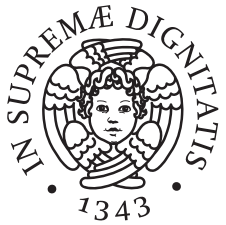

Università degli Studi di Pisa
Facoltà di Scienze Matematiche, Fisiche e Naturali
Corso di Laurea in Matematica

Tesi di Laurea Magistrale

Parity and Mean-payoff games

25 October 2019

Candidato
Dario Balboni

Relatore
Dott. **Marcello Mamino**
Università di Pisa

Anno Accademico 2018/2019

Contents

Introduction and Historical Notes

This thesis deals with aspects of the algorithmic complexity of some infinite games, called graph games. In particular we will focus on parity games and mean-payoff games. We will survey the state of the art of the problem of solving parity games and mean-payoff games efficiently, a problem which is still open. We will also see how the complexity of solving these games plays an important role in the solution of several interesting computational problems as, for instance, combinatorial linear programming and LP-type problems, scheduling with conjunctive and disjunctive constraints, formal verification of temporal logics and constraint satisfaction problems.

Solving a finite two-players game is elementary, as long as the game is presented as the graph of all possible positions. In general, however, infinite combinatorial games can be nontrivial, for instance only some infinite games (notably Borel games [Mar75]) admit optimal strategies.

Graph games are special among infinite games, in that their strategies are amenable to computation. We will now review the history of the study of graph games and describe their importance, and the progresses made in solving them computationally.

History of graph games

Possibly the first to introduce graph games has been Lord Shapley [Sha53] in the fifties, because they allow to model competitive situations that can have arbitrary duration. For example, one can consider the interaction between the market and a firm as a sequence of moves, each of which changes the state of the market and produces gains or losses for the firm. In this situation there is usually no point at which the firm wins and also there is no point at which the firm suffers a decisive loss; the firm indeed strives to get the highest long-term averaged gain.

Mathematically the outcome of a play generating a sequence $w_1, w_2, \ldots$ of gain or losses can be described as the Cesaro limit of the outcomes

$$\lim_{n\to\infty} \frac{1}{n} \sum_{i\leq n} w_i$$

where the limit is usually expressed with a $\liminf$ to encompass cases where the limit may not exist.

In general Shapley's games were multi-agent and stochastic, namely the outcome of a move is partially random. We will deal however with the simpler case of deterministic two-players game: in contrast with the general case, these subclass always admit optimal strategies, assuming that one of the player strives to maximize the outcome of the play and the other strives to minimize it. This particular case of Shapley's games is called mean-payoff games, which were introduced at the end of the seventies by Ehrenfeucht and Mycielski [EM79].

Mean-payoff games are obviously less suitable to model financial markets. They have found however applications to several areas of computational complexity. Let us mention that mean-payoff are equivalent to scheduling under and/or constraints [MSS04], the sort of problem one needs to solve for instance to prepare a train schedule, and generally when organizing tasks under constraint of overlapping or non-overlapping type. Mean-payoff games have also been used in analyzing various pivoting strategies of combinatorial simplex algorithms [All+14], where proved to be a useful tool to build strong exponential counterexamples.

A different game that turns out to be strictly related comes from a completely different background: formal verification of transition systems. A transition system is any system that can be in one of finitely many states, and that changes state receiving inputs and returning outputs; one is usually interested in proving that a transition system posseses some desiderable properties. For example, viewing a computer program as a transition system, one may be interested in proving that it does always terminate, or that it always executes correctly according to a specification.

Such problems can informally be modeled as a game, where one player is the system that tries to satisfy the constraints and the other is a supposedly malicious environment that tries to make the system fail. Most propositions that one might ask for in a transition system turn out to be expressible in μ-calculus, a formal logic introduced by Kozen [Koz82] in the eighties, which in turn is equivalent to the problem of solving a certain class of graph games called parity games [EJ91; Eme97].

Parity games and mean-payoff games are strictly related; in particular they are two steps of a hierarchy of graph games. At the present state of research, parity games appear to be easier to solve than mean-payoff games. In particular, in a recent breakthrough [Cal+17], parity games have been solved in *almost polynomial-time* while similar results are not in sight for mean-payoff games.

State of the art

In this thesis we are interested in the problem of determining the computation complexity of solving certain graph games. In general, the theory of computation complexity tries to measure in a precise way the amount of resources that are necessary to solve a class of problems with the best algorithm. This theory is peculiar in that almost no practical problem is amenable to an unconditional precise classification: in most cases, one can only

prove upper and lower bounds that result in defining hierarchies of problems.

For instance, the two most prominent classes are P, class of problems that can be solved in a polynomial number of steps in the length of the problem representation, and NP, problems for which, given a solution claim, one can verify if it is really a solution in polynomial time. It is common to consider "tractable" the problems in the class P, while problems NP-complete (maximal elements of the NP class, those that every other problem in NP can be reduced to) are usually considered untractable. For the foreseeable future there is no hope of proving $\mathrm{P} \subsetneq \mathrm{NP}$, even for those so prominent classes. This doesn't however impede of classifying problems in NP as being in P, NP-complete, or NP-intermediate.

We already mentioned parity games and mean-payoff games: they are in the class NP and, by leveraging the inherent symmetry of those games, one can easily see that they must be also inside coNP, the dual class of NP. The problems that, like these, are found in $\mathrm{NP} \cap \mathrm{coNP}$ are rare and have peculiar traits in complexity theory: from one side there is no known algorithm that proves they are in P, and from the other side if a problem in $\mathrm{NP} \cap \mathrm{coNP}$ were found to be NP-complete, a collapse of the polynomial hierarchy to Σ_1^P would occur, which is deemed improbable by experts.

Many rather think that problems in $\mathrm{NP} \cap \mathrm{coNP}$ (graph games also belong to the stricted class $\mathrm{UP} \cap \mathrm{coUP}$ [Jur98]) show tractability traits: it happened in the past that problems in these class later were found to be in P like PRIMES [AKS04] (the problem of establishing primality of a number) or fell in more tractable classes, like the factoring problem that can be solved in polynomial time on a quantum computer [Sho99].

Generally in complexity theory one considers notions of reducibility between problems, where a problem A is said to be reducible to a problem B if there is an algorithm that solves instances of problem A by calling repeatedly an oracle-subroutine for solving B. In the same way, even though the exact complexity of graph games has not yet been established, it is possible to cast them in a hierarchy under the reducibility relation [ZP96; AM09; Jur98], which can be seen in Figure 1. Condon [Con92] has been the first to raise the question of graph games being in P, a problem which is still open for the vast majority of graph games.

Recently the breakthrough of Calude et al. [Cal+17] gave rise to a series of works that adapted previously known algorithm to obtain quasi-polynomial time[1] variants for solving parity games. Among these we would like to mention the succinct progress measures by Jurdzinski and Lazic [JL17], the register-index method of Lehtinen [Leh18], a variation of the Zielonka algorithm by Parys [Par19], and an algorithm based on strategic decompositions by Daviaud et al. [DJL18].

Unfortunately the complexity of mean-payoff games is not as well constrained. There is an obvious polyspace (and hence exptime) bound on their complexity but, that apart, they

[1] We remind the reader that an algorithm is quasi-polynomial when it needs $e^{O(\log^\alpha n)}$ steps before stopping, where n is the problem description size, and $\alpha \in \mathbb{R}$ is a real number. The class of quasi-polynomial algorithms has gained a respectable status since another important but uncorrelated problem, namely graph isomorphism, was solved by Babai [Bab16].

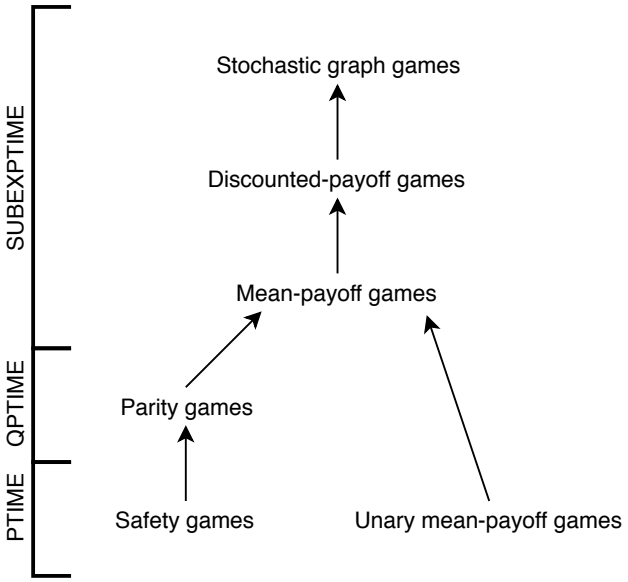

Figure 1: Main kinds of infinite graph games, with arrows meaning polynomial Turing-reductions. Stochastic games have been shown to have equivalent complexity by Andersson and Miltersen [AM09; Mam17], and the only games for which a polytime algorithm is known are safety games and unary-coded mean-payoff games [ZP96].

share the same subexponential upper bound [LP07] as the whole hierarchy of graph games, coming from the theory of LP-type problems [Lud95; Hal07].

Scope of the thesis

We concentrate on parity games and mean-payoff games. We will review known results of structural nature on these games; in particular we will prove that they are positionally determined, i.e. they possess memoryless optimal strategies. We will present reducibility results between graph games, proving that they fit in a hierarchy of problems. We will then focus on their computational complexity, proving the recent result of Calude et al. [Cal+17] and reviewing critically the principal algorithms discovered after the breakthrough. Finally we will propose a new graph game, identifying its position inside the hierarchy, and checking some of its algorithmic aspects.

In Chapter 1 we will define the basic notions for graph games and prove positional determinancy of the basic games we will consider. In Chapter 2 we introduce the notion of game reduction and prove various reductions between games, as well as their complexity status in $\mathrm{NP} \cap \mathrm{coNP}$. In Chapter 3 we present many algorithms to solve such types of games. Finally, in Chapter 4, we give the definition of a new game, which is intermediate between parity games and mean-payoff games, and prove some properties about it.

Chapter 1

Infinite duration Graph Games

1.1 Basic Definitions

We give in this section the necessary definitions for the two-players zero-sum infinite durations graph games, that will be the topic of the whole thesis.

Definition 1.1.1 (Graph Game)**:** A graph game $\mathcal{G}$ is a tuple (W, ν), where

- W is a set, called the set of weights
- $\nu : W^{\mathbb{N}} \to \mathbb{R}$ is the payoff function from sequences of weights to $\mathbb{R}$

Definition 1.1.2 (Game Arena)**:** A game arena $\mathcal{A}$ for a graph game $\mathcal{G} = (W, \nu)$ is:

- An underlying directed graph $G = (V, E)$
- A partition of the vertices V in two subsets V_{Max} and V_{Min}
- A map $w : E \to W$ from the edges to the weight set

We will abuse the word game to mean game arena when the meaning is clear by the context.

Definition 1.1.3 (Game Play)**:** Given an arena $\mathcal{A}$ for a graph game $\mathcal{G}$, a play starting from vertex v_{p_0} is a sequence $v_{p_0}, v_{p_1}, v_{p_2}, \ldots$ of vertices such that $(v_{p_i}, v_{p_i+1}) \in E$. We require that the graph G has no *cul-de-sac*, i.e. every vertex has at least one successor, to ensure that every partial play can continue[1].

Player Min moves (i.e. decides which outgoing edge to follow) when the token is on one of its vertices, i.e. $v_{p_i} \in V_{\text{Min}}$, and similarly for player Max. The sequence of vertices determines a sequence of weights: $a_i = w(v_{p_i}, v_{p_i+1})$ which can be used to

[1]Such requirement is not necessary, since if a vertex of the underlying graph has no outgoing edges, it is customary to say that the player that cannot move loses. Note though, that one can build an equivalent game by adding two additional vertices v_-, v_+ with a loop that makes player Max (respectively Min) lose, and wire each vertex where player Max cannot move to v_-, and each Min vertex with no outgoing edges to v_+. Therefore we won't deal with such cases.

compute the *value of the play* $\nu(v.) = \nu(a.) \in \mathbb{R}$.

The play is won by Max if $\nu(a.) > 0$, and by Min if $\nu(a.) < 0$, and it is a draw otherwise.

There are many questions one can ask given a game arena: can player Max win if the play starts at v_{p_0}? Which is "the maximal value" that he can obtain in a play? Can we describe a strategy that allows him to win? Those questions will be answered through the thesis, first in theoretical terms and then from the point of view of complexity theory. But let's start introducing the type of games we will be talking about.

1.2 Common graph games

We will now describe some of the major graph games that will be studied. As was already said in the introduction, those are particularly interesting because of their complexity status and their fruitful connections with other areas of mathematics.

Note that even though weight sets will be infinite, in every game arena there will only be a finite quantity of those, since the underlying graphs are finite.

Definition 1.2.1 (Parity games)**:** The weights are natural numbers $W = \mathbb{N}$. We are interested in the quantity $\rho(w.) = \limsup_n w_n$, the highest number that occurs infinitely often. The value function can be espressed as $\nu(w.) = (-1)^{\rho(w.)}\rho(w.)$, that is the Min player (also called Odd) wants $\rho(w.)$ to be an high odd number, while Max player (also called Even) wants $\rho(w.)$ to be a high even number.

Parity games have many applications to formal verification theory, because they are the model-checking games corresponding to μ-calculus, a formal logic with fix point operators [Koz82].

Definition 1.2.2 (Mean-payoff games)**:** The weights are integer numbers $W = \mathbb{Z}$. The value function is $\nu(w.) = \limsup_n \frac{1}{n}\sum_{k=0}^{n-1} w_k$, which is the limit of the mean of the weights encountered during the play.

Definition 1.2.3 (Discounted-payoff games)**:** The weights are integer numbers $W = \mathbb{Z}$ and a parameter $0 < \beta < 1$ has to be specified. The value function is then $\nu(w.) = (1-\beta)\sum_{k=0}^{\infty} \beta^k w_k$.

1.3 Strategies and Determinancy

Definition 1.3.1 (Strategy)**:** Given a game arena G, a strategy for player $i = \text{Max}, \text{Min}$ is a function describing what vertex to choose next basing on past history of the play. Formally it is a function $\sigma : \text{Hist}(V) \to V$, where $\text{Hist}(V) = \cup_{k=1}^{\infty} V^k$.

The chosen vertex must be connected through an edge to the previous vertex in the sequence, that is if $\sigma(v_{p_0}, \ldots, v_{p_k}) = v_{p_k+1}$ it must be the case that $(v_{p_k}, v_{p_k+1}) \in E$.

Definition 1.3.2 (Play consistent with a strategy)**:** A play $v_{p_0}, \ldots$ is said to be consistent with a strategy σ for player Max (respectively Min) if when the pebble is on a vertex owned by Max (Min), the next visited vertex in the play is the one chosen by the strategy. More formally if it holds that

$$v_{p_i+1} = \sigma(v_{p_0}, \ldots, v_{p_i}) \qquad \text{whenever } v_{p_i} \in V_{\text{Max}} \text{ (respectively } V_{\text{Min}})$$

Remark 1.3.3: Given a strategy σ for player Max and τ for player Min, there exists a unique play consistent with both strategies and starting from a given vetex v_{p_0}, and the sequence of vertices can be determined by the rules:

$$\begin{aligned} v_{p_i+1} &= \sigma(v_{p_0}, \ldots, v_{p_i}) \quad \text{if } v_{p_i} \in V_{\text{Max}} \\ v_{p_i+1} &= \tau(v_{p_0}, \ldots, v_{p_i}) \quad \text{if } v_{p_i} \in V_{\text{Min}} \end{aligned} \tag{1.1}$$

We will use the notation $\nu_{\sigma,\tau}(v_{p_0})$ to describe the value of the play starting from v_{p_0} which is consistent with both σ and τ, i.e. $\nu_{\sigma,\tau}(v_{p_0}) = w(v_{p_0}, \ldots)$.

Definition 1.3.4 (Determined game)**:** Given a game arena G, a game strating from $v_i \in V$ is said to be determined if $\exists a_i \in \mathbb{R}$ and there $\exists \sigma^*$ a strategy for Max and $\exists \tau^*$ a strategy for Min such that $\forall \sigma, \tau$ strategies respectively of Max and Min it holds that

$$\begin{aligned} \nu_{\sigma^*,\tau}(v_i) &\geq a_i \\ \nu_{\sigma,\tau^*}(v_i) &\leq a_i \end{aligned} \tag{1.2}$$

The value a_i is said to be the *value of the game* starting from v_i. We note that by using $\sigma = \sigma^*$ and $\tau = \tau^*$ one gets that the value of the game is $a_i = \nu_{\sigma^*,\tau^*}(v_i)$.

Definition 1.3.5 (Optimal Strategies)**:** Given a determined game arena G, if σ^* and τ^* are two strategies (respectively of Max and Min player) satisfying equation 1.2, they are said to be optimal strategies.

In other words, an optimal strategy for Max guarantees a minimal outcome against whatever strategy the other player chooses.

Definition 1.3.6 (Winning Set)**:** Given a determined game arena G we will call winning set W_{Max} for player Max (respectively W_{Min} for player Min) the set of vertices for which there exists two optimal strategies for the two players, with $a_i > 0$ (respectively $a_i < 0$).

Definition 1.3.7 (Winning Strategies)**:** Given a determined game arena G, a strategy σ for Max (respectively τ for Min) is said to be winning if it archieves at least zero (at most zero) for every vertex for which it is possible, i.e. if $\forall v \in W_{\text{Max}}$ and $\forall \tau$ strategies of Min (resp. $\forall \sigma$ strategies of Max) it holds $\nu_{\sigma,\tau}(v) > 0$ (resp. $\nu_{\sigma,\tau}(v) < 0$).

We now mention a well-known theorem of Martin [Mar75; Mar85] about determinancy of general infinite duration games, that we state without proof. Such theorem can be successfully used to prove determinancy of our games, but we will prove a more explicit result in the next section.

Remark 1.3.8 (Proving determinancy)**:** Define a *tree* T to be a set of finite sequences such that

- All prefixes of the sequences in T belong to T:

$$\langle x_1, \ldots, x_n \rangle \in T \implies \forall k \leq n \quad \langle x_1, \ldots, x_k \rangle \in T$$

- Every sequence in T can be extended to a longer sequence:

$$\langle x_1, \ldots, x_n \rangle \in T \implies \exists y \quad \langle x_1, \ldots, x_n, y \rangle \in T$$

Denote by $[T]$ the set of all infinite sequences whose initial segments are in T:

$$[T] = \{\langle x_1, \ldots, x_n, \ldots \rangle \mid \forall k \quad \langle x_1, \ldots, x_k \rangle \in T\}$$

A *game on* T is played by two players alternatingly: the first player chooses a_0, the second one chooses a_1, then the first one chooses a_2 and so on. It is required that $\langle a_0, \ldots, a_n \rangle \in T$ for each n.

Given a set $A \subseteq [T]$, we denote by $G(A, T)$ the game on T with the following winning condition: the first player wins the play $\langle a_0, a_1, \ldots \rangle$ if and only if $\langle a_0, a_1, \ldots \rangle \in A$, otherwise the second player wins.

We say that a set $A \subseteq [T]$ is Borel if it is in the Borel σ-algebra of the pointwise convergence topology, i.e. where the open sets are the plays with a predefined initial sequence of moves.

Borel Determinancy Theorem: If $A \subseteq [T]$ is Borel, then $G(A, T)$ is determined.

By modifying our definition of games only a little[2] we can use the Borel determinancy theorem to prove determinancy for our games. This determinancy result is not effective, so that we are interested in a stronger notion of determinancy that also allows one to compute optimal strategies, which will be useful in designing algorithms to solve these games.

For these reasons we are introducing the notion of positional determinancy.

1.4 Positional determinancy

Definition 1.4.1 (Memoryless strategy)**:** Given a graph game arena G, a strategy $\sigma : \text{Hist}(V) \to V$ is said to be memoryless if it is induced by a function $f : V \to V$ in

[2]We should require that the graph on which the game is played is bipartite, that is there are only edges connecting one vertex in V_{Max} to one vertex in V_{Min}. It is not difficult to transform each game arena to be bipartite without modifying the essence of the game.

the following way: $\sigma(v_0, \dots, v_i) = f(v_i)$, i.e. if the next vertex depends only on the current one, and not on past vertices.

Definition 1.4.2 (Strong optimal positional determinancy)**:** A game is said to be strong optimally positionally determined if each player has a memoryless optimal strategy for all starting vertices v_0. These two strategies can be seen as functions $\sigma : V_i \to V$ satisfying $(v_k, \sigma(v_k)) \in E$ for $i = \text{Max}, \text{Min}$.

Remark 1.4.3 (Computing the game value for positionally determined games)**:** Given two positional optimal strategies σ^*, τ^* for the two players, the value of each vertex can be computed in polynomial time for both parity, mean-payoff and discounted-payoff games: for each vertex we have a unique outgoing edge that is selected from the two strategies and the vertices visited in a play are forced to repeat.

We can thus identify in polynomial time a preperiod $v_{p_0}, \dots, v_{p_k}$ of the play, followed by a period of repeating vertices $v_{p_{k+1}}, \dots, v_{p_{k+s}}$ such that $v_{p_{k+s+1}} = v_{p_{k+1}}$.

Given such decomposition in preperiod and period, the game value can be computed for all previously outlined game types by noting that the play value can be split in a part depending on the weights encountered during the preperiod and a part depending only on the weights of the period.

For example, for β-discounted games we have

$$\nu_{\sigma^*,\tau^*} = (1-\beta)\left(\sum_{i=0}^{k} \beta^i w_i + \frac{\beta^{k+1}}{1-\beta^s} \sum_{i=k+1}^{k+s} \beta^{i-k-1} w_i\right)$$

so the only interesting part is detecting where the period begins and how long it is.

Remark 1.4.4 (Solving single-player graph games)**:** For all games that we are going to examine, solving a single-player arena is known to be feasible in polynomial time, since it usually reduces to finding an "extremal" cycle on a directed graph and which can be usually solved by a variation of the Bellman-Ford shortest-path algorithm [Bel58; FF62].

Consider for instance the case of single-player parity games: if the vertices are all owned by Even player, he will be interested in converging to the cycle with the maximal even priority. To compute the path he has to follow we can use an updating algorithm to locally compute the best strategy: start with $\mu : V \to \mathbb{N}^\perp$ given by $\mu(v) = \perp$, i.e. undefined, and iteratively compute $\mu' : V \to \mathbb{N}$ as

$$\mu'(v) = \text{Evenmax}_{(v,u)\in E} \max(w(v,u), \mu(u)) \tag{1.3}$$

by selecting at each step the maximal even priority that can be reached. Computing the transformation $n = |G|$ times, Even is able to know which is the maximal even recheable priority and to converge to it by remembering for each vertex the edge on which the maximum in Equation 1.3 is reached. Therefore computing the optimal

path for single-player parity games can be done in time $O(n^2)$, without counting the time required to do a max operation.

A similar result hold for the other classes of games. For example consider the case of single-player mean-payoff games: if the vertices are all owned by Min player, he will be interested in converging to the cycle with the minimal mean. This case has been solved by Karp [Kar78] via a characterization of the min-cycle.

1.4.1 Different types of positional determinancy

We set aside for a while the question of proving optimal positional determinancy to discuss about other possible definitions of positional determinancy. We will introduce a seemingly weeker notion of positional determinancy – that for every vertex there exists a *winning* positional strategy – and prove that it implies weak optimal positional determinancy, i.e. that there exists an *optimal* positional strategy for every starting vertex. This in turn will imply our strong notion of positional determinancy, namely that there exists a global *optimal* positional strategy, i.e. a single strategy that ensures to reach the vertex value for every vertex at once. This digression is necessary since we will be proving weak winning positional determinancy for all games, but we will need to use strong optimal positional determinancy in the algorithm chapter. A presentation of such lemma was inspired by the work of Björklund, Sandberg and Vorobyov [BSV04].

Definition 1.4.5 (Subgame specified by a strategy)**:** Given a positional strategy σ for Max we denote the subgame G_σ of G obtained by first removing from the arena $\mathcal{A}$ all the edges coming out of vertices in V_{Max} except for those agreeing with σ, and then by removing all vertices unreachable from v_0 via the edges left.

Note that plays in the game G starting in v_i and consistent with the strategy σ correspond exactly to the set of all plays in the game G_σ.

Definition 1.4.6 (Weak winning positional determinancy)**:** A game is said to be weak winning positioanlly determined if there exist a strategy σ for Max and a strategy τ for Min such that:

- If the play starts from a vertex v_0 with non-negative value, then by playing according to σ, Max can win every play starting from v_0.
- If the initial vertex v_0 has negative value, then by playing according to τ, Min can win every play starting from v_0.

Definition 1.4.7 (Weak optimal positional determinancy)**:** A game is said to be weak optimally positionally determined if for every starting vertex v with value α, there exists a strategy σ_v of Max and a strategy τ_v of Min such that:

- By playing according to σ_v, Max ensures that every play starting from v has value which is at least α.

- By playing according to τ_v, Min ensures that every play starting from v has value with is at most α.

We are now going to talk about value recovery: given the existance of weak positional winning strategies for every vertex, can we recover weak positional optimal strategy?

Definition 1.4.8 (Value shifting)**:** We say that a graph game $\mathcal{G}$ admits value shifting if, given any game arena G with a vertex v of value a, a new game arena G' with the same underlying graph as G but with different weights can be computed such that a positional strategy σ is optimal on G starting from v if and only if it is winning on G' starting from v.

Lemma 1.4.9: Let $\mathcal{G}$ be a graph game which is weak winning positionally determined and admits value shifting. Then $\mathcal{G}$ is weak optimally positionally determined.

Proof: Given a game arena G and a vertex v, we use the value shifting property to get a new arena G' such that a positional σ is optimal on G from v if and only if it is winning on G' starting from v. The notion of weak winning positional determinancy for $\mathcal{G}$ exactly says that there is a strategy σ on G' which is winning from v. □

It is now enough to prove that parity games, mean-payoff games, and discounted-payoff games all admit value shifting, because their payoff function is translation-invariant.

Definition 1.4.10 (Translation-invariant payoff)**:** A payoff function $\nu : W^{\mathbb{N}} \to \mathbb{R}$ is said to be translation-invariant if for every sequence $\alpha \in \mathbb{R}^{\mathbb{N}}$ and every constant $c \in \mathbb{R}$ one has:

$$\nu(c + \alpha) = c + \nu(\alpha)$$

where $c + \alpha$ means adding c to every weight in the sequence α.

Lemma 1.4.11: Let $\mathcal{G}$ be a graph game with translation-invariant payoff. Then it admits value shifting.

Proof: Let G be a graph game arena with a vertex v of value a. We can consider the new graph game arena $G' = G - a$ in which a is subtracted to the weight in every edge . In G' the value of v is zero, and therefore a winning strategy σ ensures the exact value of the vertex. By using the same strategy in the original arena, it ensures the original value a. □

Now we are going to see a pair of lemmas that allows us to pass from weak optimal positional determinancy to strong optimal positional determinancy, which was our initial goal. The idea of what follow is that, given a weak optimal positionally determined game arena, an equivalent game can be obtained by fixing the outgoing edge from a single vertex. Iterating such construction, one is able to fix a single outgoing edge from every vertex of one

player without changing the game values, therefore obtaining a global positional optimal strategy for him. To see this we are going to use an intermediate game, where we just fix the starting vertex value.

Definition 1.4.12 (Non-decreasing payoff)**:** A payoff function $\nu : W^{\mathbb{N}} \to \mathbb{R}$ is said to be non-decreasing if there exists a function $f : \mathbb{R}^2 \to \mathbb{R}$ such that:

- For every $a \in \mathbb{R}$ and sequence $\alpha \in \mathbb{R}^{\mathbb{N}}$ we have

$$\nu(a \frown \alpha) = f(a, \nu(\alpha))$$

  where $\frown$ denotes sequence concatenation.

- f is non-decreasing in each variable.

Remark 1.4.13 (Graph games with non-decreasing payoff)**:** We note that if a payoff is prefix-independent, i.e. if $\nu(a \frown \alpha) = \nu(\alpha)$, then it is also non-decreasing. This is the case of parity games and mean-payoff games.

Discounted-payoff also is non-decreasing, since $f(a, \nu) = (1 - \beta)a + \beta\nu$.

Definition 1.4.14 (Game with fixed node value)**:** Given a game arena G and a vertex $v^* \in V$, we can build another game $G[v^* = \alpha]$ where we postulate that whenever a play $v_{p_0}, \ldots$ reaches the vertex v, the play ends with value α, i.e. if $v_{p_i} = v^*$ then $\nu(v_{p_0}, \ldots) = \alpha$.

Lemma 1.4.15: Given a weak optimal positionally determined game arena G with non-decreasing payoff, let $v^* \in V$ be one of its vertex of value α. Then a strategy (even non-positional) σ is optimal on a vertex v in G if and only if it is optimal for the same vertex in $G[v^* = \alpha]$.

Proof: Let σ be an optimal strategy for Max from v in $G[v^* = \alpha]$. We want to prove that it can archieve at least the value $\nu_G(v)$ no matter what the counter-strategy τ of Min is. Let $\pi = v, v_{p_1}, \ldots$ be the play consistent with both σ and τ and starting from v in G. If the play does not passes through v^*, then its value is the same in G and in $G[v^* = \alpha]$.

On the other hand if a play passes through v^*, i.e. $v_{p_i} = v^*$ then we denote by ν' the value of a play in $G[v^* = \alpha]$ and it is easy to see that:

$$\begin{aligned} \nu'(\pi) &= f(w_1, f(\ldots f(w_{i-1}, \alpha)\ldots)) \\ \nu(\pi) &= f(w_1, f(\ldots f(w_{i-1}, \nu(w_i, \ldots))\ldots)) \end{aligned} \tag{1.4}$$

Since f is non-decreasing we also have for any set of real numbers $\{b_i\}_{i\in\mathbb{N}} \subseteq \mathbb{R}$ that $f(a, \min_i b_i) = \min_i f(a, b_i)$ and combining this observation with the definition of the value we obtain $\min_\tau \nu(w_i, \ldots) = \alpha$. This concludes by induction on the number of f that $\nu'(\pi) = \nu(\pi)$ since it holds that

$$f(w, \alpha) = f(w, \min_\tau \nu(w_i, \ldots)) = \min_\tau f(w, \nu(w_i, \ldots))$$

The converse statement trivially holds since α is the value of the vertex v^* in G. □

Definition 1.4.16 (Game with fixed exit edge)**:** Given a game arena G, we can define the fixed exit edge arena $G[v^* \to u]$ as the arena with all exit edges removed from v^* except the one going to u.

Lemma 1.4.17: Given a weak optimal positionally determined game arena $G[v^* = \alpha]$ with non-decreasing payoff, let σ be an optimal positional strategy for Max for the vertex v^*. Then a positional strategy σ which is optimal on $G[v^* \to \sigma(v^*)]$ is also optimal in $G[v^* = \alpha]$.

Proof: It is enough to prove that the value of each vertex remains unchanged. Because of the equality $\nu(a \cap \alpha) = f(a, \nu(\alpha))$ it is also enough to prove that the value of the vertex v^* remains unchanged. Since σ is an optimal positional strategy for Max from v^* in $G[v^* = \alpha]$, the value of the vertex v^* is the minimal value that can be attained in the single-player game $G[v^* = \alpha]_\sigma = G[v^* \to \sigma(v^*)]_\sigma$. □

Lemma 1.4.18 (Proving strong optimal positional determinancy)**:** Let G be a game arena of a weak optimal positionally determined game with non-decreasing payoff. Then G is also strong optimal positionally determined.

Proof: We prove the theorem by induction on the number $m = |E|$ of edges of the game arena. If $m = 1$ there is nothing to prove since the two notions of positional determinancy coincide.

For the inductive case, consider a vertex v^* of value α in G with at least two outgoing edges. By weak optimal positional determinancy there exists a strategy σ attaining maximal value in G from v^*. By the two previous lemmas, an optimal positional strategy ρ in the game $G[v^* \to \sigma(v^*)]$ is also optimal in the game $G[v^* = \alpha]$ and thus in the original game G. The existance of the strategy ρ is a consequence of the inductive hypothesis. □

1.4.2 Proving positional determinancy

Before proving positional determinancy of discounted-payoff games, let us state the contraction principle, which will be used in the proof.

Fact 1.4.19 (Contraction principle)**:** Let (X, d) be a complete metric space and $f : X \to X$ be an L-Lipschitz function with $L < 1$.

Then f admits a unique fix-point $\hat{x} \in X$ to which all iterated f-sequences converge, i.e. fix $x_0 \in X$ and let $x_{k+1} = f(x_k)$, then

$$d(\hat{x}, x_k) \leq \frac{L^k}{1 - L} d(x, T(x))$$

that is $x_k \to \hat{x}$ exponentially fast.

Theorem 1.4.20: Discounted-payoff games are positionally determined.

Idea of the proof: We proceed in multiple steps:

1. We write equations that should determe the value of the game from each vertex.

   To do this, suppose the current vertex is owned by player Max. Then Max will preferrably choose an edge that leads to a vertex such that the new value of the game plus the weight encoutered in the edge is maximal.

   By the contraction principle the system of equations has a unique solution.

2. Such solution induces a positional strategy for both players.

3. We then prove that by playing this strategy, Max player can force the game to have at least a value, and Min player can force the game to have at most the same value. This allows us to conclude that the game is positionally determined.

Proof: Notation: Let $w_{ij} = w(v_i, v_j)$ be the weights along the edges. Let op_i be max if $v_i \in V_{\text{Max}}$ and min if $v_i \in V_{\text{Min}}$.

Step 1: System of Equations: Consider the following system of equations:

$$\begin{aligned} x_i = &\quad \max_j \left((1-\beta)w_{ij} + \beta x_j\right) \quad \text{if } v_i \in V_{\text{Max}} \\ x_i = &\quad \min_j \left((1-\beta)w_{ij} + \beta x_j\right) \quad \text{if } v_i \in V_{\text{Min}} \end{aligned} \tag{1.5}$$

We can define the map $T : \mathbb{R}^{|V|} \to \mathbb{R}^{|V|}$ as $T(x)_i = \text{op}_i \ \{(1-\beta)w_{ij} + \beta x_j\}_{j=1,\dots,n}$. Such map is a contraction in the max-norm $||x|| = \max_i x_i$ because it is the composition of $x \mapsto c + \beta x$ which is β-Lipschitz and of $\max : \mathbb{R}^n \to \mathbb{R}$ or min which are 1-Lipschitz.

Therefore, by the contraction principle, T has a unique fixpoint x that is the iterated limit of every starting point, that is $\forall y \in \mathbb{R}^{|V|}$ we have $x = \lim_{n\to\infty} T^n(y)$.

Step 2: Induced strategies: Given the solution x to the above system of equation we can extract a pair of positional strategies (σ^*, τ^*) for the two players:

- If $v_i \in V_{\text{Max}}$ we can choose $\sigma^*(v_i) \in \text{argmax}_j((1-\beta)w_{ij} + \beta x_j)$, that is $\sigma^*(v_i) = v_j$ where v_j is a vertex such that $(1-\beta)w_{ij} + \beta x_j = \max_k((1-\beta)w_{ik} + \beta x_k)$.
- Similarly, if $v_i \in V_{\text{Min}}$ we can choose $\tau^*(v_i) \in \text{argmin}_j((1-\beta)w_{ij} + \beta x_j)$.

Step 3: Determinancy: We want to prove that by playing accordingly to σ^*, Max can be sure to win at least x_i, against every strategy of Min. We will prove by induction that

$$\left((1-\beta)\sum_{i=0}^{n-1} \beta^i w(v_{p_i}, v_{p_i+1})\right) + \beta^n x_{p_n} \geq x_{p_0}. \tag{1.6}$$

For $n = 0$ we have $\beta^0 x_{p_0} \geq x_{p_0}$ which is trivially true. Suppose now that Equation 1.6 holds for all $k \geq n$: to prove it for $n+1$ we observe that it is enough to prove

that

$$(1-\beta)w(v_{p_n}, v_{p_n+1}) + \beta x_{p_n+1} \geq x_{p_n} \tag{1.7}$$

since by adding it multiplied by β^n to Equation 1.6 one gets the inductive step.

Equation 1.7 can be easily proven:

- If $v_{p_n} \in V_{\text{Max}}$ then Max has chosen the successor node as to maximize equation 1.5 so we have that $x_{p_n} = (1-\beta)w(v_{p_n}, v_{p_n+1}) + \beta x_{p_n+1}$
- If $v_{p_n} \in V_{\text{Min}}$ then, whichever next vertex Min chooses, we surely have from equation 1.5 that $\beta x_{p_n+1} + (1-\beta)w(v_{p_n}, v_{p_n+1}) \geq \min_j(\beta x_j + (1-\beta)w(v_{p_n}, v_j)) = x_{p_n}$

Then we note that $|\beta^n x_{p_n}| \leq \beta^n ||x||$ and so in the limit $n \to \infty$ it tends to zero, so that we obtain $(1-\beta)\sum_{i=0}^{\infty} \beta^i w(v_{p_i}, v_{p_{i+1}}) \geq x_{p_0}$ as we wanted to prove.

Conclusions: Now it is easy to see that Step 3 can be repeated for Min player with a similar reasoning to prove that if he plays according to τ^*, he can ensure that the value of the play is at most x_{p_0}, thereby proving positional determinancy. □

Definition 1.4.21 (Associated DPG to a MPG)**:** Given a mean-payoff game arena G, the associated discounted-payoff game arena G_β of discount factor β has the same underlying graph, vertex ownership and weights as G, but the value of the play $v_{p_0}, v_{p_1}, \ldots$ is given by $\nu = \sum_{k=0}^{\infty} \beta^k w(v_{p_k}, v_{p_k+1})$.

We outline a procedure that will be useful in the proofs of positional determinancy, as well as later in the chapter about game reductions:

Procedure 1.4.22 (Cycle decomposition of a play)**:** Given a play $\pi = \langle v_{p_0}, \ldots \rangle$ we can decompose it into simple cycles (i.e. cycles without repeating vertices) in the following way: we maintain a stack containing a sequence of distinct nodes forming a finite path $u_0, \ldots, u_h$ on the graph, where h is the height of the stack.

Whenever the next vertex from the path π to be considered happens to be already on the stack, we remove the vertices forming the cycle from the top of the stack. Otherwise we push the new vertex onto the stack.

Theorem 1.4.23: Mean-payoff games are positionally determined.

Idea of the proof: We prove this via discounted-payoff games:

1. There exist a sequence of discount factors $\beta_n \nearrow 1$ such that $\exists \lim_{n\to\infty} v_i(\beta_n)$, where $v_i(\beta)$ is the game value of the vertex v_i for the β-discounted game and those games have the same optimal positional strategy.
2. We then deduce that every Min-reachable loop in $G_{\tilde{\sigma}}$ has a value which is bounded by below by the limit $L_i = \lim_{n\to\infty} v_i(\beta_n)$, which proves potional determinancy.

Proof: Step 1: Since positional strategies are finite, we know that there exist a sequence of β_n such that $\forall n$ we have $\tilde{\sigma}$ is an optimal positional strategy for Max in the β_n-discounted game and $\tilde{\tau}$ is an optimal positional strategy for Min in the β_n-discounted game. We now show that such sequence admits values' limit.

One can easily see that $|v_i(\beta)| \leq (1-\beta)\sum_{i=0}^{\infty}\beta^i|w_i| \leq W$, which means that, given a sequence β_n, the sequence $(v_i(\beta_n))_{n\in\mathbb{N}}$ is bounded, therefore there exists a converging subsequence β_{n_k} which admits limit $L_i = \lim_{k\to\infty} v_i(\beta_{n_k})$.

Then $\forall\varepsilon > 0$ there $\exists N$ such that $\forall n > N$ and $\forall\tau$ strategy of Min we have $\nu^{(\beta_n)}_{\tilde{\sigma},\tau}(v_i) \geq L_i - \varepsilon$ because of the definition of limit, and similarly for Min player.

Step 2: Now consider a play on the graph $G_{\tilde{\sigma}}$ starting from v_i which ends in a loop with sequence of weights w_i. We want to deduce that every min-reachable loop in $G_{\tilde{\sigma}}$ has a mean-payoff value of at least $L_i - \varepsilon$. All the limits in the following chain are done when $\beta \to 1$, $w'_j = w_{n_0+j}$.

$$\begin{aligned}
\underbrace{v_i(\beta) - \varepsilon}_{\to L_i-\varepsilon} &\leq \underbrace{(1-\beta)\sum_{i=0}^{n_0-1}\beta^i w_i}_{\to 0} + (1-\beta)\sum_{i=0}^{\infty}\sum_{j=0}^{n_1-1}\beta^{n_0+n_1 i+j}w'_j \\
&= (1-\beta)\beta^{n_0}\left(\sum_{i=0}^{\infty}\beta^{n_1 i}\right) n_1 \underbrace{\frac{1}{n_1}\sum_{j=0}^{n_1-1}\beta^j w'_j}_{\to \frac{1}{n_1}\sum w'_j} \\
&= \underbrace{\beta^{n_0} n_1 \frac{1-\beta}{1-\beta^{n_1}}}_{\to 1}\frac{1}{n_1}\sum_{j=0}^{n_1-1} w'_j \\
&= \frac{1}{n_1}\sum_{j=0}^{n_1-1} w'_j
\end{aligned}$$

This means that the mean-payoff value of each Min-reachable cycle in $G_{\tilde{\sigma}}$ starting from vertex v_i is $\geq L_i - \varepsilon$ for every ε, which means that it is $\geq L_i$.

Positional determinancy: By the previous step we have shown that, using the positional strategy $\tilde{\sigma}$, no matter what Min does, Max archieves $\geq L_i$ on every cycle. Similarly it can be shown that by using $\tilde{\tau}$, Min can ensure that every cycle Max can reach in $G_{\tilde{\tau}}$ has value $\leq L_i$.

Now using Procedure 1.4.22 about cycle decomposition of plays we can show that, by using $\tilde{\sigma}$ and no matter which strategy Min uses, Max can archieve at least L_i in the play. The symmetric argument for Min proves positional determinancy of the games. □

Lemma 1.4.24 (Subsubsequence)**:** Let (X, d) be a metric space and $\{a_n\}_{n\in\mathbb{N}} \subseteq X$ be a sequence. If every subsequence of a_n has a subsubsequence converging to a limit L, and this limit is the same for all subsubsequences, then the original sequence converges to it $a_n \to L$.

Proof: By contradiction assume a_n does not converge to L, then there exists a subsequence a_{n_k} and $\varepsilon > 0$ such that $\forall k \quad d(a_{n_k}, L) \geq \varepsilon$, but the hypothesis implies that a_{n_k} has a subsubsequence converging to L, thus contradicting the inequality. □

Remark 1.4.25 (Value of a mean-payoff game)**:** In the proof we showed that a positional strategy which is optimal frequently on a sequence G_n of discounted games is optimal also on the mean-payoff game and the value of the limit mean-payoff game is exactly the limit of the values of the discounted games which have such positional strategy as optimal.

We would now like to show that $\lim_{\beta \to 1} v_i(\beta) = L_i$, that is that the whole sequence of discounted games converges to the value of the limit mean-payoff. This follows from the *subsubsequence lemma* noting that step 1 of the above proof allows, starting from any sequence of β_n, to extract a sequence of β_{n_k} such that $\lim_{k \to \infty} v_i(\beta_{n_k}) = L_i$ where $L_i = v_i$ is the mean-payoff game value, so that if every subsequence has a subsubsequence that converges to the same limit, the whole sequence converges to it.

Remark 1.4.26 (Optimal positional strategies on an open interval $(\beta_0, 1)$)**:** We can find some positional strategies σ^* and τ^* which are optimal finally in $\beta \nearrow 1$, and not only frequently.

This can be proven by using o-minimality of the real field $(\mathbb{R}, +, \cdot, <, 0, 1)$ noting that the formula $\phi_{\sigma,\tau}(\beta)$ expressing the fact that σ is an optimal strategy for Max and τ is an optimal strategy for Min is a first-order formula in the language of the real field with variables v_i (one for each vertex, representing the values of the game) and the parameter β.

Since for each β there is a unique assignment of the v_i variables that makes the formula true, the formula is true on a finite union of intervals of the real line depending only on β. As we know that positional strategies are finite, it follows that there must be some σ^* and τ^* for which the formula is true on an open interval $(\beta_0, 1)$.

Theorem 1.4.27 (Parity games are positionally determined)**:** Gven a parity game G the winning regions W_E of Even and W_O of Odd partition the set of vertices. Moreover there is one positional strategy for Even (Odd) following which Even (Odd) can win the game starting at any vertex in its winning region.

Idea of the proof: The proof is by induction on the number of priorities, and then proceeds by contradiction on the single priority. We start by partitioning our game into three sets: W_E where Even has a positional winning strategy, W_O where Odd has a positional winning strategy and U where neither of them has a positional winning strategy. By a simple reasoning on priorities in U and the inductive hypothesis we obtain the contradiction.

Proof: We first note that the theorem holds for the base case where there is exactly one

priority. Consider then a game with d priorities, where we assume without loss of generality that d is odd. As noted in the idea, let (W_E, W_O, U) be the positional partition of G. We will show by contradiction that $U = \emptyset$.

First let τ_E and τ_O be the two positional winning strategies from W_E and W_O for Even and Odd respectively.

Now observe that if a vertex $u \in U$ is owned by Even, then u has no edge which takes it to some vertex $v \in W_E$, otherwise u will also be in W_E by setting $\sigma_E(u) = v$. Similarly if $u \in U \cap V_{\text{Odd}}$, then u has no edge which takes it to W_O.

Define $H \subseteq U$ to be the set of vertices from which player Odd can force the game to take an edge of priority d in U, and consider the graph generated by the vertices $U \setminus H$. In this subgraph, if an edge (u, v) has priority d, then u is owned by Even.

Moreover, define K to be the arena obtained by removing from the subgraph $U \setminus H$ the edges of priority d. Then the arena K has at most $d - 1$ priorities and the induction hypothesis holds. K can then be partitioned into winning regions A_O and A_E for Odd and Even respectively. Moreover there is a positional winning strategy σ_O and σ_E for Odd and Even.

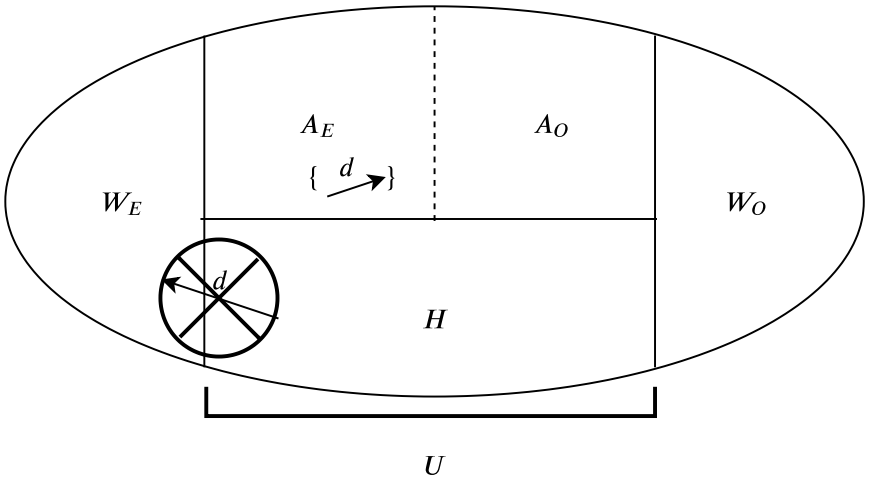

Figure 1.1: Regions of the parity game in the proof

We are now ready to show that U is empty by showing that A_O, H and A_E are empty. Suppose that $A_E \neq \emptyset$ and consider the following positional strategy for Even in the original graph: in W_E even will play according to its strategy τ_E and in A_E it plays according to the strategy σ_E. We show that this strategy is winning for Even starting the game from vertices in A_E in the original arena G.

Assume that there is a vertex in A_E from which Even cannot win: this only happens if there is a vertex $u \in A_E$ owned by Odd and from which there is an edge (u, v) such that $v \in A_O \cup W_O \cup H$. We first note that $v \notin W_O$ since otherwise u would have been in W_O as already noted. If $v \in A_O \cup H$ then the edge (u, v) should have priority d because these are the only vertices that were removed in K. But we already noted that this cannot happen, hence the strategy is a win for vertices in A_E and therefore

these should have been part of W_E.

The set $A_O \cup H$ is also empty because of the following positional strategy for Odd: in W_O he plays accordingly to his strategy in G, in A_O he plays according to his strategy in K and in H he plays the forcing strategy which takes the edge with priority d, always remaining inside of U.

Suppose by contradiction that this is not a winning strategy for Odd from $A_O \cup H$. Then there is a vertex $u \in A_O \cup H$ from which Even can escape. Even cannot force to exit from H to W_E because on H Odd has a strategy to forcing until reaching an edge with priority d inside of U, therefore the terminal vertex is either inside H or in A_O.

Therefore it must be the case that Even can force to exit from a vertex $u \in A_O$ to another vertex $v \in W_E$, but then u should have belonged to W_E already. The outlined strategy is hence a win for Odd and therefore the vertices in $A_O \cup H$ should be in W_O, which concludes the proof by contradiction. □

In order to acquire familiarity with graph games, we proved positional determinancy for them one at a time and using their peculiarities. We could have avoided such work with a theorem of Gimbert and Zielonka [GZ05] which characterizes positionally determined game by some properties of their payoff function. We provide a clear statement of their result in this part.

Remark 1.4.28 (Single-player positional determinancy)**:** The most surprising and useful corollary to their result, Gimbert and Zielonka prove that a two-player game is positionally determined if and only if in every single-player arena (i.e. when $V_{\text{Min}} = \emptyset$ or $V_{\text{Max}} = \emptyset$) the player has a positional optimal strategy.

Obviously single-player arenas are much simpler to treat, and this corollary provides a very useful tool to prove positional determinancy of new games. We now provide the full setting and a precise statement of their result.

Definition 1.4.29 (Preference relation)**:** A preference relation over a set of colours C is a bnary complete, reflexive and transitive relation over the set C^ω of infinite colour sequences.

Complete here means that $\forall x, y \in C^\omega$ either $x \sqsubseteq y$ or $y \sqsubseteq x$.

Intuitively, if $x \sqsubseteq y$ then the player whose preference relation is $\sqsubseteq$ appreciates the sequence y at least as much as the sequence x. On the other hand, if $x \sqsubseteq y$ and $y \sqsubseteq x$ then the outcomes x and y have the same value for our player, we shall say that x and y are equivalent for $\sqsubseteq$.

Notation 1.4.30 (Inverse of a preference relation)**:** We denote by $\sqsubseteq^{-1}$ the

inverse of $\sqsubseteq$: that is $x \sqsubseteq^{-1} y$ if and only if $y \sqsubseteq x$.

We shall also write $x \sqsubset y$ to denote that $x \sqsubseteq y$ but not $y \sqsubseteq x$.

In their treatment, Gimbert and Zielonka only investigate antagonistic games where the preference relation for player Min is just the inverse of the preference relation of player Max. Antagonistic games are then just pairs $(G, \sqsubseteq)$ where G is a finite arena and $\sqsubseteq$ is the preference relation of player Max.

Definition 1.4.31 (Preference relation induced by a payoff)**:** Most often preference relations are introduced by means of payoff or utility mappings: such mapping $u : C^\omega \to \mathbb{R} \cup \{-\infty, +\infty\}$ maps infinite colour sequences to extended real numbers.

This obviously induces a preference relation defined by $x \sqsubseteq_u y \Leftrightarrow u(x) \leq u(y)$.

Now let $\mathrm{Rec}(C)$ denote the family of recognizable subsets of C^*, which means that for every $L \in \mathrm{Rec}(C)$ there exists a finite subset $B \subseteq C$ such that L is a recognizable subset of B^*. For any language of finite words $L \subseteq C^*$, $\mathrm{Pref}(L)$ will stand for the set of all prefixes of the words in L.

Definition 1.4.32 ($[\cdot]$ operator)**:** The operator $[\cdot]$ associates with each language $L \subseteq C^*$ of finite words a set $[L] \subseteq C^\omega$ of infinite words:

$$[L] = \{x \in C^\omega \mid \text{every finite prefix of } x \text{ is in } \mathrm{Pref}L\}$$

We also extend the preference relation $\subseteq$ to subsets of C^ω: for $X, Y \subseteq C^\omega$:

$$X \sqsubseteq Y \Leftrightarrow \forall x \in X \quad \exists y \in Y \quad x \sqsubseteq y$$

and we also write

$$X \sqsubset Y \Leftrightarrow \exists y \in Y \quad \forall x \in X \quad x \sqsubset y.$$

Definition 1.4.33 (Monotone preference relation)**:** A preference relation $\sqsubseteq$ is said to be monotone if for all recognizable sets $M, N \in \mathrm{Rec}(C)$,

$$\exists x \in C^* \quad [xM] \sqsubset [xN] \implies \forall y \in C^* \quad [yM] \sqsubseteq [yN]$$

Definition 1.4.34 (Selective preference relation)**:** A preference relation $\sqsubseteq$ is said to be selective if for each finite word $x \in C^*$ and all recognizable languages $M, N, K \in \mathrm{Rec}(C)$,

$$[x(M \cup N)^* K] \sqsubseteq [xM^*] \cup [xN^*] \cup [xK]$$

Roughly speaking, a preference relation of Max is monotone if at each moment during the play the optimal choice of player Max between two possible futures does not depend on the preceding finite play, while the selective property expresses the fact that player Max

cannot improve his payoff by switching between different behaviours. The main result of Gimbert and Zielonka [GZ05] is the following:

Theorem 1.4.35 (Characterization of positionally determined game)**:** Given a preference relation $\sqsubseteq$, both players have optimal positional strategies for all games $(G, \sqsubseteq)$ over finite arenas G if and only if the relations $\sqsubseteq$ and its inverse $\sqsubseteq^{-1}$ are monotone and selective.

Chapter 2

Reductions between Games

2.1 Computational Problems for Graph Games

We are interested in effectively solving parity games and mean-payoff games, but we have not yet formally defined what we mean by "solving". In the literature we find several slightly different notions of solution, which can be generally summarized as follows.

Problem 2.1.1 (Qualitative Solution)**:** A graph game $\mathcal{G}$ is said to be solved qualitatively if, given a game arena, we are able to tell effectively which player wins from each starting vertex.

Problem 2.1.2 (Quantitative Solution)**:** A graph game $\mathcal{G}$ is said to be solved quantitatively if, given a game arena, we are able to compute the game-values of each vertex.

Problem 2.1.3 (Strategic Solution)**:** A graph game $\mathcal{G}$ is said to be solved strategically if, given a game arena, we are able to compute an optimal positional strategy for both players.

We aim to study the computation complexity of these problems for parity games and mean-payoff games. We have already seen in Remark 1.4.3 that for deterministic games from a *strategic solution* one can easily compute – in polynomial time – the game values, i.e. the *quantitative solution*. Moreover, it is obvious that given the value of each vertex, we are able to solve the game qualitatively since this only requires a comparison between the actual vertex value and zero.

One may also see that in case of translation-invariant payoffs, that is if adding a constant c to all weights of an arena does increase the game value of all vertices by c, being able to solve a game qualitatively allows us to also solve it quantitatively up to a finite precision by binary search on vertex values. Other connections between various types of solutions, such as Strategy Recovery, which consists in trying to go from a quantitative solution to

a strategic solution, have been analyzed in the literature. The complexity class of the various types of solutions is usually the same, therefore we will mainly focus on solving games qualitatively.

2.2 Brief Introduction to Complexity Theory

We will be mainly interested in decision problems, that is given a binary string of finite length the goal is to decide if it belongs to a certain language or not. By language we mean a fixed subset of $\{0,1\}^*$, consisting of all binary strings of finite length.

2.2.1 Turing Machines

A Turing machine is a machine with a finite number of states, that can read from an input tape, write on an output tape and has various working tapes that are read-write (see Figure 2.1).

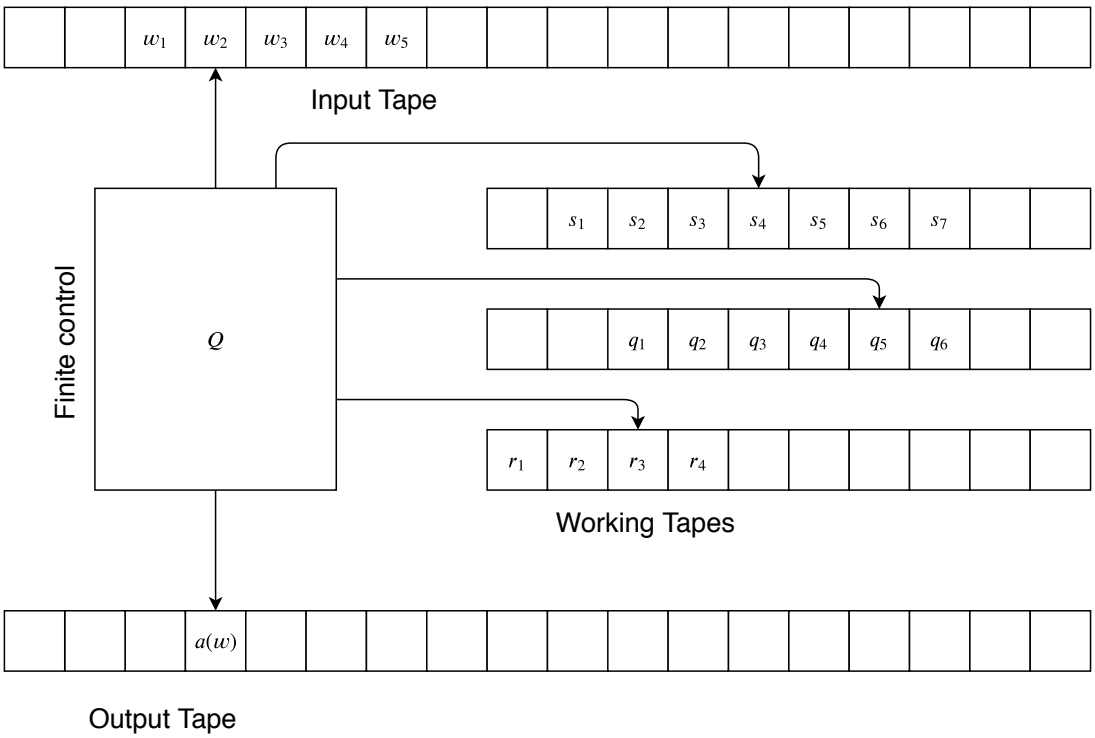

Figure 2.1: Multitape deterministic Turing Machine

Each tape has an infinite number of cells in both directions. At each time step the machine reads a symbol from each of its tapes at the current head positions. The machine then uses its current internal state q and the read symbols $s_1, \ldots, s_k$ to know in which state q' the machine will be next, which symbols $s'_1, \ldots, s'_k$ it will write on the current head positions, and in which position $m_1, \ldots, m_k$ the heads of each tape will move, either left or right.

This information is implemented as a finite table which fully describes the machine behaviour. Mathematically, each Turing Machine operating on k tapes with alphabet Σ and

interal state set S can be described by a function $S \times \Sigma^k \to \Sigma^k \times \{-1, 0, 1\}^k$.

There exists an initial state from which the machine starts and some final states in which the machine halts. The final states can be either accepting or rejecting. A run of the machine M on a string s, written on the input tape before the machine start, determines whether s belongs or not to the language specified by the machine: if the final state is accepting, s belongs to it, and if the final state is rejecting, s does not belong to the language.

Until now we have described what is known as deterministic Turing machine. Another important class of machines are the nondeterministic ones, where at each step the machine may transition to multiple internal states and thus continue executing among different execution paths at the same time.

2.2.2 Main complexity classes

We give the definition of some complexity classes for decisional problems:

- DTIME($t(n)$): A language $L \in$ DTIME($t(n)$) if there exists a *deterministic* Turing machine M that always halts in at most $t(n)$ steps (n is the input length) such that the language is the set of accepted strings of the machine.

  The most famous class in DTIME is the class P of languages which are deterministically decidable in polynomial time, that is the union of DTIME($p(n)$) for all polynomials p.

- DSPACE($t(n)$): A language $L \in$ DSPACE($t(n)$) if there exists a *deterministic* Turing machine that always halts such that the language is the set of accepted strings which use at most $t(n)$ tape cells.

- NTIME($t(n)$): A language $L \in$ NTIME($t(n)$) if there exists a *nondeterministic* Turing machine that always halts in at most $t(n)$ steps, and such that the language is the set of strings which result in at least a final accepting state, i.e. for which there is at least an accepting execution path. The class NP is the union of NTIME($p(n)$) for all polynomials p.

- coNTIME($t(n)$): A language $L \in$ coNTIME($t(n)$) if there exists a *nondeterministic* Turing machine that always halts in at most $t(n)$ steps and such that the language is the set of string for which every final state is accepting, i.e. the string is rejected if there is at least a rejecting path. The class coNP is defined analogously to NP.

2.2.3 Reductions among problems

We want to say that some computation problems are "more difficult" than others. To formally do so, we introduce the notion of reduction which basically say that problem B is more difficult than problem A if we can use a solver for B to also solve A. The exact way in which one can use such a solver is important, and different constraints on it give rise to different notions of reduction:

Definition 2.2.1 (Polynomial-time Turing reduction)**:** We say that a language A is Turing-reducible to B if there exists a deterministic machine $M \in \mathrm{P}$ which can decide A having access to an oracle for B, i.e. if the machine can call a black-box solver for the problem B as a subroutine and such calls have constant execution time.

Definition 2.2.2 (Polynomial-time many-one reduction)**:** We say that a language A is many-to-one-reducible to B if there exists a polynomial-time computable function f such that $x \in A \Leftrightarrow f(x) \in B$.

In other words, Turing reducibility means that having a subroutine for B helps in solving A. Whereas many-one reducibility means that the program solving A may only call the subroutine for B once and then just return the answer of the subrouting without doing any further calculation.

The motivation for introducing these types of reductions is the following lemma:

Proposition 2.2.3: Let $A \in \mathrm{P}$ a computational problem and let B another computational problem which is polynomial-time Turing reducible to A. Then $B \in \mathrm{P}$.

Proof: Let M_A be the deterministic Turing machine solving A, and let M_B be the deterministic Turing machine which solves B with A as oracle. We can substitute each call of M_B to the oracle for A with an execution of the machine M_A to obtain a deterministic Turing machine without oracles that is able to solve B.

In other words we can embed the states and the transitions of the machine A inside the states of the machine B corresponding to oracle calls. If the machine M_A always halts in time $p_A(n)$ and the machine M_B stops in time $p_B(n)$, then the newly built machine can call the oracle at most $p_B(n)$ times, and each call can have an input of length at most $p_B(n)$, thus the total machine running time is at most $p_A(p_B(n))p_B(n)$ which is still polynomial. □

Observe that a many-one reduction is also a Turing reduction, so that using many-one reduction allows for a finer distinction between complexities.

Remark 2.2.4 (Stability of complexity classes under reductions)**:** Note that the classes NP and coNP are not stable for polynomial-time Turing reductions, but are stable for polynomial-time many-one reductions. To see this one can observe that the complement $\overline{L}$ of a language L is Turing-reducible to L, which is exactly the distinction between the classes NP and coNP: a language $L \in \mathrm{NP}$ if and only if its negation $\overline{L} \in \mathrm{coNP}$.

On the other hand their intersection $\mathrm{NP} \cap \mathrm{coNP}$ is preserved under polynomial-time Turing reductions.

2.3 Reductions for Graph Games

We mentioned that some graph games are "more difficult" than other; now that we have introduced the computational problems, we may give a more specific notion of difficulty by using polynomial-time many-one reductions of the *qualitative solution problem*. For each reduction we will also discuss whether it is also a many-one reduction for the *strategic solution problem*.

Definition 2.3.1 (Game Translation)**:** A translation from a graph game $\mathcal{G} = (W, \nu)$ to a graph game $\mathcal{G}' = (W', \nu')$ is a family of functions $(F_n)_{n\in\mathbb{N}}$, one for each graph dimension, where $F_n : W_{\mathcal{G}} \to W_{\mathcal{G}'}$.

Remark 2.3.2 (Applying a game translation to a game arena)**:** Given a game arena $\mathcal{A} = (G, V_{\text{Max}}, V_{\text{Min}}, w)$ for the graph game $\mathcal{G}$, we can obtain a game arena for the graph game $\mathcal{G}'$ by applying to it the game translation F and thus obtaining the game arena $\mathcal{A}'$ with $G' = G$, $V'_{\text{Max}} = V_{\text{Max}}$, $V'_{\text{Min}} = V_{\text{Min}}$ and $w'(e) = F_{|G|}(w(e))$.

Note that the only thing that changes are the edge weights, as well as the payoff function. Thus this reduction is stricter than the notion of many-one reductions. Nevertheless, for our games this is already enough to prove reducibility.

Definition 2.3.3 (Strong reduction)**:** We say that a graph game $\mathcal{G}$ is *strongly reducible* to another game $\mathcal{G}'$ if and only if there exist a game translation $(F_n)_{n\in\mathbb{N}}$ that is computable in polynomial time and, given a $\mathcal{G}$-arena $\mathcal{A}$, a player wins from vertex v_i in $\mathcal{A}$ if and only if the same player wins from vertex v_i in $F(\mathcal{A})$.

In other words it allows us to restate the qualitative solution problem of $\mathcal{G}$ in terms of the same problem for $\mathcal{G}'$.

Definition 2.3.4 (Strategic reduction)**:** A graph game $\mathcal{G}$ is said to be *strategically reducible* to another game $\mathcal{G}'$ if and only if there exist a game translation $(F_n)_{n\in\mathbb{N}}$ that is computable in polynomial time and, given a $\mathcal{G}$-arena $\mathcal{A}$, a positional strategy σ_i for player $i = \text{Max}, \text{Min}$ of the arena $F(\mathcal{A})$ is optimal for the game $\mathcal{G}'$ if and only if σ_i is an optimal positional strategy of the arena $\mathcal{A}$ for the game $\mathcal{G}$.

In other words it allows us to restate the strategic solution problem of $\mathcal{G}$ in terms of the same problem for $\mathcal{G}'$.

To be able to link the given definitions to the corresponding notions of reducibility for computational problems, one has to be explicit about the way in which graph games are codified as binary strings. We note that reasonable codification of the graph structure are easily converted into one another in polynomial-time and are thus equivalent for our aim. In cases where an explicit coding is needed we assume the underlying graph coded as an adiacency matrix, and numbers written in binary.

2.4 A graph game hierarchy

We will now examine some reductions between the games we presented that, as anticipated, can be casted in a game hierarcy (see Figure 2.2).

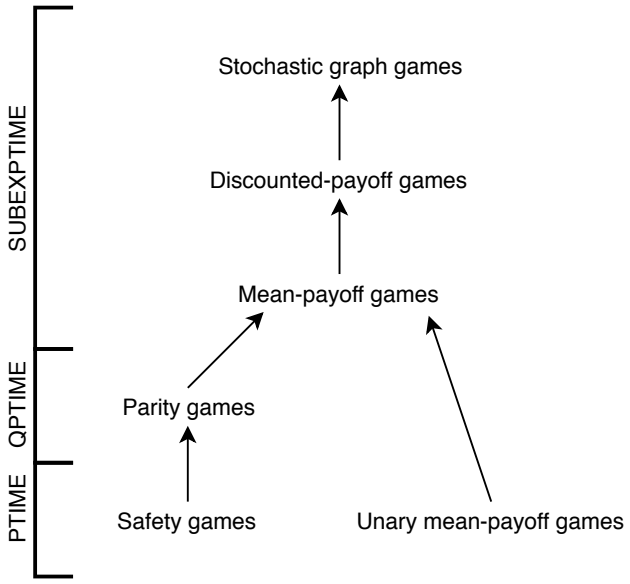

Figure 2.2: Reduction graph between many kinds of game

We recall Procedure 1.4.22 about cycle decomposition of a play that will be useful in proving the reductions between games and then we will see the actual proofs of the reductions.

Procedure 2.4.1 (Cycle decomposition of a play)**:** Given a play $\pi = \langle v_{p_0}, \ldots \rangle$ we can decompose it into simple cycles (i.e. cycles without repeating vertices) in the following way: we maintain a stack containing a sequence of distinct nodes forming a finite path $u_0, \ldots, u_h$ on the graph, where h is the height of the stack.

Whenever the next vertex from the path π to be considered happens to be already on the stack, we remove the vertices forming the cycle from the top of the stack. Otherwise we push the new vertex onto the stack.

Theorem 2.4.2 (Parity games are reducible to mean-payoff games)**:** The game translation $F_n(p) = (-1)^p n^p$ is a reduction from parity to mean-payoff games.

Idea of the proof: First we give the game translation and we prove that it works well on cycles. After that we show the reducibility by using the cycle decomposition outlined above.

Lemma 2.4.3: Let σ be a memoryless strategy for player Max in G and let $F_n(p) = (-1)^p n^p$ be the game translation to apply. Then:

1. For every simple cycle c in G_σ the highest priority of a vertex on c is even if and only if the sum of the weights of the edges on c in $F(G)$ is nonnegative.

2. If σ is a winning strategy for player Max from v_0 in G then for every simple cycle c in G_σ the highest priority of a vertex appearing on c is even.
3. If σ is a winning strategy for player Max from v_0 in $F(G)$ then for every simple cycle c in $F(G)_\sigma$ the sum of the weights of the edges of c is nonnegative.

Proof (of the lemma)**: Fact 1:** This follows trivially by the fact that the weights are spaced enough: the biggest cycle can have at most n vertices, and at most $n-1$ of them can have priority at most p (odd), while there is at least one with priority $p+1$ (even), so that the sum is positive.

Fact 2: Suppose that there exists a simple cycle c with the highest priority of a vertex on c being odd. Then player Min can force the play from v_0 to c and also to stay in c indefinitely and thus win. This however contradicts our assumption that σ is a winning strategy for player Max.

Fact 3: This is proven in the same way as Fact 2. □

Proof (of the theorem)**:** Suppose that player Max has a winning strategy from v_{p_0} in the game G and let σ be one of its positional winning strategies from v_{p_0}. We show that the strategy σ is also a winning strategy for player Max from v_{p_0} in $F(G)$. In order to do that we have to argue that for every play $\langle v_{p_0}, v_{p_1}, \ldots \rangle$ consistent with the strategy σ the following inequality holds:

$$\liminf_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} w(v_{p_{i-1}}, v_{p_i}) \geq 0$$

In the cycle decomposition of the play, whenever we remove a cycle from the top of the stack the sum of the weights of the edges on the cycle is nonnegative. In this way only the weights of the edges which are on the stack may sum up to a negative value. Hence:

$$\liminf_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} w(v_{p_{i-1}}, v_{p_i}) \geq \liminf_{n\to\infty} \frac{1}{n} \sum_{i=1}^{h(n)} w(u_{i-1}, u_i) = 0$$

because $h(n) \geq |V|$, so the absolute value of the sum $\sum_{i=1}^{h(n)} w(u_{i-1}, u_i)$ is bounded by a constant.

To finish the proof of the theorem we also have to show the reverse, i.e. if player Max has a winning strategy from v_0 in $F(G)$, then player Max also has a winning strategy from v_0 in G. This can again be proven by restricting to memoryless strategies and applying the same technique of decomposing plays in $F(G)_\sigma$ into simple cycles and using the facts from the previous lemma. □

We have proven that parity games are reducible to mean-payoff games, but note that optimal positional strategies are not preserved under this reduction: an example of this fact can be seen in Figure 2.3, where square vertices are owned by Even and circle vertices

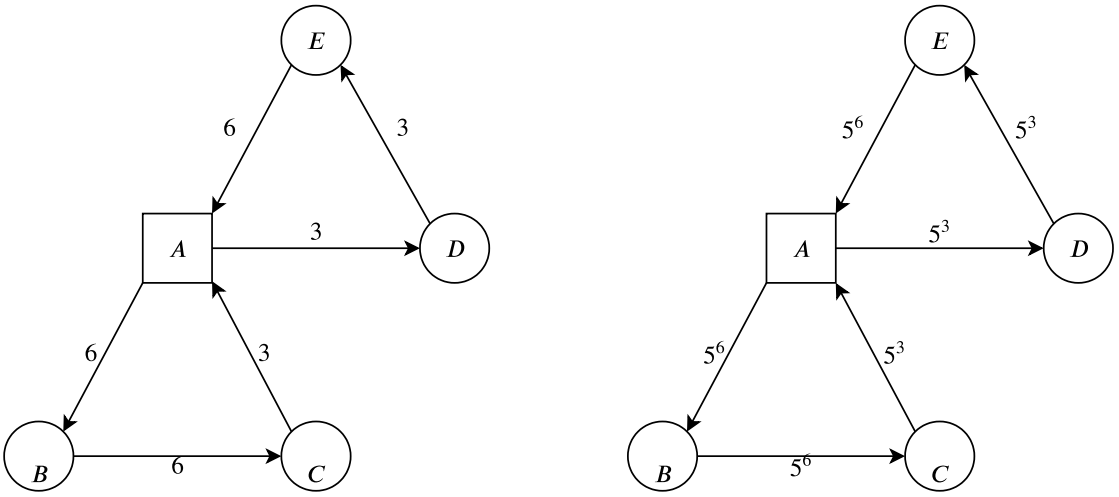

Figure 2.3: A reduction from parity games to mean-payoff games in which optimal positional strategies are not preserved.

are owned by Odd. The maximal priority in the parity game is 6 and there are two distinct cycles with a different number of occurrences of it (the cycle $ABCA$ has two occurrences, while the cycle $ADEA$ has only one).

The two strategies are equally optimal for player Even in the parity game, but the one cycling on $ADEA$ is suboptimal in the mean-payoff game.

Theorem 2.4.4 (Mean-payoff games are reducible to Discounted-payoff games)**:** The game translation on weights is the identity, but the value of β to be choosen has to satisfy

$$\beta \geq 1 - \frac{1}{4n^3W}$$

Notice that the size of a binary expansion of a β satisfying the relation $\beta \geq 1 - \frac{1}{4n^3W}$ is bounded polynomially in n and $\log W$ and thus the reduction that we obtain is a polynomial-time many-one reduction.

To prove such a theorem we need a quantitative lemma about relation between values of vertices of an arena for mean-payoff games and β-discounted games.

Lemma 2.4.5: Given a game arena $\mathcal{A}$, let $\beta \in (0, 1)$ and let $v(\beta)$ and v be respectively the value of the β-discounted game and the value of the mean-payoff game played on the same arena starting at $a \in V$. Then it holds

$$v - 2n(1-\beta)W \leq v(\beta) \leq v + 2n(1-\beta)W$$

Proof (of the theorem)**:** By choosing $\beta \geq 1 - \frac{1}{4n^3W}$ it is easy to verify that $|v(\beta) - v| \leq \frac{1}{2n(n-1)}$, and v can be obtained from $v(\beta)$ by rounding to the nearest rational with a denominator less than n.

This allows to compute the value v of the mean-payoff game and thus to compute

the winning player. □

Proof (of the lemma)**:** Consider the outcome of a discounted game in which player Max uses a positional optimal strategy for the *non-discounted game* and player Min uses a positional optimal strategy to counter the strategy of player Max. The outcome of such a game clearly supplies a lower bound on the value $v(\beta)$ of the discounted game. The play in such a case consists of path of length k, followed by a cycle of length l which is repeated indefinitely as per remark 1.4.3, where $0 \leq k \leq n-1$, $1 \leq l \leq n$ and $k + l \leq n$.

Assume for the moment that all the edge weights are non-negative and let $w_0, \ldots, w_{l-1}$ be the weights of the edges in the cycle formed. As player Max uses an optimal strategy for the non-discounted game we get that $\sum_{i=0}^{l-1} w_i \geq lv$. The outcome of the discounted game is then at least

$$\begin{aligned}(1-\beta)\beta^k\left(\sum_{i=0}^{l-1} w_i\beta^i\right)\left(\sum_{j=0}^{\infty}\beta^{jl}\right) &= \frac{(1-\beta)\beta^k}{1-\beta^l}\sum_{i=0}^{l-1} w_i\beta^i \\ &\geq \frac{(1-\beta)\beta^{k+l-1}}{1-\beta^l}\sum_{i=0}^{l-1} w_i \\ &\geq \frac{\beta(1-\beta)}{1-\beta^l}\beta^{k+l-1}v\end{aligned}$$

As $\frac{l(1-\beta)}{1-\beta^l} > 1$ and $\beta^{k+l-1} > \beta^n > 1 - n(1-\beta)$, this is at least $(1 - n(1-\beta))v$.

In the general case where the edge weights are not assumed to be non-negative, one may simply add W to each weight, thus changing the value and outcome of the game exactly by W and making all the weights non-negative.

By applying the previous inequality to the resulting non-negative game we get that

$$v(\beta) + W \geq (1 - n(1-\beta))(v + W)$$

or equivalently that

$$v(\beta) \geq v - n(1-\beta)(v+W) \geq v - 2n(1-\beta)W$$

and the opposite inequality is proved in a similary way. □

2.5 Stochastic games: introducing chance

If we allow our games to also contain random vertices (denoted by V_R), we obtain a whole set of interesting graph games. Their complexity is equivalent [AM09; Mam17] so we will only define simple stochastic games, introduced by Condon [Con92]. The reduction from discounted-payoff games to simple stochastic games was then proven by [ZP96].

Definition 2.5.1 (Simple Stochastic Games)**:** When the play is on a vertex labeled random, then the exit edge is chosen uniformly at random.

The graph also has a special vertices called the 1-sink. The game ends when the play reaches the sink vertex: player Max wins if the play reaches the 1-sink vertex and player Min wins if the play never reaches the 1-sink.

The value to be Maximized is the expected probability of a play ending in the 1-sink.

Simple Stochastic Games are used to model reactive systems, where random vertices are used to model stochastic environmental changes. We briefly note that the qualitative version asks if the expected probability p of a play ending in the 1-sink is greater than $\frac{1}{2}$.

Simple Stochastic Games also share with parity games and mean-payoff games the property of having positional strategies, a proof of which can be found in Peters and Vrieze [PV87] or in Condon [Con92].

Theorem 2.5.2 (Positional strategies for simple stochastic games)**:** Both players of a simple stochastic game have optimal pure stationary strategies, i.e. they do not make probabilistic choices in choosing a move and each player chooses the same move from a vertex every time that vertex is reached.

Now, fixed σ and τ strategies respectively of Max and Min player, consider the graph $G_{\sigma,\tau}$: let $v_{\sigma,\tau}(i)$ denote the probability of reaching the 1-sink in a play (remember that now each vertex owned by a player has only one exit), then we have the equations:

$$\begin{aligned} v_{\sigma,\tau}(i) &= \frac{1}{|i \to j|} \sum_{i \to j} v_{\sigma,\tau}(j) \text{ on average vertices} \\ v_{\sigma,\tau}(i) &= v_{\sigma,\tau}(j) \text{ on player vertices, for the unique arc } i \to j \end{aligned} \tag{2.1}$$

Remark 2.5.3 (Expanding definition of simple stochastic games)**:** Note that we can expand the definition of simple stochastic games to also include a 0-sink, a special vertex which has a single leaving edge which is a loop, since this provides a way to ensure that it is never left, and so is winning for player Min.

In this context we can consider the halting probability of a simple stochastic game, which is the probability that a play ends in one of the two sink vertices. Restricted to SSG with halting probability equal to one, the definition of SSGs can be formulated symmetrically in the two players, where the quantity to maximize is still the probability of ending in the 1-sink vertex for Max, while it is ending in the 0-sink vertex for Min.

We will call *halting simple stochastic game* this variation of simple stochastic game.

Halting simple stochastic games are much simpler to reason about and will be employed in what follows.

Lemma 2.5.4 (Unique solution of play equation for simple stochastic games)**:** Let G be a halting simple stochastic game arena, and let σ and τ be strategies of

Max and Min respectively. Equation 2.1 has a unique solution, which is also the solution of the *value equation*

$$v = Qv + b$$

where

- $b(i)$ is the probability to reach the 1-sink from vertex i in a single step
- Q is the stochastic matrix of transition probabilities in $G_{\sigma,\tau}$, i.e. $Q_{ij} = \mathrm{pr}(i \to j)$

Idea of the proof: It is easy to see that if v is a vector of non-negative entries that satisfied $v = Qv + b$, then it is also a solution to the value-equation (Equation 2.1). Moreover, the equation $v = Qv + b$ has a unique solution if and only if $(I - Q)$ is invertible.

The idea is to show that $\lim_k Q^k = 0$, from which follows that $(I - Q)$ is invertible and all entries of $(I - Q)^{-1}$ are non-negative, and the entries along the diagonal are strictly positive, which allows to prove that the unique solution of the equation $v = Qv + b$ is non-negative.

Proof: Invertibility: We will now show that $\lim_{l\to\infty} Q^l = 0$: note that the ij-th entry of Q^{mn} is the probability of reaching vertex j from vertex i in exactly mn steps. When $m = 1$, the sum of the terms of the ith row of Q^{mn} is at most 1 minus the probability of reaching a sink vertex of $G_{\sigma,\tau}$ from i in n steps. Since a sink vertex is reachable from i, there must be a path of length $\leq n$ in $G_{\sigma,\tau}$ from i to a sink. Hence the probability of reaching a sink from i in n steps is at least $\frac{1}{r^n}$, where r is the maximal number of exit edges of a vertex in V_R.

Therefore, the sum of the terms in any row of Q^n is at most $1 - \frac{1}{r^n}$, and by induction on m one obtains that the sum of the terms in any row of Q^{mn} is at most $(1 - \frac{1}{r^n})^m$. Since all terms of Q^l are non-negative for any l, it follows from this that, as $l \to \infty$, $Q^l \to 0$. Now, since $\lim_{l\to\infty} Q^l = 0$, 1 cannot be an eigenvalue of Q. Hence $I - Q$ is invertible.

Positiveness: Consider now the equation

$$(I - Q^l) = (I - Q)(I + Q + Q^2 + \ldots + Q^{l-1})$$

and multiply it by $(I - Q)^{-1}$ to the left:

$$(I - Q)^{-1}(I - Q^l) = I + Q + Q^2 + \ldots + Q^{l-1}$$

Taking the limit for $l \to \infty$, the left hand side of the equation is $(I-Q)^{-1}$; hence the limit of the right hand side must exist and the equality $(I-Q)^{-1} = I+Q+Q^2+\ldots$ holds. This means that every entry of $(I-Q)^{-1}$ is non-negative and the entries along the diagonal are strictly positive. □

Theorem 2.5.5 (Discounted-payoff games are reducible to simple stochastic games)**:** For every given discounted-payoff game with discount factor λ there is a polynomially sized equivalent simple stochastic game.

Idea of the proof: We first rescale the weight range to get all positive rational weights, noting that it can be done since discounted payoff games are linear games. A transition of the discounted-payoff game can then be simulated using the trick shown in Figure 2.4.

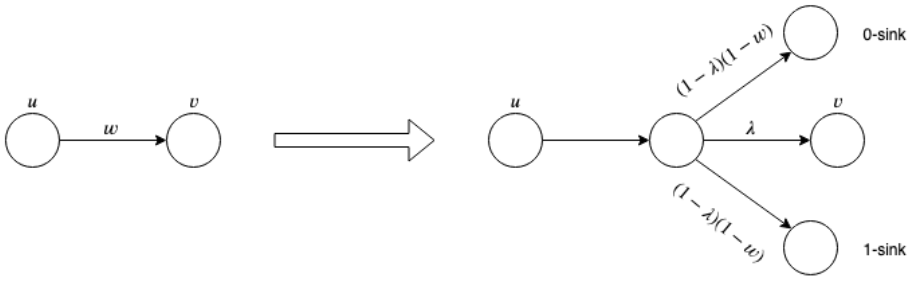

Figure 2.4: Simulating a transition of a discounted-payoff game

Proof: Let $G = (V_{\text{Max}} \cup V_{\text{Min}} \cup V_R, E)$ be a discounted-payoff game with discounting factor λ.

If we add a constant c to all the weights of the game, the value of the game is increased by c, and if we multiply all the weights of the game by a constant $c > 0$, the value of the game is multiplied by c.

We can therefore scale the weights so that they will all be rational numbers in the interval $[0, 1]$. If the original weights were in the range $\{-W, \ldots, 0, \ldots, W\}$, then the new weights will be rational numbers with denominators and numerators in the range $\{0, 1, \ldots, 2W\}$.

We construct in the following way a simple stochastic game $G' = (V', E')$ with the same value as the scaled discounted-payoff game G with discounting factor λ: each edge (u, v) with weight w in G is replaced by the construction shown in Figure 2.4, where the added vertex is a random vertex with fixed escaping probabilities.

The simple stochastic game G' is halting, as in each transition there is a probability of $1 - \lambda$ of reaching a sink vertex. The values of the vertices of the discounted-payoff game G satisfy Equation 1.5. The values of the vertices of the simple stochastic game G' satisfy the set of equations given in Lemma 2.5.4.

These two sets of equations become identical once the intermediate variables, that correspond to the intermediate vertices introduced by the described transformation, are eliminated. As this set of equations has a unique solution, the values of the two games are equal. Moreover, the transformation of G to G' can clearly be carried out

in polynomial time, which completes the description of the reduction. □

2.6 Complexity Status of Graph Games

As we mentioned in the introduction, all of the games we have seen are in the complexity class NP ∩ coNP, and even in the more restricted class UP ∩ coUP, where the checkable certificates are required to be unique. In this section we will prove that all games lie in NP ∩ coNP: having alredy proven reductions between games, it will suffice to show that the qualitative problem for simple stochastic games is in NP ∩ coNP.

Moreover, restricting to halting simple stochastic games, it suffices to show that the problem is in NP; from this we know it must also belong to coNP because of the symmetry between players in halting simple stochastic games.

Theorem 2.6.1: Quantitative solution of simple stochastic games is in NP ∩ coNP.

Proof: As already noted, it is enough to prove containment in NP because of the inherent symmetry of the game. Using Lemma 2.5.4 one may prove the result with the reformulation: non-deterministically claim a value v for vertices, values for b and for non-random places of Q which satisfy constraints on being a positonal strategy and then check it by using Equation 2.1. The only possible problem is that one should be sure that the values of vetices can be written in a polynomial number of bits in the graph description. This is what we prove in the next lemma, originally proved by Condon [Con92]. □

Lemma 2.6.2: The value of a simple stochastic game G with n non-sink vertices is of the form $\frac{p}{q}$ where p and q are integers such that $0 \leq p, q \leq n^n$.

Proof: Consider the solution to the equation $(I - Q)v = b$, for Q, v, b defined as in the previous lemma. By Lemma 2.5.4 we know that $\det(I - Q) \neq 0$, so defining $A = I - Q$, by the cramer rule we know that $v = \frac{\det(A_i)}{\det(A)}$ where A_i is the matrix formed by replacing the i-th column of A by the column vector b.

If we denote by r_i the outdegree of the node i in the game $G_{\sigma,\tau}$, it is trivial to note that $\forall i, j$ one has $Q_{ij} = 0 \vee Q_{ij} = \frac{1}{r_i}$ and $b_i = 0 \vee b_i = \frac{1}{r_i}$. Therefore all matrices A_i and A have, on their k-th row, at most $r_k + 1$ rational positive numbers, everyone of which has a denominator of r_k and a numerator $\leq r_k$. We will now prove that every square matrix M which satisfies such condition has a determinant which is a rational number $\frac{p}{q}$ with $0 \leq p \leq R^n$ where $R = \max_k r_k$ and $q = \prod_i r_i$, from which the required bound will be derive.

The proof will be by induction on the size n of the matrix: it is clear that such bound holds if $n = 1$. For the inductive case consider the laplace row-expansion of

the determinant of M on the first row:

$$\det(M) = \sum_{i=0}^{r_1} \frac{\alpha_i}{r_1} \det(M_i) = \frac{1}{r_1} \sum_{i=0}^{r_1} \alpha_i \frac{\beta_i}{q_i} = \frac{1}{q} \sum_{i=0}^{r_1} \alpha_i \beta_i$$

where the second equality follows by the inductive hyphothesis. This proves the required bound on the numerator.

Let now $q = \prod_i r_i$ and let p and p_i be the numerators of the determinants of A and A_i respectively. Then the game values $v = \frac{\det(A_i)}{\det(A)} = \frac{p_i}{p}$, each of which is $\leq R^n \leq n^n$. □

A proof that the deterministic games are in UP ∩ coUP can be obtained by carefully analyzing discounted-payoff games: we would like to obtain a unique polynomial-sized certificate for the solution of discounted-payoff games; we already know that the values of a discounted-payoff game determine two optimal positional strategies for the players and as such also the respective winning sets. One must only prove that the values of the games vertices can be written using only a polynomial number of bits in the size of the graph, which was proven by Jurdziński [Jur98].

We conclude the section by showing that the qualitative, quantitative and strategic problems are polynomial-time Turing reducible one another. We show a way to recover vertex values in a game with a translation-invariant payoff (Definition 1.4.10) and a polynomial number of possible values. We then show how, given a procedure computing the value of a vertices for any arena, it is possible to extract a strategic solution by recursively removing edges and checking whether the vertex values change.

Remark 2.6.3: We note that a vertex in a fixed parity game arena or in a mean-payoff game arena G can assume a small number of values:

- In the case of a parity game, the possible values are the different priorities, so there are only $d \leq n$ possibilities.
- In the case of a mean-payoff game, the possible values are the means of a cycle, so they are fractions with denominator not greater than n and numerator which is the sum of at most n different weights up to W, therefore there are at most $nW \cdot n$ possibilities.

Theorem 2.6.4: Let $\mathcal{G}$ be a graph game with translation-invariant payoff and with at most $r(n)$ possible values. Given an algorithm that computes whether a player wins in a graph game arena G from a vertex v in time $O(f(n))$, there exists an algorithm for computing a single vertex value in the arena which runs in time $O(f(n) \log r(n))$.

Proof: Note that the possible values are ordered and Max player wins from a vertex if and only if the vertex value is greater than or equal to zero. Therefore the required complexity can be obtain by performing a binary search on the set of possible values, and asking for each candidate value α if Max player wins from vertex v in the shifted game arena $G - \alpha$, where all weights are shifted of $-\alpha$. □

Theorem 2.6.5: Given an algorithm that computes the vertex values of a graph game arena G in time $O(f(n))$, there exists an algorithm for computing positional optimal strategies for both players which runs in $O(nf(n)\log n)$

Proof: Start by computing the values $v(a)$ for each vertex $a \in V$. If all vertices $a \in V_{\text{Max}}$ have outdegree one, then player Max has a unique strategy and this strategy is positional and optimal.

Otherwise, consider any vertex $a \in V_{\text{Max}}$ with outdegree $d > 1$. Remove any $\lceil d/2 \rceil$ of the edges leaving a, and recompute the value of a, $v'(a)$ say, for the resulting graph. If $v'(a) = v(a)$ then there is a positional optimal strategy for Max which does not use any of the removed edges; if on the contrary $v'(a) \neq v(a)$ then there is a positional optimal strategy for this player using one of the removed edges. Whichever is the case, we can now restrict attention to a subgraph G' with at least $\lfloor d/2 \rfloor$ fewer edges.

Let $d(a)$ be the initial outdegree of vertex $a \in V$. The computation of a positional optimal strategy for both players thus requires

$$\sum_{a \in V} \log d(a) = \log \left(\prod_{a \in V} d(a) \right) \leq \log \left(\frac{\sum_{a \in V} d(a)}{|V|} \right)^{|V|} = |V| \log \frac{|E|}{|V|} \leq n \log n$$

calls to the value computing algorithm, thus obtaining the bound. □

Chapter 3

Solving Graph Games

Among the various graph games we have presented, parity games are currently the most computationally tractable, i.e. can be solved in less than subexponential time. Higher games on the hierarchy, such as mean-payoff and simple stochastic games, have not as strong upper bounds and ad-hoc algorithms archieve only a pseudo-polynomial running time (i.e. exponential). On the other hand, lower games on the hierarchy, such as safety games, have longly been known to be solvable in linear time.

In this part of the thesis we are going to see various algorithms that can be used to solve parity and mean-payoff games. We will concentrate on parity because after the recent work of Calude et al. [Cal+17] many adaptations of previously known algorithm have been devised to run in quasi-polynomial time. Among these we must cite Parys [Par19] for adapting the early algorithm of Zielonka for parity games [Zie98]; Jurdziński and Lazić [JL17] for creating the Succinct Progress Measure algorithm, an adaptation of a former algorithm of Jurdziński [Jur00]; Lehtinen [Leh18] that invented a new approach to the problem by exploiting the strong connection between parity games and the modal μ-calculus.

This will give us the opportunity to present them beside the original idea, to give a better overview of the current situation and of how the tractability results were obtained. We will also explore the framework of universal trees by Czerwiński, Daviaud, Fijalkow, Jurdziński, Lazić and Parys [Cze+19] which fully explains the similarities of some of the various algorithms and proves that the currently known techniques for solving parity games cannot be used to beat the quasi-polynomial running time.

Concerning mean-payoff games we will expose two algorithms: a value-iteration procedure by Zwick and Paterson [ZP96] which is a clever adaptation of the classical Bellman-Ford algorithm for shortest path; and another algorithm based on potential theory for weighted graphs due to Gurvich, Karzanov and Khachivan [GKK88] which allows to compute a canonical form for mean-payoff games where locally optimal strategies are also globally optimal strategies. Both run in pseudo-polynomial time, i.e. exponential time.

In the chapter we also explain the connection of graph games with the theory of LP-type problems due to Halman [Hal07] which obtains the currently strongest bound valid for the

whole family of problems, namely randomized subexponential time.

3.1 Algorithms for Parity Games

In the treatment of algorithms for parity the two players will be called Even and Odd, since Min and Max are counterintuitive in this setting. We remind the reader that Even corresponds to Max, while Odd corresponds to Min. We also assume that priorities are written on vertices and not on edges.

3.1.1 Zielonka Recursive algorithm

This was the first algorithm to solve parity games [Zie98] and today is still the practically fastest algorithm known for parity games [Dij18]. We present it before the other algorithms because its operation can be clearly understood from a single picture and the algorithm itself only needs a single definition to be stated, which makes it ideal as foundation to get a wider understanding of the other algorithms.

The original presentation of Zielonka is rather technical and is carried out in a more descriptive complexity setting, which we have not dealt with. Recently Parys [Par19] simplified the presentation of this classical algorithm and also adapted it to obtain a quasi-polynomial variant.

Definition 3.1.1 (Attractor Set)**:** Given a set of nodes N in the game graph G, and a player $P = \text{Max}, \text{Min}$, we define the attractor of N to be the set of nodes of the graph from which P can force to reach a node from N.

In other words, it is the smallest set such that:

- $N \subseteq \text{Attr}_P(G, N)$
- If $v \in V_P$ and some of its successor are in $\text{Attr}_P(G, N)$, then $v \in \text{Attr}_P(G, N)$
- If $v \in V_{\neg P}$ and all its successors are in $\text{Attr}_P(G, N)$, then $v \in \text{Attr}_P(G, N)$

We now outline the algorithm that computes the winning region of one player, which only makes use of attractor computations and later prove its correctness.

The procedure $\text{Solve}_E(G, h)$ in returns the winning region of Even $\text{Win}_E(G)$, if h is an even number that is greater or equal than all priorities appearing in G. The analogous procedure $\text{Solve}_O(G, h)$ is identical with the roles of Even and Odd exchanged.

With $G \setminus S$ we mean the game obtained by removing from G all nodes in S and all edges leading to nodes in S or starting from nodes in S. We use this construct only when S is an attractor; in such a case, if all successors of a node v are removed, then v is also removed.

Structure of winning sets: The algorithm can be clearly understood while looking at Figure 3.1: let h be the highest priority used in G and assume that it is even. We know that

Algorithm 1 Zielonka Algorithm [Zie98]

```
1: procedure SOLVE_E(G, h)                    ▷ h is an even upper for priorities on G
2:     repeat
3:         N_h = {v ∈ G | π(v) = h}           ▷ Nodes with the highest priority
4:         H = G \ Attr_E(G, N_h)             ▷ New game: reaching priority h → win
5:         W_O = Solve_O(H, h − 1)            ▷ In W_O we lose before reaching priority h
6:         G = G \ Attr_O(G, W_O)             ▷ Possibly N_h ∩ Attr_O(G, W_O) ≠ ∅
7:     until W_O ≠ ∅
8: end procedure
```

the game graph G can be divided into two parts: $\mathrm{Win}_E(G)$ and $\mathrm{Win}_O(G)$ by determinancy of the game.

In $\mathrm{Win}_E(G)$ we can distinguish the attractor of nodes with priority h – denoted A_E. Note that Odd either loses inside $\mathrm{Win}_E(G) \setminus A_E$ or enters A_E, which causes him to see a node with priority h, and then the game continues in some node of $\mathrm{Win}_E(G)$.

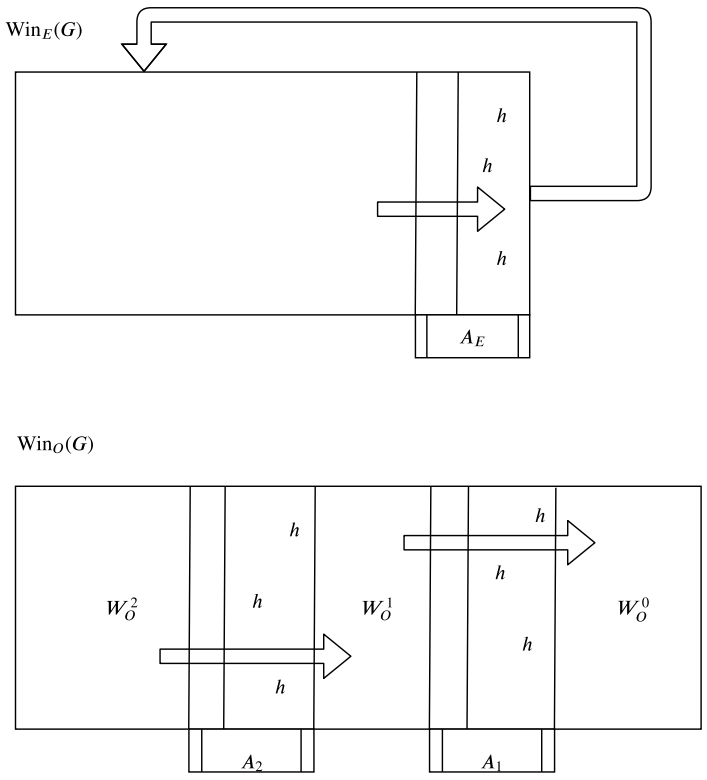

Figure 3.1: The structure of winning regions in a parity game

On the other hand, the winning region of Odd $\mathrm{Win}_O(G)$ can be divided into multiple parts: the part W_O^0, where Odd can win without ever seeing a node of priority h; then we have nodes of priority h from which Even is forced to enter W_O^0, and their attractor, denoted A_1; then we have a part W_O^1, where Odd can ensure that the play is either winning for him inside W_O^1 or enters A_1; in other words, from nodes of W_O^1 Odd can win while seeing h at most once.

In the same way we can define W_O^2 as those remaining vertices from which Odd can either win remaining in W_O^2 or can force the play to enter A_2, from which he knows he can win. Similarly, in the general case we define W_O^{i+1} as the set of vertices in $G \setminus W_O^i$ from which Odd has a forcing strategy to A_{i+1} or can win remaining inside of W_O^{i+1}.

Discovering the regions: While running the Algorithm 3.1.1 the partition of G is not known and has to be discovered. To this aim the algorithm removes first all nodes of priority h. The first call to $\text{Solve}_O(H, h-1)$ at line 5 returns the set W_O^0 of nodes where Odd wins without seeing a node of priority h. We can then remove them from the game, toghether with their attractor A_1, because they are won by Odd.

In the following step $\text{Solve}_O(H, h-1)$ returns the set W_O^0 relative to the subgame, which corresponds to the set W_O^1 in the original game. Eventually the algorithm will stop and it is easy to see that Even wins in the remaining graph. The procedure thus returns $\text{Win}_E(G)$.

Quasi-polynomial adaptation: A variation of the algorithm, due to Parys [Par19], archieves a quasi-polynomial running time by a counting procedure. Indeed, by looking at Figure 3.1, one can see that at most one of the parts W_O^i can be of size larger than $\frac{n}{2}$.

So the idea of the variation is to search only for winning regions of size at most $\frac{n}{2}$ when looking for W_O^i: this will usually be enough except for at most one W_O^i. When the algorithm finds no set of size at most $\frac{n}{2}$, we can search once for a W_O^i of an arbitrary size, being sure that subsequent W_O^i will be again of size at most $\frac{n}{2}$. Of course, due to the recursive nature of the algorithm, we notice that every W_O^i can be further subdivided in a similar way, splitting on the priority $h-2$.

To exploit the observation we can pass two precision parameters in the recursive call, p_E and p_O, which denote the fact that we are searching for a winning set of size at most p_E for Even and at most p_O for Odd. The algorithm is started with parameters $p_E = p_O = n$, the number of nodes of G, and can be summarized as follows:

- We first look for sets W_O^i of size at most $\left\lfloor \frac{p_O}{2} \right\rfloor$. If the returned set is empty, this means either that W_O^i is empty, or that it is of size greater than $\left\lfloor \frac{p_O}{2} \right\rfloor$.
- Then we once search for a set W_O^i of size at most p_O. Note that this can happen at most once because if we find an empty W_O^i here, the procedure has ended.
- Finally we look again for sets W_O^i of size at most $\left\lfloor \frac{p_O}{2} \right\rfloor$, because the algorithm already found a set of greater size.

Letting h be the maximal priority, we can denote by $R(h, l)$ the number of executions of the new Solve_E and Solve_O procedures performed during one call to $\text{Solve}_E(G, h, p_E, p_O)$, where $l = \lfloor \log p_E \rfloor + \lfloor \log p_O \rfloor$. Clearly $R(0, l) = R(h, 0) = 0$, and for $h, l \geq 1$ it holds

$$R(h, l) \leq 1 + nR(h-1, l-1) + R(h-1, l)$$

because after (almost) every call to Solve_O at least one node is removed from G, so that we have at most n calls to Solve_O with decreased precision (where l decreases by one), plus at most one time where we have to call Solve_O with full precision.

Solving the recursion we obtain that $R(h,l) \leq n^l \cdot \binom{h+l}{l} - 1 \leq n^l(h+l)^l$ which is quasi-polynomial in n and h since we start the algorithm with $l = 2\lfloor \log n \rfloor$ and an execution of the Solve_O procedure (without counting recursive calls) only costs polynomial time.

3.1.2 Progress Measures

Progress measures are a special kind of value-iteration algorithm of which many variants are known; in this section we will describe the technique of small progress measures [Jur00], due to Jurdziński and Lazić. In this section a parity game is assumed to be winning for the player who owns the *least* priority which occurs infinitely often, and not the greatest one; this change is motivated by the simplification which occurs in the notation about ordering of tuples. To the same end we will also assume that the parity games have priorities written on vertices, and not on arcs. We will denote the priority function by $\pi : V \to \mathbb{N}$.

The idea of the technique is to characterize the cycles reachable from each vertex using a measure such that the measure assigned to a vertex contains the maximal number of times an odd priority can be seen if player Odd moves over the graph, until a vertex with *lower* even priority is seen. Note that this is very similar to what was done by the previous algorithm, as it is evident from Figure 3.1.

To find such a measure $\mu : V \to \{1, \ldots, n\}^d$ we proceed by an approximation algorithm: we define the notion of progressive edge as those edges which Even can take advantageously (those which make μ decrease in a specific sense) and we iteratively lift the μ value at each vertex by choosing the least measure that makes it progressive. If we are forced to lift a certain vertex for too many times we know that it must be loosing for Even, and this gives a bound on the algorithm running time.

Definition 3.1.2 (Closed strategy)**:** A positional strategy σ for Even is closed on a set $W \subseteq V$ if for all $v \in W$ we have:

- if $v \in V_{\text{Even}}$ then $\sigma(v) \in W$ and
- if $v \in V_{\text{Odd}}$ then $(v, w) \in E$ implies $w \in W$.

In other words, this means that Odd cannot escape from W, while Even can decide to remain in W using σ. We also note that each play consistent with σ and starting in W always stays within W.

Definition 3.1.3 (Cycles)**:** A cycle is a path $v_1, \ldots, v_n$ with $v_1 = v_n$.

We say that a cycle $v_1, \ldots, v_n$ is an i-cycle for $i = \max\{\pi(v_j) \mid 1 \leq j \leq n\}$, i.e. if i is the *smallest* priority occurring on the cycle. We will also call "even" the i-cycles for even i.

Note, as was remarked in the reduction chapter, that a strategy σ closed on W is winning for Even from W if and only if all cycles in W_σ are even.

Notation 3.1.4: Let $\alpha \in \mathbb{N}^d$ be a d-tuple of non-negative integers.

- We number its components from 0 to $d-1$, i.e. $\alpha = (\alpha_0, \alpha_1, \ldots, \alpha_{d-1})$
- $<, \leq, =, \neq, \geq, >$ on tuples denote lexicographic ordering
- We define trucated comparison between tuples as comparison between their initial segments:

$$(n_0, n_1, \ldots, n_k) \gtreqless_i (m_0, m_1, \ldots, m_l) \quad \Leftrightarrow \quad (n_0, n_1, \ldots, n_i) \gtreqless (m_0, m_1, \ldots, m_i)$$

Note that if $i > k$ or $i > l$, the tuples may be suffixed with zeros.

Note that, for $i > j$, $\alpha \geq_i \beta \implies \alpha \geq_j \beta$. Moreover inequalities are compositional: $\alpha >_i \beta$ and $\beta \geq_i \gamma$ implies $\alpha >_i \gamma$.

Notation 3.1.5: In what follows the parity game will be called G, and d will stand for a bound on the maximal priority defined as $d = \max\{\pi(v) \mid v \in V\} + 1$. Now for every $i \in \mathbb{N}$, denote the vertices of priority i as $V_i = \{v \in V \mid \pi(v) = i\}$ and their number by $n_i = |V_i|$.

Definition 3.1.6: Define $\mathbb{M}_G \subseteq \mathbb{N}^d$ such that it is the finite set of d-tuples, with zeros on even positions, and non-negative integers bounded by n_i on odd positions i.

$$\mathbb{M}_G = \{(0, \alpha_1, 0, \alpha_3, \ldots, \alpha_{d-1}) \mid 0 \leq \alpha_{2k+1} \leq n_{2k+1}\}$$

We now introduce the notion of a progress measure for a single-player game. We will show that the play is won by even if and only if there exists a witness parity progress measure.

Definition 3.1.7 (Parity progress measure)**:** Let G be a single-player parity game; then the function $\rho : V \to \mathbb{M}_G$ is a parity progress measure for G if for all edges $(v, w) \in E$ it holds that:

- $\rho(v) \geq_{\pi(v)} \rho(w)$ if $\pi(v)$ is even
- $\rho(v) >_{\pi(v)} \rho(w)$ if $\pi(v)$ is odd

If an edge $(v, w) \in E$ satisfies the above inequalities we say that it is progressive.

The idea in defining a progressive edge is to characterize vertices that can reach even cycles, where the ρ is non-increasing, while keeping away from odd cycles, on which ρ is strictly decreasing, and as it turns out parity progress measures are witnesses for a graph to have only even cycles.

The interpretation of tuples we gave before also explains why we are comparing tuples until a certain point: let's consider the case where $\pi(v) = 1$, which requires $\rho(v) >_1 \rho(w)$ or, in other words, $(\alpha_0, \alpha_1) > (\beta_0, \beta_1)$, where $\rho(v)_1 = (\alpha_0, \alpha_1)$ and $\rho(w)_1 = (\beta_0, \beta_1)$. But recall that $\alpha_0 = \beta_0 = 0$, so we are imposing that the number of odd priorities yet to be seen are lowered when we step on a vertex with odd priority, which was our initial idea.

Lemma 3.1.8: Let ρ be a parity progress measure for G. Then all cycles in G are even.

Proof: Suppose by contradiction that there is an odd cycle $v_1, \ldots, v_k = v_1$ in G, and let $p = \min_i \pi(v_i)$ be the smallest priority on the cycle, and suppose that it occurs in v_1. Then we have the chain of inequalities

$$\rho(v_1) >_{\pi(v_1)} \rho(v_2) \geq_{\pi(v_2)} \rho(v_3) \geq_{\pi(v_3)} \cdots \geq_{\pi(v_{k-1})} \rho(v_k) = \rho(v_1)$$

which is easily seen to imply

$$\rho(v_1) >_p \rho(v_2) \geq_p \rho(v_3) \geq_p \ldots \geq_p \rho(v_k) = \rho(v_1)$$

which gives the contradiction $\rho(v_1) >_p \rho(v_1)$. □

On the other hand we also show that, given a parity game where all cycles are even, there exists a parity progress measure defined on it. This constitutes the basis for the algorithm: we will search for a parity progress measure by using an iterative scheme based only on the structure of the set of measures $V \to \mathbb{M}_G$. If we find a parity progress measure we will know how to recover a strategy for Even.

Lemma 3.1.9: Let G be a parity game where all cycles are even. Then there exists a parity progress measure $\rho : V \to \mathbb{M}_G$ for G such that if $\pi(v)$ is odd then $\rho(v) >_{\pi(v)} (0, \ldots, 0)$.

Proof: We proceed by induction on the number of vertices in G. We first suppose that there are at least some vertices of the two lowest priorities: $V_0 \cup V_1 \neq \emptyset$. Otherwise one can scale down the priority function of G by two. The base of the induction is with the graph consisting of a single vertex, which is trivial.

We suppose that $V_0 \neq \emptyset$. In this case we obtain by induction a parity progress measure $\rho : V \backslash V_0 \to \mathbb{M}_G$ for the subgraph $G \mid_{V \backslash V_0}$, and by the positiveness hypothesis we can set $\rho(v) = (0, \ldots, 0)$ for every $v \in V_0$.

On the contrary suppose that $V_0 = \emptyset$ and $V_1 \neq \emptyset$. We then claim that we can partition the vertices of the graph in two groups W_1 and W_2 such that there is no edge going from W_1 to W_2 in G. Having done this, one can obtain two parity progress measures ρ_1 and ρ_2 for the subgraphs $G_1 = G \mid_{W_1}$ and $G_2 = G \mid_{W_2}$ respectively. Let $n'_i = |V_i \cap W_1|$ and $n''_i = |V_i \cap W_2|$; since there are no edges from W_1 to W_2, the conditions of parity progress measure are trivially satisfied if we let $\rho = \rho_1 \cup (\rho_2 + (0, n'_1, 0, n'_3, 0, \ldots) : V \to \mathbb{M}_G$. Indeed, the shift of a progress measure remains a progress measure and on edges from W_2 to W_1 we are ensuring that $\rho(v) \geq_{\pi(v)} \rho(w)$ by using the inequality $\rho_1 \leq (0, n'_1, 0, n'_3, \ldots)$.

To prove that such partition exists take any vertex of the lowest priority $u \in V_1$ and let $U \subseteq V$ be the set of vertices from which there is a non-trivial path to u in G. Then if $U = \emptyset$ a partition is given by $W_1 = \{u\}$ and $W_2 = V \setminus \{u\}$, which has no edges from W_1 to W_2 by the definition of U; otherwise the partition is given by $W_1 = U$ and $W_2 = V \setminus U$, which is non-trivial because if $u \in U$ then we would have a cycle where the minimal priority is $\pi(u) = 1$, which would be an odd cycle. □

We now turn to expanding the definition of progress measure to account also for two-player games. Definition 3.1.7 requires that for each odd priority, we see a *lower* one at a later point, therefore it can only possibly exists for even cycles. In an odd cycle the number of odd priorities to be seen before an higher one could be infinite; we will then introduce a value $\top$, greater than every other value in $\mathbb{M}_G$, to mean that the current vertex isn't in the winning region of Even.

Definition 3.1.10: Define $\mathbb{M}_G^\top = \mathbb{M}_G \cup \{\top\}$ where:

- $m < \top$, $m <_i \top$ for all $m \in \mathbb{M}_G$ and for all i.
- $\top =_i \top$ for all i.

Definition 3.1.11: If $\rho : V \to \mathbb{M}_G^\top$ and $(v, w) \in E$ is an edge, then $\mathrm{Prog}(\rho, v, w)$ is the least $m \in \mathbb{M}_G^\top$ such that $m \geq \rho(v)$ and

- $m \geq_{\pi(v)} \rho(w)$ if $\pi(v)$ is even
- $m >_{\pi(v)} \rho(w)$ or $m = \rho(w) = \top$ if $\pi(v)$ is odd

Note that in this way we are making the edge (v, w) progressive.

In this way we can set the value of a vertex from which only odd cycles are reachable to $\top$, while other vertices will have a finite value. Intuitively we are searching for the right value m to assign to $\rho(v)$ such that it locally satisfies the definition of parity progress measure; and this will allow us to build an iterative algorithm that locally improves functions from V to $\mathbb{M}_G^\top$ until they become parity progress measures.

We have to slightly modify the definition of parity progress measure to also account for the moves of the other player.

Definition 3.1.12 (Game parity progress measure)**:** Let G be a parity game. A function $\rho : V \to \mathbb{M}_G^\top$ is a game parity progress measure if for all vertices $v \in V$, it holds that:

- if $v \in V_{\mathrm{Even}}$ then $\exists (v, w) \in E$ such that (v, w) is progressive in the sense of Definition 3.1.7.
- if $v \in V_{\mathrm{Odd}}$, then $\forall (v, w) \in E$ we have (v, w) is progressive.

Note that if ρ is a game parity progress measure, then $\rho(v) \neq \top$ if and only if Even can force to reach even cycles and this explains how we can reduce the solution of the qualitative problem for parity games to searching for a game parity progress measure.

Definition 3.1.13 (Induced strategy from game parity progress measures)**:** Let G be a parity game, and $\rho : V \to \mathbb{M}_G^\top$ be a game parity progress measure.

Define the induced strategy $\overline{\rho} : V_{\text{Even}} \to V$ for player Even by setting

$$\overline{\rho}(v) \in \operatorname*{argmin}_w \{\rho(w) \mid (v, w) \in E\}$$

that is $\rho(v)$ is a successor w of v that minimizes $\rho(w)$.

Let $||\rho|| = \{v \in V \mid \rho(v) \neq \top\}$ the vertices from which the reachable cycles are even.

Lemma 3.1.14 (Induced strategy is winning)**:** If ρ is a game parity progress measure, then $\overline{\rho}$ is a winning strategy for player Even from $||\rho||$.

Proof: We note that all vertices we will be considering are inside $||\rho||$: the starting vertex obviously is, and for each vertex v one can note that each successor chosen by Odd satisfies $\rho(v) \geq \rho(w)$ so that $\rho(w) \neq \top$, and since Even choses the w that minimizes $\rho(w)$, one also has $\rho(w) \neq \top$ for $w = \overline{\rho}(v)$.

Then one can easily see that strategy $\overline{\rho}$ is closed on $||\rho||$ and that ρ restricted to $||\rho||$ is a parity progress measure on $G_{\overline{\rho}}$ restricted to $||\rho||$. By the lemma 3.1.8 we can then conclude that all cycles in $G_{\overline{\rho}}$ restricted to $||\rho||$ are even, which is the statement of this lemma. □

Lemma 3.1.15 (Existence of a game parity progress measure)**:** Let W_E be the winning set of player Even in a parity game G. Then there exists a game parity progress measure $\rho : V \to \mathbb{M}_G^\top$ such that $||\rho|| = W_E$.

Proof: The existence again follows from the analogue statement for parity progress measures (Lemma 3.1.9): we know there that all cycles in the parity game $G\mid_{W_E}$ are even, hence by the lemma there is a parity progress measure $\rho : W_E \to \mathbb{M}_G$ for the restricted game. Extending $\rho \mid_{V \setminus W_E} = \top$ makes ρ into a game parity progress measure. □

We characterize now game parity progress measures as fixed points of monotone operators in the finite complete lattice of functions $V \to \mathbb{M}_G^\top$ ordered pointwise. By the Knaster-Tarski fixpoint theorem a least game parity progress measure μ exists and is computable by the fixed point iteration algorithm.

We first recall the Knaster-Tarski fixpoint theorem and the notion of complete lattices.

Definition 3.1.16 (Complete Lattice)**:** A complete lattice is a partially ordered set $(L, \sqsubseteq)$ in which every subset of L has a greatest lower bound and a least upper bound in L. The least upper bound of the empty set is denoted by $\bot$ and the greatest lower bound of the empty set is $\top$.

Definition 3.1.17 (Monotonic function)**:** A function $f : L \to L$ is said to be monotonic if $\forall x, y \in L\ x \sqsubseteq y \implies f(x) \sqsubseteq f(y)$

Definition 3.1.18 (Fixed points)**:** Given a monotonic function $f : L \to L$, a point

$x \in L$ is said to be:

- prefixed point if $f(x) \sqsubseteq x$
- postfixed point if $x \sqsubseteq f(x)$
- fixed point if $x = f(x)$.

Theorem 3.1.19 (Knaster-Tarski fixpoint)**:** For any complete lattice $(L, \sqsubseteq)$ and any monotonic function $f : L \to L$ the fixed points form a complete lattice. In particular the least and greatest fixed points exists.

Remark 3.1.20 (Fixpoint theorem with finite lattices)**:** Since our lattice is finite and complete, the result of Knaster-Tarski can be easily proven by a simple inductive arguments. Consider the sequence $x_0 = \bot$, $x_{k+1} = f(x_k)$: by the definition of $\bot$ we have $x_0 \leq x_1$ and by monotonicity of f we obtain $\forall k\ x_k \leq x_{k+1}$. The x_i then form an increasing sequence in the lattice $x_0 \leq x_1 \leq \ldots$.

Let $x^* = \sup_i \{x_i\}$ be their supremum and we now show that it is a fixpoint: since the elements in the lattice are finite, the increasing sequence of the x_i is definitely constant, and therefore $x^* = x_i$ so we have that $x_i = x_{i+1} = f(x_i)$ which implies $x^* = f(x^*)$ and it is a fixpoint.

We also note that it is the least fixpoint: let y be another fixpoint of f, then it is true that $x_0 = \bot \leq y$. By monotonicity of f we have that $x_k = f^k(x_0) \leq f^k(y) = y$ and by taking the supremum we obtain $x^* \leq y$.

Therefore we have proven that the least fixpoint exists and we also obtained an algorithm for computing it: start with $x_0 = \bot$ and iteratively compute $x_{k+1} = f(x_k)$. It will happen that $x_{k+1} = x_k$, and we will have found the desired fixpoint.

Definition 3.1.21 (Functional lattice)**:** We define a complete lattice structure $\sqsubseteq$ on the set of functions $V \to \mathbb{M}_G^\top$ by comparing them pointwise:

$$\mu \sqsubseteq \rho \Leftrightarrow \forall v \in V\ \mu(v) \leq \rho(v)$$

We will also write $\mu \sqsubset \rho$ to mean $\mu \sqsubseteq \rho$ and $\mu \neq \rho$.

We get back to the progress measure algorithm: we define a lift operator for each vertex and show that it is equivalent for a function $\mu : V \to \mathbb{M}_G^\top$ to be a game parity progress measure or to be a prefixed point of all lift operators.

Definition 3.1.22 (Lifting progress measures)**:** We define the lifting operator $\mathrm{Lift}(\rho, v)(u)$ for $v \in V$ as follows:

- $\rho(u)$ if $u \neq v$
- $\min \{\mathrm{Prog}(\rho, v, w) \mid (v, w) \in E\}$ if $u = v \in V_{\mathrm{Even}}$
- $\max \{\mathrm{Prog}(\rho, v, w) \mid (v, w) \in E\}$ if $u = v \in V_{\mathrm{Odd}}$

Lemma 3.1.23 (Lifting is monotone)**:** The operator $\mathrm{Lift}(\cdot, v)$ is $\sqsubseteq$-monotone.

Proof: Suppose that $\mu \sqsubseteq \rho$, then we have to show that $\forall u, v \in V$ we have that

$$\mathrm{Lift}(\mu, v)(u) \leq \mathrm{Lift}(\rho, v)(u)$$

If $u \neq v$ then we have $\mathrm{Lift}(\mu, v)(u) = \mu(u)$ and $\mathrm{Lift}(\rho, v)(u) = \rho(u)$ which is verified. If $u = v \in V_{\mathrm{Even}}$ then we obtain the required inequality by proving that if $\mu \sqsubseteq \rho$, then $\mathrm{Prog}(\mu, v, w) \leq \mathrm{Prog}(\rho, v, w)$, and by reminding that max and min are monotone operators.

Suppose $m \in \mathbb{M}_G^\top$ satisfies $m \geq_{\pi(v)} \rho(w)$ if $\pi(v)$ is even; then $m \geq_{\pi(v)} \rho(w) \geq \mu(w)$ which implies that $\mathrm{Prog}(\rho, v, w) \geq_{\pi(v)} \mu(w)$. Moreover if m satisfies $m >_{\pi(v)} \rho(w)$ or $m = \rho(w) = \top$ if $\pi(v)$ is odd then $m >_{\pi(v)} \rho(w) \geq \mu(w)$ or $m = \top = \rho(w) \geq \mu(w)$, so that we also get that either $m >_{\pi(v)} \mu(w)$ or that $\top = \mu(w)$.

We have thus proven that m satisfied the conditions to be in the minimum for $\mathrm{Prog}(\mu, v, w)$: thus $m \geq \mathrm{Prog}(\mu, v, w)$ which gives the required inequality. □

Lemma 3.1.24 (Game parity progress measures as fixpoints)**:** A function $\rho : V \to \mathbb{M}_G^\top$ is a game parity progress measure if and only if $\forall v \in V \quad \mathrm{Lift}(\rho, v) \sqsubseteq \rho$.

Proof: $\boxed{\Rightarrow}$ Let ρ be a game parity progress measure: we show that it is a prefixed point: if $u \neq v$ then $\mathrm{Lift}(\rho, v)(u) = \rho(u)$, which proves inequalities on such vertices. On the other hand if $u = v$ from the definition of game parity progress measure we know that:

- If $v \in V_{\mathrm{Even}}$ then $\rho(v) \geq_{\pi(v)} \mathrm{Prog}(\rho, v, w)$ for some w. Since the lift considers the min of such progs we obtain that $\rho(v) \geq_{\pi(v)} \mathrm{Lift}(\rho, v)(u)$.
- If $v \in V_{\mathrm{Odd}}$ then $\rho(v) >_{\pi(v)} \mathrm{Prog}(\rho, v, w)$ for all w, and we again obtain its description as a prefixed point.

$\boxed{\Leftarrow}$ This part of the proof is obtained similarly as it is enough to reverse the steps of the other direction. □

The least game parity progress measure can now be computed using fixed point approximation from Remark 3.1.20.

Algorithm 2 Small Progress Measures [Jur00]

```
1: procedure SOLVE
2:     μ(v) = (0, ..., 0)   ∀v ∈ V
3:     while μ ⊏ Lift(μ, v) for some v ∈ V do
4:         μ → Lift(μ, v)
5:     end while
6: end procedure
```

Remark 3.1.25 (Algorithm running time)**:** Note that the running time of the algorithm is essentially bounded by the size of $\mathbb{M}_G^\top$ since, excluding the time to compute Lifts which is polynomial, the while loop can be executed at most $|\mathbb{M}_G|$ times for each vertex because of the monotonicity of Lift operators and the ordering of functions.

Unfortunately the algorithm as stated has exponential runtime in the number of priorities d since it is trivial to compute that

$$|\mathbb{M}_G| = \prod_{i \text{ odd}} (n_i + 1) \leq \left(\frac{1}{\lfloor d/2 \rfloor} \sum_{i \text{ odd}} (n_i + 1) \right)^{\lfloor d/2 \rfloor} \leq \left(\frac{n}{\lfloor d/2 \rfloor} \right)^{\lfloor d/2 \rfloor}$$

by the arithmethic-geometric mean inequality. Moreover, it is not difficult to find a game which exibits worst-case behaviour.

The variant called "Succinct Progress Measures" by Jurdziński and Lazić [JL17] uses a generalization on the notion of progress measure to get the running time down to quasi-polynomial. Namely one can consider progress measures as functions which take values in an arbitrary partially ordered set T. The iterative algorithm can be carried out in the same way as the previous one, so one can try to find a more succinct coding of the functions $V \to \mathbb{M}_G$ to lower the running time; the coding must be chosen carefully to ensure that the iterative algorithm still converges to the least progress measure. Jurdziński and Lazić introduced the notion of succinct tree coding to this aim.

Definition 3.1.26 (Ordered tree)**:** An ordered tree is a prefix-closed set of sequences of elements of a linearly ordered set T.

A *node* is any such sequence of elements; the maximal nodes w.r.t. prefix ordering are called *leaves*; the *root* of the tree is the empty sequence. A sequence $(a_0, \ldots, a_n, a_{n+1})$ is called *child* of $(a_0, \ldots, a_n)$. We say that the depth of a node $(a_0, \ldots, a_n)$ is $n + 1$ and the tree inherits an order structure which is defined lexicographically. In the ordering a father is always less than any of its children.

Definition 3.1.27 (Ordered-tree coding)**:** An ordered tree coding is an order-preserving relabelling of branching directions, allowing for the relabelling at various nodes to differ from one another, i.e. to be adaptive.

The aim of coding trees is to provide a succinct (i.e. short) code for each leaf in the tree. We now want to find an ordered tree as small as possible that can code for all trees of depth h with at most l leaves. To do so we ask how many bits are required to code the path to each node in an ordered tree?

A main technical contribution of the paper [JL17] is to provide an adaptive ordered-tree coding with $(\lceil \lg h \rceil + 1)\lceil \lg l \rceil$ bits for every ordered tree with height h and l leaves. To introduce it we will need the notion of bounded counters:

Definition 3.1.28 (Bounded adaptive counters)**:** We define the set $B_{g,h}$ of g-

bounded adaptive h-counters to consist of h-tuples of binary strings whose total length is at most g.

As examples $(0, \varepsilon, 1, 0)$ is a 3-bounded 4-counter; $(0, 1, \varepsilon, \varepsilon, 0)$ is a 3-bounded 5-counter; $(01, \varepsilon, 10)$ is a 4-bounded 3-counter.

Definition 3.1.29 (Order on bounded counters)**:** We define an ordering on binary strings and extend the ordering to $B_{g,h}$ lexicographically.

For both binary digits $b = 0, 1$ and for all binary strings s, s' the ordering rules are:

$$0s < \varepsilon, \qquad \varepsilon < 1s, \qquad bs < bs' \text{ iff } s < s' \tag{3.1}$$

Lemma 3.1.30 (Succinct tree coding)**:** For every ordered tree T of height h and with at most l leaves there is a tree coding in which every navigation path is an $\lceil \lg l \rceil$-bounded adaptive i-counter, where $i \leq h$ is the length of the path

Idea of the proof: By induction on l and h we consider a child M of the root that evenly divides the two sets of leaves $L_<$ whose first component is strictly smaller than M and $L_>$ whose first component is strictly larger than M. Also denote by L_M the subtree rooted in M.

The code $\text{code}_T(l)$ is then obtained in the following way:

- If $l \in L_<$ and $\text{code}_{L_<}(l) = (a_0, \ldots, a_k)$ then $\text{code}_T(l) = (a_0 0, a_1, \ldots, a_k)$
- If $l \in L_>$ and $\text{code}_{L_>}(l) = (a_0, \ldots, a_k)$ then $\text{code}_T(l) = (a_0 1, a_1, \ldots, a_k)$
- If $l \in L_M$ and $\text{code}_{L_M}(l) = (a_0, \ldots, a_k)$ then $\text{code}_T(l) = (\varepsilon, a_0, \ldots, a_k)$

where justapposition is the append operation for binary strings. Observe that the size of the two subtrees $L_<$ and $L_>$ is halved at every step of the recurtion, i.e. $|L_<|, |L_>| \leq \frac{|T|}{2}$.

We observe that the tree coding bears remarkable similarities with Huffman coding used in compression schemes.

Definition 3.1.31 (Succinct progress measure)**:** Is a mapping $\mu : V \to T$ where T is an ordered tree such that every navigation path in T is a $\lceil \lg \eta \rceil$-bounded adaptive i-counter for some $0 \leq i \leq d/2$ where η is the number of vertices with an odd priority.

Truncations and progressiveness of edges in succinct progress measures are defined analogously to small progress measures.

Definition 3.1.32: Let $S_{\eta,d}$ be a linearly ordered set of bounded adaptive multi-counters

$$S_{\eta,d} = \cup_{i=0}^{d/2} B_{\lceil \lg \eta \rceil, i}$$

and let $S_{\eta,d}^{\top}$ denote the same set with an extra top element $\top$.

Lemma 3.1.33 (Existence of a succinct progress measure)**:** Let W_E be the winning set of player Even in a parity game G. Then there exists a succinct game parity progress measure $\rho : V \to S^{\top}_{\eta,d}$ such that $||\rho|| = W_E$.

Proof: Observe that $\mathbb{M}^t_G op$ is a linearly ordered set, and therefore a small progress measure $\mu : V \to \mathbb{M}^{\top}_G$ con be encoded succinctly using the tree coding of Lemma 3.1.30. The existance of a small progress measure is ensured by Lemma 3.1.15, and applying the succinct tree coding we obtain exactly a succinct progress measure for the game. □

Theorem 3.1.34: There is a succinct progress measure lifting algorithm which solves parity games in quasi-polynomial time.

Proof: From the previous lemma we have the existence of a succinct progress measure. We can then use Algorithm 3.1.2 on succinct progress measures to obtain an algorithm whose running time is essentialy decided by the size of the set $S^t_{\eta,d} op$.

From a simple estimate one can see that

$$\left|S^{\top}_{\eta,d}\right| \leq 2^{\lceil \log \eta \rceil} \binom{\lceil \log \eta \rceil + d/2 + 1}{d/2} \leq 2^{\lceil \log \eta \rceil} (\lceil \log \eta \rceil + d/2 + 1)^{\lceil \log \eta \rceil}$$

and therefore its size is quasi-polynomial. □

3.1.3 Lehtinen Register-Index

In this section we explore the algorithm of Lehtinen [Leh18] which exploits the strong connection between parity games and modal μ-calculus obtaining a very elegant method for solving parity games in quasi-polynomial time.

Her proof is based on the definition of auxiliary "register-games" that have the same winner as the original parity games and which can then be encoded themselves as parity games with a quasi-polynomial number of vertices but only a logarithmic number of priorities. Combining such transformation with another algorithm for solving games with a low number of priorities, the desired runtime is obtained.

We begin by introducing the notion of register-index of a parity game, a measure of complexity which captures how many priorities the winner of a parity game needs to keep in memory in order to produce a witness of their victory. Note that we will be considering games which can have *coule-de-sac*, i.e. vertices with no exits. In case a play ends on one of these vertices, the player owning it loses.

Definition 3.1.35 (Induced k-register game)**:** Given a parity game arena G with priorities on the vertices, induce a new game $R^k(G)$ where in addition to winning the parity game, player Even has to witness her victory using a fixed amount of registers, i.e. memory.

The new game is played on the same graph G, with added data of k registers, each

one holding a number in $H = \{0, \ldots, d\}$ where d is the maximal priority in G, assumed odd. Register value will be denoted by $x_1, \ldots, x_k$. Registers may be *reset*. The registers are also ranked according to how long it has been since their last reset; their rankings will be denoted by $r_1, \ldots, r_k \in \{1, \ldots, k\}$.

The initial state of the registers is $x_1 = \ldots = x_k = 0$ and $r_i = i$. Each register records the highest priority that has occurred in the parity game since it was last reset. Player Even is given control of the registers. At each move the registers are first updated, and after player Even can choose to reset a register, let's say i. If the register contains the priority x_i, this produces output $2r_i$ if x_i is even and $2r_i + 1$ otherwise. When a register i is reset, its rank is also resetted $r_i = 1$ and the rank of the other registers increases by one. Even may also choose not to reset any register at a certain move.

If Even resets registers infinitely often, an infinite output sequence will be produced in $\{0, \ldots, 2k + 1\}^\omega$. To be declared the winner, Even has to produce an infinite sequence of outputs that satisfies the parity condition, i.e. whose maximal priority occurring infinitely often is even.

Remark 3.1.36 (The register game is a parity game)**:** Note that this new game has a parity winning condition on its output sequence and therefore can be explicitly transformed into another parity game, which will also be denoted by $R^k(G)$, where vertices also keep the register state $V' = V \times H^k$ and where for every edge $v \to v'$ in the game G, edges $(v, \hat{x}) \to (v', \hat{x}')$ are placed for every possible move concerning register update that Even can follow. Those edges emit a priority according to the register update rules described above.

Register ranking can be encoded by ordering them accordingly to their rank, which is the reason why the ranking is not explicitly added in the graph vertices.

In a certain sense we are encoding a proof system that allows Even to produce inside the game a certificate of its win, and we are requiring that he simultaneously wins in the parity game, and is able to produce a valid certificate of its win.

In order to extract a quasi-polynomial algorithm from such construction, we need to establish the following facts:

1. If Even can win, then he can also produce the required certificate.

2. The new game can be solved in quasi-polynomial time.

Definition 3.1.37 (Underlying play)**:** Given a play in $R^k(G)$ we define the underlying play as the projection onto the first element of each visited position.

Definition 3.1.38 (Interval of a position)**:** The interval of a position v at which Even resets a register i is the fragment of the play since the previous reset of register i.

Remark 3.1.39 (Inducing positional strategies)**:** We note that a positional winning strategy on G for Even coupled with a register resetting strategy induces a positional strategy in the game $R^k(G)$.

Note that the size of the register game is $n \cdot d^k$ with $2k+1$ priorities so that if we are able to bound k logarithmically in n we will obtain that $R^k(G)$ is a game with $n \cdot d^{\ln n}$ vertices and $2 \lg n + 1$ priorities, which can then be solved using any other algorithm to obtain a pseudo-polinomial running time.

Definition 3.1.40 (Register-index of an arena)**:** The register-index of an arena G, denoted by $\mathrm{RI}(G)$, is the minimal k for which for every vertex $v_i \in V^G$ the register game $R^k(G)$ is winning for Even from $(v_i, \hat{0})$ if and only if G is winning for Even from vertex v_i.

The definition is well posed, i.e. there exists a natural k for which G and $R^k(G)$ have the same winner from positions v and $(v, \hat{0})$ respectively. It can be shown that $k = n$ suffices, but we will show a much more stronger result with the next lemma.

Let σ be a positional winning strategy for Even in the arena G, and let v be any vertex from the winning region of even in G. Then in each game $R^k(G_\sigma)$, a play starting from $(v, \hat{0})$ is entirely determined by the underlying strategy of Odd and the resetting strategy of Even.

Definition 3.1.41 (Defensive register index)**:** A defensive resetting strategy is a resetting strategy in which the maximally ranked register, i.e. that of ranking k, is never reset when it contains an odd priority if it contained a priority q or an higher even priority in the initial configuration.

The defensive register-index of an arena G, denoted by $\mathrm{DRI}(G)$, is the minimal k for which for every vertex $v_i \in V^G$ the register game $R^k(G)$ is winning for Even with a defensive strategy from $(v_i, \hat{0})$ if and only if G is winning for Even from vertex v_i.

Since it's obvious that a defensive resetting strategy is also a resetting strategy, it follows that the register-index is bounded by the defensive register index: $\mathrm{RI}(G) \leq \mathrm{DRI}(G)$.

Lemma 3.1.42 (Recurrence on connected components)**:** Let σ be a winning positional strategy in a parity game arena G. Let q be the maximal even priority in G_σ. Let $C_0, \ldots, C_k$ be the strongly connected components of the game arena G'_σ where vertices of priorities q and $q-1$ are removed from G_σ, and let $r_i = \mathrm{DRI}(C_i)$ be their defensive register-index. If $r_i < r$ for every $i = 1, \ldots, k$ and $r_0 = r$, then the defensive register-index of the whole arena is less than or equal to r: $\mathrm{DRI}(G) \leq r$.

Idea of the proof: Observe that a play on G_σ starting from a vertex v that is winning for Even on G either stays definitely in one of the C_i or passes infinitely often through

an arc of priority q.

The base case with $r = 2$ is simple because it is enough to notice that single-vertex arenas have register-index equal to one.

Then the strategy of Even using r registers is to initially reset all registers containing odd priorities higher than q, which must be in finite number. Thereafter he resets the highest ranking register whenever q occurs, thus emitting an even priority of $2r$, and plays his winning strategy in each component $C_1, \ldots, C_k$ using only the lowest ranking registers from 1 to r_i. When in C_0 instead Even uses his defensive strategy there using all of the registers.

In this way, if the play (now entirely controlled by Odd) definitely remains in one of the C_i for $i \neq 0$, the resetting winning strategy there is obviously also winning for the play on G because of prefix independence of game payoff.

On the other hand, if Odd continues exiting from C_i, the play passes infinitely often through an arc of priority q, and Even emits a priority $2r$ infinitely often. We must prove that he never emits an higher odd priority to finish the proof. In components $C_1, \ldots, C_k$ the highest ranking register remains untouched; and we enter component C_0 only when the highest register contains q or an higher even priority, thus by the defensive strategy existing here we never emit an higher odd priority. Infact, if there are more than two register, after we pass from a vertex of priority q, the registers' values are $x_i = q$ (or maybe higher even priorities coming from external games) and, after resetting the highest ranking register, the new highest ranking register still contains q or an higher even priority.

Note that by the lemma it is possible to say that the register-index of any game arena G is finite, and the above lemma provides an algorithm to compute it recursively.

Remark 3.1.43 (The register-index is logarithmic in game size)**:** As a corollary to the previous lemma we obtain that the register index of a game is logarithmically bounded by the size of the game, i.e. $\mathrm{RI}(G) \leq 1 + \log_2 |G|$.

It is easy to see this by induction on $\mathrm{RI}(G)$ and by secondary induction on $|G|$: if $|G| = 1$ the inequality is satisfied. Let now $r + 1 = \mathrm{RI}(G)$; by the previous lemma we know that the connected components of G_σ must be:

- Either a single strongly connected component C_0 of register index $r + 1$, and we conclude by secondary induction on $|G|$ by restricting our consideration to C_0.

- or G has at least two strongly connected components whose register-index is r. Those components have disjoint vertex sets.

By induction we thus obtain that $|G| \geq 2^{\mathrm{RI}(G)-1}$.

Theorem 3.1.44: Parity games are solvable in quasi-polynomial time.

Proof: By the previous remark $r = \mathrm{RI}(G) \leq 1 + \log_2 n$, where n is the number of vertices of the arena. We already noted that $R^k(G)$ is a parity game of size nd^k and with $2k+1$ priorities, so by setting $k = 1 + \log_2 n$ we obtain an equivalent parity game of size $n2^{O(\log n \cdot \log d)} \leq n2^{O(\log^2 n)}$ and with $O(\log_2 n)$ priorities.

By using now one of the previous algorithms, for example small progress measures, we can solve such a game in time $O(2^{O(\log_2^3 n)}p(n))$, for a suitable polynomial $p(n)$. □

3.2 Lower bounds via universal trees

In the previous section we presented various quasi-polynomial algorithms. It is reasonable to ask whether such algorithms are sharing a common underlying structure.

Czerwiński, Daviaud, Fijalkow, Jurdziński, Lazić and Parys [Cze+19] answer such question: all these solution strategies are based on automata separating "even" words (plays visiting even cycles) from "odd" words. They prove that such automata must explore a quasi-polynomial search space to work correctly in all cases, and thus that they share a quasi-polynomial lower bound. In other words, this prevents these algorithms from archieving a polynomial running time.

We briefly show the intuition of their result applied to the case of progress measure lifting algorithms.

Definition 3.2.1 ((l,h)-universal ordered tree)**:** An (l,h)-universal ordered tree is an ordered tree such that every ordered tree of height at most h and with at most l leaves can be isomorphically embedded into it injectively and in an order-preserving way.

Remark 3.2.2 (Intuition leading to universal trees)**:** A game parity progress measure on a game with n vertices and at most d distinct edge priorities is a mapping from the vertices to nodes in an ordered tree of height at most $d/2$ and with at most n leaves. Such progress measure can then be seen as a mapping into a $(n, d/2)$-universal tree. The progress measure lifting algorithm runtime is moreover bound only by the size of the universal tree.

Theorem 3.2.3 (Smallest size of universal trees)**:** For all positive integers l and h, every (l,h)-universal tree has at least $\binom{\lfloor \ln l \rfloor + h - 1}{h-1}$ leaves, which is at least $l^{\ln(h/\ln l)-1}$ provided that $2h \leq l$.

Proof: The proof is by induction on (l,h). Let $g(l,h)$ be a lower bound on the size of leaves of an (l,h)-universal tree. Then we show that $g(l,h) \geq \sum_{\delta=1}^{l} g\big(\lfloor \frac{l}{\delta} \rfloor, h-1\big)$ obtaining a recursive definition from which it is easy to prove that $g(l,h) \geq \binom{\lfloor \ln l \rfloor + h - 1}{h-1}$.

Let T be a (l,h)-universal tree and $\delta \in \{1, \ldots, l\}$. We claim that the number of nodes at depth $h-1$ of degree $\geq \delta$ is at least $g\big(\lfloor \frac{l}{\delta} \rfloor, h-1\big)$.

Indeed, let T_δ be the subtree of T obtained by removing all leaves and all nodes at depth $h-1$ of degree less than δ: the leaves of the tree T_δ have depth exactly $h-1$.

Universality of T_δ: Then T_δ is $(\lfloor \frac{l}{\delta} \rfloor, h-1)$-universal: let t be a tree with $\lfloor \frac{l}{\delta} \rfloor$ leaves all at depth $h-1$. To each leaf of t we can append δ children, yielding the tree t_+ which has $\lfloor \frac{l}{\delta} \rfloor \cdot \delta \leq l$ leaves all at depth h.

Since T is (l,h)-universal, t_+ embeds into T and this induces an embedding of t into T_δ, since the leaves of t have degree δ in t_+.

Counting leaves: Let l_δ be the number of nodes at depth $h-1$ with degree exactly δ. We proved so far that the number of nodes at depth $h-1$ of degree $\geq \delta$ is at least $g\big(\lfloor \frac{l}{\delta} \rfloor, h-1\big)$, so $\sum_{i=\delta}^{l} l_i \geq g\big(\lfloor \frac{l}{\delta} \rfloor, h-1\big)$.

Thus the number of leaves of T is:

$$\sum_{i=1}^{l} l_i \cdot i = \sum_{\delta=1}^{l} \sum_{i=\delta}^{l} l_i \geq \sum_{\delta=1}^{l} g\left(\left\lfloor \frac{l}{\delta} \right\rfloor, h-1 \right) = g(l,h)$$

which ends the proof. □

Both progress measure lifting algorithms perform an iterative search for a progress measure and their search spaces are limited by restricting the labels considered in candidate tree labellings to leaves in a specific $(n, d/2)$-universal tree.

The algorithms correctness also relies precisely on the universality property of the ordered trees from which the candidate labels are taken from: the algorithm computes a sequence of tree labellings that is monotonically increasing while being pointwise lexicographically smaller than or equal to every progress measure. Moreover, the worst-case run-time bound for the algorithm is determined by the size of the universal tree used, up to a small polynomial factor.

The quasi-polynomial lower bound on the size of universal trees forms a barrier that needs to be overcome in order to further improve the complexity of solving parity games.

3.3 Algorithms for mean-payoff games

We now analyze the main algorithms for mean-payoff games. Unfortunately not many are known and their runtime is pseudo-polynomial, namely is polynomial in the number of vertices n but linear in the size of the biggest weight W, and therefore is exponential in the problem description. The most known alogrithm is that of Zwick and Paterson [ZP96] based on approximating each vertex value by an iterative sum, which will be explained in this section. In the next section we will then consider an algorithm based on potential theory for weighted graphs due to Gurvich, Karzanov and Khachivan [GKK88] which allows to compute an easy to solve canonical form for mean-payoff games.

3.3.1 Zwick and Paterson value-iteration

Before introducing the idea of the algorithm, we remark a finiteness construction for a positionally determined game.

Remark 3.3.1 (Finite game from a positionally determined game)**:** Given a positionally determined game G we can derive from it a finite game with the same positional strategies. We let the game start from a vertex v_i and the two players play as usual, but the game ends as soon as a vertex is visited twice, and its payoff is computed as the play would cycle forever on the ending cycle.

The goal of the algorithm is to find, for each vertex $a \in V$, the value $v(a)$ of the finite and infinite games that start at a. To reach this goal we consider a third version of the game: this time the two players play the game for exactly k steps constructing a path of length k, and the weight of this path is the outcome of the game. The length of the game is known in advance to both players: we let $v_k(a)$ be the value of this game started at vertex $a \in V$, where player Max or Min plays forst according to the vertex ownership.

Theorem 3.3.2 (Computing the truncated values of the game)**:** The values $v_k(a)$, for every $a \in V$, can be computed in $O(k \cdot |E|)$ time.

Proof: It is easy to see that for every $a \in V$ and every $k \geq 1$ we have that

$$v_k(a) = \max_{a \to b} \{w(a,b) + v_{k-1}(b)\} \qquad \text{if } a \in V_{\text{Max}}$$

$$v_k(a) = \min_{a \to b} \{w(a,b) + v_{k-1}(b)\} \qquad \text{if } a \in V_{\text{Min}}$$

and clearly $v_0(a) = 0$ for every vertex. The values $v_k(a)$ for every vertex can then be easily computed using these recursive formulas in $O(k \cdot |E|)$ time. □

It seems intuitively clear that $\lim_{k\to\infty} v_k(a)/k = v(a)$, where $v(a)$ is the value of the infinite game that starts at a. The next theorem shows that this is indeed the case:

Theorem 3.3.3: For every $a \in V$ we have:

$$k \cdot v(a) - 2nW \leq v_k(a) \leq k \cdot v(a) + 2nW$$

Proof: Let σ be a positional optimal strategy for player Max in the finite game starting at a. We show that if Max plays using the strategy σ then the outcome of a k-step game is at least $(k-n) \cdot v(a) - nW$.

We basically used the cycle decomposition technique described in Procedure 1.4.22 and push vertices on a stack when they are visited. Whenever a cycle is formed, it follows from the fact that σ is an optimal strategy for Max in the finite game, that the mean weight of the cycle formed is at least $v(a)$. They are all removed and the process continues.

At each stage the stack contains at most n edges and the weight of each of them is at least $-W$. Max can therefore ensure that the total weight of the edges encountered in a k-step game starting from a is at least $(k-n)\cdot v(a) - nW$, which is at least $k \cdot v(a) - 2nW$ as $v(a) \leq W$.

Applying a similar reasoning for player Min we obtain the other inequality. □

We can now describe the algorithm for computing the exact values of the finite and infinite games:

Theorem 3.3.4 (Computing values in a mean-payoff game)**:** The values $v(a)$ for every vertex $a \in V$, corresponding to the infinite and finite games that start at all the vertices of V can be computed in $O(|V|^3|E|W)$ time.

Proof: Compute the values $v_k(a)$ for every $a \in V$ for $k = 4n^3W$. This can be done, according to a previous theorem, in $O(|V|^3|E|W)$ time. For each vertex $a \in V$, compute an estimate $v'(a) = v_k(a)/k$.

We then get the inequalities:

$$v'(a) - \frac{1}{2n(n-1)} < v'(a) - \frac{2nW}{k} \leq v(a) \leq v'(a) + \frac{2nW}{k} < v'(a) + \frac{1}{2n(n-1)}$$

The value $v(a)$ is a rational number with a denominator whose size is at most n, therefore the minimum distance between two possible values of $v(a)$ is at least $\frac{1}{n(n-1)}$. The exact value of $v(a)$ is then the unique rational number with denominator of size at most n that lies in the interval $v'(a) \pm \frac{1}{2n(n-1)}$, which is easily found in polynomial time by exaustive enumeration. □

3.3.2 Canonical form for Mean-payoff games

We will search in this section for a peculiar form for mean-payoff games, where locally optimal strategies are also globally optimal; such form is shown to exist for every mean-payoff game by using the theory of discounted-payoff games. Later we analyze an algorithm to transform a given mean-payoff game into its canonical form, obtaining a pseudo-polynomial algorithm.

Definition 3.3.5 (Potentials)**:** A potential for a game G is a function $p : V^G \to \mathbb{Z}$.

Definition 3.3.6 (Potential transformations)**:** Given a potential function p, the transformed game $p(G)$ has the same underlying graph and vertex ownership as G, but the weights are changed according to the rule $w_{p(G)}(a,b) = w_G(a,b) - p(a) + p(b)$ for a mean-payoff game and to the rule $w_{p(g)}(a,b) = w_G(a,b) - p(a) + \beta p(b)$ for a discounted-payoff game.

Remark 3.3.7: The value of the mean-payoff game G and of $p(G)$ is the same for ev-

ery starting vertex, and the same holds for discounted-payoff games. Moreover, a positional strategy for one player is optimal in G if and only if it is optimal in $p(G)$.

Proof: Observe that the value of the cycle c in G and in $p(G)$ doesn't change and use positional determinancy. □

Definition 3.3.8 (Locally Optimal Strategy)**:** A locally optimal strategy is a strategy where the player, starting from a vertex v owned by him, moves its token on an outgoing edge that has higher immediate reward (i.e. one with the most positive weight if the player is Max, and one with the most negative weight if the player is Min).

Definition 3.3.9 (Canonical Form)**:** We say that a mean-payoff game G is in canonical form if every locally optimal positional strategy is globally optimal.

Definition 3.3.10 (Weak Canonical Form)**:** We say that a mean-payoff game G is in weak canonical form if every locally optimal positional strategy which remains on vertices with the same value is also globally optimal.

Theorem 3.3.11 (Every MPG has a Canonical Form)**:** We will now prove that every mean-payoff game admits a potential transformation p that transforms it into a game in canonical form.

Obviously games in canonical form are easily solved by choosing the locally optimal strategy for the player, so we will later be also interested in the complexity of computing such potential transformations.

Idea of the proof: We will prove the statement via associated discounted-payoff games in multiple steps:

1. We can a pair of strategies that is optimal for a sequence of $\beta_n \to 1$ so that the values of the vertices in the β-discounted games become rational functions of β along the sequence.
2. Expanding $(1-\beta)v_i(\beta)$ in power series near $\beta = 1$, we obtain a potential transformation for discounted games.
3. The same potential transformation with $\beta = 1$ transforms the mean-payoff game in a particularly convenient form, that will be called weak canonical form.
4. We will then transform the weak canonical form into a canonical form.

Proof: Notation: Let $W = \max_{i,j} |w_{ij}|$ be the maximum of the arc weights. Note also that we will write β for convenience, but the formulas are valid only for a sequence of $\beta_n \to 1$.

We will denote with $v_i(\beta)$ the value of the vertex v_i in the β-discounted game, as

defined previously by Equation 1.5.

Step 1: Fix a strategy σ that is optimal for a sequence of $\beta_n \to 1$ (that is known to exists thank to previous remark 1.4.25). Now Equation 1.5 can be written as

$$v_i(\beta) = w_{i\sigma(i)} + \beta v_{\sigma(i)}(\beta) \tag{3.2}$$

Note that the $v_i(\beta)$ are rational function of β on the sequence β_n because they are the solution to the above system of linear equation (use Cramer rule).

Step 2: We know that $|(1-\beta)v_i(\beta)| = \left|(1-\beta)\sum_k \beta^k w_{i_k\sigma(i_k)}\right| \leq W$ so $(1-\beta)v_i(\beta)$ is bounded near $\beta = 1$ and can be expanded in power series as $(1-\beta)v_i(\beta) = a_i + b_i(1-\beta) + o(1-\beta)$, so that we obtain $v_i(\beta) = \frac{1}{1-\beta}a_i + b_i + o(1-\beta)$ and substituting that into Equation 3.2 we get

$$\frac{1}{1-\beta}a_i + b_i + o(1-\beta) = w_{i\sigma(i)} + \beta\left(\frac{1}{1-\beta}a_{\sigma(i)} + b_{\sigma(i)} + o(1)\right)$$

that can be easily rewritten as

$$\frac{1}{1-\beta}a_i + b_i + o(1-\beta) = w_{i\sigma(i)} - a_{\sigma(i)} + \frac{1}{1-\beta}a_{\sigma(i)} + \beta b_{\sigma(i)} + o(1-\beta)$$

that is an equality for a sequence $\beta_n \to 1$, so the coefficients must satisfy

$$a_i = a_{\sigma(i)}, \qquad b_i = w_{i\sigma(i)} - a_{\sigma(i)} + \beta b_{\sigma(i)} + o(1-\beta)$$

Letting $\beta \to 1$ in the second equation we obtain that a necessary condition for σ to be an optimal positional strategy near $\beta = 1$ is to satisfy: $a_i = a_{\sigma(i)}$ and $a_{\sigma(i)} = w_{i\sigma(i)} + b_{\sigma(i)} - b_i$.

Step 3: The map $i \mapsto b_i$ is a potential transformation. The transformed system can be partitioned into sets of vertices with equal a_i, which are known to be the values of the vertices in the mean-payoff game as we have proven in remark 1.4.25.

Let's now consider the vertex i and suppose it belongs to player Max. Its game value is a_i, so it has no edges going to a vertex of greater value, otherwise Max could choose that in a non-positional strategy and get a greater game value from vertex i, which contradicts the definition of game value.

By the same reasoning it has at least one edge going to a vertex j with the same value $a_j = a_i$, and Max has no reason to choose edges going to a vertex with lower value.

The modified weights in the arena are $w'_{ij} = w_{ij} + b_j - b_i$. We will now see that by choosing the outgoing edge with the highest weight among those with the same a value, Max can realize an optimal strategy.

Infact, by taking the limit of Equation 1.5 multiplied by $(1-\beta)$ we have

$$(1-\beta)v_i(\beta) = \max((1-\beta)w'_{ij} + (1-\beta)\beta v_j(\beta))$$

$$a_i = \max(a_j)$$

So we can see also from the equations that there is at least one edge to a vertex with same game value, and there is no edge to a vertex with higher game value. Now by substituting $a_i = a_j = a$ in the equation, restricting Max to consider vertices with same game value, we get that $a = \max(w_{ij} + \beta b_j - b_i)$ and so in the limit $a = \max(w'_{ij})$, so there is an edge such that $\exists k \quad w'_{ik} = a$.

By choosing this edge for each vertex beloging to him, Max will always choose outgoing edges with weight equal to the current vertex value. By a similar reasoning, player Min will also have $a = \min(w'_{ij})$, so that whatever vertex it chooses, it has weight $\geq a$.

Then, in a play starting from a vertex v_i with value a_i, Max can choose all weights $w = a$ and Min can only choose $w \geq a$, so that we have that the value of the game where Max plays its positional strategy σ as now defined, and Min plays whatever τ is $\nu_{\sigma,\tau} = \limsup_n \frac{1}{n} \sum_{k=0}^{n-1} w_k \geq a = \nu_{\sigma^*,\tau^*}$ which is the value of the vertex, so we obtain that σ is also an optimal positional strategy.

Step 4: The game we have obtained is not in canonical form since an edge going between vertices with different value has no restrictions on the weights it carries. To put it in canonical form a very simple transformation has to be done: consider the potential $p(v_i) = Ca_i$, where $C > 0$ is a constant to be specified later.

By using such potential transformation, we have $w''_{ij} = w'_{ij}$ if $a_i = a_j$, and $w''_{ij} = w'_{ij} + C(a_j - a_i)$ if $a_i \neq a_j$. In this way, weights between vertices with the same game value are kept unchanged. On the contrary, weights going to a vertex of higher value $a_j \geq a_i$ will increase and weights going to a vertex of lower value will decrease.

By choosing C big enough we can ensure that arcs going to a vertex of lower value $a_j \leq a_i$ will have $w''_{ij} < a_i$ and thus won't be chosen by a locally optimal strategy. □

Computing the canonical potential transformation

In the previous section we solved mean-payoff games via associated discounted-payoff games. We are now going to explain how to obtain a canonical potential when the optimal positional strategies are known [ZP96]: this will allow us to develop an algorithm to find a canonical potential in psudopolynomial time.

Algorithm 3.3.12 (Canonical potential from an optimal positional strategy)**:** Let G be our mean-payoff game arena and suppose that it has value $a \geq 0$ for a fixed vertex v_0. We remind the reader that such value can be computed in polynomial time by knowing only one positional strategy.

Consider the graph G_{σ^*} obtained from G by removing all outgoing edges from vertices belonging to Max, except those specified by σ^* (the vertices of G_{σ^*} are $\{(v, w) \in V^2 \mid v \in V_{\text{Min}} \text{ or } w = \sigma^*(v)\}$): since the game has positive value from v_0,

there are no negative cycles in G_{σ^*} that are reachable from v_0. The same is obviously true if we subtract a from each of G weights, since now the new arena G' has value 0 starting from v_0.

Define the length of a path to be the sum of the weights encountered in the path. Let $h(v)$ be the minimum oriented distance in the weighted graph G'_{σ^*} from v_0 to the vertex v, that is well defined because of the absence of negative cycles: such function can be computed (by a breadth first search with priority) in polynomial time and is a valid potential transformation since for each vertex v in the graph, and for each edge $v \to^{\alpha} z$ in G'_{σ^*} it holds that $h(v) \geq h(z) + w'(v, z)$ by the definition of minimum distance. By convention let $h(v) = \infty$ when the node v is not reachable from v_0.

By the fact that we have specified an initial optimal strategy we obtain that $\forall v \in V_{\text{Max}}$, $\exists v \to^{\alpha} z$ such that $h(v) \geq h(z) + w'(v, z)$ and for Min it holds the same but for every outgoing edge: $\forall v \in V_{\text{Min}}$, $\forall v \to^{\alpha} z$ we have $h(v) \geq h(z) + w'(v, z)$. Substituting back the original weights, from $h(v) \geq h(z) + w'(v, z)$ we obtain $h(v) + a \geq h(z) + w(v, z)$, therefore h is the desired potential transformation, and is a certificate for the game to have value a.

Remark 3.3.13 (Bound on the potential transformation)**:** We observe that, since in the algorithm h is a sum of at most n edge weights in G', it holds $|h(v)| \leq n(W + |a|) \leq 2nW$ or $h(v) = \infty$.

Algorithm 3.3.14 (Canonical Transformation in pseudopolynomial time)**:** We already know that every mean-payoff game admits a canonic potential and we will now build an updating algorithm to compute such potential. For this algorithm we suppose that the game arena G is ergodic (i.e. all vertices have the same value $a \geq 0$).

First define the map $T : (V \to \mathbb{Q}) \to (V \to \mathbb{Q})$ by:

$$\begin{aligned} T(h)(v) &= \min_{v'}(h(v') - a + w(v, v')) \quad \text{if } v \in V_{\text{Max}} \\ T(h)(v) &= \max_{v'}(h(v') - a + w(v, v')) \quad \text{if } v \in V_{\text{Min}} \end{aligned} \tag{3.3}$$

Note that the map T is monotone and a valid potential function is a prefixed point $T(\overline{h}) \geq \overline{h}$. By Remark 3.1.20 we can compute such a prefixed point by starting with a function $h_0(v) = 2nW \quad \forall v \in V$ and define $h_k = T^k(h_0)$.

If $h_i = h_{i+1}$ then h_i is a valid potential: suppose infact that $h(v) = T(h)(v)$ for every vertex v. We have in particular that $h(v) = \min_{v'}(h(v') - a + w(v, v'))$ for a vertex $v \in V_{\text{Max}}$, which means that exists an edge $v \to v'$ for which we have $h(v) + a \geq h(v') + w(v, v')$ which is the exact definition of potential function.

Note also that the sum $\sum_v h(v)$ decreases strictly at each updating step.

Moreover, if at a certain point we have $h(v) < 0$ then is has to be $h(v) = -\infty$ because of the bound we obtained previously for the potential transformation. This procedure

then can be carried out in time $O(n^2W)$ by observing that, even if the numbers are rational, their denominator is at most n since the only rational number which can be summed or subtracted is a which is a mean of at most n integral numbers.

Remark 3.3.15: We note the similarity between this potential updating algorithm and the progress measure lifting algorithm: in both cases a value is assigned to a vertex and an iterative algorithm updates it until it gets to a prefixed point of a certain function. The form of the inequalities defining the notion of progressive edge also show a remarkable similarity with the inequalities of the definition of potential function.

3.4 LP-type problems

It is widely known that the satisfability problem for linear programming is solvable in polynomial time, for example with the ellipsoid method [Kha79], but it is still open whether it is feasible to solve linear programming in combinatorial polynomial time. The best general result in this direction is by Matouvšek, Sharir and Welzl [MSW96] and Clarkson [Cla95], which presented subexponential algorithm for the task.

Their algorithm solves a greater class of problems, known as LP-type problems, which have been very important to yield linear or close to linear time algorithms for various problems in computational geometry and location theory. Moreover in 2007 Halman [Hal07] showed that they also have application in finding solutions to graph games.

In particular, Halman shows that a simple stochastic game can be formulated as an LP-type problem and hence he obtains a strongly subexponential solution for mean-payoff, discounted-payoff and simple stochastic games by combining the already cited algorithms of Matouvšek, Sharir and Welzl [MSW96] and Clarkson [Cla95].

In this section we state what an LP-type problem is and then we discuss how a simple stochastic game can be seen as an LP-type problem.

Definition 3.4.1 (Abstract problem)**:** An abstract problem is a tuple (H, ω) where H is a finite set of elements (which we call constraints) and ω is an objective function from 2^H to some totally ordered set Λ which contains a special maximal element ∞ and a minimal element $-\infty$. The goal is to compute $\omega(H)$.

Definition 3.4.2 (LP-type problem)**:** An LP-type problem is an abstract problem (H, ω) that obeys the following conditions (when we write inequality symbols we mean under the ordered set Λ):

- **Monotonicity**: For all $F \subseteq G \subseteq H$ it holds $\omega(F) \leq \omega(G)$
- **Locality**: For all $F \subseteq G \subseteq H$ such that $\omega(F) = \omega(G) \neq -\infty$ and for each $h \in H$, if $\omega(G \cup \{h\}) > \omega(G)$ then $\omega(F \cup \{h\}) > \omega(F)$.

Definition 3.4.3 (Basis)**:** A basis B is a set $B \subseteq H$ such that $\omega(B') < \omega(B)$ for all

proper subsets $B' \subsetneq B$.

Definition 3.4.4: Let $G \subseteq H$ be arbitrary:

- G is said to be infeasible if $\omega(G) = \infty$
- G is said to be unbounded if $\omega(G) = -\infty$
- A basis for G is a basis B such that $\omega(B) = \omega(G)$.

Definition 3.4.5 (Combinatorial dimension)**:** The combinatorial dimension d of an LP-type problem is the maximum size of any basis for any feasible subfamily G.

An LP-type problem is said to be fixed dimensional if d is a constant independent of the size n of the problem.

An LP-type algorithm takes a d-dimensional LP-type problem (H, ω) and returns a basis B for H. The randomized algorithm of Sharir and Welzl gets as an input the set of contraints H and a candidate basis $C \subseteq H$. C is not necessarily a basis for H, but it can be viewed as some auxiliary information one gets for the computation of the solution which has no influence on the output of the procedure, but influences its efficiency.

The algorithm uses two kind of primitive operations: a basis computation which takes a family G of at most $d+1$ constraints and finds a basis for G, and a violation test which takes a basis B and a constraint h and returns false if and only if B is a basis for $B \cup \{h\}$.

Let (H, ω) be a d-dimensional LP-type problem and let $n = |H|$. Let t_b the time required for a basis computation and t_v the time required for a violation test. Then the overall randomized running time of carefully combining the two algorithms [SW92] and [Cla95] results in a time complexity of $O(e^{O(\sqrt{d \ln d})}(t_v n + t_b \log n))$. By specifying in which way a simple stochastic game is an LP-type problem and providing polynomial algorithms for the two primitive operations in such setting, we will obtain the required bound.

The following lemma, already proved by Derman [Der70], reduces a single-player simple stochastic game (i.e. where only Min and random vertices are allowed) to a linear programming problem, which can then be solved in polynomial time. This lemma is the analogue of Remark 1.4.4 for simple stochastic mean-payoff games.

Lemma 3.4.6: Let G be a simple stochastic game with no max vertices that halts with probability one. Label the vertices of G such that $V = \{1, \ldots, n\}$ and label the 0-sink by $n-1$ and the 1-sink with n. Then the optimal strategy for player Min can

be found by solving the following linear program:

$$\begin{aligned} \text{maximize } & \sum_{i=1}^{n} v(i) \\ \text{subject to } & v(i) \leq v(j) && \text{if } i \in V_{\text{Min}} \wedge (i,j) \in E \\ & v(i) = \sum_{(i,j)\in E} \text{pr}(i,j)v(j) && \text{if } i \in V_R \\ & v(n-1) = 0 \\ & v(n) = 1 \end{aligned} \tag{3.4}$$

where $\text{pr}(i,j)$ represent the transition probability from vertex i to vertex j.

Idea of the proof: The various constraints ensure that the $v(i)$ values represent the values of the vertices in the simple stochastic game: the second equation is the value equation for random vertices, while the first one roughly means that all Min-successors of a vertex have greater value. The value of the single vertex is then maximized in the extremal condition, ensuring that for $i \in V_{\text{Min}}$ it holds $v(i) = \min_{(i,j)\in E} v(j)$.

Moreover, if we know the solution of the linear program 3.4, we can find an optimal strategy τ for Min in the following way: for $i \in V_{\text{Min}}$ we set $\tau(i) = j$ where $(i,j) \in E$ and $v(i) = v(j)$, which exists by the maximization objective.

To convert a simple stochastic game to an LP-type problem it will now suffice to define a set S such that each subset of S give rise to a strategy σ of Max player, and the function $\omega : 2^S \to \Lambda$ will be the sum of the values that the strategy archieves on each vertex, which can be computed by solving the above linear program on the graph G_σ.

Definition 3.4.7 (LP-type problem associated with a simple stochastic game)**:** Let G be a simple stochastic game and S be the set of edges outgoing from vertices of Max; for a vertex v define $E(v) = \{(v,w) \in E\}$ as the outgoing edges from that vertex.

For every $S' \subseteq S$ we define $G(S')$ as the game where, between the edges outgoing from vertices of Max, only those in S' are kept. By imposing on S' the condition that every max vertex has an outgoing edge in S' we can make sure that every non-sink vertex in $G(S')$ has at least one outgoing edge and that if G halts with probability one then so does $G(S')$.

Let now $\Lambda = \mathbb{R} \cup \{\infty, -\infty\}$ and let $\sigma(S'), \tau(S')$ be an arbitrary pair of optimal strategies in $G(S')$. We define $\omega : 2^S \to \Lambda$ as follows:

$$\omega(S') = \begin{cases} -\infty & \text{if } \exists i : E(i) \cap S' = \emptyset \\ \sum_{z\in V} \nu_{\sigma(S'),\tau(S')}(z) & \text{otherwise} \end{cases} \tag{3.5}$$

Note that if σ, τ is a pair of strategies in the game G, and since it is not always true that

$S_\sigma \subseteq S'$ then σ, τ is not necessarily a pair of strategies in $G(S')$. Conversely if σ, τ is a pair of optimal strategies in $G(S')$, it is a pair of strategies in G which is not necessarily optimal since the set of possible strategies for player Max in G is in general strictly bigger than those in $G(S')$.

There is fortunately a sufficient condition that ensures the optimality of σ, τ in G, which allows to effectively recover the solutions of the simple stochastic games from a basis of the LP-type problem formulation.

Lemma 3.4.8: Let G be a simple stochastic game that halts with probability one. Let $G(S')$ be a simple stochastic sub-game of G and σ, τ a pair of optimal strategies in $G(S')$. If $\omega(S) = \omega(S')$ then σ, τ is a pair of optimal strategies in G as well.

To conclude the reformulation as LP-type problem it is enough to prove monotonicity and locality: the first one is trivial, while the second one uses a technical lemma on unstability of vertices. To obtain a subexponential algorithm then Halman proves that the associated LP-type problem has dimension $d = |V_{\text{Max}}|$, which combined with the initial estimates provides the complexity bound of $O(e^{O(\sqrt{n \ln n})})$.

Chapter 4

A new Game

We are going to define a new graph game and prove that the problem of solving it lies inbetween parity games and mean-payoff games. This could help in finding an algorithm to solve mean-payoff games because this graph game helps separate the main algorithmic difficulties found in solving mean-payoff games. We will then analyze adaptations of existing algorithms to asses their effectiveness against our new game.

4.1 Stacked unary mean-payoff games

Recall that, in dealing with computational problems, we are always assuming that instances of a problem are coded, say by binary strings, in a canonical way. Integers are usually coded in binary positional notation, so that $\text{size}(n) = \lceil \log_2 n \rceil$. In order to make hard problems appear easier one can code integers in unary notation, so that $\text{size}(n) = n$. By artificially expanding the size of the problem representation, one can thus lower its complexity.

Definition 4.1.1 (Unary mean-payoff game)**:** A unary mean-payoff game is a mean-payoff game where weights are coded in unary.

Observe that this definition makes a pseudo-polynomial algorithm for mean-payoff games into a polynomial algorithm for unary mean-payoff game.

Definition 4.1.2 (Stacked unary mean-payoff game)**:** Weights are polynomials with integer coefficients $W = \mathbb{Z}[x]$.

Given a play $p_0, \ldots$ we consider the space of all accumulation points (which are polynomials) of the sequence $a_n = \frac{1}{n} \sum_{i=0}^{n-1} w_i$ in the pointwise convergence topology. We define the value of the play to be the maximum of those limit polynomials in the lexicographic order (i.e. as if $x = \infty$), and we will denote such limit operation by $\text{maxlexlim}_{n\to\infty} a_n$.

Notation 4.1.3: We will denote by $P(d, W)$ the set of polynomials in $\mathbb{Z}[x]$ of degree

at most d and with coefficients bounded in the range $[-W, W]$. In what follows we will also denote by W the minimal bound on the polynomial coefficient of a stacked unary mean-payoff arena's weights.

A similar game where the weights are tuples of integer ordered lexicographically is equivalent. Polynomials were chosen because they make proofs more clear and easier to reason about.

We are now going to prove positional determinancy for Stacked unary mean-payoff in two ways: the first will be an approximation result via mean-payoff games similar to the proof of Theorem 1.4.23; the second one will use the result of Gimbert and Zielonka [GZ05] (Remark 1.4.28)with the observation that the value of a play in necessarily less or equal to the highest mean of a reachable cycle. We will then prove that the complexity of stacked unary mean-payoff lies inbetween that of parity games and mean-payoff games, by showing a chain of reductions.

Remark 4.1.4 (Well-definedness of the play value)**:** We prove that the maximum in the lexicographic order of the limit polynomials exists. The set of polynomials with bounded coefficient and fixed degree is a compact set in the pointwise convergence topology, and therefore also the limit points of $\{a_n\}_{n\in\mathbb{N}}$, being a closed subset, are a compact set which will be denoted by $S \subseteq P(d, W)$.

Now consider the map $[x^i] : P(d, W) \to \mathbb{R}$ which associates to each polynomial its coefficient of x^i, which is clearly continuous w.r.t. the pointwise convergence in $P(d, W)$. We now prove by induction on d that each compact set $S \subseteq P(d, W)$ in the pointwise convergence topology has a lexicographic maximum.

For $d = 0$, we consider $B = [x^0]S$ which is compact in the real line, and therefore has a maximum which coincides with the lexicographic one. Suppose the thesis holds for d, we show that it also holds for $d{+}1$: let $B = [x^{d+1}]S$ which is compact on the real line, and let $m \in B$ be its maximum. This is also the maximum coefficient of degree x^{d+1} of a polynomial in S. Consider then the set $R = \{p \in S \mid [x^{d+1}]p = m\}$, which is closed in $P(d{+}1, W)$ and therefore compact. All of the polynomials in R have m as their first coefficient; therefore we know that $\max_{\text{Lex}} S = \max_{\text{Lex}} R = mx^{d+1} + \max_{\text{Lex}} f(R)$, where $f(p) = p - mx^{d+1}$.

Lemma 4.1.5: The value of a play is less than or equal to the highest mean value of a simple cycle.

Proof: Let $r_1, \ldots, r_s$ be the sums of the edge values of each simple cycle $c_1, \ldots, c_s$, and $l_1, \ldots, l_s$ their lengths. By the cycle decomposition (Procedure 1.4.22) a partial play value can be written as:

$$S_n = \frac{1}{n}\sum_{i\le n} w_i = \frac{\sum_j a_j^{(n)} r_j + B^{(n)}}{\sum_j a_j^{(n)} l_j + k^{(n)}}$$

where $a_j^{(n)}$ is the number of times the cycle c_j has been stepped over, $B^{(n)}$ is the sum of edges which remained in the stack, and $k^{(n)}$ is the number of edges on the stack.

Let now R be a polynomial which is lex-greater than every mean value of a simple cycle, i.e. $\forall i \quad R \geq_{\text{Lex}} \frac{r_i}{l_i}$, then it holds

$$S_n = \frac{\sum_j a_j^{(n)} l_j \frac{r_j}{l_j} + B^{(n)}}{\sum_j a_j^{(n)} l_j + k^{(n)}} \leq_{\text{Lex}} \frac{\left(\sum_j a_j^{(n)} l_j\right) R + B^{(n)}}{\sum_j a_j^{(n)} l_j + k^{(n)}} = \frac{(n - k^{(n)})R + B^{(n)}}{n} = R_n$$

We can now use a similar reasoning to that of the Remark 4.1.4, and start by analyzing this inequality between the top-most coefficients $[x^d]S_n \leq [x^d]R_n$. We have two cases: if $\lim_n [x^d]S_n = [x^d]S < [x^d]R$ then we are done since this already proves $S <_{\text{Lex}} R$. Otherwise if $[x^d]S = [x^d]R$, it is not too hard to analyze the inequality between limits to prove that most of the cycles taken are from cycles where $\frac{[x^d]r_j}{l_j} = [x^d]R$. More precisely, let $b_j = \lim_{n\to\infty} \frac{a_j^{(n)}}{n}$ then we have $\sum_j b_j = 0$ and if $b_j > 0$ then $\frac{[x^d]r_j}{l_j} = [x^d]R$. Now by induction on d one can prove the statement of the lemma. □

Theorem 4.1.6: Stacked unary mean-payoff games are positionally determined.

Idea of the proof: We approximate the given game by a sequence of mean-payoff games and use their determinancy to get a pair of strategies which are frequently optimal on the sequence of games. We then show that such strategies are also optimal on our original game.

Proof: Let $G(x)$ be a stacked unary mean-payoff game, and denote by $G(n)$ for $n \in \mathbb{N}$ the resulting mean-payoff obtained by the weight translation $\mathcal{F}_n(p(x)) = p(n)$. Since the mean-payoff game sequence $\{G(n)\}_{n\in\mathbb{N}}$ is made of positionally determined game and the positional strategies are a finite number, there must exists a couple (σ^*, τ^*) of strategies of Max and Min respectively, which are frequently optimal on $G(n)$, i.e. $\exists\{n_k\}_{k\in\mathbb{N}}$ such that (σ^*, τ^*) is a couple of optimal strategies for $G(n_k)$.

We will now prove that the couple of strategies (σ^*, τ^*) is optimal on $G(x)$. Indeed, consider any counter-strategy τ of Min (even non-positional) to σ^* and σ of Max to τ^*. Then by the above Lemma 4.1.5 we know that each play on $G_{\sigma^*}(x)$ has a score which is greater than or equal the minimal cycle value in $G_{\sigma^*}(x)$. Let $\text{mcv}(G(x))$ be the minimal cycle value of $G(x)$. We should prove that, for big enough $k \in \mathbb{N}$

$$\text{mcv}(G_{\sigma^*}(x))(k) = \text{mcv}(G_{\sigma^*}(k)) \tag{4.1}$$

from which it will follow that

$$\mathcal{V}_{\sigma^*,\tau}^{G(x)}(k) \geq \text{mcv}(G_{\sigma^*}(x))(k) = \text{mcv}(G_{\sigma^*}(k)) = \mathcal{V}_{\sigma^*,\tau^*}^{G(k)}$$

for large enough k, which implies positional determinancy.

Proving Equation 4.1 is simple: we first notice that, given any two fixed-degree polynomials $p(x), q(x)$, it holds $p(x) \leq_{\text{Lex}} q(x)$ if and only if $\text{FIN}_k \ p(k) \leq q(k)$;

now let S be the set of mean cycle values of all simple cycles in $G_{\sigma^*}(x)$, which is a finite set. By this fact we know that $\exists k$ such that $\forall n > k$ and $\forall p, q \in S$ we have $p \leq_{\text{Lex}} q \Leftrightarrow p(n) \leq q(n)$ and therefore Equation 4.1 is proven. □

Remark 4.1.7 (Reproving Positional determinancy)**:** Another way to prove positional determinancy of the game is to refer to Remark 1.4.28 and observe that single-player games are trivially positionally determined since Max player has only interest in reaching the cycle with highest mean, and analogously Min player is interested only in reaching the cycle with the most negative mean. We will now prove this for games where only Max plays.

Consider first the reachable cycle with the highest mean-value and restrict the single-player game to the reachable part of the graph. By Lemma 4.1.5 the value of the play is less than this highest mean-value, so the strategy of Max player should be to cycle on its vertices, and to converge as fast as possible to the relevant cycle from every other vertex. It is then enough to note that such strategy is positional.

Lemma 4.1.8: Let σ be a memoryless winning strategy for player Max from v_0 in the stacked unary mean-payoff G. Then for every siple cycle c in G_σ reachable from v_0 the sum of the weights of the edges of c is non-negative.

Proof: Suppose that there exists a simple cycle c with negative sum of edge weights. Then player Min can force the play from v_0 to c and also to stay in c indefinitely and thus win, contradicting the assumption about σ. □

Theorem 4.1.9: Parity games are reducible to stacked unary mean-payoff.

Idea of the proof: Consider the reduction $p \mapsto (-1)^p x^p$ from a parity game G to the stacked unary mean-payoff G'. If the highest value in a cycle in G is even then the mean of the same cycle in G' is positive. Then we use the tecnique of cycle decomposition to prove the theorem.

Proof: Consider $F(p) = (-1)^p x^p$ as the reduction from the parity game G to $F(G)$. It is easy to prove that if the highest value in a cycle in G is even then the mean of the same cycle in $F(G)$ is positive: suppose the priorities of the cycle are $p_1, \ldots, p_k$. Suppose now that $\forall i \quad p_k \geq p_i$ and p_k is even: then we have $\sum_i F(p_i) = \sum_i (-1)^{p_i} x^{p_i}$, where x^{p_k} is the highest grade that appears, with positive coefficient since p_k is even.

Now we remind the reader of the cycle decomposition technique (Procedure 1.4.22) and proceed to completing the proof: suppose player Max wins in G from v_i with positional strategy σ. Then in G_σ the highest value in each cycles is even, and the sum of the weights of every cycle in $F(G)_\sigma$ is non-negative, which means that player Max wins in $F(G)$ from v_i with the same positional strategy and which proves the reduction. □

Theorem 4.1.10: Stacked unary mean-payoff games are reducible to mean-payoff

Idea of the proof: Consider the reduction $p(x) \mapsto p(N)$ where $N > nW$ from a stacked unary mean-payoff G to the resulting mean-payoff game G'. The mean of a cycle in G is positive if and only if the mean of the same cycle in G' is positive. We then use the technique of cycle decomposition to prove the theorem.

Proof: Consider $F(p(x)) = p(N)$ with $N > nW$ as the reduction from the stacked unary mean-payoff game G to $F(G)$. It is easy to prove that if the mean-value of a cycle in G is non-negative, then also the mean of the same cycle in $F(G)$ is non-negative: suppose the weights of the cycle are $w_1(x), \ldots, w_k(x)$ and suppose that $e(x) = \sum_i w_i(x) \geq 0$; we have that $e(x) \in \mathbb{Z}[x]$ and coeff$(e) \in [-nW, nW]$. Therefore $E = \sum_i w_i(N) \geq 0$ by a previous observation on the lexicographic order and evaluating polynomials at high natural numbers.

Using now the cycle decomposition technique (Procedure 1.4.22), suppose player Max wins in G from v_i with positional strategy σ. Then in G_σ the mean-value of each cycle is non-negative, so the mean-value of each cycle in $F(G)_\sigma$ is non-negative, which means that player Max wins in $F(G)$ from v_i with the same positonal strategy. □

In essence Stacked unary mean-payoff encode "sparse" mean-payoff games, i.e. those where the payoffs decompose into sums of numbers differing by at least one order of magnitude (by at least a factor of $N = nW$). Suppose for example that all weights of a mean-payoff games are taken from the set $\{3, 101, 10002\}$. Then it is clear that if the mean-payoff has fewer than 30 vertices, we can inversely use the above reduction: we could transform weights into $3 \to 3$, $101 \to x + 1$, $10002 \to x^2 + 2$ (i.e. as if $x = 100$) and then solve the easier stacked unary mean-payoff. Such observation leads to the following question:

Problem 4.1.11: To which mean-payoff games a similar transformation can be applied?

The answer to this question may shed light on possible ways to solve mean-payoff games and so they should be the subject for further investigations.

Remark 4.1.12 (Mean-payoff games are "strictly" more difficult than stacked unary mean-payoff)**:** One might suspect that if the weights of a mean-payoff gmae are exponentially distributed, then in essence, it is a stacked unary mean-payoff game. This is not the case, infact we note that each mean-payoff game can be reduced to a mean-payoff with only zeros and powers of two on the edges by replacing each edge with weight $e = \sum_{i<\lceil \log W \rceil} 2^{d_i}$ with $\lceil \log W \rceil$ edges of weights 2^{d_i}.

4.2 Resolution methods

We will now address possible resolution methods of stacked unary mean-payoff games, which are adapted from methods solving parity games. We highlight problems in adapting the respective quasi-polynomial solution methods, thereby also making strong arguments that the step from parity games to stacked unary mean-payoff is not trivial too.

4.2.1 Adaptation of Small Progress Measure

Notation 4.2.1: Let $\alpha \in \mathbb{N}^{d+1}$ be a $(d+1)$-tuple of non-negative integers.

- We number its components from 0 to d, i.e. $\alpha = (\alpha_d, \ldots, \alpha_1, \alpha_0)$
- $<, \leq, =, \neq, \geq, >$ on tuples denote lexicographic ordering
- We define the sum between tuples to be carried componentwise

Definition 4.2.2: Define $\mathbb{M}_G \subseteq \mathbb{N}^d$ such that it is the finite set of $(d+1)$-tuples with all components bounded between 0 and nW:

$$\mathbb{M}_G = \{(\alpha_d, \ldots, \alpha_0) \mid 0 \leq \alpha_i \leq nW\}$$

Definition 4.2.3 (Progress Measure)**:** Let G be a stacked unary mean-payoff game; then the function $\rho : V \to \mathbb{M}_G$ is a parity progress measure for G if every edge $(v, u) \in E$ is progressive, i.e. if

$$\rho(v) + w(v, u) \geq \rho(u)$$

The idea of a progressive edge is that it should characterize cycles with mean greater than zero, which is also the statement of the next lemma.

Lemma 4.2.4: Let ρ be a progress measure for G. Then all cycles in G have non-negative mean.

Proof: Suppose by contradition that there is a negative cycle $v_1, \ldots, v_k = v_1$ in G. Then we have the chain of inequalities

$$\rho(v_1) = \rho(v_k) \leq \rho(v_{k-1}) + w(v_{k-1}, v_k) \leq \ldots \leq \rho(v_1) + \sum_i w(v_{i-1}, v_i)$$

which gives the contradiction $\sum_i w(v_{i-1}, v_i) \geq 0$. □

Lemma 4.2.5: Let G be a stacked unary mean-payoff game where all cycles have non-negative mean. Then there exists a progress measure $\rho : V \to \mathbb{M}_G$ for G.

Proof: The proof is by induction on the number of strongly connected components of G. For the base case suppose that G consists only of a single strongly connected component and let v be a point on this component. Then G is a weighted graph with non-negative cycles and therefore one can define the distance between two vertices as the

length of the shortest path connecting those points, which exists and is non-negative for every couple of vertices and is also bounded by $(n-1)\cdot W$. We can therefore define $\mu(w) = d(v,w)$ which satisfied the triangular inequality $\mu(u)+c(u,w) \geq \mu(w)$.

For the inductive case, let $C_0, \ldots, C_k$ be the strongly connected components of the graph G, and let $\mu_0, \ldots, \mu_k$ be their respective progress measures computed with the outlined procedure. By taking the quotient graph of G collapsing each C_i to a point, we can see that the resulting graph G' is acyclic; hence there must exists a strongly connected component without ingoing edges (from other components), let it be C_0.

The inductive hypothesis guarantees the existence of a progress measure μ defined on $G \setminus C_0$ with values in $[0, (|G \setminus C_0| - 1)W]$. To extend it to a progress measure to all of G it is enough to consider $\mu' = (\mu_0 + K) \cup \mu$ for a certain constant $K \in \mathbb{N}$. In this way edges inside of C_0 and inside of $G \setminus C_0$ are progressive for μ' because they are progressive for the respective measures μ_0 and μ. Moreover if an edge (u,v) goes from C_0 to $G \setminus C_0$ the edge is progressive if and only if $\mu_0(u)+K+w(u,v) \geq \mu(v)$. Therefore it is enough to set $K = \max_{(u,v)\in E} \mu(v) - \mu_0(u) + w(u,v) \leq (|G \setminus C_0| - 1)W + W = |G \setminus C_0|W$ which gives the required bound on μ': $mu'(v) \in [0, (|G| - 1)W]$. □

We can now extend the definiton of progress measure and expose the lifting algorithm.

Definition 4.2.6: Define $\mathbb{M}_G^\top = \mathbb{M}_G \cup \{\top\}$ by setting

$$\forall m \in \mathbb{M}_G \quad m < \top$$

Definition 4.2.7: If $\rho : V \to \mathbb{M}_G^\top$ and $(v,u) \in E$ is an edge, then $\mathrm{Prog}(\rho, v, u)$ is the least $m \in \mathbb{M}_G^\top$ such that $m + w(v,u) \geq \rho(u)$ and $m \geq \rho(v)$.

Definition 4.2.8 (Game progress measure)**:** Let G be a stacked unary mean-payoff game. A function $\rho : V \to \mathbb{M}_G^\top$ is a game progress measure if for all vertices $v \in V$ we have:

- If $v \in V_{\mathrm{Max}}$ then $\exists (v,u) \in E$ that is progressive
- If $v \in V_{\mathrm{Min}}$ then $\forall (v,u) \in E$ are progressive

Definition 4.2.9 (Induced strategy from game progress measure)**:** Let G be a stacked unary mean-payoff game and $\rho : V \to \mathbb{M}_G^\top$ a game parity progress measure.

Define the induced strategy $\overline{\rho} : V_{\mathrm{Max}} \to V$ for player Max by setting

$$\overline{\rho}(v) = \operatorname*{argmin}_w \{\rho(w) \mid (v,w) \in E\}$$

that is $\rho(v)$ is a successor w of v that minimizes $\rho(w)$.

Let moreover $||\rho|| = \{v \in V \mid \rho(v) \neq \top\}$ the vertices from which the recheable cycles are non-negative.

Lemma 4.2.10 (Induced strategy is winning for Max)**:** If ρ is a game progress measure, then $\overline{\rho}$ is a winning strategy for player Max from $||\rho||$.

Proof: We note that all vertices we will be considering are inside $||\rho||$: the starting vertex obviously is, and for each vertex v one can note that each successor chosen by Min satisfied $\rho(v) + c(v, w) \geq \rho(w)$ so that $\rho(w) \neq \top$, and since Max choses the w that minimizes $\rho(w)$, one also has $\rho(w) \neq \top$ for $w = \overline{\rho}(v)$.

Then it can easily be seen that strategy $\overline{\rho}$ is closed on $||\rho||$ and that ρ restricted to $||\rho||$ is a progress measure on $G_{\overline{\rho}}$ restricted to $||\rho||$. By the lemma 4.2.4 we can then conclude that all cycles in $G_{\overline{\rho}}$ restricted to $||\rho||$ are non-negative, which is the statement of this lemma. □

Lemma 4.2.11 (Existence of a parity progress measure)**:** Let W_{Max} be the winning set of player Max in a stacked unary mean-payoff G. Then there exists a game progress measure $\rho : V \to \mathbb{M}_G^\top$ such that $||\rho|| = W_{\text{Max}}$.

Proof: The existence again follows from the analogue statement for progress measures (Lemma 4.2.5): we know that all cycles in the game $G\,|_{W_{\text{Max}}}$ are non-negative, hence there is a progress measure $\rho : W_{\text{Max}} \to \mathbb{M}_G$ for the restricted game. Extending $\rho\,|_{V \setminus W_{\text{Max}}} = \top$ makes ρ into a game progress measure. □

Definition 4.2.12 (Lifting of a progress measure)**:** Define the lifting operator $\text{Lift}(\rho, v)(u)$ for $v \in V$ as follows:

- $\rho(u)$ if $u \neq v$
- $\min\{\text{Prog}(\rho, v, u) \mid (v, u) \in E\}$ if $u = v \in V_{\text{Max}}$
- $\max\{\text{Prog}(\rho, v, u) \mid (v, u) \in E\}$ if $u = v \in V_{\text{Min}}$

Lemma 4.2.13 (Lifting is monotone)**:** The operator $\text{Lift}(\cdot, v)$ is monotone in the functional lattice.

Proof: This is proven in exactly the same way as Lemma 3.1.23 since the operations described here have the same monotonicity properties as those in the description of the Small Progress Measures algorithm. □

Lemma 4.2.14 (Game progress measures as fixpoints)**:** A function $\rho : V \to \mathbb{M}_G^\top$ is a game progress measure if and only if $\forall v \in V \quad \text{Lift}(\rho, v) \sqsubseteq \rho$

Proof: $\boxed{\Rightarrow}$ Let ρ be a game progress measure: we show that it is a prefixed point: if $u \neq v$ then $\text{Lift}(\rho, v)(u) = \rho(u)$, which proves inequalities on such vertices. On the other hand if $u = v$ from the definition of game progress measure we know that:

- If $v \in V_{\text{Max}}$ then $\rho(v) + c(v, w) \geq \text{Prog}(\rho, v, w)$ for some w. Since the lift considers the min of such progs we obtain that $\rho(v) + c(v, w) \geq \text{Lift}(\rho, v)(u)$

- If $v \in V_{\text{Min}}$ then $\rho(v) + c(v, w) \geq \text{Prog}(\rho, v, w)$ for all w, and we again obtain its description as a prefixed point.

$\boxed{\Leftarrow}$ This part of the proof is obtained similarly as it is enough to reverse the steps of the other direction. □

Again the least game progress measure can be computed using fixed point approximation from Remark 3.1.20.

Remark 4.2.15 (Algorithm running time)**:** As in the case of Small Progress Measure for parity games, the running time of the algorithm is essentially bounded by the size of $\mathbb{M}_G^\top$, which unfortunately is exponential: $\left|\mathbb{M}_G^\top\right| = W^{d+1}$.

Unfortunately we have not been able to generalize the succinct progress measure algorithm for stacked unary mean-payoff games. In this variant of progress measures there is also the need to be able to sum numbers, other than to be able to compare them. The main technical difficulty we have encountered was indeed in encoding sums into a succinct scheme.

4.3 Value-iteration algorithms

We already observed in Remark 3.3.15 that the algorithm of Zwick and Paterson for the solution of mean-payoff games and the progress measures algorithms bear a shocking similarity. In this section we introduce the concept of value-iteration algorithm [CH08] and explore more in detail such similarity.

In a value-iteration algorithm each vertex of a graph is assigned a value, and the values are iteratively improved until a fixpoint is reached. Moreover the improvement function is local, meaning that the new improved value at a vertex depends on the old values at neighbouring vertices.

Definition 4.3.1 (Valued Graph)**:** A valued graph (G, D) consists of:

- A finite graph G, where each vertex has at least one successor.
- A complete lattice of values D.

Definition 4.3.2 (Valuation)**:** A valuation is a function $\nu : V \to D$ from the graph vertices to the value lattice.

The ordering on values is lifted to valuations in a pointwise fashion: for two valuations μ, ν we write $\mu \leq \nu \Leftrightarrow \forall v \in V \quad \mu(v) \leq \nu(v)$. Note that valuations themselves form a complete lattice.

We remind the reader that a chain C in a lattice is a set of elements which are all pairwise comparable.

Definition 4.3.3 (Value Improvement)**:** An improvement function $\text{Imp} : D^V \to D^V$ is a function on valuations which satisfies the following requirements:

- Monotone: $\forall v_1, v_2 \in D^V$ if $v_1 \leq v_2$ then $\text{Imp}(v_1) \leq \text{Imp}(v_2)$.
- Continuous: for every chain $C = \langle v_0, v_1, \ldots \rangle$ of valuations, we require $\text{Imp}(\lim C) = \lim \text{Imp}(C)$. Note that by monotonicity $\text{Imp}(C)$ is a chain itself.
- Directed: it holds either $v \leq \text{Imp}(v)$ for all valuations or $v \geq \text{Imp}(v)$ for all valuations.
- Local: $\forall s \in S$ states and all valuations $v_1, v_2 \in D^V$ if $v_1(s') = v_2(s')$ for all successors $s' \in E(s)$, then $\text{Imp}(v_1)(s) = \text{Imp}(v_2)(s)$.

It it not difficult to prove that given a value improvement function with these properties, the least fixpoint of it exists and is computable by Knaster-Tarski approximations.

In the algorithm of Zwick and Paterson, such improvement function is the map T in Equation 3.3 which was defined by:

$$\begin{aligned} T(h)(v) = &\quad \min_{v'}(h(v') - a + w(v, v')) \quad \text{if } v \in V_{\text{Max}} \\ T(h)(v) = &\quad \max_{v'}(h(v') - a + w(v, v')) \quad \text{if } v \in V_{\text{Min}} \end{aligned} \tag{4.2}$$

Now that we have exposed a variation of small progress measure, the similarity between the two can be more clearly seen. Note though that each Lift_v function is an improvement function, and we are searching for a fixpoint of all of them at the same time. $\text{Lift}(\rho, v)(u)$ is defined in Definition 4.2.12 as:

$$\begin{aligned} \rho(u) &\quad \text{if } u \neq v \\ \min\{\text{Prog}(\rho, v, w) \mid (v, w) \in E\} &\quad \text{if } u = v \in V_{\text{Max}} \\ \max\{\text{Prog}(\rho, v, w) \mid (v, w) \in E\} &\quad \text{if } u = v \in V_{\text{Min}} \end{aligned} \tag{4.3}$$

By unwinding the definition of Prog in Definition 4.2.7 one can see that the definition can be in fact written as:

$$\begin{aligned} \text{Lift}_v(\rho)(v) = &\quad \min_{v'}(\rho(v') - w(v, v')) \quad \text{if } v \in V_{\text{Max}} \\ \text{Lift}_v(\rho)(v) = &\quad \max_{v'}(\rho(v') - w(v, v')) \quad \text{if } v \in V_{\text{Min}} \end{aligned} \tag{4.4}$$

which bears remarkable similarity with the previous equations.

4.4 Conclusions

We saw that recently the tractability bound of graph games moved from below parity games to slightly above parity games. Nevertheless, the solution of mean-payoff games seems to be yet out of reach of current methods.

To the aim of simplifying the testing of new resolution methods it seems logical to add new degrees of difficulty in the game hierarchy. We propose to study stacked unary mean-payoff

games as new node in the hierarchy; a game which shares the same relevant properties of parity and mean-payoff games (positionally determined, zero sum, perfect information), and is of intermediate difficulty.

We have seen how the progress measure algorithm can be adapted to stacked unary mean-payoff games, giving an exponential time algorithm for the solution. We also proved that solving mean-payoff games is "strictly" more difficult than solving the two intermediate problems of solving stacked unary mean-payoff games and solving mean-payoff games with an oracle for stacked unary mean-payoff games.

It is highly probable that neither of those two problems is trivial because quasi-polynomial algorithms for parity games do not seem to apply nor adapt nicely to stacked unary mean-payoff games and on the other hand there are some mean-payoff games of polynomial growth that do not come from the reduction from stacked unary mean-payoff games.

It is then natural to ask:

Problem 4.4.1: What is the complexity of solving stacked unary mean-payoff games?

Problem 4.4.2: What is the complexity of solving mean-payoff games with an oracle for the solution of unary stacked mean-payoff games?

Our hope is that stacked unary mean-payoff could contribute in separating the algorithmic difficulties found in solving mean-payoff games in two more manageable steps.

Acknowledgements

Writing a thesis is a task that spans many months and which consists in understanding a big amount of literature works related to a topic, and then expose such concepts in a form that can be better understood by everyone which is interested in knowing more about the field. Yet such work is seldomly read by others, if at all, and it mainly constitutes a rite of passage for the author.

I am thus very grateful to my supervisor Marcello Mamino, that not only helped me in understanding graph games and technical issues in their algoritmic solutions, but also spent countless hours in suggesting improvements in the topics' presentations, which required him to fully read my thesis multiple times. He also gave me an opportunity to grasp what a research work is like, by challenging me with currently unanswered questions about graph games and by providing possible pathways to a solution or building counterexamples to a proposed idea.

Other people also helped me with fruitful discussions about the theorems exposed in the thesis, and more generally helped me in growing up as a mathematician: they engaged in stimulating discussions and brought many interesting topics to my attention; we spent long days (and many nights) studying together, and we also sustained each other to keep going on despite the pressure of these years. It is for these reasons that they are a significant part of my formation years, and so I sincerely thank (in no particular order) Dario Ascari, Andrea Marino, Gianluca Tasinato, Federico Franceschini, Andrea Caberletti, Umberto Pappalettera, Andrea Clini, Luca Capizzi, Marco Costa, Federico Belliardo, Piero Lafiosca, Fabio Zoratti, Enrico Polesel and Manuele Cusumano; I hope they will forever smile remembering me saying "quale freccia stai facendo?".

I would also like to thank my family and my friends: even if they didn't help in writing this thesis, they have helped me a lot with their support in my personal life. There would be too many names to add here, and I will surely forget someone anyhow, so that I prefer not to write them.

Finally, I would like to thank what inspired me to persue the study of mathematics. I experienced lot of frustration, and it was not easy for me to focus and finish university, but looking back now I see that I enjoyed it.

Dario Balboni

Bibliography

[AKS04] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. "PRIMES is in P". In: *Annals of mathematics* (2004), pp. 781–793.

[All+14] Xavier Allamigeon et al. "Combinatorial simplex algorithms can solve mean payoff games". In: *SIAM Journal on Optimization* 24.4 (2014), pp. 2096–2117.

[AM09] Daniel Andersson and Peter Bro Miltersen. "The complexity of solving stochastic games on graphs". In: *International Symposium on Algorithms and Computation*. Springer. 2009, pp. 112–121.

[Bab16] László Babai. "Graph isomorphism in quasipolynomial time". In: *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*. ACM. 2016, pp. 684–697.

[Bel58] Richard Bellman. "On a routing problem". In: *Quarterly of applied mathematics* 16.1 (1958), pp. 87–90.

[BSV04] Henrik Björklund, Sven Sandberg, and Sergei Vorobyov. "Memoryless determinacy of parity and mean payoff games: a simple proof". In: *Theoretical Computer Science* 310.1-3 (2004), pp. 365–378.

[Cal+17] Cristian S Calude et al. "Deciding parity games in quasipolynomial time". In: *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*. ACM. 2017, pp. 252–263.

[CH08] Krishnendu Chatterjee and Thomas A Henzinger. "Value iteration". In: *25 Years of Model Checking*. Springer, 2008, pp. 107–138.

[Cla95] Kenneth L Clarkson. "Las Vegas algorithms for linear and integer programming when the dimension is small". In: *Journal of the ACM (JACM)* 42.2 (1995), pp. 488–499.

[Con92] Anne Condon. "The complexity of stochastic games". In: *Information and Computation* 96.2 (1992), pp. 203–224.

[Cze+19] Wojciech Czerwiński et al. "Universal trees grow inside separating automata: Quasi-polynomial lower bounds for parity games". In: *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics. 2019, pp. 2333–2349.

[Der70] Cyrus Derman. *Finite state Markovian decision processes*. Tech. rep. 1970.

[Dij18] Tom van Dijk. "Oink: An implementation and evaluation of modern parity game solvers". In: *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer. 2018, pp. 291–308.

[DJL18] Laure Daviaud, Martin Jurdziński, and Ranko Lazić. "A pseudo-quasi-polynomial algorithm for mean-payoff parity games". In: *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*. ACM. 2018, pp. 325–334.

[EJ91] E Allen Emerson and Charanjit S Jutla. "Tree automata, mu-calculus and determinacy". In: *[1991] Proceedings 32nd Annual Symposium of Foundations of Computer Science*. IEEE. 1991, pp. 368–377.

[EM79] Andrzej Ehrenfeucht and Jan Mycielski. "Positional strategies for mean payoff games". In: *International Journal of Game Theory* 8.2 (1979), pp. 109–113.

[Eme97] E Allen Emerson. "Model checking and the mu-calculus". In: *DIMACS series in discrete mathematics* 31 (1997), pp. 185–214.

[FF62] LR Ford and DR Fulkerson. "Flows in Networks". In: (1962).

[GKK88] Vladimir A Gurvich, Alexander V Karzanov, and LG Khachivan. "Cyclic games and an algorithm to find minimax cycle means in directed graphs". In: *USSR Computational Mathematics and Mathematical Physics* 28.5 (1988), pp. 85–91.

[GZ05] Hugo Gimbert and Wiesław Zielonka. "Games where you can play optimally without any memory". In: *International Conference on Concurrency Theory*. Springer. 2005, pp. 428–442.

[Hal07] Nir Halman. "Simple stochastic games, parity games, mean payoff games and discounted payoff games are all LP-type problems". In: *Algorithmica* 49.1 (2007), pp. 37–50.

[JL17] Marcin Jurdziński and Ranko Lazić. "Succinct progress measures for solving parity games". In: *2017 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. IEEE. 2017, pp. 1–9.

[Jur00] Marcin Jurdziński. "Small progress measures for solving parity games". In: *Annual Symposium on Theoretical Aspects of Computer Science*. Springer. 2000, pp. 290–301.

[Jur98] Marcin Jurdziński. "Deciding the winner in parity games is in UP ∩ co-UP". In: *Information Processing Letters* 68.3 (1998), pp. 119–124.

[Kar78] Richard M Karp. "A characterization of the minimum cycle mean in a digraph". In: *Discrete mathematics* 23.3 (1978), pp. 309–311.

[Kha79] Leonid G Khachiyan. "A polynomial algorithm in linear programming". In: *Doklady Academii Nauk SSSR*. Vol. 244. 1979, pp. 1093–1096.

[Koz82] Dexter C Kozen. *Results on the Propositional M [mu]-calculus*. Århus Universitet, Matematisk Institut, 1982.

[Leh18] Karoliina Lehtinen. "A modal μ perspective on solving parity games in quasi-polynomial time". In: *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*. ACM. 2018, pp. 639–648.

[LP07] Yuri M Lifshits and Dmitri S Pavlov. "Potential theory for mean payoff games". In: *Journal of Mathematical Sciences* 145.3 (2007), pp. 4967–4974.

[Lud95] Walter Ludwig. "A subexponential randomized algorithm for the simple stochastic game problem". In: *Information and computation* 117.1 (1995), pp. 151–155.

[Mam17] Marcello Mamino. "Strategy recovery for stochastic mean payoff games". In: *Theoretical Computer Science* 675 (2017), pp. 101–104.

[Mar75] Donald A Martin. "Borel determinacy". In: *Annals of Mathematics* (1975), pp. 363–371.

[Mar85] Donald A Martin. "A purely inductive proof of Borel determinacy". In: *Recursion Theory, Proceedings of Symposia in Pure Mathematics*. Vol. 42. American Mathematical Society. 1985, pp. 303–308.

[MSS04] Rolf H Möhring, Martin Skutella, and Frederik Stork. "Scheduling with AND/OR precedence constraints". In: *SIAM Journal on Computing* 33.2 (2004), pp. 393–415.

[MSW96] Jiří Matoušek, Micha Sharir, and Emo Welzl. "A subexponential bound for linear programming". In: *Algorithmica* 16.4-5 (1996), pp. 498–516.

[Par19] Paweł Parys. "Parity Games: Zielonka's Algorithm in Quasi-Polynomial Time". In: *arXiv preprint arXiv:1904.12446* (2019).

[PV87] Hans JM Peters and Okko Jan Vrieze. "Surveys in game theory and related topics". In: *CWI Tracts* (1987).

[Sha53] Lloyd S Shapley. "Stochastic games". In: *Proceedings of the national academy of sciences* 39.10 (1953), pp. 1095–1100.

[Sho99] Peter W Shor. "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer". In: *SIAM review* 41.2 (1999), pp. 303–332.

[SW92] Micha Sharir and Emo Welzl. "A combinatorial bound for linear programming and related problems". In: *Annual Symposium on Theoretical Aspects of Computer Science*. Springer. 1992, pp. 567–579.

[Zie98] Wieslaw Zielonka. "Infinite games on finitely coloured graphs with applications to automata on infinite trees". In: *Theoretical Computer Science* 200.1-2 (1998), pp. 135–183.

[ZP96] Uri Zwick and Mike Paterson. "The complexity of mean payoff games on graphs". In: *Theoretical Computer Science* 158.1-2 (1996), pp. 343–359.