n+6)^{3n(n+2)} \leqslant (10qm^{(i)})^{3n(n+2)} \leqslant (m^{(i)})^{6n(n+2)} \leqslant m^{(i+1)}$. This bound is crude but sufficient for our results. We note that even finer bounds on the iteration of $m \mapsto 36q^2(4m + 8n + 6)^{3n(n+2)}$ lead to doubly-exponential complexities. $\qquad\square$

Thanks to Lemma 7.3, it will follow that $\boldsymbol{V}^i$, obtained by applying $i$ times the operator $\mathcal{F}$ over $\boldsymbol{V}^0$ of bounded complexity, can be stored using a space depending polynomially on $m_0$ and $\beta_0$, exponentially on $n$, and doubly-exponentially on $i$. Moreover, the operations that we have described can be performed with a time complexity at most linear in the size of the output. This provides a doubly-exponential upper bound on the complexity (with respect to $i$) of computing $\boldsymbol{V}^i$. This concludes the proof of Theorem 7.1.

## 7.4. **Exponential vs doubly-exponential.**
Let us now discuss the result of [1], where an exponential upper bound on the bounded value problem is obtained with a non-symbolic algorithm on a slightly different setting (with non-negative weights only, without final weights, in a concurrent setting).

Whereas the concurrent setting they use generalizes ours, the sign of weights has seemingly no impact in the proofs, and the symbolic version requires minor changes related to the continuity of value functions and to the way guards are handled. These changes should not affect the complexity significantly. The main difference with their work is that their partition object is nested, it forms a tree structure where cells are partitionned into sub-cells as needed. Our techniques can be extended to nested partitions in the same way, and this has been detailed in [17, Chapter 10]. However, even with nested partitions the complexity of our techniques is still doubly-exponential, and the reason for this exponential gap is not apparent.

As a tentative answer, we make the following observation. If the game has no resets, i.e. $\mathcal{Y} = \emptyset$ on all edges, the complexity of our approach becomes exponential.[18] In [1], the way one should deal with resets is not detailed, it is therefore left open whether we could obtain an exponential bound or whether their solution is in fact doubly-exponential.

As an additional point of interest, it is shown in [17, Chapter 10.1.4] that there exists a tube partition $P$, of complexity $m$, such that if $P'$ of complexity $m'$ is obtained after applying $\mathcal{F}$, then $m' = \Theta(m^{n-1})$. This is the root of the issue, as we would need $m' = \mathcal{O}(m)$ in order to obtain an exponential bound when nesting $\mathcal{F}$.

7.5. **Bounding partial derivatives.** In the previous analysis, we explained that constants (partial derivatives and additive constants) grow polynomially at each elementary step, which is enough for an exponential upper bound (double-exponential growth of their value, stored in binary). This rough analysis will not be fine enough for some of our results. In particular, the approximation results of Section 8 will be sensitive to the partial derivatives in a linear (and not logarithmic) way. In this section, we study the growth of these partial derivatives more closely. This time, our focus will not be on the space required to store affine equations, but rather on mathematical properties of the value functions, namely their Lipschitz-continuity, closely related to bounds on partial derivatives. As a result, we revert to denoting affine equations as terms $\mathsf{y} = a_1 \mathsf{x}_1 + \cdots + a_n \mathsf{x}_n + b$ with rational constants instead of using integers with a separately stored denominator $a_\mathsf{y}$.

**Definition 7.11.** A value function $\boldsymbol{V} : L \times \mathbb{R}_{\geqslant 0}^{\mathcal{X}} \to \mathbb{R}_\infty$ is said to be $\Lambda$-*Lipschitz-continuous on regions* when, for all regions $r$, $|\boldsymbol{V}(\ell, \nu) - \boldsymbol{V}(\ell, \nu')| \leqslant \Lambda \|\nu - \nu'\|_\infty$ for all valuations $\nu, \nu' \in r$, where $\|\nu\|_\infty = \max_{\mathsf{x} \in \mathcal{X}} |\nu(\mathsf{x})|$ is the $\infty$-norm of vector $\nu \in \mathbb{R}^{\mathcal{X}}$. The function $\boldsymbol{V}$ is said to be Lipschitz-continuous on regions if it is $\Lambda$-Lipschitz-continuous on regions, for some $\Lambda$.

Since final weight functions $\mathsf{fwt}$ are piecewise affine and continuous on regions, they are Lipschitz-continuous on regions, so is $\boldsymbol{V}^0$. We will maintain as an invariant that $\boldsymbol{V}^i$ is Lipschitz-continuous on regions:

**Proposition 7.12.** *If every final weight in a WTG $\mathcal{G}$ is $\Lambda$-Lipschitz-continuous on regions (and piecewise affine), then $\mathsf{Val}_{\mathcal{G}}^i$ is $\Lambda\Lambda'$-Lipschitz-continuous on regions, with $\Lambda'$ polynomial in $w_{\max}^L$ and $n = |\mathcal{X}|$, and exponential in $i$.*

Note that for a piecewise affine function with finitely many pieces, being $\Lambda$-Lipschitz-continuous on regions is equivalent to being continuous on regions and having all partial derivatives bounded by $\Lambda$ in absolute value. The rest of this section is dedicated to proving Proposition 7.12, as a corollary of the following more precise result.

**Lemma 7.13.** *If for all $\ell \in L$, $\boldsymbol{V}_\ell$ is piecewise affine with finitely many pieces that have all their partial derivatives bounded by $\Lambda$ in absolute value, then for all $\ell \in L$, $\mathcal{F}(\boldsymbol{V})_\ell$ is continuous on regions and piecewise affine with partial derivatives bounded by $\max(\Lambda, |\mathsf{wt}(\ell)| + (n-1)\Lambda)$ in absolute value.*

*Proof.* We will show that for every region $r$, $\mathcal{F}(\boldsymbol{V})$ restricted to $r$ has those properties. Note that they are transmitted over finite min and max operations. The continuity on regions is easy to prove because it is stable by infimum and supremum. There exists a partition

---

[18]This requires nested partitions and a more involved complexity analysis detailed in [17, Chapter 10]. In this case, the $\mathsf{Unreset}_\mathcal{Y}$ steps can be skipped, and in Proposition 7.9 we end up with a *linear* growth of the complexity with respect to $m$, instead of *polynomial*.

value function $(P_\ell, F_\ell)$ for each $\ell \in L$ that represents $\boldsymbol{V}$. As explained before, a crucial property is that, for a given valuation $\nu$, the delays $d$ that need to be considered in the infimum and supremum of $\mathcal{F}(\boldsymbol{V})_\ell(\nu)$ correspond to the intersection points of the diagonal half line containing the time successors of $\nu$ and borders of cells (if $\nu^b$ is such a valuation, $d = \|\nu^b - \nu\|_\infty$ is the associated delay). In particular, there is a finite number of such borders, and the final $\mathcal{F}(\boldsymbol{V})_\ell$ function can be written as a finite nesting of finite min and max operations over affine terms, each corresponding to a choice of delay and an edge to take. Formally, there are several cases to consider to define those terms, depending on delay and edge choices. For each available edge $e$, those terms can either be:

• If a delay 0 is taken and all clocks in $\mathcal{Y} \subseteq \mathcal{X}$ are reset by $e$, then

$$\mathsf{wt}_\Sigma((\ell, \nu) \xrightarrow{0} (\ell, \nu) \xrightarrow{e} (\ell', \nu[\mathcal{Y} := 0])) = \mathsf{wt}_\Sigma(e) + \boldsymbol{V}_{\ell'}(\nu[\mathcal{Y} := 0])$$

• If a delay $d > 0$ (leading to valuation $\nu^b$ on border $B$) is taken and all clocks in $\mathcal{Y} \subseteq \mathcal{X}$ are reset by $e$, then

$$\mathsf{wt}_\Sigma((\ell, \nu) \xrightarrow{d} (\ell, \nu^b) \xrightarrow{e} (\ell', \nu^b[\mathcal{Y} := 0])) = \mathsf{wt}_\Sigma(\ell) \cdot d + \mathsf{wt}_\Sigma(e) + \boldsymbol{V}_{\ell'}(\nu^b[\mathcal{Y} := 0])$$
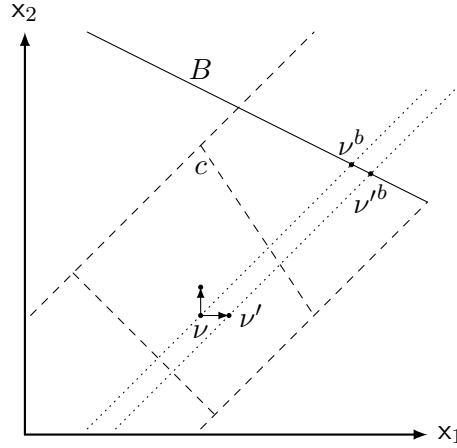


Figure 17: A cell $c$ as described in the proof of Lemma 7.13. Dashed lines are borders of $c$, dotted lines are proof constructions.

In the first case, the resulting partial derivatives are 0 for clocks in $\mathcal{Y}$, and the same as the partial derivatives in $\boldsymbol{V}_{\ell'}$ for all other clocks, which allows us to conclude that they are bounded by $\Lambda$. We now consider the second case. We argue that it can be decomposed as a delay followed by an edge of the first case, meaning that we can assume $\mathcal{Y} = \emptyset$ without loss of generality.

There are again two cases: the border $B$ being inside a region or on the frontier of a region.

If the border is not the frontier of a region, it is the intersection points of two affine pieces of $\boldsymbol{V}_{\ell'}$ whose equations (in the space $\mathbb{R}^{n+1}$ whose $n$ first coordinates are the clocks $(\mathsf{x}_1, \ldots, \mathsf{x}_n)$ and the last coordinate $\mathsf{y}$ corresponds to the value $\boldsymbol{V}_{\ell'}(\mathsf{x}_1, \ldots, \mathsf{x}_n)$) can be written $\mathsf{y} = a_1\mathsf{x}_1 + \cdots + a_n\mathsf{x}_n + b$ (before the border) and $\mathsf{y} = a'_1\mathsf{x}_1 + \cdots + a'_n\mathsf{x}_n + b'$ (after the border). Therefore, valuations at the borders all fulfill the equation

$$(a'_1 - a_1)\mathsf{x}_1 + \cdots + (a'_n - a_n)\mathsf{x}_n + b' - b = 0 \tag{7.4}$$

We let $A = (a_1' - a_1) + \cdots + (a_n' - a_n)$. Consider that $\ell$ is a location of $\mathsf{Min}$ (the very same reasoning applies to the case of a location of $\mathsf{Max}$). Since $\mathcal{F}$ computes an infimum, we know that the function mapping the delay $d$ to the weight obtained from reaching $\nu + d$ is decreasing before the border and increasing after. These functions are locally affine which implies that their slopes verify:

$$\mathsf{wt}(\ell) + a_1 + \cdots + a_n \leqslant 0 \quad \text{and} \quad \mathsf{wt}(\ell) + a_1' + \cdots + a_n' \geqslant 0 \tag{7.5}$$

We deduce from these two inequalities that $A \geqslant 0$. The case where $A = 0$ would correspond to the case where the border contains a diagonal line, which is forbidden, and thus $A > 0$. Consider now a valuation $\nu$ and another valuation $\nu'$ of coordinates $(\nu(\mathsf{x}_1), \ldots, \nu(\mathsf{x}_{k-1}), \nu(\mathsf{x}_k) + \lambda, \nu(\mathsf{x}_{k+1}), \ldots, \nu(\mathsf{x}_n))$ with $\lambda$ small enough. The delays $d$ and $d'$ needed to arrive to the border starting from these two valuations are such that $\nu + d$ and $\nu' + d'$ both verify (7.4). We can then deduce

$$d' - d = \lambda \frac{a_k - a_k'}{A}$$

It is now possible to compute the partial derivative of $\mathcal{F}(\boldsymbol{V})_\ell$ in the $k$-th coordinate using the limit when $\lambda$ tends to $0$ of the quotient

$$\frac{\mathcal{F}(\boldsymbol{V})_{\ell,\nu'} - \mathcal{F}(\boldsymbol{V})_{\ell,\nu}}{\lambda} = \frac{\mathsf{wt}(\ell)(d' - d) + \boldsymbol{V}_{\ell',\nu'+d'} - \boldsymbol{V}_{\ell',\nu+d}}{\lambda} \tag{7.6}$$

We may compute it by using the equations of the affine pieces before or after the border. We thus obtain

$$\frac{\mathcal{F}(\boldsymbol{V})_{\ell,\nu'} - \mathcal{F}(\boldsymbol{V})_{\ell,\nu}}{\lambda} = \frac{a_k - a_k'}{A} \big( \mathsf{wt}(\ell) + a_1 + \cdots + a_n \big) + a_k$$

$$\frac{\mathcal{F}(\boldsymbol{V})_{\ell,\nu'} - \mathcal{F}(\boldsymbol{V})_{\ell,\nu}}{\lambda} = \frac{a_k - a_k'}{A} \big( \mathsf{wt}(\ell) + a_1' + \cdots + a_n' \big) + a_k'$$

In the case where $a_k \geqslant a_k'$, the first equation, with (7.5), allows us to obtain that the quotient (and thus the partial derivative) is at most $a_k$. The second equation, with (7.5), allows us to obtain that the partial derivative is at least $a_k'$. In case $a_k' \geqslant a_k$, we obtain similarly that the partial derivative is at most $a_k'$ and at least $a_k$. Since $a_k$ and $a_k'$ are bounded in absolute value by $\Lambda$, so is the partial derivative.

We now come back to the case where the border is on the frontier of a region. Then, it is a segment of a line of equation $\mathsf{x}_k = c$ for some $k$ and $c$. Moreover, $\boldsymbol{V}_{\ell'}$ contains at most three values for points of $B$: the limit coming from before the border, the value at the border, and the limit coming from after the border. The computation of $\mathcal{F}(\boldsymbol{V})$ considers values obtained from all three and takes the minimum (or maximum).

Now, let $\mathsf{y} = a_1 \mathsf{x}_1 + \cdots + a_n \mathsf{x}_n + b$ be the equation defining the affine piece of $\boldsymbol{V}_{\ell'}$ before the border (resp. at the border, after the border). Consider a valuation $\nu$ and another valuation $\nu'$ of coordinates $(\nu(\mathsf{x}_1), \ldots, \nu(\mathsf{x}_{j-1}), \nu(\mathsf{x}_j) + \lambda, \nu(\mathsf{x}_{j+1}), \ldots, \nu(\mathsf{x}_n))$ with $\lambda$ small enough. The delays $d$ and $d'$ needed to arrive to the border starting from these two valuations are such that $\nu + d$ and $\nu' + d'$ both verify $\mathsf{x}_k = c$. We can then deduce that $d' - d = 0$ if $j \neq k$ and $d' - d = -\lambda$ if $j = k$. It is now possible to compute the partial derivative of $\mathcal{F}(\boldsymbol{V})_\ell$ in the $j$-th coordinate using (7.6) again. We may compute it by using the equations of the affine

piece before the border (resp. at the border, after the border). Then,

$$\boldsymbol{V}_{\ell',\nu+d} = a_1(\mathsf{x}_1 + d) + \cdots + a_n(\mathsf{x}_n + d) + b = \sum_{i=1, i\neq k}^{n} a_i(\mathsf{x}_i + d) + a_k c + b$$

$$\boldsymbol{V}_{\ell',\nu'+d'} = \sum_{i=1, i\neq k}^{n} a_i(\mathsf{x}_i + d') + a_k c + b$$

We thus obtain

$$\frac{\mathcal{F}(\boldsymbol{V})_{\ell,\nu'} - \mathcal{F}(\boldsymbol{V})_{\ell,\nu}}{\lambda} = \begin{cases} a_j & \text{if } j \neq k \\ -\mathsf{wt}(\ell) - \sum_{i=1, i\neq k}^{n} a_i & \text{otherwise} \end{cases}$$

Then, the partial derivatives are bounded, in absolute value, by $|\mathsf{wt}(\ell)| + (n-1)\Lambda$. □

As a corollary, we obtain Proposition 7.12.

## 8. Computing values

In this section we conclude the proofs of Theorems 3.3 and 3.8, with a triply-exponential upper bound on computing (resp. approximating) values in divergent (resp. almost-divergent) weighted timed games.

These upper bounds are obtained by computations performed on the semi-unfolding $\mathcal{T}(\mathcal{G})$ described in Section 6. Whereas the computation is direct for divergent WTGs, we then extend it to almost-divergent WTGs by first explaining how to compute value approximations in kernels.

8.1. **Divergent WTGs.** We first explain how to compute the values of a divergent WTG $\mathcal{G}$, thus proving Theorem 3.3. By definition, in such a divergent WTG, there are no 0-cycles, and thus the kernel $\mathsf{K}$ is empty. In this case, the semi-unfolding $\mathcal{T}(\mathcal{G})$ is a tree of depth $i$ exponential in $\mathcal{G}$, and thus of doubly-exponential size. Proposition 6.1 ensures that the values of $\mathcal{T}(\mathcal{G})$ coincide with the values of $\mathcal{G}$. By Theorem 7.1, we can compute the (exact) values in $\mathcal{T}(\mathcal{G})$ in time doubly-exponential in $i$ and exponential in the size of $\mathcal{T}(\mathcal{G})$. We thus obtain a triply-exponential algorithm computing the value of a divergent WTG. Equivalently, this shows that the value problem is in 3-EXPTIME for divergent WTGs.

8.2. **Approximation of kernels.** To generalise our computations to almost-divergent WTGs, we start by approximating a kernel $\mathsf{K}$ of a WTG $\mathcal{G}$ by extending the region-based approximation schema of [10]. More precisely, we thus consider here a WTG $\mathcal{G}$ composed of a kernel (a subgraph of a region game containing only 0-cycles, as defined in Section 5.2) and some target locations equipped with final weights. In the setting of [10], not only cycles but all finite plays in kernels have weight 0, allowing for a reduction to a finite game. In our setting, we have to approximate the timed dynamics of runs, and therefore resort to the corner-point abstraction (as shown to the right of Figure 11).

Our goal is to compute an $\varepsilon$-approximation of the value of the kernel (in a given initial configuration). Since final weight functions are piecewise affine with a finite number of pieces and continuous on regions, they are $\Lambda$-Lipschitz-continuous on regions, for a given constant

$\Lambda \geqslant 0$. The technique to obtain the approximation is to consider regions of a refined enough granularity: we thus pick

$$N(\varepsilon, \Lambda) = \left\lceil \frac{w_{\max}^L |L| |\mathsf{Reg}(\mathcal{X}, M)| + \Lambda}{\varepsilon} \right\rceil \tag{8.1}$$

later denoted $N$ when the parameters $\varepsilon$ and $\Lambda$ are clear from context.

Consider then the corner-point abstraction $\Gamma_N(\mathcal{G})$ described in Section 2.2, with locations of the form $(\ell, r, \mathsf{v})$ such that $\mathsf{v}$ is a corner of the $1/N$-region $r$. Two plays $\rho$ of $\mathcal{G}$ and $\rho'$ of $\Gamma_N(\mathcal{G})$ are said to be $1/N$-*close* if they follow the same path $\mathsf{p}$ in $\mathcal{R}_N(\mathcal{G})$. In particular, at each step the configurations $(\ell, \nu)$ in $\rho$ and $(\ell', r', \mathsf{v}')$ in $\rho'$ (with $\mathsf{v}'$ a corner of the $1/N$-region $r'$) satisfy $\ell = \ell'$ and $\nu \in r'$, and the edges taken in both plays have the same discrete weights. Close plays have *close* weights, in the following sense:

**Lemma 8.1.** *For all $1/N$-close plays $\rho$ of $\mathcal{G}$ and $\rho'$ of $\Gamma_N(\mathcal{G})$,*

$$|\mathsf{wt}_{\mathcal{G}}(\rho) - \mathsf{wt}_{\Gamma_N(\mathcal{G})}(\rho')| \leqslant \varepsilon$$

*Proof.* Since $\rho$ and $\rho'$ encounter the same locations of $\mathcal{G}$, one reaches a target location if and only if the other does. In the case where they do not reach a target location, both weights are infinite, and thus equal. We now look at the case where both plays reach a target location, moreover in the same step.

Consider the region path $\mathsf{p}$ of the run $\rho$: it can be decomposed into a simple path with maximal cycles in it. The number of such maximal cycles is bounded by $|L \times \mathsf{Reg}(\mathcal{X}, M)|$ and the remaining simple path has length at most $|L \times \mathsf{Reg}(\mathcal{X}, M)|$. Since all cycles of a kernel are 0-cycles, the parts of $\rho$ that follow the maximal cycles have weight exactly 0.

Consider the same decomposition for the play $\rho'$. Cycles of $\mathsf{p}$ do not necessarily map to cycles over locations of $\Gamma_N(\mathcal{G})$, since the $1/N$-regions could be distinct. However, Lemma 2.10 shows that, for all those cycles of $\mathsf{p}$, there exists a sequence of finite plays of $\mathcal{G}$ whose weight tends to the weight of $\rho'$. Since all those finite plays follow a cycle of the region game $\mathcal{R}(\mathcal{G})$ (with $\mathcal{G}$ being a kernel), they all have weight 0. Hence, the parts of $\rho'$ that follow the maximal cycles of $\mathsf{p}$ have also weight exactly 0.

Therefore, the difference $|\mathsf{wt}_{\mathcal{G}}(\rho) - \mathsf{wt}_{\Gamma_N(\mathcal{G})}(\rho')|$ is concentrated on the remaining simple path of $\mathsf{p}$: on each edge of this path, the maximal weight difference is $1/N \times w_{\max}^L$ since $1/N$ is the largest difference possible in time delays between plays that stay $1/N$-close (since they stay in the same $1/N$-regions). Moreover, the difference between the final weight functions is bounded by $\Lambda/N$, since the final weight function $\mathsf{fwt}$ is $\Lambda$-Lipschitz-continuous and the final weight function of $\Gamma_N(\mathcal{G})$ is obtained as limit of $\mathsf{fwt}$. Summing the two contributions, we obtain as upper bound the constant $(w_{\max}^L |L| |\mathsf{Reg}(\mathcal{X}, M)| + \Lambda)/N \leqslant \varepsilon$. $\square$

In particular, if we start in some configurations $(\ell, \nu)$ of $\mathcal{G}$, and $((\ell, r, \mathsf{v}), \mathsf{v})$ of $\Gamma_N(\mathcal{G})$, with $\nu \in r$, since both players have the ability to stay $1/N$-close all along the plays, a bisimulation argument allows us to obtain that the values of the two games are also close in $(\ell, \nu)$ and $((\ell, r, \mathsf{v}), \mathsf{v})$, respectively:

**Lemma 8.2.** *For all locations $\ell \in L$, $1/N$-regions $r$, valuations $\nu \in r$ and corners $\mathsf{v}$ of $r$,*

$$|\mathsf{Val}_{\mathcal{G}}(\ell, \nu) - \mathsf{Val}_{\Gamma_N(\mathcal{G})}((\ell, r, \mathsf{v}), \mathsf{v})| \leqslant \varepsilon$$

*Proof.* To obtain the result, we prove that

$$\mathsf{Val}_{\mathcal{G}}(\ell, \nu) \leqslant \mathsf{Val}_{\Gamma_N(\mathcal{G})}((\ell, r, \mathsf{v}), \mathsf{v}) + \varepsilon \quad \text{and} \quad \mathsf{Val}_{\Gamma_N(\mathcal{G})}((\ell, r, \mathsf{v}), \mathsf{v}) \leqslant \mathsf{Val}_{\mathcal{G}}(\ell, \nu) + \varepsilon$$

By definition and determinacy of turn-based WTGs, this is equivalent to proving these two inequalities:

$$\inf_{\sigma_{\mathsf{Min}}} \sup_{\sigma_{\mathsf{Max}}} \mathsf{wt}_{\mathcal{G}}(\mathsf{play}((\ell,\nu),\sigma_{\mathsf{Min}},\sigma_{\mathsf{Max}})) \leqslant \inf_{\sigma'_{\mathsf{Min}}} \sup_{\sigma'_{\mathsf{Max}}} \mathsf{wt}_{\Gamma_N(\mathcal{G})}(\mathsf{play}(((\ell,r,\mathsf{v}),\mathsf{v}),\sigma'_{\mathsf{Min}},\sigma'_{\mathsf{Max}})) + \varepsilon$$

$$\sup_{\sigma'_{\mathsf{Max}}} \inf_{\sigma'_{\mathsf{Min}}} \mathsf{wt}_{\Gamma_N(\mathcal{G})}(\mathsf{play}(((\ell,r,\mathsf{v}),\mathsf{v}),\sigma'_{\mathsf{Min}},\sigma'_{\mathsf{Max}})) \leqslant \sup_{\sigma_{\mathsf{Max}}} \inf_{\sigma_{\mathsf{Min}}} \mathsf{wt}_{\mathcal{G}}(\mathsf{play}((\ell,\nu),\sigma_{\mathsf{Min}},\sigma_{\mathsf{Max}})) + \varepsilon$$

To show the first inequality, it suffices to show that for all $\sigma'_{\mathsf{Min}}$, there exists $\sigma_{\mathsf{Min}}$ such that for all $\sigma_{\mathsf{Max}}$, there is $\sigma'_{\mathsf{Max}}$ verifying

$$|\mathsf{wt}_{\mathcal{G}}(\mathsf{play}((\ell,\nu),\sigma_{\mathsf{Min}},\sigma_{\mathsf{Max}})) - \mathsf{wt}_{\Gamma_N(\mathcal{G})}(\mathsf{play}(((\ell,r,\mathsf{v}),\mathsf{v}),\sigma'_{\mathsf{Min}},\sigma'_{\mathsf{Max}}))| \leqslant \varepsilon \qquad (8.2)$$

For the second, it suffices to show that for all $\sigma'_{\mathsf{Max}}$, there exists $\sigma_{\mathsf{Max}}$ such that for all $\sigma_{\mathsf{Min}}$, there is $\sigma'_{\mathsf{Min}}$ verifying the same equation (8.2). We will detail the proof for the first, the second being syntactically the same, with both players swapped.

Equation (8.2) can be obtained from Lemma 8.1, under the condition that the plays $\mathsf{play}((\ell,\nu),\sigma_{\mathsf{Min}},\sigma_{\mathsf{Max}})$ and $\mathsf{play}(((\ell,r,\mathsf{v}),\mathsf{v}),\sigma'_{\mathsf{Min}},\sigma'_{\mathsf{Max}})$ are $1/N$-close. Therefore, we fix a strategy $\sigma'_{\mathsf{Min}}$ of $\mathsf{Min}$ in the game $\Gamma_N(\mathcal{G})$, and we construct a strategy $\sigma_{\mathsf{Min}}$ of $\mathsf{Min}$ in $\mathcal{G}$, as well as two mappings $f\colon \mathsf{FPlays}_{\mathcal{G}}^{\mathsf{Min}} \to \mathsf{FPlays}_{\Gamma_N(\mathcal{G})}^{\mathsf{Min}}$ and $g\colon \mathsf{FPlays}_{\Gamma_N(\mathcal{G})}^{\mathsf{Max}} \to \mathsf{FPlays}_{\mathcal{G}}^{\mathsf{Max}}$ such that:

- for all $\rho \in \mathsf{FPlays}_{\mathcal{G}}^{\mathsf{Min}}$, $\rho$ and $f(\rho)$ are $1/N$-close, and if $\rho$ is consistent with $\sigma_{\mathsf{Min}}$ and starts in $(\ell,\nu)$, then $f(\rho)$ is consistent with $\sigma'_{\mathsf{Min}}$ and starts in $((\ell,r,\mathsf{v}),\mathsf{v})$;
- for all $\rho' \in \mathsf{FPlays}_{\Gamma_N(\mathcal{G})}^{\mathsf{Max}}$, $g(\rho')$ and $\rho'$ are $1/N$-close, and if $\rho'$ is consistent with $\sigma'_{\mathsf{Min}}$ and starts in $((\ell,r,\mathsf{v}),\mathsf{v})$, then $g(\rho')$ is consistent with $\sigma_{\mathsf{Min}}$ and starts in $(\ell,\nu)$.

We build $\sigma_{\mathsf{Min}}$, $f$, and $g$ by induction on the length $k$ of plays, over prefixes of plays of length $k-1$, $k$ and $k$, respectively. For $k = 0$ (plays of length 0 are those restricted to a single configuration), we let $f(\ell,\nu) = ((\ell,r,\mathsf{v}),\mathsf{v})$ and $g((\ell,r,\mathsf{v}),\mathsf{v}) = (\ell,\nu)$, leaving the other values arbitrary (since we will not use them).

Then, we suppose $\sigma_{\mathsf{Min}}$, $f$, and $g$ built until length $k-1$, $k$ and $k$, respectively (if $k = 0$, $\sigma_{\mathsf{Min}}$ has not been build yet), and we define them on plays of length $k$, $k+1$ and $k+1$, respectively. For every $\rho \in \mathsf{FPlays}_{\mathcal{G}}^{\mathsf{Min}}$ of length $k$, we note $\rho' = f(\rho)$. Consider the decision $(d',e') = \sigma'_{\mathsf{Min}}(\rho')$ and $\rho'_+$ the prefix $\rho'$ extended with the decision $(d',e')$. By timed bisimulation, there exists $(d,e)$ such that the prefix $\rho_+$ composed of $\rho$ extended with the decision $(d,e)$ builds $1/N$-close plays $\rho_+$ and $\rho'_+$. We let $\sigma_{\mathsf{Min}}(\rho) = (d,e)$. If $\rho_+ \in \mathsf{FPlays}_{\mathcal{G}}^{\mathsf{Min}}$, we also let $f(\rho_+) = \rho'_+$, and otherwise we let $g(\rho'_+) = \rho_+$. Symmetrically, consider $\rho' \in \mathsf{FPlays}_{\Gamma_N(\mathcal{G})}^{\mathsf{Max}}$ of length $k$, and $\rho = g(\rho')$. For all possible decisions $(d',e')$, by timed bisimulation, there exists a decision $(d,e)$ in the prefix $\rho$ such that the respective extended plays $\rho'_+$ and $\rho_+$ are $1/N$-close. We then let $g(\rho'_+) = \rho_+$ if $\rho_+ \in \mathsf{FPlays}_{\mathcal{G}}^{\mathsf{Max}}$ and $f(\rho_+) = \rho'_+$ otherwise. We extend the definition of $f$ and $g$ arbitrarily for other prefixes of plays. The properties above are then trivially verified.

We then fix a strategy $\sigma_{\mathsf{Max}}$ of $\mathsf{Max}$ in the game $\mathcal{G}$, which determines a unique play $\mathsf{play}((\ell,\nu),\sigma_{\mathsf{Min}},\sigma_{\mathsf{Max}})$. We construct a strategy $\sigma'_{\mathsf{Max}}$ of $\mathsf{Max}$ in the game $\Gamma_N(\mathcal{G})$ by building the unique play $\mathsf{play}(((\ell,r,\mathsf{v}),\mathsf{v}),\sigma'_{\mathsf{Min}},\sigma'_{\mathsf{Max}})$ we will be interested in, such that each of its prefixes is in relation, via $f$ or $g$, to the associated prefix of $\mathsf{play}((\ell,\nu),\sigma_{\mathsf{Min}},\sigma_{\mathsf{Max}})$. Thus, we only need to consider a prefix of play $\rho' \in \mathsf{FPlays}_{\Gamma_N(\mathcal{G})}^{\mathsf{Max}}$ that starts in $((\ell,r,\mathsf{v}),\mathsf{v})$ and is consistent with $\sigma'_{\mathsf{Min}}$, and $\sigma'_{\mathsf{Max}}$ built so far. Consider the play $\rho = g(\rho')$, starting in $(\ell,\nu)$ and consistent with $\sigma_{\mathsf{Min}}$, and $\sigma_{\mathsf{Max}}$ (by assumption). For the decision $(d,e) = \sigma_{\mathsf{Max}}(\rho)$ (letting $\rho_+$ be the extended prefix), the definition of $f$ and $g$ ensures that there exists a decision

$(d', e')$ after $\rho'$ that results in an extended play $\rho'_+$ that is $1/N$-close, via $f$ or $g$, with $\rho_+$. We thus can choose $\sigma'_{\mathsf{Max}}(\rho') = (d', e')$.

We finally have built two plays $\mathsf{play}((\ell, \nu), \sigma_{\mathsf{Min}}, \sigma_{\mathsf{Max}})$ and $\mathsf{play}((\ell', \nu'), \sigma'_{\mathsf{Min}}, \sigma'_{\mathsf{Max}})$ that are $1/N$-close, as needed, which concludes this proof. $\qquad\square$

We can thus obtain an $\varepsilon$-approximation of $\mathsf{Val}_{\mathcal{G}}(\ell, \nu)$ by computing $\mathsf{Val}_{\Gamma_N(\mathcal{G})}((\ell, r, \mathsf{v}), \mathsf{v})$ for any corner $\mathsf{v}$ of $r$. Recall that $\Gamma_N(\mathcal{G})$ can be considered as an untimed weighted game (with reachability objective). Thus we can apply the result of [15], where it is shown that the optimal values of such games can be computed in pseudo-polynomial time (i.e. polynomial with respect to the number of locations, at most $(|\mathcal{X}| + 1)|L||\mathsf{Reg}_N(\mathcal{X}, M)|$, and the weights of transitions $w_{\max}$ encoded in unary, instead of binary): more precisely, the value $\mathsf{Val}_{\Gamma_N(\mathcal{G})}$ is obtained using the operator $\mathcal{F}$ of (2.2) (page 16), as the $i$-th iteration with $i = ((2(|\mathcal{X}| + 1)|L||\mathsf{Reg}_N(\mathcal{X}, M)| - 1)w_{\max} + 1)(|\mathcal{X}| + 1)|L||\mathsf{Reg}_N(\mathcal{X}, M)|$, polynomial in $|L|$, $w_{\max}$, $M$ and $N$, and exponential in $|\mathcal{X}|$. We then define an $\varepsilon$-approximation of $\mathsf{Val}_{\mathcal{G}}$, named $\mathsf{Val}'_N$, on each $1/N$-region by interpolating the values of its $1/N$-corners in $\Gamma_N(\mathcal{G})$ with a piecewise affine function: If $\nu$ is a valuation that belongs to the $1/N$-region $r$, then $\nu$ can be expressed as a (unique) convex combination of the $1/N$-corners $\mathsf{v}$ of $r$, so that $\nu = \sum_{\mathsf{v}} \alpha_{\mathsf{v}} \mathsf{v}$ with $\alpha_{\mathsf{v}} \in [0, 1]$ for all $\mathsf{v}$, and we let $\mathsf{Val}'_N(\ell, \nu) = \sum_{\mathsf{v}} \alpha_{\mathsf{v}} \mathsf{Val}_{\Gamma_N(\mathcal{G})}((\ell, r, \mathsf{v}), \mathsf{v})$ for all locations $\ell$ of $\mathcal{G}$.

Moreover, we can control the growth of the Lipschitz constant of the approximated value for further use.

**Lemma 8.3.** $\mathsf{Val}'_N$ *is an $\varepsilon$-approximation of* $\mathsf{Val}_{\mathcal{G}}$, *i.e.* $\|\mathsf{Val}'_N - \mathsf{Val}_{\mathcal{G}}\|_\infty \leqslant \varepsilon$. *Moreover,* $\mathsf{Val}'_N$ *is piecewise affine with a finite number of pieces and* $2(w_{\max}^L |L||\mathsf{Reg}(\mathcal{X}, M)| + \Lambda)$-*Lipschitz-continuous over regions.*

*Proof.* By construction, the approximated value is piecewise affine with one piece per $1/N$-region. To prove the Lipschitz constant, it is then sufficient to bound the difference between $\mathsf{Val}_{\Gamma_N(\mathcal{G})}((\ell, r, \mathsf{v}), \mathsf{v})$ and $\mathsf{Val}_{\Gamma_N(\mathcal{G})}((\ell, r, \mathsf{v}'), \mathsf{v}')$, for $\mathsf{v}$ and $\mathsf{v}'$ two corners of a $1/N$-region $r$. We can pick any valuation $\nu$ in $r$ and apply Lemma 8.2 twice, between $\nu$ and $\mathsf{v}$, and between $\nu$ and $\mathsf{v}'$. We obtain $|\mathsf{Val}_{\Gamma_N(\mathcal{G})}((\ell, r, \mathsf{v}), \mathsf{v}) - \mathsf{Val}_{\Gamma_N(\mathcal{G})}((\ell, r, \mathsf{v}'), \mathsf{v}')| \leqslant 2(w_{\max}^L |L||\mathsf{Reg}(\mathcal{X}, M)| + \Lambda)/N = 2(w_{\max}^L |L||\mathsf{Reg}(\mathcal{X}, M)| + \Lambda)\|\mathsf{v} - \mathsf{v}'\|_\infty.$ $\qquad\square$

The time complexity needed to compute $\mathsf{Val}'_N$ is polynomial in $|L|$, $w_{\max}$, $M$ and $N$, and exponential in $|\mathcal{X}|$.

### 8.3. Approximation of almost-divergent WTGs.

We now explain how to approximate the value of an almost-divergent WTG $\mathcal{G}$, thus proving Theorem 3.8. After having computed the semi-unfolding $\mathcal{T}(\mathcal{G})$ described in Section 6, we perform a bottom-up computation of the approximation. The addition of kernels (with respect to the case of divergent WTGs studied before) requires us to compute an approximation instead of the actual value. Notice that the techniques used in Theorem 7.1, applied for $i = 1$ step (and not for the whole tree as for divergent WTGs), allow us to compute the value of a non-kernel node of $\mathcal{T}(\mathcal{G})$, depending on the values of its children. There is no approximation needed here, so that if we have computed $\varepsilon$-approximations of all its children, we can compute an $\varepsilon$-approximation of the node. More formally, this is justified by the following lemma with $i = 1$:

**Lemma 8.4.** *Let $\mathcal{G}$ and $\mathcal{G}'$ be two WTGs that only differ on the final weight functions* $\mathsf{fwt}$ *and* $\mathsf{fwt}'$. *Then,* $\|\mathsf{Val}_{\mathcal{G}} - \mathsf{Val}_{\mathcal{G}'}\|_\infty \leqslant \|\mathsf{fwt} - \mathsf{fwt}'\|_\infty$. *Moreover, for all $i \in \mathbb{N}$,* $\|\mathsf{Val}_{\mathcal{G}}^i - \mathsf{Val}_{\mathcal{G}'}^i\|_\infty \leqslant \|\mathsf{fwt} - \mathsf{fwt}'\|_\infty$.

*Proof.* Consider two strategies $\sigma_{\mathsf{Min}}$ and $\sigma_{\mathsf{Max}}$ for both players in $\mathcal{G}$ and $\mathcal{G}'$ (since both games are identical up-to the final weight functions, they share the same sets of strategies for both players). Then, from an initial configuration $(\ell, \nu)$, the plays $\mathsf{play}_{\mathcal{G}}((\ell, \nu), \sigma_{\mathsf{Min}}, \sigma_{\mathsf{Max}})$ and $\mathsf{play}_{\mathcal{G}'}((\ell, \nu), \sigma_{\mathsf{Min}}, \sigma_{\mathsf{Max}})$ are the same, and their weight only differ by the final weight functions taken at the same configurations. Thus,

$$|\mathsf{wt}(\mathsf{play}_{\mathcal{G}}((\ell, \nu), \sigma_{\mathsf{Min}}, \sigma_{\mathsf{Max}})) - \mathsf{wt}(\mathsf{play}_{\mathcal{G}'}((\ell, \nu), \sigma_{\mathsf{Min}}, \sigma_{\mathsf{Max}}))| \leqslant \|\mathsf{fwt} - \mathsf{fwt}'\|_{\infty}$$

In particular,

$$\mathsf{wt}(\mathsf{play}_{\mathcal{G}}((\ell, \nu), \sigma_{\mathsf{Min}}, \sigma_{\mathsf{Max}})) \leqslant \mathsf{wt}(\mathsf{play}_{\mathcal{G}'}((\ell, \nu), \sigma_{\mathsf{Min}}, \sigma_{\mathsf{Max}})) + \|\mathsf{fwt} - \mathsf{fwt}'\|_{\infty}$$

so that, taking infimum over all $\sigma_{\mathsf{Min}}$ and supremum over all $\sigma_{\mathsf{Max}}$ gives

$$\mathsf{Val}_{\mathcal{G}}(\ell, \nu) \leqslant \mathsf{Val}_{\mathcal{G}'}(\ell, \nu) + \|\mathsf{fwt} - \mathsf{fwt}'\|_{\infty}$$

A symmetrical argument allows us to conclude that $\|\mathsf{Val}_{\mathcal{G}} - \mathsf{Val}_{\mathcal{G}'}\|_{\infty} \leqslant \|\mathsf{fwt} - \mathsf{fwt}'\|_{\infty}$. Using the bounded horizon versions of $\mathsf{Val}$ and $\mathsf{wt}$ in the previous proof, we also obtain $\|\mathsf{Val}_{\mathcal{G}}^i - \mathsf{Val}_{\mathcal{G}'}^i\|_{\infty} \leqslant \|\mathsf{fwt} - \mathsf{fwt}'\|_{\infty}$ for any $i$. $\qquad\square$

We now explain in details the full process of approximation of the value $\mathsf{Val}_{\mathcal{G}}(\ell_0, \nu_0)$ of an almost-divergent WTG $\mathcal{G}$: it is a bottom-up computation on the tree $T$ rooted in $(\ell_0, r_0)$ (with $r_0$ the region of $\nu_0$) that we used to describe the semi-unfolding $\mathcal{T}(\mathcal{G})$. By Proposition 6.1, the value we want to approximate is equal to $\mathsf{Val}_{\mathcal{T}(\mathcal{G})}((\tilde{\ell}_0, r_0), \nu_0)$. For a node $s$ in $T$, let $\boldsymbol{V}_s$ denote the exact value function of the corresponding node in $\mathcal{T}(\mathcal{G})$. In particular, the value function at the root of $T$ is equal to $\nu \mapsto \mathsf{Val}_{\mathcal{T}(\mathcal{G})}((\tilde{\ell}_0, r_0), \nu)$. Our algorithm iteratively computes an approximated value function $\boldsymbol{V}_s'$ for all nodes $s$ of $T$.

To obtain an adequate $\varepsilon$-approximation of $\mathsf{Val}_{\mathcal{G}}(\ell_0, \nu_0)$, we will thus need to guarantee a precision in kernels that depend on the number of kernels we visit in the semi-unfolding. Let $\alpha$ be the maximal number of kernels along a branch of the tree $T$. For a given node $s$ in $T$, we also let $\alpha(s)$ be the maximal number of kernels along the branches of the subtree rooted in $s$. Finally, let us denote by $h(s)$ the maximal length of a branch in the subtree rooted in $s$.

We will maintain, along the bottom-up computation, that $\boldsymbol{V}_s'$ is a $\Lambda_s$-Lipschitz-continuous mapping on regions such that

$$\Lambda_s \leqslant \max(2, |\mathcal{X}|)^{h(s)} (2 w_{\max}^L |L| |\mathsf{Reg}(\mathcal{X}, M)| + \Lambda) \quad \text{and} \quad \|\boldsymbol{V}_s - \boldsymbol{V}_s'\|_{\infty} \leqslant \alpha(s)\varepsilon/\alpha \quad (8.3)$$

where $\Lambda$ is the maximal Lipschitz constant of the final weight functions of $\mathcal{G}$. In particular, at the root of $T$ where $\alpha(s) = \alpha$, we indeed recover an $\varepsilon$-approximation of the value of the game in configuration $(\ell_0, \nu_0)$.

For leaf nodes $s$ of $T$, $\boldsymbol{V}_s$ and $\boldsymbol{V}_s'$ are equal to the final weight function $\mathsf{fwt}$ of $\mathcal{T}(\mathcal{G})$, and we thus get the invariant (8.3) for free (knowing that $\alpha(s) = h(s) = 0$).

For an internal node $s$ of $T$ (that either gives rise to a single state $(\ell, r)$ of $\mathcal{T}(\mathcal{G})$, or to a kernel $\mathsf{K}_{\ell, r}$), let us suppose that the value of all children $s'$ of $s$ in $T$ have been computed and verify the invariant (8.3).

If $s$ is a node of the form $(\ell, r)$ (that is not part of a kernel), we define two WTGs $\widetilde{\mathcal{G}}$ and $\widetilde{\mathcal{G}}'$ that contain the state $(\ell, r)$ as well as its children in $T$. The children $s'$ are made target states with respective final weight functions given by $\boldsymbol{V}_{s'}$ and $\boldsymbol{V}_{s'}'$. Moreover, we know that

$$\alpha(s) = \max_{s'} \alpha(s') \text{ and } h(s) = \max_{s'} h(s') + 1$$

By definition, $\boldsymbol{V}_s$ is equal to $\nu \mapsto \mathsf{Val}_{\widetilde{\mathcal{G}}}(s, \nu)$ and thus to $\nu \mapsto \mathsf{Val}^1_{\widetilde{\mathcal{G}}}(s, \nu)$ since $\widetilde{\mathcal{G}}$ is acyclic of depth 1. Our approximation algorithm consists in letting $\boldsymbol{V}'_s = \nu \mapsto \mathsf{Val}^1_{\widetilde{\mathcal{G}}'}(s, \nu)$. By Lemma 8.4 (with $i = 1$ and $\varepsilon$ being $\max_{s'} \alpha(s')\varepsilon/\alpha$), $\|\boldsymbol{V}_s - \boldsymbol{V}'_s\|_\infty \leqslant \alpha(s)\varepsilon/\alpha$. By Lemma 7.13, $\boldsymbol{V}'_s$ is $\Lambda_s$-Lipschitz-continuous on regions with $\Lambda_s \leqslant \max(\max_{s'} \Lambda_{s'}, |\mathsf{wt}(\ell)| + (|\mathcal{X}| - 1) \max_{s'} \Lambda_{s'})$: with the help of the invariant (8.3) for all children $s'$, and since $|\mathsf{wt}(\ell)| \leqslant w^L_{\max}$ and $h(s') \leqslant h(s) - 1$, we obtain $\Lambda_s \leqslant \max(2, |\mathcal{X}|)^{h(s)}(2w^L_{\max} |L||\mathsf{Reg}(\mathcal{X}, M)| + \Lambda)$.

If $s$ is a node of the form $\mathsf{K}_{\ell, r}$, we define two WTGs $\widetilde{\mathcal{G}}$ and $\widetilde{\mathcal{G}}'$, that contain the locations of the kernel $\mathsf{K}_{\ell, r}$ of $\mathcal{T}(\mathcal{G})$, as well as the children of $s$ in $T$ (reached by output edges of $\mathsf{K}_{\ell, r}$). The children $s'$ are made target states of respective final weight functions $\boldsymbol{V}_{s'}$ and $\boldsymbol{V}'_{s'}$. Moreover, we know that

$$\alpha(s) = \max_{s'} \alpha(s') + 1 \text{ and } h(s) = \max_{s'} h(s') + 1$$

Thus, games $\widetilde{\mathcal{G}}$ and $\widetilde{\mathcal{G}}'$ are identical, except for their final weight functions that are $\varepsilon$-close. Thus, we know by Lemma 8.4 that $\|\mathsf{Val}_{\widetilde{\mathcal{G}}} - \mathsf{Val}_{\widetilde{\mathcal{G}}'}\|_\infty \leqslant \max_{s'} \alpha(s')\varepsilon/\alpha$. Moreover, by definition, $\boldsymbol{V}_s$ is equal to $\nu \mapsto \mathsf{Val}_{\widetilde{\mathcal{G}}}(s, \nu)$. Our approximation algorithm consists in letting $\boldsymbol{V}'_s$ be equal to an $\varepsilon/\alpha$-approximation of $\mathsf{Val}_{\widetilde{\mathcal{G}}'}$, obtained by Lemma 8.3 with a granularity $N(\varepsilon/\alpha, \max_{s'} \Lambda_{s'})$: we thus have $\|\mathsf{Val}_{\widetilde{\mathcal{G}}'}(s, \cdot) - \boldsymbol{V}'_s\|_\infty \leqslant \varepsilon/\alpha$, and $\boldsymbol{V}'_s$ is $\Lambda_s$-Lipschitz-continuous on regions with $\Lambda_s \leqslant 2(w^L_{\max} |L||\mathsf{Reg}(\mathcal{X}, M)| + \max_{s'} \Lambda_{s'})$. By triangular inequality, we deduce that $\|\boldsymbol{V}_s - \boldsymbol{V}'_s\|_\infty \leqslant \max \alpha(s')\varepsilon/\alpha + \varepsilon/\alpha = \alpha_s\varepsilon/\alpha$. Moreover, with the help of invariant (8.3) for $\Lambda_{s'}$, we once again obtain that $\Lambda_s \leqslant \max(2, |\mathcal{X}|)^{h(s)}(2w^L_{\max} |L||\mathsf{Reg}(\mathcal{X}, M)| + \Lambda)$.

We thus obtain an algorithm that faithfully computes an $\varepsilon$-approximation of the value of the game. Let us now discuss the complexity of the algorithm. Overall, the biggest Lipschitz constant for $\boldsymbol{V}'$ is $\max(2, |\mathcal{X}|)^h(2w^L_{\max} |L||\mathsf{Reg}(\mathcal{X}, M)| + \Lambda)$ with $h$ the height of the semi-unfolding that is $|\mathcal{R}(\mathcal{G})|(3|\mathcal{R}(\mathcal{G})|w_{\max} + 2w^{\mathsf{t}}_{\max} + 2) + 1$ as noticed in (6.1) (page 39). This Lipschitz constant is thus at most doubly-exponential with respect to the size of $\mathcal{G}$. Therefore, the biggest granularity $N$ used in kernel approximations (described in (8.1)) can be globally bounded as doubly-exponential in the size of $\mathcal{G}$ and polynomial in $1/\varepsilon$. This entails that each kernel approximation can be performed in time doubly-exponential in the size of $\mathcal{G}$, and polynomial in $1/\varepsilon$. As the height of the semi-unfolding is at most exponential in the size of $\mathcal{G}$, the number of kernel approximations needed is at most doubly-exponential. The rest of the algorithm consist in applying Theorem 7.1, outside of the kernels. To obtain the overall complexity, we prove for kernel nodes an equivalent to Proposition 7.9, explaining how one step of computation on the semi-unfolding increases the complexity of the encoding of piecewise affine value functions.

**Lemma 8.5.** *For a kernel WTG $\tilde{\mathcal{G}}'$, if all piecewise affine value functions $\boldsymbol{V}'_{s'}$ of target children $s'$ of $s$ in $T$ have complexity at most $\langle m, \beta \rangle$, then $\boldsymbol{V}'_s$, obtained by Lemma 8.3 with granularity $N$, has complexity at most $\langle m', \beta' \rangle$ with $m' \in \mathcal{O}(MNn)$ and $\beta' = (w_{\max} + n\beta)(n + 1)!(MN)^{n+2}$, where $n$ denotes the number of clocks.*

*Proof.* The bound on $m'$ comes from the fact that value function $\boldsymbol{V}'_s$ is obtained by interpolating the value computed for all $1/N$-corners: equations needed to describe all $1/N$-regions are of the form $N\mathsf{x}_i = \alpha$ with $\alpha \in \{0, 1, \ldots, MN\}$, and $N(\mathsf{x}_1 + \cdots + \mathsf{x}_n) = k$ with $k \in \{-MN, -MN + 1, \ldots, MN\}$. There are $(MN + 1)n + 2MN + 1 \in \mathcal{O}(MNn)$ of them.

These equations use constants bounded by $MN$. To bound $\beta'$, we also need to bound the constants appearing in the partition value function $F$ describing $\boldsymbol{V}'_s$ on each base cell

(that is a $1/N$-region). We deduce from [15, Proposition 19] that the value of each of the corners in the $1/N$-region game $\Gamma_N(\mathcal{G})$ is the value of an acyclic path in the untimed game $\Gamma_N(\mathcal{G})$ (since Max always has a memoryless optimal strategy), and is thus the sum of a final weight and of the weight of at most $(n+1)|L||\mathsf{Reg}_N(\mathcal{X}, M)|$ transitions (this is the number of states of the untimed game).

- The final weight is obtained by taking the value of the partition value function $F_{s'}$ of one of the children of $s$ in $T$ at a corner of a $1/N$-region. If $F_{s'}$ is described by the equation $a_\mathsf{y}\mathsf{y} = a_1\mathsf{x}_1 + \cdots + a_n\mathsf{x}_n + b$ (with integers $|a_\mathsf{y}|, |a_1|, \ldots, |a_n|, |b| \leqslant \beta$) then the value at corner $(x_1 \mapsto k_1/N, \ldots, x_n \mapsto k_n/N)$, with natural numbers $0 \leqslant k_1, \ldots, k_n \leqslant NM$, is of the form $(a_1 k_1 + \cdots + a_n k_n)/Na_\mathsf{y} + b/a_\mathsf{y}$. It can thus be written $A/B$ with $|A|$ and $|B|$ integers at most $nMN\beta$.
- Each transition of $\Gamma_N(\mathcal{G})$ has a weight of the form $d\,\mathsf{wt}(\ell) + \mathsf{wt}(e)$ with $d$ a delay that separates two corners: thus $d \in \{0, 1/N, 2/N, \ldots, MN/N\}$, and the weight can be written as $A/B$ with $|A|$ and $|B|$ integers at most $MNw_{\max}^L + Nw_{\max}^E$.

In total, corners have values that can be written as $A/B$ with $|A|$ and $|B|$ integers at most $MN(w_{\max} + n\beta)$.

The partition value function $F$ describing $\boldsymbol{V}_s'$ on a $1/N$-region $r$ is obtained by interpolating the values of each of its corners. Fix $\nu \in r$: it can be expressed as a (unique) convex combination of the $1/N$-corners $\mathsf{v}_1, \ldots, \mathsf{v}_{n+1}$ of $r$, so that $\nu = \sum_{i=1}^{n+1} \alpha_i \mathsf{v}_i$ with $\alpha_i \in [0, 1]$. Then, we let $[\![F]\!](\nu) = \sum_{i=1}^{n+1} \alpha_i \mathsf{Val}_{\Gamma_N(\mathcal{G})}((s, r, \mathsf{v}), \mathsf{v})$. To further bound $\beta'$, we need to express the coefficients $\alpha_i$ in terms of $\nu(\mathsf{x}_1), \ldots, \nu(\mathsf{x}_n)$. Indeed, we have the matricial relation

$$
\begin{pmatrix} \nu(\mathsf{x}_1) \\ \vdots \\ \nu(\mathsf{x}_n) \\ 1 \end{pmatrix} = \begin{pmatrix} \mathsf{v}_1(\mathsf{x}_1) & \mathsf{v}_2(\mathsf{x}_1) & \cdots & \mathsf{v}_{n+1}(\mathsf{x}_1) \\ \vdots & \vdots & & \vdots \\ \mathsf{v}_1(\mathsf{x}_n) & \mathsf{v}_2(\mathsf{x}_n) & \cdots & \mathsf{v}_{n+1}(\mathsf{x}_n) \\ 1 & 1 & \cdots & 1 \end{pmatrix} \times \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_n \\ \alpha_{n+1} \end{pmatrix}
$$

the last line coming from the fact that the sum of coefficients $\alpha_i$ must be equal to 1. The square matrix $\mathcal{M}$ of this equality can be shown invertible, since no three corners of a $1/N$-region are aligned. Therefore, each coefficient $\alpha_i$ can be written as an affine expression over $\nu(\mathsf{x}_1), \ldots, \nu(\mathsf{x}_n)$ whose coefficients are given by the inverse of matrix $\mathcal{M}$. By Cramer's rule, this inverse can be written as the cofactor matrix $\mathcal{C}$ by the determinant of $\mathcal{M}$. Coefficients of $\mathcal{C}$ are determinants of $n \times n$ submatrices of $\mathcal{M}$. Since coefficients of $\mathcal{M}$ are of the form $k/N$ with $k \in \{0, \ldots, MN\}$, coefficients of $\mathcal{C}$ are of the form $k/N^n$ with $k \in \{0, 1, \ldots, n!(MN)^n\}$. The determinant of $\mathcal{M}$ is similarly of the form $k'/N^{n+1}$ with $k' \in \{1, \ldots, (n+1)!(MN)^{n+1}\}$. We can thus write each $\alpha_i$ as an affine coefficient of $\nu(\mathsf{x}_1), \ldots, \nu(\mathsf{x}_n)$ whose coefficients are of the form $Nk/k'$ with $k \in \{0, 1, \ldots, n!(MN)^n\}$ and $k' \in \{1, \ldots, (n+1)!(MN)^{n+1}\}$. Multiplying these coefficients with the values of corners, we obtain that each partial derivative and additive constant of the equalities defining $F$ on each $1/N$-region is bounded by $MN(w_{\max} + n\beta)(n+1)!(MN)^{n+1}$. $\qquad\square$

Pairing this lemma with Proposition 7.9 that explains how the complexity grows along a non-kernel node, we obtain (as in Corollary 7.10) the maximal complexity of the partition value functions obtained from the leaves of the semi-unfolding in $i$ steps. Notice that the $m$-complexity is reset each time we go through a kernel, while the $\beta$-complexity continues to grow polynomially in $\beta$ (but exponential in $n$ which does not change the final computations). Thus, we conclude once again that the partition value functions after $i$ steps have a complexity doubly-exponential in $i$, and thus (since the height $h$ of the semi-unfolding is

exponential in the size of the game), we obtain a triply-exponential algorithm computing an $\varepsilon$-approximation of the value of an almost-divergent WTG, which concludes the proof of Theorem 3.8. This complexity is polynomial in $1/\varepsilon$ as $N$ is linear in $1/\varepsilon$. An example of execution of the approximation schema can be found in Appendix A.

## 9. Symbolic algorithms

The previous approximation result suffers from several drawbacks. It relies on the SCC decomposition of the region automaton. Each of these SCCs have to be analysed in a sequential way, and their analysis requires an a priori refinement of the granularity of regions. This approach is thus not easily amenable to implementation. We instead prove in this section that the symbolic approach based on the value iteration paradigm, i.e. the computation of iterates of the operator $\mathcal{F}$, recalled in page 15, is an approximation schema, as stated in Theorem 3.9.

9.1. **Symbolic approximation algorithm.** Notice that configurations with value $+\infty$ are stable through value iteration, and do not affect its other computations. Since Theorem 3.9 assumes the absence of configurations of value $-\infty$, we will therefore consider in the following that all configurations have finite value in $\mathcal{G}$.

Consider first a game $\mathcal{G}$ that is a kernel, with final weight functions that are $\Lambda$-Lipschitz-continuous on regions. By Lemma 8.2, there exists an integer $N$ such that solving the untimed weighted game $\Gamma_N(\mathcal{G})$ computes an $\varepsilon/2$-approximation of the value of $1/N$ corners. We denote $N(\varepsilon, \Lambda)$ this granularity, and recall that

$$N(\varepsilon, \Lambda) = \left\lceil \frac{2w_{\max}^L |L||\mathsf{Reg}(\mathcal{X}, M)| + 2\Lambda}{\varepsilon} \right\rceil .$$

Using the results of [15] for untimed weighted games, we know that those values are obtained after a finite number of steps of (the untimed version of) the value iteration operator, depending on the number $|L||\mathsf{Reg}_{N(\varepsilon,\Lambda)}(\mathcal{X}, M)|(|\mathcal{X}| + 1)$ of locations of the region game. More precisely, if one considers a number of iterations

$$P(\varepsilon, \Lambda) = |L||\mathsf{Reg}_{N(\varepsilon,\Lambda)}(\mathcal{X}, M)|(|\mathcal{X}| + 1)(2(|L||\mathsf{Reg}_{N(\varepsilon,\Lambda)}(\mathcal{X}, M)|(|\mathcal{X}| + 1) - 1)w_{\max} + 1),$$

then $\mathsf{Val}_{\Gamma_{N(\varepsilon,\Lambda)}(\mathcal{G})}^{P(\varepsilon,\Lambda)}((\ell, r, \mathsf{v}), \mathsf{v}) = \mathsf{Val}_{\Gamma_{N(\varepsilon,\Lambda)}(\mathcal{G})}((\ell, r, \mathsf{v}), \mathsf{v})$. From this observation, we deduce the following property of $P(\varepsilon, \Lambda)$:

**Lemma 9.1.** *If $\mathcal{G}$ is a kernel with no configurations of infinite value, then $|\mathsf{Val}_{\mathcal{G}}(\ell, \nu) - \mathsf{Val}_{\mathcal{G}}^{P(\varepsilon,\Lambda)}(\ell, \nu)| \leqslant \varepsilon$ for all configurations $(\ell, \nu)$ of $\mathcal{G}$.*

*Proof.* We already know that $\mathsf{Val}_{\Gamma_{N(\varepsilon,\Lambda)}(\mathcal{G})}^{P(\varepsilon,\Lambda)}((\ell, r, \mathsf{v}), \mathsf{v}) = \mathsf{Val}_{\Gamma_{N(\varepsilon,\Lambda)}(\mathcal{G})}((\ell, r, \mathsf{v}), \mathsf{v})$ for all configurations $((\ell, r, \mathsf{v}), \mathsf{v})$ of $\Gamma_{N(\varepsilon,\Lambda)}(\mathcal{G})$. Moreover, Lemma 8.2 ensures that $|\mathsf{Val}_{\mathcal{G}}(\ell, \nu) - \mathsf{Val}_{\Gamma_{N(\varepsilon,\Lambda)}(\mathcal{G})}((\ell, r, \mathsf{v}), \mathsf{v})| \leqslant \varepsilon/2$ whenever $\nu$ is in the $\frac{1}{N(\varepsilon,\Lambda)}$-region $r$. Therefore, we only need to prove that $|\mathsf{Val}_{\mathcal{G}}^{P(\varepsilon,\Lambda)}(\ell, \nu) - \mathsf{Val}_{\Gamma_{N(\varepsilon,\Lambda)}(\mathcal{G})}^{P(\varepsilon,\Lambda)}((\ell, r, \mathsf{v}), \mathsf{v})| \leqslant \varepsilon/2$ to conclude. This is done as for Lemma 8.2, since Lemma 8.1 (that we need to prove Lemma 8.2) does not depend on the length of the plays $\rho$ and $\rho'$, and both runs reach the target state in the same step, i.e. both before or after the horizon of $P(\varepsilon, \Lambda)$ steps. $\qquad\square$

Once we know that value iteration converges on kernels, we can use the semi-unfolding of Section 6 to prove that it also converges on non-negative SCCs when all values are finite.

**Lemma 9.2.** *If $\mathcal{G}$ is a non-negative SCC with no configurations of infinite value, we can compute a bound $P_+(\varepsilon, \Lambda)$ such that $|\mathsf{Val}_{\mathcal{G}}(\ell, \nu) - \mathsf{Val}_{\mathcal{G}}^{P_+(\varepsilon, \Lambda)}(\ell, \nu)| \leqslant \varepsilon$ for all configurations $(\ell, \nu)$ of $\mathcal{G}$.*

*Proof.* The idea is to unfold every kernel of the semi-unfolding game $\mathcal{T}(\mathcal{G})$ according to its bound in Lemma 9.1. More precisely, consider a non-negative SCC $\mathcal{G}$, a precision $\varepsilon$, and an initial configuration $(\ell_0, \nu_0)$. Let $\mathcal{T}(\mathcal{G})$ be its finite semi-unfolding (obtained from the labelled tree $T$, as in Section 6), such that $\mathsf{Val}_{\mathcal{G}}(\ell_0, \nu_0) = \mathsf{Val}_{\mathcal{T}(\mathcal{G})}((\tilde{\ell}_0, r_0), \nu_0)$. Let $\alpha$ be the maximum number of kernels along one of the branches of $T$.

We describe a bottom-up transformation of $T$ into $T'$, that keeps track of the Lipschitz constants and the precision of the approximation, and define a new weighted timed game $\mathcal{T}'(\mathcal{G})$ from $T'$ by applying the method used to create $\mathcal{T}(\mathcal{G})$ in Section 6.

Leaf nodes are left unchanged, with final weight functions that are $\Lambda$-Lipschitz-continuous on regions. Note that the associated state in $\mathcal{T}(\mathcal{G})$ has the same value. For an internal node $s$ of $T$ (that gives rise to a single state $(\ell, r)$ of $\mathcal{T}(\mathcal{G})$, or a kernel $\mathsf{K}_{\ell,r}$), let us suppose that the value of all children $s'$ of $s$ in $T'$ are $\Lambda_1$-Lipschitz-continuous on regions, for some $\Lambda_1$, and that they are $\varepsilon_1$-close to their values in $\mathcal{T}(\mathcal{G})$.

If $s$ is a node of the form $(\ell, r)$ (that is not part of a kernel), we keep it unchanged in $T'$, and therefore by Lemma 8.4 the value of $s$ in $\mathcal{T}'(\mathcal{G})$ is $\varepsilon_1$-close to its value in $\mathcal{T}(\mathcal{G})$, and is $\max(\Lambda_1, |\mathsf{wt}(\ell)| + (|\mathcal{X}| - 1)\Lambda_1)$-Lipschitz-continuous on regions by Lemma 7.13.

If $s$ is a node of the form $\mathsf{K}_{\ell,r}$, we replace it by an unfolding of $\mathsf{K}_{\ell,r}$ of depth $P(\frac{\varepsilon}{\alpha}, \Lambda_1)$, where $P$ is the bound of Lemma 9.1 for the kernel $\mathsf{K}_{\ell,r}$. Then, by Lemmas 9.1 and 8.4, the value of $s$ in $\mathcal{T}'(\mathcal{G})$ is $\varepsilon_1 + \frac{\varepsilon}{\alpha}$-close to its value in $\mathcal{T}(\mathcal{G})$, and is $\Lambda_2$-Lipschitz-continuous on regions by Proposition 7.12, with $\Lambda_2$ polynomial in $w_{\max}^L$ and $|\mathcal{X}|$, and exponential in $P(\frac{\varepsilon}{\alpha}, \Lambda_1)$. In particular, note that $P(\frac{\varepsilon}{\alpha}, \Lambda_1)$ is polynomial in $\Lambda_1$, therefore $\Lambda_2$ is exponential in $\Lambda_1$.

At the root of $T'$, we recover a node whose value function in $\mathcal{T}'(\mathcal{G})$ is $\varepsilon$-close to its value in $\mathcal{T}(\mathcal{G})$, and that is $\Lambda'$-Lipschitz-continuous on regions, with $\Lambda'$ bounded by a tower of $\alpha$ exponentials.

Observe that $\mathcal{T}'(\mathcal{G})$ is not a semi-unfolding, it is instead a (complete) unfolding of $\mathcal{R}(\mathcal{G})$, of a certain bounded depth $P_+(\varepsilon, \Lambda)$ (at most the depth of $T$ times the greatest bound $P$ obtained on kernels, thus non-elementary).

Finally, let us show that the value computed by $\mathsf{Val}_{\mathcal{G}}^{P_+(\varepsilon, \Lambda)}$ on the root state $(\tilde{\ell}_0, r_0)$ is bounded between $\mathsf{Val}_{\mathcal{G}}$ and $\mathsf{Val}_{\mathcal{T}'(\mathcal{G})}$, which allows us to conclude. Consider $\mathcal{T}''(\mathcal{G})$, the (complete) unfolding of $\mathcal{R}(\mathcal{G})$ with unfolding depth $P_+(\varepsilon, \Lambda)$, where kernels are also unfolded. By construction, $\mathsf{Val}_{\mathcal{T}''(\mathcal{G})}((\tilde{\ell}_0, r_0), \nu_0) = \mathsf{Val}_{\mathcal{T}''(\mathcal{G})}^{P_+(\varepsilon, \Lambda)}((\tilde{\ell}_0, r_0), \nu_0)$. Then, we can prove that $\mathsf{Val}_{\mathcal{T}''(\mathcal{G})}^{P_+(\varepsilon, \Lambda)}((\tilde{\ell}_0, r_0), \nu_0) = \mathsf{Val}_{\mathcal{G}}^{P_+(\varepsilon, \Lambda)}(\ell_0, \nu_0)$ (same strategies at bounded horizon $P_+(\varepsilon, \Lambda)$), which implies $\mathsf{Val}_{\mathcal{G}}(\ell_0, \nu_0) = \mathsf{Val}_{\mathcal{R}(\mathcal{G})}((\ell_0, r_0), \nu_0) \leqslant \mathsf{Val}_{\mathcal{T}''(\mathcal{G})}((\tilde{\ell}_0, r_0), \nu_0)$ (the function $\mathsf{Val}^k$ decreases monotonously as $k$ increases). By another monotonicity argument, we can also prove $\mathsf{Val}_{\mathcal{T}''(\mathcal{G})}((\tilde{\ell}_0, r_0), \nu_0) \leqslant \mathsf{Val}_{\mathcal{T}'(\mathcal{G})}((\tilde{\ell}_0, r_0), \nu_0)$: the tree $\mathcal{T}''(\mathcal{G})$ contains $\mathcal{T}'(\mathcal{G})$ as a prefix, and therefore one can construct $\mathcal{T}'(\mathcal{G})$ from $\mathcal{T}''(\mathcal{G})$ by replacing sub-trees of $\mathcal{T}''(\mathcal{G})$ by stopped leaves of final weight $+\infty$ (remember that we are in a non-negative SCCs). Bringing everything together we obtain $|\mathsf{Val}_{\mathcal{G}}^{P_+(\varepsilon, \Lambda)}(\ell_0, \nu_0) - \mathsf{Val}_{\mathcal{G}}(\ell_0, \nu_0)| \leqslant \varepsilon$. $\square$

Proving the same property on non-positive SCCs requires more work, because the semi-unfolding gives final weight $-\infty$ to stop leaves, which does not integrate well with value iteration (initialisation at $+\infty$ on non-target states). However, by unfolding those SCCs slightly more (at most $|\mathcal{R}(\mathcal{G})|$ more steps), we can obtain the desired property with a similar bound $P_-$.

**Lemma 9.3.** *If $\mathcal{G}$ is a non-positive SCC with no configurations of infinite value, we can compute $P_-(\varepsilon, \Lambda)$ such that $|\mathsf{Val}_{\mathcal{G}}(\ell, \nu) - \mathsf{Val}_{\mathcal{G}}^{P_-(\varepsilon, \Lambda)}(\ell, \nu)| \leqslant \varepsilon$ for all configurations $(\ell, \nu)$ of $\mathcal{G}$.*

*Proof.* Consider a non-positive SCC $\mathcal{G}$, a precision $\varepsilon$, and an initial configuration $(\ell_0, \nu_0)$. Let $\mathcal{T}(\mathcal{G})$ be its finite semi-unfolding (obtained from the labelled tree $T$, as in Section 6), such that $\mathsf{Val}_{\mathcal{G}}(\ell_0, \nu_0) = \mathsf{Val}_{\mathcal{T}(\mathcal{G})}((\tilde{\ell}_0, r_0), \nu_0)$.

We now change $T$, by adding a subtree under each stopped leaf: the complete unfolding of $\mathcal{R}(\mathcal{G})$, starting from the stopped leaf, of depth $|\mathcal{R}(\mathcal{G})|$. Let us name $T^+$ this unfolding tree. We then construct $\mathcal{T}^+(\mathcal{G})$ as before, based on $T^+$. Since we are in a non-positive SCC, $\mathcal{T}^+(\mathcal{G})$ must have final weight $-\infty$ on its stopped leaves. It is easy to see that $\mathsf{Val}_{\mathcal{G}}(\ell_0, \nu_0) = \mathsf{Val}_{\mathcal{T}^+(\mathcal{G})}((\tilde{\ell}_0, r_0), \nu_0)$ still holds (the proof of Lemma 9.2 was based on branches being long enough, and we increased the lengths). We now perform a small but crucial change: the final weight of stopped leaves in $\mathcal{T}^+(\mathcal{G})$ is set to $+\infty$ instead of $-\infty$. Trivially $\mathsf{Val}_{\mathcal{T}(\mathcal{G})}((\tilde{\ell}_0, r_0), \nu_0) \leqslant \mathsf{Val}_{\mathcal{T}^+(\mathcal{G})}((\tilde{\ell}_0, r_0), \nu_0)$ (we increased the final weight function). Let us prove that

$$\mathsf{Val}_{\mathcal{T}^+(\mathcal{G})}((\tilde{\ell}_0, r_0), \nu_0) \leqslant \mathsf{Val}_{\mathcal{T}(\mathcal{G})}((\tilde{\ell}_0, r_0), \nu_0) \,.$$

For a fixed $\eta > 0$, consider $\sigma_{\mathsf{Min}}$ an $\eta$-optimal strategy for player $\mathsf{Min}$ in $\mathcal{T}(\mathcal{G})$. Let us define $\sigma_{\mathsf{Min}}^+$, a strategy for $\mathsf{Min}$ in $\mathcal{T}^+(\mathcal{G})$, by making the same choice as $\sigma_{\mathsf{Min}}$ on the common prefix tree, and once a node that is a stopped leaf in $\mathcal{T}(\mathcal{G})$ is reached, we switch to a positional attractor strategy of $\mathsf{Min}$ towards target states. Consider any strategy $\sigma_{\mathsf{Max}}^+$ of $\mathsf{Max}$ in $\mathcal{T}^+(\mathcal{G})$, and let $\sigma_{\mathsf{Max}}$ be its projection in $\mathcal{T}(\mathcal{G})$. Let $\rho^+$ denote the (maximal) play

$$\mathsf{play}_{\mathcal{T}^+(\mathcal{G})}(((\ell_0, r_0), \nu_0), \sigma_{\mathsf{Min}}^+, \sigma_{\mathsf{Max}}^+)) \,,$$

and $\rho$ be $\mathsf{play}_{\mathcal{T}(\mathcal{G})}(((\ell_0, r_0), \nu_0), \sigma_{\mathsf{Min}}, \sigma_{\mathsf{Max}}))$. By construction, $\rho^+$ does not reach a stopped leaf in $\mathcal{T}^+(\mathcal{G})$. If the play $\rho^+$ stays in the common prefix tree of $T$ and $T^+$, then $\rho = \rho^+$, and

$$\mathsf{wt}_{\mathcal{T}^+(\mathcal{G})}(\rho^+) \leqslant \mathsf{Val}_{\mathcal{T}(\mathcal{G})}((\tilde{\ell}_0, r_0), \nu_0) + \eta \,.$$

If it does not, then $\rho^+$ has a prefix that reaches a stopped leaf in $\mathcal{T}(\mathcal{G})$: this must be $\rho$. This implies (see Lemma 6.4) that

$$\mathsf{wt}_{\mathcal{T}^+(\mathcal{G})}(\rho^+) < -|\mathcal{R}(\mathcal{G})| w_{\max} - w_{\max}^{\mathsf{t}} \leqslant \mathsf{Val}_{\mathcal{T}(\mathcal{G})}((\tilde{\ell}_0, r_0), \nu_0) \,.$$

Since this holds for all $\eta > 0$, we proved $\mathsf{Val}_{\mathcal{T}^+(\mathcal{G})}((\tilde{\ell}_0, r_0), \nu_0) \leqslant \mathsf{Val}_{\mathcal{T}(\mathcal{G})}((\tilde{\ell}_0, r_0), \nu_0)$, which finally implies that the two values are equal.

Then, we can follow the proof of Lemma 9.2 (with $T^+$ and $\mathcal{T}^+(\mathcal{G})$ instead of $T$ and $\mathcal{T}(\mathcal{G})$) in order to conclude.  $\square$

Now, if we are given an almost-divergent WTG $\mathcal{G}$ and a precision $\varepsilon$, we can add the bounds for value iteration obtained from each SCC by Lemmas 9.2 and 9.3, and obtain a final bound $P$ such that for all $k \geqslant P$, $\mathsf{Val}_{\mathcal{G}}^k$ is an $\varepsilon$-approximation of $\mathsf{Val}_{\mathcal{G}}$. This concludes the proof of Theorem 3.9.

9.2. **Discussion.** Overall, this leads to an upper bound complexity that is of the order of a tower of $\alpha$ exponentials, with $\alpha$ exponential in the size of the input WTG. However, we argue that this symbolic procedure is more amenable to implementation than the previous approximation schema. First, it avoids the three already mentioned drawbacks (SCC decomposition, sequential analysis of the SCCs, and refinement of the granularity of regions) of the previous approximation schema. Then, it allows one to directly launch the value iteration algorithm on the game $\mathcal{G}$, and we can stop the computation whenever we are satisfied enough by the approximation computed: however, there are no guarantees whatsoever on the quality of the approximation before the number of steps $P$ given above.

If $\mathcal{G}$ is not guaranteed to be free of configurations of value $-\infty$, then we must first perform the SCC decomposition of $\mathcal{R}(\mathcal{G})$, and, as $\mathcal{G}$ is almost-divergent, identify and remove regions whose value is $-\infty$, by Proposition 5.5. Then, we can apply the value iteration algorithm.

We also note that if $\mathcal{G}$ is a divergent WTG, the unfolding of kernels is not needed, so that Lemmas 9.2 and 9.3 construct bounds $P_+(\varepsilon, \Lambda)$ and $P_-(\varepsilon, \Lambda)$ allowing one to compute the exact values of $\mathcal{G}$. Moreover, these bounds become exponential in the size of $\mathcal{G}$ instead of being non-elementary, so that the overall complexity of the symbolic algorithm is 3-EXPTIME, matching the result of Section 8.

As a final remark, notice that our correctness proof strongly relies on Section 8.2, and thus would not hold with the approximation schema of [10] (which does not preserve the continuity on regions of the computed value functions, in turn needed to define final weights on $\frac{1}{N(\varepsilon, \Lambda)}$-corners).

## 10. Strategy synthesis

We are also interested in the synthesis problem, that asks for an $\varepsilon$-optimal strategy of Min. In this section, we will prove the following result:

**Theorem 10.1.** *Let $\varepsilon > 0$ and let $\mathcal{G}$ be a divergent WTG. We can compute in triple exponential time an $\varepsilon$-optimal strategy for* Min.

Recall that in the value iteration algorithm of Section 2.6, one step of the game is summarised by the operator $\mathcal{F}$ mapping each value function $\boldsymbol{V}$ to a value function $\boldsymbol{V}' = \mathcal{F}(\boldsymbol{V})$ defined by

$$\boldsymbol{V}'_\ell(\nu) = \begin{cases} \mathsf{fwt}(\ell, \nu) & \text{if } \ell \in L_\mathsf{t}, \\ \sup_{(\ell,\nu) \xrightarrow{d,e} (\ell',\nu')} \left[ d \cdot \mathsf{wt}(\ell) + \mathsf{wt}(e) + \boldsymbol{V}_{\ell'}(\nu') \right] & \text{if } \ell \in L_\mathsf{Max}, \\ \inf_{(\ell,\nu) \xrightarrow{d,e} (\ell',\nu')} \left[ d \cdot \mathsf{wt}(\ell) + \mathsf{wt}(e) + \boldsymbol{V}_{\ell'}(\nu') \right] & \text{if } \ell \in L_\mathsf{Min}. \end{cases} \tag{10.1}$$

Intuitively, $\varepsilon$-optimal strategies can be extracted from the value iteration operator, by mapping each play that ends in $(\ell, \nu)$ with $\ell \in L_\mathsf{Min}$ to a choice $(d, e)$ such that the transition $(\ell, \nu) \xrightarrow{d,e} (\ell', \nu')$ is $\varepsilon$-optimal. However, the choice depends on the step $i$ in the value iteration computation. Formally, if $A$ is a set, $f$ is a mapping from $A$ to $\mathbb{R}_\infty$ and $\varepsilon > 0$, let $\mathrm{arginf}_{a \in A}^\varepsilon f(a)$ denote the set of elements $a^* \in A$ such that $f(a^*) \leqslant \inf_{a \in A} f(a) + \varepsilon$. Then, let us name $\sigma_\mathsf{Min}^{i,\varepsilon}$ a strategy defined from the application of (10.1) in $\boldsymbol{V}^i = \mathcal{F}(\boldsymbol{V}^{i-1})$, so that for all finite plays $\rho$ ending in $(\ell, \nu)$ and $\ell \in L_\mathsf{Min}$,

$$\sigma_\mathsf{Min}^{i,\varepsilon}(\rho) \in \mathrm{arginf}_{(\ell,\nu) \xrightarrow{d,e} (\ell',\nu')}^\varepsilon \left( d \cdot \mathsf{wt}(\ell) + \mathsf{wt}(e) + \boldsymbol{V}_{\ell'}^{i-1}(\nu') \right) \tag{10.2}$$

Now, let $\sigma_{\mathsf{Min}}^{\star,i,\varepsilon}$ denote a strategy that maps every finite play $\rho$ ending in $L_{\mathsf{Min}} \setminus L_{\mathsf{t}}$ to $(d, e) = \sigma_{\mathsf{Min}}^{j,\varepsilon}(\rho)$, with $j = \max(0, i - |\rho|)$.

**Proposition 10.2.** *The strategy $\sigma_{\mathsf{Min}}^{\star,i,\varepsilon}$ is $\varepsilon i$-optimal for $\mathsf{Min}$ during the first $i$ steps:*

$$|\mathsf{Val}^i((\ell,\nu), \sigma_{\mathsf{Min}}^{\star,i,\varepsilon/i}) - \mathsf{Val}^i(\ell,\nu)| \leqslant \varepsilon$$

*Proof.* Let us show by induction on $i$ that for all configurations $(\ell, \nu)$ in $\mathcal{G}$, $\mathsf{Val}_{\mathcal{G}}^i((\ell,\nu), \sigma_{\mathsf{Min}}^{\star,i,\varepsilon}) \leqslant V_\ell^i(\nu) + \varepsilon i$. If $i = 0$, both sides are equal to either $+\infty$ or to $\mathsf{fwt}(\ell,\nu)$ if $\ell \in L_{\mathsf{t}}$. Let us assume that $\mathsf{Val}_{\mathcal{G}}^{i-1}((\ell,\nu), \sigma_{\mathsf{Min}}^{\star,i-1,\varepsilon}) \leqslant V_\ell^{i-1}(\nu) + \varepsilon(i-1)$.

We note that for all strategies $\sigma_{\mathsf{Min}}$, if $\ell \in L_{\mathsf{Max}}$ then

$$\mathsf{Val}_{\mathcal{G}}^i((\ell,\nu), \sigma_{\mathsf{Min}}) = \sup_{(\ell,\nu) \xrightarrow{d,e} (\ell',\nu')} \left[ d \cdot \mathsf{wt}(\ell) + \mathsf{wt}(e) + \mathsf{Val}_{\mathcal{G}}^{i-1}((\ell',\nu'), \sigma_{\mathsf{Min}}') \right],$$

where $\sigma_{\mathsf{Min}}'$ appends $(\ell,\nu) \xrightarrow{d,e} (\ell',\nu')$ in front of paths and then calls $\sigma_{\mathsf{Min}}$. In particular, if $\sigma_{\mathsf{Min}} = \sigma_{\mathsf{Min}}^{\star,i,\varepsilon}$ then $\sigma_{\mathsf{Min}}'$ matches the definition of $\sigma_{\mathsf{Min}}^{\star,i-1,\varepsilon}$. Then, $\mathsf{Val}_{\mathcal{G}}^i((\ell,\nu), \sigma_{\mathsf{Min}}^{\star,i,\varepsilon}) \leqslant V_\ell^i(\nu) + \varepsilon(i-1)$.

Similarly, if $\ell \in L_{\mathsf{Min}} \setminus L_{\mathsf{t}}$ then

$$\mathsf{Val}_{\mathcal{G}}^i((\ell,\nu), \sigma_{\mathsf{Min}}) = d \cdot \mathsf{wt}(\ell) + \mathsf{wt}(e) + \mathsf{Val}_{\mathcal{G}}^{i-1}((\ell',\nu'), \sigma_{\mathsf{Min}}')$$

where $(\ell,\nu) \xrightarrow{d,e} (\ell',\nu')$ is the decision made by $\sigma_{\mathsf{Min}}$ on $(\ell,\nu)$ and $\sigma_{\mathsf{Min}}'$ appends $(\ell,\nu) \xrightarrow{d,e} (\ell',\nu')$ in front of paths and then calls $\sigma_{\mathsf{Min}}$. Then,

$$\mathsf{Val}_{\mathcal{G}}^i((\ell,\nu), \sigma_{\mathsf{Min}}^{\star,i,\varepsilon}) - \varepsilon \leqslant d \cdot \mathsf{wt}(\ell) + \mathsf{wt}(e) + \mathsf{Val}_{\mathcal{G}}^{i-1}((\ell',\nu'), \sigma_{\mathsf{Min}}^{\star,i-1,\varepsilon}) \leqslant V_\ell^i(\nu) + \varepsilon(i-1)$$

Finally, if $\ell \in L_{\mathsf{t}}$ then $\mathsf{Val}_{\mathcal{G}}^i((\ell,\nu), \sigma_{\mathsf{Min}}^{\star,i,\varepsilon}) = V_\ell^i(\nu) = \mathsf{fwt}(\ell,\nu)$.  $\square$

We now explain how to extract some strategies $\sigma_{\mathsf{Min}}^{i,\varepsilon}$ from the partition value functions, in order to solve the synthesis problems on acyclic and divergent games (games where a known $i \in \mathbb{N}$ implies $\mathsf{Val} = \mathsf{Val}^i$).

We will use affine inequalities over $n+1$ variables to encode constraints on the choice of delays. Formally, an affine delay inequality is an equation $I$ of the form $a_{\mathsf{d}}\mathsf{d} \prec a_1 \mathsf{x}_1 + \cdots + a_n \mathsf{x}_n + b$, with all $a_i$, $b$ and $a_{\mathsf{d}}$ integers of $\mathbb{Z}$, $a_{\mathsf{d}} \neq 0$ and $\prec \in \{<, \leqslant\}$. We say that $I$ is a lower bound if $a_{\mathsf{d}} < 0$, and that it is an upper bound if $a_{\mathsf{d}} > 0$.

**Definition 10.3.** A *partition strategy function* $S$ defined over a partition $P$ is a mapping from the base cells of $P$ to a tuple $(e, I_{\mathsf{l}}, I_{\mathsf{u}}, p)$, where $e$ is an edge of $\mathcal{G}$, $I_{\mathsf{l}}$ and $I_{\mathsf{u}}$ are two affine delay inequalities that are respectively lower and upper bounds, and finally $p \in \{\mathsf{l}, \mathsf{u}\}$ selects one of the two.

Given a precision $\varepsilon > 0$, a partition strategy function encodes a mapping from $c_P$ to pairs $(e, d^\varepsilon)$ denoted $[\![S]\!]^\varepsilon$, with $e \in E$ and $d^\varepsilon \in \mathbb{R}_{\geqslant 0}$. Let $\nu \in c_P$ and $S([\nu]_P)$ be equal to $(e, I_{\mathsf{l}}, I_{\mathsf{u}}, p)$, with $I_{\mathsf{l}}$ defined by $a_{\mathsf{d}}\mathsf{d} \prec a_1\mathsf{x}_1 + \cdots + a_n\mathsf{x}_n + b$ and $I_{\mathsf{u}}$ defined by $a_{\mathsf{d}}'\mathsf{d} \prec' a_1'\mathsf{x}_1 + \cdots + a_n'\mathsf{x}_n + b'$. Let $D \subseteq \mathbb{R}_{\geqslant 0}$ denote the interval of delays $\mathsf{d}$ that satisfy both $a_{\mathsf{d}}\mathsf{d} \prec a_1\nu(\mathsf{x}_1) + \cdots + a_n\nu(\mathsf{x}_n) + b$ and $a_{\mathsf{d}}'\mathsf{d} \prec' a_1'\nu(\mathsf{x}_1) + \cdots + a_n'\nu(\mathsf{x}_n) + b'$. We denote by $d_{\mathsf{l}}$ and $d_{\mathsf{u}}$ the lower and upper bounds of $D$. Let $D^\varepsilon$ denote $D \cap (d_{\mathsf{l}} - \varepsilon, d_{\mathsf{l}} + \varepsilon)$ if $p = \mathsf{l}$, and $D \cap (d_{\mathsf{u}} - \varepsilon, d_{\mathsf{u}} + \varepsilon)$ if $p = \mathsf{u}$. We denote by $d_{\mathsf{l}}^\varepsilon$ and $d_{\mathsf{u}}^\varepsilon$ the lower and upper bounds of $D^\varepsilon$. Then, $[\![S]\!]^\varepsilon(\nu) = (e, d^\varepsilon)$ with $d^\varepsilon = \frac{d_{\mathsf{l}}^\varepsilon + d_{\mathsf{u}}^\varepsilon}{2} \in D^\varepsilon$.

We argue that partition strategy functions can be used to compute a positional strategy $\sigma_{\mathsf{Min}}^{i,\varepsilon}$ given an encoding of $\mathsf{Val}_{\mathcal{G}}^{i-1}$ as partition value functions, so that (10.2) holds.

We assume that for all $\ell \in L$, $\boldsymbol{V}_\ell^i$ is piecewise affine with finitely many pieces and is $\Lambda$-Lipschitz-continuous over regions. By Proposition 7.12, we know that $\Lambda$ is at most exponential in the size of $\mathcal{G}$ and $i$.

**Proposition 10.4.** *For all $\varepsilon > 0$ and $i \in \mathbb{N}_{>0}$, we can compute in time doubly-exponential in $i$ and exponential in the size of $\mathcal{G}$ a partition $P_\ell^i$ and a partition strategy function $S_\ell^i$ for each location $\ell \in L_{\mathsf{Min}} \backslash L_{\mathsf{t}}$, so that $[\![S_\ell^i]\!]^{\frac{\varepsilon}{\Lambda}}(\nu) = \sigma_{\mathsf{Min}}^{i,\varepsilon}(\ell, \nu)$.*

*Proof.* In Section 7, we detailed how to compute piecewise affine value functions $(P_\ell^i, F_\ell^i)$ encoding $\boldsymbol{V}_\ell^i$ for all locations $\ell$ and a fixed horizon $i$. We will now explain how $S_\ell^i$ can be obtained as a side-product of the step $\boldsymbol{V}^i = \mathcal{F}(\boldsymbol{V}^{i-1})$.

Recall that if $\ell \in L_{\mathsf{Min}} \backslash L_{\mathsf{t}}$, it holds that

$$\boldsymbol{V}_\ell^i = \min_{e=(\ell,g,\mathcal{Y},\ell')} \mathsf{Pre}_e(\mathsf{Guard}_g(\mathsf{Unreset}_{\mathcal{Y}}(\boldsymbol{V}_{\ell'}^{i-1})))$$

where $e$ ranges over the edges in $\mathcal{G}$ that starts from $\ell$. We can identify, for every base cell $c$ of the resulting partition $P_\ell^i$, an edge $e$ so that $\boldsymbol{V}_\ell^i$ restricted to domain $c$ is equal to $\mathsf{Pre}_e(\mathsf{Guard}_g(\mathsf{Unreset}_{\mathcal{Y}}(\boldsymbol{V}_{\ell'}^{i-1})))$. This is the edge that $S_\ell^i$ will play on $c$. For the choice of delay, let us explain how to obtain the delay inequalities $I_{\mathbf{l}}$ and $I_{\mathbf{u}}$, and the selection choice $p$, from the computation of $\mathsf{Pre}_e$.

Let $(P, F)$ be an atomic tube partition encoding $\mathsf{Guard}_g(\mathsf{Unreset}_{\mathcal{Y}}(\boldsymbol{V}_{\ell'}^{i-1}))$. If $c_b$ is a base cell of $P$, recall that $\mathcal{C}_{c_b}$ is the set of cells reachable from $c_b$ by time elapse, and that if $c \in C_{c_b}$, $\mathcal{B}_{c_b}(c)$ is the set of borders of $c$ reachable from $c_b$ by time-elapse. In particular, $\mathcal{B}_{c_b}(c)$ contains either one non-diagonal border (a single choice of delay reaches $c$ from any $\nu \in c_b$) or two non-diagonal borders (in which case there is a range of delays reaching $c$). Then, $\mathsf{Pre}_e([\![F]\!])$ restricted to domain $c_b$ equals $\min([\![F]\!], \min_{c \in \mathcal{C}_{c_b}} \min_{B \in \mathcal{B}_{c_b}(c)}[\mathsf{Pre}_{e,c,B}(F)])$. These minimum operators are applied over piecewise affine value functions, and we can identify, for every base cell $c'$ of the output, the arguments that optimise them. In particular, the base cells $c'$ where $\mathsf{Pre}_e([\![F]\!]) = [\![F]\!]$ correspond to a choice of delay zero, so that $S_\ell^i$ maps $c'$ to $(e, -\mathsf{d} \leqslant 0, \mathsf{d} \leqslant 0, \mathbf{l})$. Alternatively, the base cells $c'$ where $\mathsf{Pre}_e([\![F]\!]) = \mathsf{Pre}_{e,c,B}(F)$ for some $c \in \mathcal{C}_{c_b}$ and some $B \in \mathcal{B}_{c_b}(c)$ correspond, for all $\nu \in c'$, to a choice of delay arbitrarily close to $d_{\nu,B}$, the delay that lets valuation $\nu$ reach $[\![B]\!]$ by time-elapse. Recall that if $B$ is described by $a_1\mathsf{x}_1 + \cdots + a_n\mathsf{x}_n + b = 0$ and $A = a_1 + \cdots + a_n$, then

$$-Ad_{\nu,B} = a_1 \cdot \nu(\mathsf{x}_1) + \cdots + a_n \cdot \nu(\mathsf{x}_n) + b$$

There are now two cases. If $\mathcal{B}_{c_b}(c) = \{B\}$, then $S_\ell^i$ maps $c'$ to $(e, I_{\mathbf{l}}, I_{\mathbf{u}}, \mathbf{l})$, with $I_{\mathbf{l}}$ defined by $-A\mathsf{d} \leqslant a_1\mathsf{x}_1 + \cdots + a_n\mathsf{x}_n + b$ and $I_{\mathbf{u}}$ defined by $A\mathsf{d} \leqslant -a_1\mathsf{x}_1 - \cdots - a_n\mathsf{x}_n - b$. Otherwise $\mathcal{B}_{c_b}(c) = \{B_{\mathbf{l}}, B_{\mathbf{u}}\}$, with $B_{\mathbf{l}}$ the lower border and $B_{\mathbf{u}}$ the upper one.[19] As $B_{\mathbf{l}}$ and $B_{\mathbf{u}}$ are borders of $c$, they correspond to affine inequalities $a_1\mathsf{x}_1 + \cdots + a_n\mathsf{x}_n + b \prec 0$ and $a_1'\mathsf{x}_1 + \cdots + a_n'\mathsf{x}_n + b' \prec' 0$, respectively. Let $A = a_1 + \cdots + a_n$ and $A' = a_1' + \cdots + a_n'$. We let $I_{\mathbf{l}}$ be defined by $-A\mathsf{d} \prec a_1\mathsf{x}_1 + \cdots + a_n\mathsf{x}_n + b$ and $I_{\mathbf{u}}$ be defined by $A'\mathsf{d} \prec' -a_1'\mathsf{x}_1 - \cdots - a_n'\mathsf{x}_n - b'$. Then we let $S_\ell^i$ map $c'$ to $(e, I_{\mathbf{l}}, I_{\mathbf{u}}, p)$, where $p = \mathbf{l}$ if $B = B_{\mathbf{l}}$ and $p = \mathbf{u}$ if $B = B_{\mathbf{u}}$.

It follows that $[\![S_\ell^i]\!]^\varepsilon(\nu)$ corresponds to a choice of edge $e$ and delay $d$ from $(\ell, \nu)$ leading to a valuation $(\ell', \nu')$, so that $\|(\nu + d) - (\nu + d_{\nu,B})\|_\infty < \varepsilon$, with $d_{\nu,B}$ the optimal delay

---

[19]We can assume without loss of generality that $d_{\nu,B_{\mathbf{l}}} \leqslant d_{\nu,B_{\mathbf{u}}}$ for all $\nu \in c_b$ as the tube partition $(P, F)$ is atomic.

for minimizing the $\mathsf{Pre}_e$ step. As the output piecewise affine functions are $\Lambda$-Lipschitz-continuous over regions, $|\boldsymbol{V}_\ell^i(\nu + d) - \boldsymbol{V}_\ell^i(\nu + d_{\nu,B})| < \Lambda\varepsilon$. Then by definition of $\mathsf{Pre}_e$ we obtain

$$\llbracket S_\ell^i \rrbracket^\varepsilon(\nu) \in \operatorname*{arginf}_{(\ell,\nu)\xrightarrow{d,e}(\ell',\nu')}{}^{\Lambda\varepsilon} \left(d \cdot \mathsf{wt}(\ell) + \mathsf{wt}(e) + \boldsymbol{V}_{\ell'}^{i-1}(\nu')\right)$$

$\square$

By Proposition 10.2, we can therefore solve the synthesis problem in triple exponential time for all weighted time games $\mathcal{G}$ so that $\mathsf{Val}_{\mathcal{G}} = \mathsf{Val}_{\mathcal{G}}^i$ with $i$ at most exponential in the size of $\mathcal{G}$. This holds for acyclic games, where $i$ is bounded by $|\mathcal{R}(\mathcal{G})|$, and also for divergent games, where $i$ can be bounded by the results of Section 9.1 (Lemmas 9.2 and 9.3 in the special case with no kernel), assuming that an exponential time pre-computation using Proposition 5.5 is used to remove the valuations of value $-\infty$. This concludes the proof of Theorem 10.1.

## 11. WEIGHTED TIMED GAMES WITH NO CLOCKS

In this section, we study weighted timed games where the set of clocks is empty. Their semantics is a finite transition system, so we will simply call them (finite) weighted games. For notational convenience, we omit guards, resets, valuations and delays from all notations, and merge the notions of locations and configurations into the notion of states. We also assume that all delays are null and that weights are integers.

**Definition 11.1.** A *weighted game*[20] is a tuple $\mathcal{G} = \langle S_{\mathsf{Min}}, S_{\mathsf{Max}}, S_{\mathsf{t}}, T, \mathsf{wt}, \mathsf{fwt}\rangle$ with $S = S_{\mathsf{Min}} \uplus S_{\mathsf{Max}}$ a finite set of states split between players $\mathsf{Min}$ and $\mathsf{Max}$, $T \subseteq S \times S$ a finite set of transitions $s \to s'$ from state $s$ to state $s'$, $\mathsf{wt}\colon T \to \mathbb{Z}$ a weight function associating an integer weight with each transition[21], $S_{\mathsf{t}} \subseteq S_{\mathsf{Min}}$ a set of target states for player $\mathsf{Min}$, and $\mathsf{fwt}\colon S_{\mathsf{t}} \to \mathbb{Z}_\infty$ is a function mapping each target state to a final weight of $\mathbb{Z}_\infty = \mathbb{Z} \cup \{-\infty, +\infty\}$.

A cycle is a finite play $\rho$, of length at least 1, such that $\mathsf{first}(\rho) = \mathsf{last}(\rho)$.

11.1. **Solving weighted games.** Let $\mathcal{G}$ be a finite weighted game. The value iteration algorithm described in Section 2.6 can be applied on $\mathcal{G}$ as a greatest fixpoint computation over a vector of $|S|$ numbers, as detailed in [15]. This algorithm computes the value of any finite weighted game in pseudo-polynomial time. In particular, along the fixpoint computation states with a value of $-\infty$ will be associated with a sequence of values that converges towards $-\infty$. When the value of such a state gets low enough, it is recognised as a $-\infty$ state and is set to its fixpoint value directly.

---

[20]Weighted games are called *min-cost reachability games* in [15].
[21]If $t = s \to s'$ is a transition in $\mathcal{G}$, we denote $\mathsf{wt}(s, s')$ the weight $\mathsf{wt}(t)$.

11.1.1. *Optimal strategies.* Let us fix an initial state $s$. By definition of lower and upper values, there exists for each player $\mathsf{P} \in \{\mathsf{Min}, \mathsf{Max}\}$ a sequence of strategies $(\sigma_{\mathsf{P}}^i)_{i \in \mathbb{N}}$ such that $\lim_{i \to \infty} \mathsf{Val}_{\mathcal{G}}(s, \sigma_{\mathsf{P}}^i) = \mathsf{Val}_{\mathcal{G}}(s)$ and such that the sequence $(\mathsf{Val}_{\mathcal{G}}(s, \sigma_{\mathsf{Min}}^i))_{i \in \mathbb{N}}$ is non-increasing over $\mathbb{Z}_{\infty}$, while $(\mathsf{Val}_{\mathcal{G}}(s, \sigma_{\mathsf{Max}}^i))_{i \in \mathbb{N}}$ is non-decreasing. If the sequence $(\mathsf{Val}_{\mathcal{G}}(s, \sigma_{\mathsf{P}}^i))_{i \in \mathbb{N}}$ stabilizes for all $i \geqslant k$, then $\sigma_{\mathsf{P}}^k$ is an optimal strategy of player $\mathsf{P}$ for $s$, i.e. $\mathsf{Val}_{\mathcal{G}}(s, \sigma_{\mathsf{P}}^k) = \mathsf{Val}_{\mathcal{G}}(s)$. Therefore, if $\mathsf{Val}_{\mathcal{G}}(s) > -\infty$ then $\mathsf{Min}$ must have an optimal strategy for $s$ (an infinite decreasing sequence over $\mathbb{Z}$ stabilizes), and if $\mathsf{Val}_{\mathcal{G}}(s) < +\infty$ then $\mathsf{Max}$ has an optimal strategy for $s$. Moreover, if an optimal strategy for $\mathsf{P}$ exists for all states $s$, then they can be combined into an (overall) optimal strategy for $\mathsf{P}$.

In fact, there always exists a *positional* strategy $\sigma_{\mathsf{Max}}^{\star}$ for $\mathsf{Max}$ that is optimal, even if some states have value $+\infty$. The strategy $\sigma_{\mathsf{Max}}^{\star}$ can be obtained in the value iteration algorithm, by memorizing for every state $s \in S_{\mathsf{Max}}$ the transition that maximizes $\max_{s \to s'} \left[ \mathsf{wt}(s, s') + \boldsymbol{V}_{s'} \right]$ in the last application of $\mathcal{F}$. However, this does not hold for $\mathsf{Min}$, as there might be no sequence of positional strategies for player $\mathsf{Min}$ whose value at $s$ converges towards $\mathsf{Val}_{\mathcal{G}}(s)$. In [15], it is shown that value iteration can also compute an optimal strategy for $\mathsf{Min}$ (or a sequence of strategies in the $-\infty$ case), by switching between two positional strategies $\sigma_{\mathsf{Min}}^{\star}$ and $\sigma_{\mathsf{Min}}^{\dagger}$: $\sigma_{\mathsf{Min}}^{\star}$ accumulates negative weight by following negative cycles, and $\sigma_{\mathsf{Min}}^{\dagger}$ ensures reaching a target. The optimal strategy of $\mathsf{Min}$ follows the decisions of $\sigma_{\mathsf{Min}}^{\star}$, until switching to the decisions of $\sigma_{\mathsf{Min}}^{\dagger}$ when the length of the play is greater than a finite bound $k$. These strategies thus require finite memory in the form of a counter.

An interesting case happens if $\mathcal{G}$ has no cycles of negative cumulative weight, e.g. if weights are non-negative.

**Lemma 11.2.** *If $\mathcal{G}$ has no cycles of negative cumulative weight, then both players have optimal strategies that are positional. Moreover, $\mathsf{Val}_{\mathcal{G}} = \mathsf{Val}_{\mathcal{G}}^{|S|}$ and the optimal strategies can be computed in polynomial time.*

*Proof.* Any strategy of $\mathsf{Min}$ is optimal on states of value $+\infty$, so we will ignore those. As there are no negative cycles, value $-\infty$ can only be obtained through reaching a target with final weight $-\infty$. As a consequence, $\mathsf{Min}$ has an optimal strategy $\sigma_{\mathsf{Min}}$ that switches between $\sigma_{\mathsf{Min}}^{\star}$ and $\sigma_{\mathsf{Min}}^{\dagger}$ when the length of the play is greater than some $k$, as detailed in [15]. The strategy $\sigma_{\mathsf{Min}}^{\star}$ is only compatible with cycles of negative cumulative weight, and $k \geqslant |S| + 1$, therefore $\sigma_{\mathsf{Min}}^{\dagger}$ is never used. Thus, $\mathsf{Min}$ has an optimal strategy $\sigma_{\mathsf{Min}}^{\star}$ that is positional. Then, consider $\mathsf{Val}_{\mathcal{G}}^{|S|}$, the value with bounded horizon $|S|$. For every state $s$, we have $\mathsf{Val}_{\mathcal{G}}^{|S|}(s, \sigma_{\mathsf{Min}}^{\star}) = \mathsf{Val}_{\mathcal{G}}(s)$, so that $\mathsf{Val}_{\mathcal{G}}^{|S|} \leqslant \mathsf{Val}_{\mathcal{G}}$. As $\mathsf{Val}_{\mathcal{G}} \leqslant \mathsf{Val}_{\mathcal{G}}^k$ holds for any $k \geq 0$, $\mathsf{Val}_{\mathcal{G}} = \mathsf{Val}_{\mathcal{G}}^{|S|}$. Finally, since value iteration converges in $|S|$ steps, the computation of optimal strategies described in [15] runs in polynomial time. $\square$

11.2. **Divergent and almost-divergent weighted games.** Our contribution on finite weighted games is to solve in polynomial time the value problem, for a subclass of finite weighted games. This class corresponds to the games that are *almost-divergent* when seen as timed games with zero clocks. To the best of our knowledge, this is the first attempt to solve a non-trivial class of weighted games with arbitrary weights in polynomial time.

Let us first define the class of divergent weighted games in the untimed setting:

**Definition 11.3.** A weighted game $\mathcal{G}$ is divergent if every cycle $\rho$ of $\mathcal{G}$ satisfies $\mathsf{wt}_\Sigma(\rho) \neq 0$.

In particular, if $\mathcal{G}$ is seen as a weighted timed game with no clocks, then $\mathcal{R}(\mathcal{G})$ is isomorphic to $\mathcal{G}$, so that divergent weighted games are divergent weighted timed games. We will obtain the following results:

**Theorem 11.4.** *The value problem over finite divergent weighted games is* PTIME-*complete. Moreover, deciding if a given finite weighted game is divergent is an* NL-*complete problem when weights are encoded in unary, and is in* PTIME *when they are encoded in binary.*

With divergent weighted games, we described a class where the value problem is polynomial instead of pseudo-polynomial. This gain in complexity came at a cost, the absence of cycles of weight 0. We argue that some of those cycles can be allowed, and apply the almost-divergent notion to the untimed setting.

**Definition 11.5.** A weighted game $\mathcal{G}$ is *almost-divergent* if every 0-cycle $\rho$ of $\mathcal{G}$ satisfies the following property: for every decomposition of $\rho$ into smaller cycles $\rho'$ and $\rho''$, $\rho'$ and $\rho''$ are 0-cycles.

We will obtain the following results, extending Theorem 11.4:

**Theorem 11.6.** *The value problem over finite almost-divergent weighted games is* PTIME-*complete. Moreover, deciding if a given finite weighted game is almost-divergent is an* NL-*complete problem when weights are encoded in unary, and is in* PTIME *when they are encoded in binary.*

We prove Theorems 11.4 and 11.6 at the same time, by proving a PTIME upper bound for the value problem on almost-divergent game, a PTIME lower bound on divergent games, and finally by studying the decision problems associated with membership to the divergent and almost-divergent classes.

11.2.1. *Polynomial upper bound.* Our algorithm solving almost-divergent games in polynomial time follows the semi-unfolding scheme, that we detailed in the timed setting as a proof of Theorem 3.8. The SCC characterisation of almost-divergence and the notion of kernel apply, and the only changes concern the complexity analysis. In particular, with 0 clocks, we have $|\mathcal{R}(\mathcal{G})| = |\mathcal{G}|$, therefore computing the kernel and the states of value $+\infty$ or $-\infty$ can be done in polynomial time. Moreover, the semi-unfolding of Section 6 has polynomial depth.

In order to compute the value of a state $s_0$ of $\mathcal{G}$, one could construct the semi-unfolding of root $s_0$, and compute its value. Indeed, every cycle in $\mathcal{T}(\mathcal{G})$ belongs to K, so they must all be 0-cycles, therefore by Lemma 11.2 we can compute $\mathsf{Val}_{\mathcal{T}(\mathcal{G})}(\tilde{s}_0)$ in time polynomial in the size of $\mathcal{T}(\mathcal{G})$. However, this would be an exponential time algorithm, since the number of nodes in $T$ can be exponential in $|S|$. We argue that this exponential blow-up can be avoided: when two nodes of $T$ are at the same depth and are labelled by the same state they can be merged, producing a graph $T$ that is acyclic instead of tree-shaped, with at most quadratically many states. This does not change the value of the resulting weighted game $\mathcal{T}(\mathcal{G})$ at its root, because the two merged nodes had the same sub-tree, and therefore were states with the same value in $\mathcal{T}(\mathcal{G})$. This optimization on the construction of $\mathcal{T}(\mathcal{G})$ is performed on-the-fly, while the semi-unfolding is constructed, such that constructing $\mathcal{T}(\mathcal{G})$ (and solving it by Lemma 11.2) can be done in time polynomial in the size of $\mathcal{G}$.

11.2.2. *Polynomial lower bound.* Let us show that the value problem is PTIME-hard on divergent weighted games. This comes from a reduction (in logarithmic space) of the problem of solving finite games with reachability objectives [21]. To a reachability game, we simply set the weight of every transition to 1 and the final weight of every target to 0, making it a divergent weighted game. Then, Min wins the reachability game if and only if the value in the weighted game is lower than $|S|$. The same reduction can be used to show the PTIME-hardness of the $+\infty$ and $-\infty$-value problems on divergent weighted games.

11.2.3. *Deciding divergence and almost-divergence.* Let us study the *membership problem* for divergent and almost-divergent weighted games, i.e. the decision problem that asks if a given weighted game is divergent or almost-divergent, and prove the results of Theorems 11.4 and 11.6.

We start with almost-divergent games, and will rely on the characterization of almost-divergent games in term of SCCs given in Proposition 4.4. First, we note that simple cycles are enough to ensure that an SCC is non-negative (resp. non-positive), providing us with an efficient way to check this property:

**Lemma 11.7.** *An SCC $S$ is non-negative (resp. non-positive) if and only if every simple cycle in $S$ is non-negative (resp. non-positive). Moreover, deciding if an SCC is non-negative (resp. non-positive) is in NL when weights are encoded in unary, and is in PTIME when they are encoded in binary.*

*Proof.* The direct implication holds by definition. Reciprocally, let us assume that every simple cycle in $S$ is non-negative (resp. non-positive), and prove that every cycle $\rho$ in $S$ is non-negative (resp. non-positive). The cycle $\rho$ can be decomposed into simple cycles, all belonging to $S$. Therefore they are all non-negative (resp. non-positive). As the cumulative weight of $\rho$ is the sum of the cumulative weights of these simple cycles, $\rho$ must be non-negative (resp. non-positive).

As a corollary, an SCC $S$ is non-negative (resp. non-positive) if and only if every cycle in $S$, of length at most $|S|$, is non-negative (resp. non-positive).

To decide if a strongly connected $\mathcal{G}$ is non-negative (resp. non-positive), we outline two procedures: one is deterministic and will provide the polynomial upper bound on time-complexity, the other will guess a logarithmic number of bits and provide NL membership.

The deterministic algorithm proceeds as follows: with Floyd-Warshall's algorithm, one can compute the shortest paths (resp. greatest paths) matrix $M$ in cubic time, such that $M(s, s')$ contains the minimal (resp. maximal) value in $\{\mathsf{wt}(\rho) \mid \rho \text{ simple path from } s \text{ to } s'\}$. Then, $S$ is non-negative (resp. non-positive) if and only if for every state $s$ it holds that $M(s, s) \geqslant 0$ (resp. $M(s, s) \leqslant 0$). If there exists a state $s$ such that $M(s, s) < 0$ (resp. $M(s, s) > 0$), then $S$ is not non-negative (resp. not non-positive) as there is a negative (resp. positive) cycle. Conversely, if $M(s, s) \geqslant 0$ (resp. $M(s, s) \leqslant 0$), we know that all simple paths from $s$ to $s$ have non-negative (resp. non-positive) weight.

Let us now assume that weights are encoded in unary, and present a non-deterministic procedure. Then, note that a (binary) register containing integer values in $[-B, B]$, with $B$ polynomial in $w_{\max}$ and $|S|$, requires a number of bits at most logarithmic in the size of $\mathcal{G}$. An SCC is *not* non-negative (resp. not non-positive), if and only if it contains a cycle of positive (resp. negative) cumulative weight, of length bounded by $|S|$. We can guess such a cycle $\rho$ on-the-fly, keeping in memory its cumulative weight (smaller than $B = w_{\max} \times |S|$ in absolute value), its initial state, and its current length, all in logarithmic space. If the length of the

cycle exceeds $|S|$, the guess is invalid. Similarly, we can verify that the last state equals the first, and that the computed cumulative weight is indeed positive (resp. negative). Therefore, deciding if $S$ is non-negative (resp. not non-positive) is in coNL = NL [22, 31]. Note that when weights are encoded in binary this procedure only gives coNP membership. □

Let us now explain why the membership problem is an NL-complete problem when weights are encoded in unary. First, to prove the membership in NL, notice that a weighted game is *not almost-divergent* if and only if there is a positive cycle and a negative cycle, both of length at most $|S|$, and belonging to the same SCC. This can be tested in NL, using a non-deterministic procedure similar to the one from Lemma 11.7. We first guess a starting state for both cycles. Verifying that those are in the same SCC can be done in NL by using standard reachability analysis. Then, we once again guess the two cycles on-the-fly, keeping in memory their cumulative weights in logarithmic space. Therefore, testing divergence is in coNL = NL [22, 31].

The NL-hardness (indeed coNL-hardness, which is equivalent [22, 31]) is shown by a reduction of the reachability problem in a finite automaton. More precisely, we consider a finite automaton with a starting state and a different target state without outgoing transitions. We construct from it a weighted game by distributing all states to Min, and equipping all transitions with weight 0. We also add a loop with weight 1 on the initial state, one with weight $-1$ on the target state, and a transition from the target state to the initial state with weight 0. Then, the game is not almost-divergent if and only if the target can be reached from the initial state in the automaton.

When weights are encoded in binary, the previous decision procedure gives NP membership. However, we can achieve a PTIME upperbound by computing the strongly connected components and then using Lemma 11.7 to check that each SCC is either non-negative or non-positive.

This concludes the proofs of Theorem 11.6. For the membership results in Theorem 11.4, we note that a weighted game $\mathcal{G}$ is divergent if and only if it is almost-divergent and its kernel is empty, i.e. $\mathcal{G}$ does not contain any simple cycle of weight zero. Variants of the previous techniques can be used to either compute the kernel in polynomial time, guess a 0-cycle of length at most $|S|$ in NL when weights are encoded in unary, or prove NL-hardness.

## 12. Conclusion

We have obtained new results for several controller synthesis problems on timed automata, that we now summarise. Our study of weighted timed games belongs to a series of works that explore the frontier of decidability. We introduced the first decidable class of weighted timed games with arbitrary weights and no restrictions on the number of clocks. We have given an approximation procedure for a larger class of weighted timed games, where the exact problem becomes undecidable. In addition, we have proved the correctness of a symbolic approximation schema, that does not start by splitting exponentially every region, but only does so when necessary. We argue that this paves the way towards an implementation of value approximation for weighted timed games. Such tool would likely struggle with instances of moderate size, but could help with the design and testing of alternative approaches that trade theoretical guarantees with performance.

Another perspective is to extend this work to the concurrent setting, where both players play simultaneously and the shortest delay is selected. It should be noted that several known

results on weighted timed games with non-negative weights [9, 1, 10] are stated in such a concurrent setting. We did not consider this setting in this work because concurrent WTGs are not determined, and several of our proofs rely on this property for symmetrical arguments (mainly to lift results of non-negative strongly connected components to non-positive ones).

A long-standing open problem is the approximation of weighted timed games, i.e. whether one can compute an arbitrarily close approximation of the value of a given game. We successfully solved this problem on the class of almost-divergent games, but we were not able to extend further our techniques to more general games. As a first step, we could try to consider the slightly larger class of 0-isolated games, where we ask for every cycle of the region game to have a weight either $\geqslant 1$, or $\leqslant -1$, or exactly 0. We do not have approximation results on this 0-isolated class, and as such it forms a natural intermediate step between the best known decidable class and the general case. However we must prepare ourselves to possibly negative answers (the value of a weighted timed game could be *non approximable*). Therefore, pursuing better lower bounds in various settings could help in the future, in order to close the remaining complexity gaps.

## References

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

[2] Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

[3] Rajeev Alur, Salvatore La Torre, and George J. Pappas. Optimal paths in weighted timed automata. *Theoretical Computer Science*, 318(3):297–322, 2004.

[4] Eugene Asarin and Oded Maler. As soon as possible: Time optimal control for timed automata. In *Hybrid Systems: Computation and Control*, volume 1569 of *Lecture Notes in Computer Science*, pages 19–30. Springer, 1999.

[5] Gerd Behrmann, Ansgar Fehnker, Thomas Hune, Kim G. Larsen, Judi Romijn, and Frits W. Vaandrager. Minimum-cost reachability for priced timed automata. In *Proceedings of the 4th International Workshop on Hybrid Systems: Computation and Control (HSCC'01)*, volume 2034 of *Lecture Notes in Computer Science*, pages 147–161. Springer, 2001.

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

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

[8] Patricia Bouyer, Ed Brinksma, and Kim G. Larsen. Optimal infinite scheduling for multi-priced timed automata. *Formal Methods in System Design*, 32(1):3–23, 2008.

[9] Patricia Bouyer, Franck Cassez, Emmanuel Fleury, and Kim G. Larsen. Optimal strategies in priced timed game automata. In *Proceedings of the 24th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'04)*, volume 3328 of *LNCS*, pages 148–160. Springer, 2004.

[10] Patricia Bouyer, Samy Jaziri, and Nicolas Markey. On the value problem in weighted timed games. In *Proceedings of the 26th International Conference on Concurrency Theory (CONCUR'15)*, volume 42 of *Leibniz International Proceedings in Informatics*, pages 311–324. Leibniz-Zentrum für Informatik, 2015.

[11] Marius Bozga, Conrado Daws, Oded Maler, Alfredo Olivero, Stavros Tripakis, and Sergio Yovine. Kronos: A model-checking tool for real-time systems. In Alan J. Hu and Moshe Y. Vardi, editors, *Computer Aided Verification (CAV 1998), Proceedings*, pages 546–550, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[12] Romain Brenguier, Franck Cassez, and Jean-François Raskin. Energy and mean-payoff timed games. In *Proceedings of the 17th International Conference on Hybrid Systems: Computation and Control (HSCC'14)*, pages 283–292. ACM, 2014.

[13] Thomas Brihaye, Véronique Bruyère, and Jean-François Raskin. On optimal timed strategies. In *Proceedings of the Third international conference on Formal Modeling and Analysis of Timed Systems (FORMATS'05)*, volume 3829 of *LNCS*, pages 49–64. Springer, 2005.

[14] Thomas Brihaye, Gilles Geeraerts, Axel Haddad, Engel Lefaucheux, and Benjamin Monmege. Simple priced timed games are not that simple. In *Proceedings of the 35th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'15)*, volume 45 of *LIPIcs*, pages 278–292. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2015.

[15] Thomas Brihaye, Gilles Geeraerts, Axel Haddad, and Benjamin Monmege. Pseudopolynomial iterative algorithm to solve total-payoff games and min-cost reachability games. *Acta Informatica*, 2016.

[16] Thomas Brihaye, Gilles Geeraerts, Shankara Narayanan Krishna, Lakshmi Manasa, Benjamin Monmege, and Ashutosh Trivedi. Adding negative prices to priced timed games. In *Proceedings of the 25th International Conference on Concurrency Theory (CONCUR'14)*, volume 8704, pages 560–575. Springer, 2014.

[17] Damien Busatto-Gaston. *Symbolic controller synthesis for timed systems: robustness and optimality*. Theses, Aix Marseille Université, December 2019.

[18] Thomas Dueholm Hansen, Rasmus Ibsen-Jensen, and Peter Bro Miltersen. A faster algorithm for solving one-clock priced timed games. In *Proceedings of the 24th International Conference on Concurrency Theory (CONCUR'13)*, volume 8052 of *Lecture Notes in Computer Science*, pages 531–545. Springer, 2013.

[19] Frédéric Herbreteau, Gerald Point, and Thanh-Tung Tran. Tchecker. http://www.labri.fr/perso/herbrete/tchecker/index.html, 2019.

[20] Frédéric Herbreteau, B. Srivathsan, and Igor Walukiewicz. Efficient emptiness check for timed Büchi automata. In *Computer Aided Verification (CAV 2010), Proceedings*, volume 6174 of *Lecture Notes in Computer Science*, pages 148–161. Springer, 2010.

[21] Neil Immerman. Number of quantifiers is better than number of tape cells. *Journal of Computer and System Sciences*, 22(3):384–406, 1981.

[22] Neil Immerman. Nondeterministic space is closed under complementation. *SIAM Journal on Computing*, 17:935–938, 1988.

[23] Marcin Jurdziński and Ashutosh Trivedi. Reachability-time games on timed automata. In *Proceedings of the 34th International Colloquium on Automata, Languages and Programming (ICALP'07)*, volume 4596 of *LNCS*, pages 838–849. Springer, 2007.

[24] Leonid Khachiyan, Endre Boros, Konrad Borys, Khaled Elbassioni, Vladimir Gurvich, Gabor Rudolf, and Jihui Zhao. On short paths interdiction problems: Total and node-wise limited interdiction. *Theory of Computing Systems*, 43(2):204–233, 2008.

[25] Kim G. Larsen, Gerd Behrmann, Ed Brinksma, Ansgar Fehnker, Thomas Hune, Paul Pettersson, and Judi Romijn. As cheap as possible: Efficient cost-optimal reachability for priced timed automata. In *Proceedings of the 13th International Conference on Computer Aided Verification (CAV'01)*, volume 2102 of *Lecture Notes in Computer Science*, pages 493–505. Springer, 2001.

[26] Kim G. Larsen, Paul Pettersson, and Wang Yi. Uppaal in a nutshell. *International Journal on Software Tools for Technology Transfer*, 1(1):134–152, Dec 1997.

[27] Jiri Matousek. *Lectures on Discrete Geometry*. Springer-Verlag, Berlin, Heidelberg, 2002.

[28] Anuj Puri. Dynamical properties of timed automata. *Discrete Event Dynamic Systems*, 10(1-2):87–113, 2000.

[29] Michał Rutkowski. Two-player reachability-price games on single-clock timed automata. In *Proceedings of the Ninth Workshop on Quantitative Aspects of Programming Languages (QAPL'11)*, volume 57 of *Electronic Proceedings in Theoretical Computer Science*, pages 31–46, 2011.

[30] Walter J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences*, 4(2):177–192, 1970.

[31] Róbert Szelepcsényi. The method of forced enumeration for nondeterministic automata. *Acta Informatica*, 26(3):279–284, 1988.

## APPENDIX A. EXAMPLE OF AN EXECUTION OF THE APPROXIMATION SCHEMA

Consider the WTG $\mathcal{G}$ in Figure 10 and some $\varepsilon > 0$. We want to compute an $\varepsilon$-approximation of its value in location $\ell_0$ for the valuation $(x_1 = 0, x_2 = 0)$, denoted $\mathsf{Val}_{\mathcal{G}}(\ell_0, (0,0))$. In this example, we will use $\varepsilon = 15$ because the computations would not be readable with a smaller precision. Since in this example each location $\ell$ of $\mathcal{G}$ leads to a unique state $(\ell, r)$ of $\mathcal{R}(\mathcal{G})$, we will refer to states of $\mathcal{R}(\mathcal{G})$ by their associated location label. As explained in Example 3.7, $\mathcal{R}(\mathcal{G})$ contains one SCC made of two simple cycles, $\mathsf{p}_1 = \ell_1 \to \ell_2 \to \ell_1$ is a positive cycle and $\mathsf{p}_2 = \ell_1 \to \ell_3 \to \ell_4 \to \ell_1$ is a 0-cycle.

Therefore, $\mathcal{R}(\mathcal{G})$ only contains non-negative SCCs and is almost-divergent. Since all states are in the attractor of Min towards $L_t$, all cycles are non-negative and the final weight function is bounded (on all reachable regions), there are no configurations in $\mathcal{R}(\mathcal{G})$ with value $+\infty$ or $-\infty$.

We let the kernel $\mathsf{K}$ be the sub-game of $\mathcal{R}(\mathcal{G})$ defined by $\mathsf{p}_2$, and we construct a semi-unfolding $\mathcal{T}(\mathcal{G})$ of $\mathcal{R}(\mathcal{G})$ of equivalent value. We should unfold the game until every stopped branch contains a state seen at least $3|\mathcal{R}(\mathcal{G})|w_{\max} + 2w_{\max}^{\mathsf{t}} + 2 = 3 \times 3 \times 4 + 2 \times 1 + 2 = 40$ times. We will unfold with bound 4 instead of 40 for readability (it is enough on this example). Thus the infinite branch $(\ell_1\ell_2)^\omega$ is stopped when $\ell_1$ is reached for the fourth time, as depicted in Figure 18.
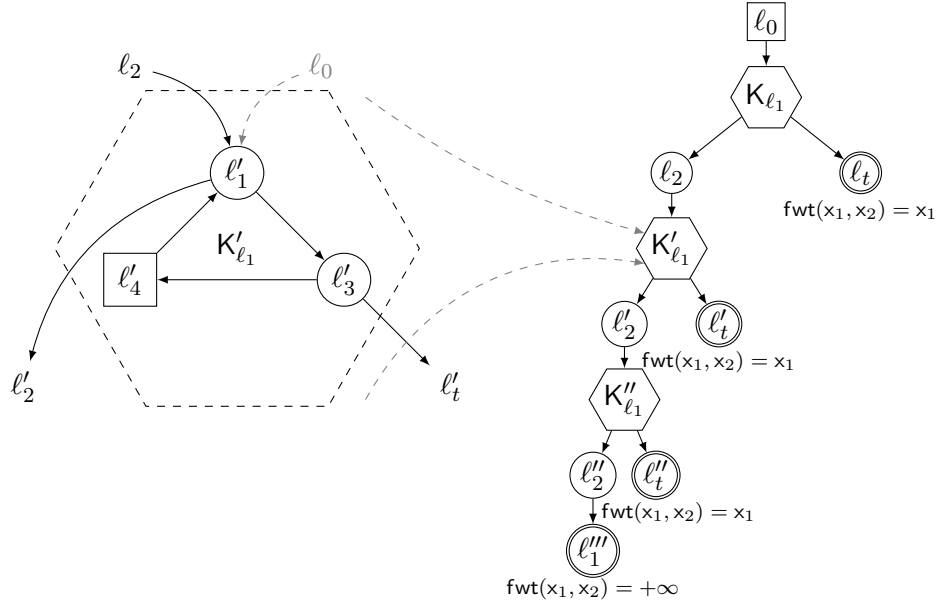


Figure 18: The kernel $\mathsf{K}$ (with input state $\ell_1$), and a semi-unfolding $\mathcal{T}(\mathcal{G})$ such that $\mathsf{Val}_{\mathcal{G}}(\ell_0, (0,0)) = \mathsf{Val}_{\mathcal{T}(\mathcal{G})}(\ell_0, (0,0))$. We denote by $\ell_i$, $\ell_i'$ and $\ell_i''$ the locations in $\mathsf{K}$, $\mathsf{K}'$ and $\mathsf{K}''$.

Let us now compute an approximation of $\mathsf{Val}_{\mathcal{T}(\mathcal{G})}$. Let us first remove the states of value $+\infty$: $\ell_1'''$ and $\ell_2''$. Then, we start at the bottom and compute an $(\varepsilon/3)$-approximation of the value of $\ell_1''$ in the game defined by $\mathsf{K}_{\ell_1}''$ and its output edge to $\ell_t''$. Following Section 8.2, we

should use $N \geqslant 3(4+1)/\varepsilon$ and compute values in the $1/N$-corners game $\mathcal{C}_N(\mathsf{K}''_{\ell_1})$ in order to obtain an $(\varepsilon/3)$-approximation of the value function. For $\varepsilon = 15$ we will use $N = 1$ (in this case the computation happens to be exact and would also hold with a small $\varepsilon$). We construct this corner game, and obtain the finite (untimed) weighted game in Figure 19.
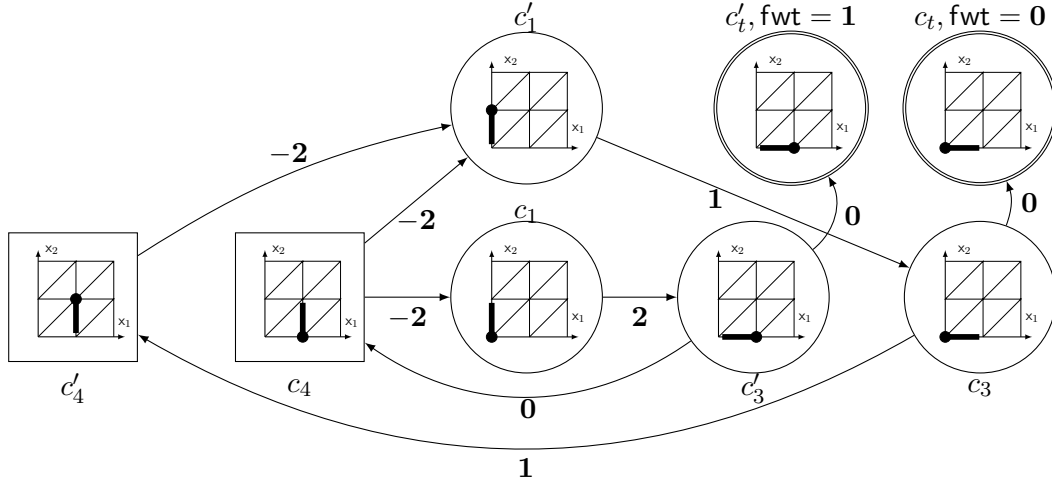


Figure 19: The finite weighted game obtained from $\mathcal{C}_1(\mathsf{K}''_{\ell_1})$, where $c_i$ and $c'_i$ are the corners of $\ell''_i$ in $\mathcal{T}(\mathcal{G})$.

We can compute the values in this game to obtain $\mathsf{Val}(c'_1) = 1$ and $\mathsf{Val}(c_1) = 3$. We then define a value for every configuration in state $\ell''_1$ by linear interpolation, obtaining:

$$(\mathsf{x}_1, \mathsf{x}_2) \mapsto 3 - 2\mathsf{x}_2 \,.$$

This happens to be exactly $(\mathsf{x}_1, \mathsf{x}_2) \mapsto \mathsf{Val}_{\mathcal{T}(\mathcal{G})}(\ell''_1, (\mathsf{x}_1, \mathsf{x}_2))$ in this case, but would only be an $\varepsilon/3$-approximation of it in general. Now, we can compute an $\varepsilon/3$-approximation of $\mathsf{Val}_{\mathcal{T}(\mathcal{G})}(\ell'_2)$ with one step of value iteration, obtaining

$$(\mathsf{x}_1, \mathsf{x}_2) \mapsto \inf_{0 < d < 2 - \mathsf{x}_1} (-1) \times d + 1 + 3 - 2(0 + d) = 3\mathsf{x}_1 - 2 \,.$$

The next step is computing an $\varepsilon/3$-approximation of the value of $\ell'_1$ in the game defined by $\mathsf{K}'_{\ell_1}$ and its output edges to $\ell'_t$ and $\ell'_2$, of respective final weight functions $(\mathsf{x}_1, \mathsf{x}_2) \mapsto \mathsf{x}_1$ and $(\mathsf{x}_1, \mathsf{x}_2) \mapsto 3\mathsf{x}_1 - 2$. This will give us a $2\varepsilon/3$-approximation of $\mathsf{Val}_{\mathcal{T}(\mathcal{G})}(\ell'_1)$.

Following Section 8.2 once again, we should use $N \geqslant 3(5+3)/\varepsilon$ and compute values in the $1/N$-corners game $\mathcal{C}_N(\mathsf{K}'_{\ell_1})$. For $\varepsilon = 15$ this gives $N = 2$ (which will once again keep the computation exact). We can construct a finite (untimed) weighted game as in Figure 19, and obtain a value for each $1/2$-corner of state $\ell'_1$:
- On the $1/2$-region $(\mathsf{x}_1 = 0, 0 < \mathsf{x}_2 < 1/2)$, corner $(0,0)$ has value 2 and corner $(0, 1/2)$ has value 2.
- On the $1/2$-region $(\mathsf{x}_1 = 0, \mathsf{x}_2 = 1/2)$, corner $(0, 1/2)$ has value 2.
- On the $1/2$-region $(\mathsf{x}_1 = 0, 1/2 < \mathsf{x}_2 < 1)$, corner $(0, 1/2)$ has value 2 and corner $(0, 1)$ has value 1.

From these results, we define a piecewise affine function by interpolating the values of corners on each 1/2-region, and obtain

$$(x_1, x_2) \mapsto \begin{cases} 2 & \text{if } x_2 \leqslant 1/2 \\ 3 - 2x_2 & \text{otherwise} \end{cases}$$
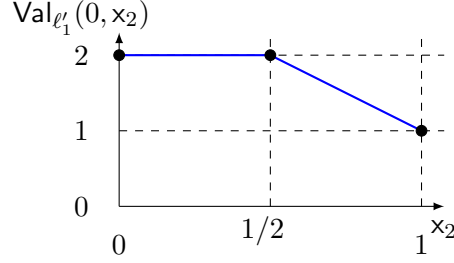
as depicted in Figure 20.



Figure 20: The value function $(x_1, x_2) \mapsto \mathsf{Val}_{\mathcal{T}(\mathcal{G})}(\ell'_1, (x_1, x_2))$, projected on $x_1 = 0$. Black dots represent the values obtained for 1/2-corners using the corner-point abstraction.

This gives us a $2\varepsilon/3$-approximation of $(x_1, x_2) \mapsto \mathsf{Val}_{\mathcal{T}(\mathcal{G})}(\ell'_1, (x_1, x_2))$ (in fact exactly $\mathsf{Val}_{\mathcal{T}(\mathcal{G})}(\ell'_1)$). Now, we can compute a $2\varepsilon/3$-approximation of $\mathsf{Val}_{\mathcal{T}(\mathcal{G})}(\ell_2)$ on region $(1 < x_1 < 2, x_2 = 0)$ with one step of value iteration, obtaining :

$$(x_1, x_2) \to \inf_{0 < d < 2 - x_1} \begin{cases} 3 - d & \text{if } d \leqslant 1/2 \\ 4 - 3d & \text{otherwise} \end{cases} = \begin{cases} 3x_1 - 2 & \text{if } x_1 \leqslant 3/2 \\ x_1 + 1 & \text{otherwise} \end{cases}$$

Then, we need to compute an $\varepsilon/3$-approximation of the value of $\ell_1$ in the game defined by $\mathsf{K}_{\ell_1}$ and its output edges to $\ell_t$ and $\ell_2$, of respective final weight functions $(x_1, x_2) \mapsto x_1$ and $(x_1, x_2) \mapsto 3x_1 - 2$ if $x_1 \leqslant 3/2$; $x_1 + 1$ otherwise. This will give us an $\varepsilon$-approximation of $\mathsf{Val}_{\mathcal{T}(\mathcal{G})}(\ell_1)$.

Following Section 8.2 one last time, we should use $N \geqslant 3(5 + 3)/\varepsilon$ and compute values in the $1/N$-corner game $\mathcal{C}_N(\mathsf{K}_{\ell_1})$. This time, let us use $N = 3$ to showcase an example where the computed value is not exact. We can construct a finite (untimed) weighted game as in Figure 19, and obtain a value for each 1/3-corner of state $\ell'_1$. From these results, we define a piecewise affine function by interpolation, as depicted in Figure 21.
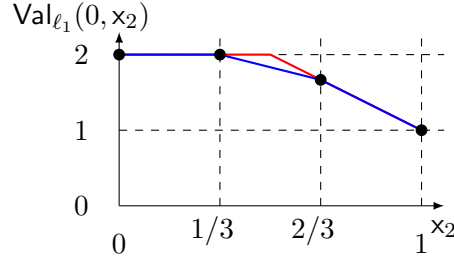
Figure 21: The value function $(x_1, x_2) \mapsto \mathsf{Val}_{\mathcal{T}(\mathcal{G})}(\ell_1, (x_1, x_2))$, projected on $x_1 = 0$, is depicted in red. Black dots represent the values obtained for 1/3-corners using the corner-points abstraction, and the derived approximation of the value function is depicted in blue

Finally, from this $\varepsilon$-approximation of $\mathsf{Val}_{\mathcal{T}(\mathcal{G})}(\ell_1)$, we can compute an $\varepsilon$-approximation of $\mathsf{Val}_{\mathcal{T}(\mathcal{G})}(\ell_0)$ using one step of value iteration, and conclude. On our example this ensures

$$\mathsf{Val}_{\mathcal{T}(\mathcal{G})}(\ell_0, (0, 0)) = \sup_{0 < d < 1} \mathsf{Val}_{\mathcal{T}(\mathcal{G})}(\ell_1, (0, d)) \in [2 - \varepsilon, 2 + \varepsilon].$$