Succinct Counting and Progress Measures for Solving Infinite Games

Katrin Martine Dannert

September 2017

Masterarbeit im Fach Mathematik
vorgelegt der
Fakultät für Mathematik, Informatik und Naturwissenschaften
der Rheinisch-Westfälischen Technischen Hochschule Aachen

1. Gutachter: Prof. Dr. Erich Grädel
2. Gutachter: Prof. Dr. Martin Grohe

Eigenständigkeitserklärung

Hiermit versichere ich, die Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt zu haben, alle Stellen, die wörtlich oder sinngemäß aus anderen Quellen übernommen wurden, als solche kenntlich gemacht zu haben und, dass die Arbeit in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegt wurde.

Aachen, den

Katrin Dannert

Contents

Introduction

In this Master's thesis we will take a look at infinite games and at methods for deciding their winner. The first games we will consider are parity games. Parity games are games of the form $\mathcal{G} = (V, V_0, V_1, E, \mathrm{val})$, where $\mathrm{val}\colon V \to \mathbb{N}$ is a function assigning a value to each node. The winner of a play in a parity game is the one of whose parity the largest value occurring infinitely often is. Deciding parity games has long been known to be in NP $\cap$ Co-NP but it is not known, whether they are decidable in polynomial time. The best previously known bound was $n^{m/3+\mathcal{O}(1)}$ [Sch07], where n denotes the number of nodes and m the largest occurring value. In 2016, Calude, Jain, Khoussainov, Li and Stephan published a preprint of their paper *Deciding Parity Games in Quasipolynomial Time* [Cal17], the final version of which was published in 2017. In this paper, Calude et al. present a method which, as the title suggests, solves parity games in quasipolynomial time and space, i.e. in $2^{\mathcal{O}(\log(n)^c)}$ for some constant c. It uses a very succinct way of tracking winning statistics for one player, which allows for an alternating polylogarithmic space algorithm, which can be translated into a quasipolynomial time algorithm as well as an FPT-algorithm using a reachability game.
That paper was quickly followed by a paper called *Succinct progress measures for solving parity games* [JL17] by authors Jurdziński and Lazić. They employ a different method, using progress measures and a lifting algorithm to solve parity games in quasipolynomial time using only quasilinear space. In presenting a very succinct way to code ordered trees, they are able to define their highly succinct progress measures. By now, Fearnley et al. [Fea17] have shown that the method by Calude et al. can be adapted to use only quasilinear space as well.
In this thesis we will present both methods as well as the adaptation by Fearnley et al. and use these methods for two further purposes. First, we will give two algorithms, based on the algorithm developed by Calude et al., that solve Streett-Rabin games in FPT, a time bound which follows from the fact that their method can solve parity games in FPT. Then we consider model checking games of the modal μ-calculus, which are parity games, and see what results we can derive by applying the methods by Calude et al. and by Jurdziński and Lazić.

In the first chapter of this thesis we will summarise the main basic results about the games and the logic considered, for instance the positional determinacy of parity games, the *latest appearence record* for Muller games and Streett-Rabin games and the fact that the model checking games for the modal μ-calculus are parity games.
In the second chapter, we will present in detail the results both of Calude et al. and of Jurdziński and Lazić and we will use the adaptation by Fearnley et al. to compare the two.
In the third chapter we will develop algorithms that decide Streett-Rabin games in FPT, both for winning conditions given as Muller conditions, where the parameter is the number of colours, and for pair conditions, where the parameter is the number of pairs.
In the last chapter, we will take a look at how to apply the methods for solving parity games to model checking games for the modal μ-calculus.

Chapter 1

Basics and Conventions

In the first two subsections of this section we will document the notations and conventions used in this thesis and define a few of the basic notions that appear throughout it. The following four sections cover three different but related types of games, namely parity games, Muller games and Streett-Rabin games, and the modal μ-calculus, a logic whose model checking games are parity games. As mentioned, the aim of this thesis is to present in detail new methods for solving parity games and to try and apply them to Streett-Rabin games and to the modal μ-calculus. In order to do this, we need a basic understanding of the games and the logic in question, so these sections will cover the definitions and main results about them, most of which will be used in later chapters.

1.1 Conventions and Notation

In this thesis we write $\mathbb{N}$ for the natural numbers $\{0, 1, 2, 3, \dots\}$ and $\mathbb{N}_{>0}$ for the set $\{1, 2, 3, \dots\}$. When we write $\log(n)$ for some number n, we mean the logarithm with base 2. We use standard Notation for sets and functions. Let V, W be sets. We write $V \backslash W$ for the set $\{v \in V : v \notin W\}$ and $\mathcal{P}(V)$ for the power set of a set V.
When we speak of graphs, we mean finite *directed graphs*, i.e. graphs of the form $G = (V, E)$, where V with $|V| < \infty$ is the set of nodes and $E \subseteq V \times V$ is the set of edges. We allow edges of the form (v, v) and we allow $(v, w) \in E$ and $(w, v) \in E$ for two nodes $v, w \in V$, but we do not allow multiple identical edges (v, w) from a node v to a node w. We will usually use the letter n for the number of nodes in a graph.
For a given function $f \colon V \to W,\ v \mapsto f(v)$ and a subset $V' \subseteq V$ we write $f|_{V'}$ for the restriction $f|_{V'} \colon V' \to W,\ v \mapsto f(v)$ of f to V'. We write $f(V)$ for the image of V under f.
When we speak of a partial function $p \colon V \to W$ we mean a set $p \subseteq V \times W$ such that for each $v \in V$ there is at most one $w \in W$ such that $(v, w) \in p$. When we do not specify that we speak of a partial function, we mean a total function, i.e. a function in the usual sense.

1.2 Basics on Games and Complexity Theory

In this section we will give a few basic terms about games and complexity theory. The reader will probably be familiar with most of them.

1.2.1 Games

In this thesis we will consider several different games. A game in our sense is usually played by two players, 0 and 1, on a *game graph* $G = (V, V_0, V_1, E)$, where V is the set of nodes, $E \subseteq V \times V$ the set of edges and $V = V_0 \cup V_1$ is a partition, i.e. $V_0 \cap V_1 = \emptyset$, of V into the set V_0 of positions where Player 0 moves and the set V_1 of positions, where it is Player 1's move. Sometimes, the game graph is not given explicitly but the game is for instance played on a formula', in which case the game graph is given implicitly by giving a set of positions and giving the legal moves for Player 1 and Player 2. This specifies the nodes, the partition into V_0 and V_1 and the edges, which represent the legal moves.
A *play* π in a game $\mathcal{G}$ with game graph $G = (V, V_0, V_1, E)$ is a sequence $\pi = v_1 v_2 v_3 \cdots$ of nodes in V or rather of positions in the game, such that $(v_i, v_{i+1}) \in E$ for all i, i.e. the move from v_i to v_{i+1} is a legal move for all i. To complete the definition of the game, *winning conditions* for both players have to be specified. The winning conditions determine for each play π, which player wins it.
One example of a game is a *reachability game*. It is played by two players, 0 and 1, on a game graph $G = (V, V_0, V_1, E)$. The winning conditions are the following: A set $W \subseteq V$ is specified and Player 0 wins a play $\pi = v_1 v_2 v_3 \cdots$, if at some point v_i in π we have $v_i \in W$. Player 1 wins otherwise. This winning condition for Player 1 is called a *safety condition*, since it is to stay in the safe' region $V \backslash W$.
A *strategy* for Player i is a partial function $f\colon \{v_l v_{l+1} \cdots v_{l+k} \in V^k \colon l, k \in \mathbb{N},\ v_k \in V_i\} \to V$ such that $(v_{l+k}, f(v_l \cdots v_{l+k})) \in E$ for all $(v_l \cdots v_{l+k}, f(v_l \cdots v_{l+k})) \in f$, i.e. each move from v_{l+k} to $f(v_l \cdots v_{l+k})$, if defined, has to be a legal move. In most circumstances, but not all, we will be considering *positional strategies*. A positional strategy for Player i is a strategy that only depends on the current node, i.e. a strategy of the form $f\colon V_i \to V$. In a reachability game for instance, all strategies can without loss of generality be assumed to be positional.
A play $\pi = v_1 v_2 v_3 \ldots$ is *consistent* with or played *according to* a strategy f for Player i, $i \in \{0, 1\}$, if for each subsequence $v_l v_{l+1} \cdots v_{l+k}$ of π with $(v_l v_{l+1} \cdots v_{l+k}, f(v_l v_{l+1} \cdots v_{l+k})) \in f$ we have $v_{l+k+1} = f(v_l v_{l+1} \cdots v_{l+k})$.
A strategy f for Player i, $i \in \{0, 1\}$, is a *winning strategy* or *winning* from node $v \in V$, if every play in the game with first node v that is consistent with f is won by Player i. If Player i has a winning strategy from node $v \in V$, we say that she *wins* the game from starting node v or that she is the *winner* of the game with starting node v. The set W_i of nodes $v \in V$ such that Player i wins the game from node v is called the *winning region* of Player i. A game is *determined*, if $W_0 \cup W_1 = V$ and $W_0 \cap W_1 = \emptyset$.

1.2.2 Complexity

Here we will briefly summarize some complexity classes which are relevant in this thesis. It is assumed, that the reader is familiar with the classes P and NP.
The most important class considered in this thesis is *quasipolynomial time*. A problem can be decided in quasipolynomial time, if it can be decided in time $2^{\mathcal{O}(\log(n)^c)}$ for some constant c, where n is the size of the instance of the problem.
A problem can be solved in *quasilinear time*, if it can be solved in time $\mathcal{O}(n \log^k(n))$ for some $k > 0$.
Another class, briefly mentioned in a later chapter, is the class UP of problems solvable in unambiguous non-deterministic polynomial time, i.e. solvable in polynomial time by an unambiguous Turing machine. For more details, see [HR97].
The last important complexity class discussed in this thesis is that of *fixed parameter tractable* problems (FPT). A problem is fixed parameter tractable with parameter k, if it can be solved in time $f(k) \cdot n^{\mathcal{O}(1)}$, where f is a computable function and n the size of the instance of the problem.

1.3 Parity Games

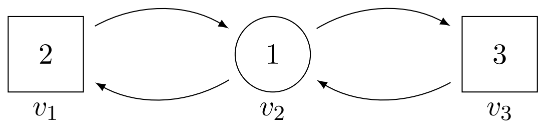
Parity games form a very important class of games that has been intensely studied and is still the focus of many studies today. One reason is that they are, as mentioned, the model checking games for the modal μ-calculus, which is an immensely useful logic. But parity games are also interesting in themselves, mainly because of their complexity or rather our uncertainty about said complexity. It is known that parity games are in NP $\cap$ Co-NP and they can be shown to by solvable in quasipolynomial time, a result which is the main focus of this thesis. But it is not known, whether parity games can be decided in polynomial time and so this is one of very few common problems that are known to be both in NP and in Co-NP but not known to be in P. But before we tackle the complexity of parity games, let us start with the definition.

1.3.1 Definition: A *parity game* is an infinite game $\mathcal{G} = (V, V_0, V_1, E, \mathrm{val}\colon V \to \mathbb{N}_{>0})$ on a graph (V, E) with $|V| = n < \infty$, $V_0 \cap V_1 = \emptyset$, $V_0 \cup V_1 = V$, and $E \subseteq V \times V$. Additionally we have $|\,\mathrm{val}(V)| = m < \infty$. For $v \in V$ we call $\mathrm{val}(v)$ the *value* or *priority* of v.
There are two players, namely 0 and 1. From a node $v \in V_i$, $i \in \{0, 1\}$, Player i moves to any node $w \in V$ of her choice, such that $(v, w) \in E$.
The winner is determined in the following way: If a player cannot move, she loses and her opponent wins. An infinite play π is won by Player 0 if the largest value $a \in \mathrm{val}(V)$ occurring infinitely often in π is even. Otherwise the play is won by Player 1.

1.3.2 Remark: Since the number of nodes and the number of different values will play an important role in large parts of this paper, we will reserve the letters n and m, respectively, for them. So from now on, in any given parity game, n refers to the number of nodes and m to the number of different values (or the largest occurring value, as we will see later), unless it is said otherwise.

The following simple example will make the definition of a parity game clearer.

1.3.3 Example: Consider the following game graph with $V_0 = \{v_2\}$, $V_1 = \{v_1, v_3\}$, $\mathrm{val}(v_1) = 2$, $\mathrm{val}(v_2) = 1$ and $\mathrm{val}(v_3) = 3$:



Player 0 can win from any starting node, by always moving to v_1, when the game arrives at v_2. A winning play with v_3 as the starting node played according to that strategy would look as follows:

$$\pi = v_3\ v_2\ v_1\ v_2\ v_1\ v_2\ v_1\ v_2\ v_1\ v_2\ v_1\ v_2\ v_1\ v_2\ v_1\ v_2\ v_1\ \ldots$$

The highest value appearing infinitely often in π is $\mathrm{val}(v_1) = 2$, so Player 0 wins π.

Note that in a parity game different nodes do not have to have different values as in the example. We can make a few assumptions on the game graphs that do not alter the parity games on them in any relevant way. For one, we can assume that all plays are infinite, which is a result of the next lemma. Also, using the lemma after that, we can assume that if the number of different values is m then the set of values is $\{1, \ldots, m\}$.

1.3.4 Lemma: Let $\mathcal{G} = (V, V_0, V_1, E, \mathrm{val}\colon V \to \mathbb{N}_{>0})$ be a parity game. If there is a node in $v \in V$ such that $\{w \in V \colon (v, w) \in E\} = \emptyset$ we can add a new node z with $\mathrm{val}(z) := \mathrm{val}(v) + 1$ to V (without loss of generality to V_0) and edges $(v, z), (z, v)$ to E. Then for all $w \in V$, Player 0 wins $\mathcal{G}' = (V \cup \{z\}, V_0 \cup \{z\}, V_1, E \cup \{(v, z), (z, v)\}, \mathrm{val}\colon V \cup \{z\} \to \mathbb{N}_{>0})$ from starting node w if and only if she wins $\mathcal{G}$ from starting node w. Additionally, if $v \in V_i$, $i \in \{0, 1\}$, then Player $1 - i$ wins $\mathcal{G}'$ from starting node z.

Proof. Let $v \in V_i$, $i \in \{0, 1\}$. Then Player $1 - i$ wins any play in $\mathcal{G}$ that reaches v. The plays in $\mathcal{G}$ that do not reach v are plays in $\mathcal{G}'$ that do not reach v, since the induced subgraphs of $V \backslash \{v\}$ in $\mathcal{G}$ and $V' \backslash \{v, z\}$ are identical and we can map a play π to the identical play π' in $\mathcal{G}'$.
Assume without loss of generality that $v \in V_0$. Otherwise just flip the roles of Player 0 and Player 1. Suppose that Player 0 has a winning strategy f from node $w \in V$. Then any play π in $\mathcal{G}$ with starting node w played according to f will not reach v and therefore $f' \colon V_0' \to V'$, where $f'(x) = f(x)$ for all $x \in V \backslash \{v\}$ and $f'(z) = v$ (if $z \in V_0$), is a winning strategy for Player 0 in $\mathcal{G}'$ from starting node w.
Suppose now that Player 1 has a winning strategy f from node w in $\mathcal{G}$. Define $f' \colon V_1' \to V'$ analogous to the definition above and let π' be a play in $\mathcal{G}'$ played according to f'. If π' does not reach v then there is a corresponding play π in $\mathcal{G}$ consistent with f and therefore Player 1 wins π'. If π' reaches $v \in V'$, then at this point the play enters an infinite loop containing only the nodes v and z, since these are the only remaining legal moves. Since $\mathrm{val}(z) = \mathrm{val}(v) + 1$ is odd, Player 1 wins π'. (This also proves the additional claim.) It follows that f' is a winning strategy for Player 1 from w.
If Player i, $i \in \{0, 1\}$, has a winning strategy f' in $\mathcal{G}'$ from node $w \in V' \backslash \{z\}$, then define $f \colon V_i \to V,\ x \mapsto f'(x)$. Let π be a play in $\mathcal{G}$ consistent with f. If π does not contain v, then Player i wins π since it is identical to a play π' in $\mathcal{G}'$ consistent with f'. If π contains v, Player 1 wins π and π ends at the first occurrence of v. Let π' be the play in $\mathcal{G}'$ with $\pi' = \pi wvwvw \cdots$. Then π' is consistent with f' and won by Player 1 (in particular, this case arises only if $i = 1$). $\square$

Note that one could alternatively add an edge (v, v) to each node v without successors and change the value to 1, if $v \in V_1$ and to 2, if $v \in V_0$.

1.3.5 Remark: Using Lemma 1.3.4, we will assume from now on, without loss of generality, that all plays in our parity games are infinite.

1.3.6 Lemma: Let $\mathcal{G} = (V, V_0, V_1, E, \text{val}\colon V \to \mathbb{N}_{>0})$ be a parity game with $|\text{val}(V)| = m$. Then there is a parity game $\mathcal{G}' = (V, V_0, V_1, E, \text{val}'\colon V \to \{1, \ldots, m\})$ such that for all plays π in $\mathcal{G}$ Player 0 wins π if and only if she wins the identical play π' in $\mathcal{G}'$ or for all plays π in $\mathcal{G}$ Player 0 wins π if and only if Player 1 wins the identical play π' in $\mathcal{G}'$.

Proof. We define $\mathcal{G}'$ in the following way. Let $m_1 + 1 < m_2$ be values of nodes in $\mathcal{G}$ such that there is no node in $\mathcal{G}$ with a value m_3 and $m_1 < m_3 < m_2$ (skip this step if no such values exist). If m_1 and m_2 both have the same parity then we label all nodes in $\mathcal{G}'$ whose corresponding nodes in $\mathcal{G}$ have value m_1 or m_2 with the value m_1. If m_1 and m_2 are not of the same parity then $m_1 + 2k < m_2 < m_1 + 2(k+1)$ for some $k > 0$. In this case label all nodes $\mathcal{G}'$ whose corresponding nodes in $\mathcal{G}$ have value m_2 with the value $m_1 + 1$.
Repeat this process as long as there exist pairs of values with the above properties. Since there are only finitely many values occurring in $\mathcal{G}$ and with each application of the above paragraph all nodes of a certain value get a smaller value, this process will terminate. When that happens, all occurring values will be consecutive. Let m' be the smallest occurring value. If $m' = 2k + 1$ is odd, reduce the value of each node by $2k$ to obtain the game $\mathcal{G}'$. If $m' = 2k$ is even, reduce the value of each node by $2k - 1$. We claim that in the first of these two cases, for all plays π in $\mathcal{G}$ Player 0 wins π if and only if she wins the identical play π' in $\mathcal{G}'$, and in the second, for all plays π in $\mathcal{G}$ Player 0 wins π if and only if Player 1 wins the identical play π' in $\mathcal{G}'$.
Consider the first case and let π be a play in $\mathcal{G}'$ won by Player 0. Let π' be the play along the corresponding nodes in $\mathcal{G}'$. Let a be the largest value in π that occurs infinitely often and let v be a node with $\text{val}(v) = a$. By construction of $\mathcal{G}'$, the value a' of v in $\mathcal{G}'$ is $a - 2k$ for some $k \in \mathbb{N}$, since the value is reduced by an even number in every step. Thus a' is even. Additionally, let w be a node in $\mathcal{G}$ with $\text{val}(w) < a$. Then a' is larger than or equal to the value of w in $\mathcal{G}'$, since no value is ever reduced below the next occurring value in that step which is smaller. Hence a' is the largest value occurring infinitely often in π' and Player 0 wins π'.
Now let π' be a play in $\mathcal{G}'$ won by Player 0 and a' the largest value occurring infinitely often in π'. Let v be a node with value a' in $\mathcal{G}'$ and let a be the value of v in $\mathcal{G}$. Since as seen above the difference between a and a' is even, a is even. Additionally, let w be a node in $\mathcal{G}$ with $\text{val}(w) > a$. Then a' is smaller than or equal to the value of w in $\mathcal{G}'$, since no value is ever reduced below the next occurring value which is smaller. Hence no nodes with a value larger than a in $\mathcal{G}$ occur infinitely often in π' and π and therefore Player 0 wins π.
Now consider the second case and let π be a play in $\mathcal{G}'$ won by Player 0. Let π' be the play along the corresponding nodes in $\mathcal{G}'$. Let a be the largest value in π that occurs infinitely often and let v be a node with $\text{val}(v) = a$. By construction of $\mathcal{G}'$, the value a' of v in $\mathcal{G}'$ is $a - (2k - 1)$ for some $k \in \mathbb{N}$, since the value is reduced by an even number in every step from the first paragraph of this proof and by an odd number in the last step. Thus a' is odd. Additionally, let w be a node in $\mathcal{G}$ with $\text{val}(w) < a$. Then a' is larger than or equal to the value of w in $\mathcal{G}'$, since no

value is ever reduced below the next occurring value which is smaller. Hence a' is the largest value occurring infinitely often in π' and Player 1 wins π'.
Now let π' be a play in $\mathcal{G}'$ won by Player 1 and a' the largest value occurring infinitely often in π'. Let v be a node with value a' in $\mathcal{G}'$ and let a be the value of v in $\mathcal{G}$. Since as seen above the difference between a and a' is odd, a is even. Additionally, let w be a node in $\mathcal{G}$ with $\mathrm{val}(w) > a$. Then a' is smaller than or equal to the value of w in $\mathcal{G}'$, since no value is ever reduced below the next occurring value which is smaller. Hence no nodes with a value larger than a in $\mathcal{G}$ occur infinitely often in π' and π and therefore Player 0 wins π. $\square$

1.3.7 Remark: Making use of Lemma 1.3.6 we will from now on assume, that for every parity game $\mathcal{G} = (V, V_0, V_1, E, \mathrm{val}\colon V \to \mathbb{N}_{>0})$ with $|\,\mathrm{val}(V)| = m$ we have $\mathrm{val}(V) = \{1, \ldots, m\}$.

It is not completely obvious that from any given node one of the two players has a winning strategy. But this is actually the case and follows from a more general result by Martin (see [Mar75]) which states that Borel games are determined.

1.3.8 Proposition: Parity games are determined.

1.3.1 Positional determinacy

Not only are parity games determined, but they are positionally determined, meaning that the winner has a winning strategy that only depends on the current node and not on the previous path of the play. This property is extremely important for complexity results about parity games and will be used extensively in proofs in later chapters. It also separates parity games from Muller games and Streett-Rabin games, as we will see in the next two sections.
This subsection, largely based on [Küs02], is aimed at proving the positional determinacy. Formally, it is defined as follows.

1.3.9 Definition: A game $\mathcal{G}$ with players 0 and 1, where W_i, $i \in \{0, 1\}$, denotes the winning region of player i, is called *positionally determined*, if it is determined and for each $w \in W_i$, $i \in \{0, 1\}$, Player i has a positional winning strategy from node w.

As already mentioned, this applies to parity games so the aim of this subsection will be to prove the following theorem.

1.3.10 Theorem: Parity games are positionally determined.

The proof is relatively long and we will therefore split it into several smaller proofs. But first we need two important notions when talking about finding winning strategies for games on graphs. Those notions are attractors and traps. The results and proofs concerning attractors and traps can also be found in [Grä16]

1.3.11 Definition: Let $G = (V, V_0, V_1, E)$ be a graph, $i \in \{0,1\}$ and $X \subseteq V$. The *i-attractor* of X is defined as

$$\mathrm{Attr}_i(X) = \{v \in V \colon \text{ Player } i \text{ has a strategy to reach } X \text{ from } v\}.$$

This abstract definition can be made a little more concrete by the following remark which gives us an actual construction of the attractor for a set X. Note that as we are now considering the graph in the context of a reachability game, all strategies can and will be assumed to be positional strategies.

1.3.12 Remark: For $v \in V$ let $vE = \{w \in V \colon (v,w) \in E\}$ denote the set of successors of v. The i-attractor of a set X can be computed by defining $X_0 = X$ and

$$X_{j+1} = X_j \cup \{v \in V_i \colon vE \cap X_j \neq \emptyset\} \cup \{v \in V_{1-i} \colon vE \subseteq X_j\}$$

for all $j \in \mathbb{N}$. Then $\mathrm{Attr}_i(X) = \bigcup_{j\in\mathbb{N}} X_j$.

Proof. Let $v \in \mathrm{Attr}_i(X)$. Then there is a $k \in \mathbb{N}$ such that Player i has a strategy to reach X in at most k steps. We prove that $v \in \bigcup_{j\in\mathbb{N}} X_j$ by induction over k. If $k = 0$ then $v \in X$ and therefore $v \in X_0 \subseteq \bigcup_{j\in\mathbb{N}} X_j$. Now suppose $k > 0$ and let f be a strategy to reach X from v in at most k steps for Player i. If $v \in V_i$, Player i has a strategy to reach X from $f(v)$ in at most $k-1$ steps. It follows that $f(v) \in \bigcup_{j\in\mathbb{N}} X_j$ by induction hypothesis. Then there is a unique $j \in \mathbb{N}$ such that $f(v) \in X_j \backslash X_{j-1}$ or $f(v) \in X = X_0$, in which case we set $j = 0$. Hence, $vE \cap X_j \neq \emptyset$ and $v \in X_{j+1} \subseteq \bigcup_{j\in\mathbb{N}} X_j$. If on the other hand $v \in V_{1-i}$, Player i has a strategy to reach X from any $w \in vE$ in at most $k-1$ steps. It follows by induction hypothesis that $w \in \bigcup_{j\in\mathbb{N}} X_j$ for all $w \in vE$. Then there exists $j \in \mathbb{N}$ such that $w \in X_j \backslash X_{j-1}$ for all $w \in vE$ or $w \in X_j := X_0$ for all $w \in vE$, since vE is finite. Hence, $vE \subseteq X_j$ and therefore $v \in X_{j+1} \subseteq \bigcup_{j\in\mathbb{N}} X_j$.
Conversely, let $v \in \bigcup_{j\in\mathbb{N}} X_j$. Then by construction there is $k \in \mathbb{N}$ such that $v \in X_k \backslash X_{k-1}$ or $v \in X_k := X_0$. We prove $v \in \mathrm{Attr}_i(X)$ by induction over k. If $k = 0$ then $v \in X_0 = X$ and the strategy to reach X is trivial. Now suppose $k > 0$. If $k \in V_i$, then by definition of X_k there exists $w \in vE \cap X_{k-1}$ and by induction hypothesis Player i has a strategy $f \colon V_i \cap X_{k-1} \to V$ to reach X from w. Define a new strategy $f' \colon V_i \cap X_k \to V$ by setting $f'(x) = f(x)$ for all $x \in X_{k-1}$, $f'(v) = w$ and $f'(x)$ as an arbitrary successor of x for all $x \in V_i \cap (X_k \backslash (X_{k-1} \cup \{v\}))$. Then f' is a strategy for Player i to reach X from v. If on the other hand $v \in V_{1-i}$, then by definition of X_k, $vE \subseteq X_{k-1}$ and therefore Player i has a strategy to reach X from any $w \in vE$ by induction hypothesis. Then obviously such a strategy, arbitrarily extended to any $x \in V_i \cap X_k$, is a strategy

for Player i to reach X from v. □

The strategy to reach X from any node in $\mathrm{Attr}_i(X)$ will be important to prove the positional determinacy of parity games, so we give it a name.

1.3.13 Definition: Player i always has a positional strategy to reach X from $v \in \mathrm{Attr}_i(X)$. We call such a strategy an *attractor strategy.*

A somewhat complementary idea to the notion of an attractor is that of a trap. A trap is a set where one of the players can prevent the other from ever leaving. This of course also happens by a positional strategy, as stated in the next remark.

1.3.14 Definition: Let $G = (V, V_0, V_1, E)$ be a graph and $i \in \{0, 1\}$. An *i-trap* is a set $Y \in V$ such that $vE \subseteq Y$ for all $v \in Y \cap V_i$ and $vE \cap Y \neq \emptyset$ for all $v \in V_{1-i} \cap Y$.

1.3.15 Remark: On an i-trap Y, Player $1 - i$ has a positional strategy to keep any play beginning in Y in Y.

Proof. Define a positional strategy $f \colon Y \cap V_{1-i} \to Y$ by defining $f(v)$ as some $w \in vE \cap Y$ for all $v \in Y \cap V_{1-i}$, which exists by definition of Y. □

We already mentioned that attractors and traps are complementary ideas. In particular, the complement of an attractor is a trap.

1.3.16 Remark: The complement of an i-attractor for some set $X \subseteq V$ is an i-trap.

Proof. Let $A = \mathrm{Attr}_i(X)$ and $v \in V \backslash A$. Then $v \notin X_j$ for any $j \in \mathbb{N}$. If $v \in V_i$ it follows that $vE \cap X_j = \emptyset$ for all $j \in \mathbb{N}$, since otherwise $v \in X_{j+1}$, and therefore $vE \subseteq V \backslash \bigcup_{j \in \mathbb{N}} X_j = V \backslash A$. If $v \in V_{1-i}$, it follows that $vE \nsubseteq X_j$ for all $j \in \mathbb{N}$, since otherwise $v \in X_{j+1}$. Hence there is $w \in vE$ such that $w \notin X_j$ for all $j \in \mathbb{N}$ by construction of the sequence of the X_j. It follows that $w \notin \bigcup_{j \in \mathbb{N}} X_j = A$ and thus $vE \cap V \backslash A \neq \emptyset$. □

Now we go back to parity games. The following definition, again taken from [Küs02], describes a subset of the winning region of Player i, which is also a trap. It is fittingly called a *paradise*, since a set of winning positions where one can remain with a positional strategy might be considered the best one can hope for, if one tries to win a game.

1.3.17 Definition: Let $\mathcal{G} = (V, V_0, V_1, E, \mathrm{val})$ be a parity game and $i \in \{0, 1\}$. A set $U \subseteq V$ is an i-paradise if

- U is a $(1 - i)$-trap and

- Player i has a positional strategy f_i on U to win any play consistent with f_i that stays in U.

To answer the question of how we can use this to prove our positional determinacy theorem, consider the following: If we can prove that there is a partition of the set of all nodes into such a paradise for each of the two players, then obviously those would have to be the winning regions. But by definition, the respective player has a positional winning strategy on her paradise. Thus, there would be a positional winning strategy for either winning region and therefore for any given node (for one of the two players).
So let us prove this statement.

1.3.18 Theorem: Let $\mathcal{G} = (V, V_0, V_1, E, \mathrm{val})$ be a parity game. Then V is partitioned into a 0-paradise and a 1-paradise.

Proof. Let m be the maximal value that occurs in $\mathcal{G}$. First, assume $m = 1$. Then Player 1 wins from any node with any strategy, in particular with a positional one. Thus we have that V is a 1-paradise and $\emptyset$ is a 0-paradise.
Now assume $m > 1$. Without loss of generality let m be even (otherwise switch the roles of Players 0 and 1). Using transfinite induction we will define a sequence of 1-paradises W_1^ξ with corresponding positional winning strategies f_1^ξ on those 1-paradises. For $\nu < \xi$ we will have $W_1^\nu \subseteq W_1^\xi$ and f_1^ξ will be an extension of f_1^ν.
As the first set we simply choose the empty set $W_1^0 = \emptyset$ and for a limit ordinal ν we define $W_1^\nu = \bigcup_{\xi<\nu} W_1^\xi$.
Claim: W_1^ν as defined above is a 1-paradise if W_1^ξ are 1-paradises for all $\xi < \nu$.
Obviously if W_1^ξ for all $\xi < \nu$ are 0-traps, then so is W_1^ν because if for all $\xi < \nu$ we have $vE \subseteq W_1^\xi$ for all $v \in W_1^\xi \cap V_i$ and $vE \cap W_1^\xi \neq \emptyset$ for all $v \in V_{1-i} \cap W_1^\xi$, then $vE \subseteq \bigcup_{\xi<\nu} W_1^\xi$ for all $v \in \bigcup_{\xi<\nu} W_1^\xi \cap V_i$ and $vE \cap \bigcup_{\xi<\nu} W_1^\xi \neq \emptyset$ for all $v \in V_{1-i} \cap \bigcup_{\xi<\nu} W_1^\xi$. Let f_ξ be a positional winning strategy for Player 1 on W_1^ξ that keeps the game in W_1^ξ for all $\xi < \nu$. Define a positional strategy f_ν on W_1^ν by $f_\nu(v) = f_\chi(v)$, where χ is the smallest ordinal such that $v \in W_1^\chi$, for all $v \in W_1^\nu \cap V_1$. Let $\pi = v_1 v_2 v_3 \cdots$ be a play consistent with f_ν and $v_1 \in W_1^\nu$. We need to show that $v_k \in W_1^\nu$ for all $k \in \mathbb{N}_{>0}$ and that Player 1 wins π. We show the first claim by a simple induction over k. The case $k = 1$ follows from the assumption that $v_1 \in W_1^\nu$. So let $k > 1$. Then there exists a minimal $\chi < \nu$ such that $v_{k-1} \in W_1^\chi$, since by induction hypothesis $v_{k-1} \in W_1^\nu$. If $v_{k-1} \in V_0$ then $v_k \in vE \subseteq W_1^\chi \subseteq W_1^\nu$, since W_1^χ is a 0-trap. If $v_{k-1} \in V_1$, then $v_k = f_\nu(v_{k-1}) = f_\chi(v_{k-1}) \in W_1^\chi \subseteq W_1^\nu$. This proves the first claim. To see that π is won by Player 1 note that π will at some point be played exclusively consistent with f_ξ for some $\xi < \nu$, since for any $v_k \in W_1^\chi$, $v_{k+1} \in W_1^\eta$ for $\eta \leq \chi$, and therefore Player 1 wins π.
By induction hypothesis, f_1^ξ is an extension of f_1^χ if $\chi < \xi$. Therefore we can define for a limit ordinal ν the strategy f_1^ν as the union of the strategies f_1^ξ for $\xi < \nu$. By the exact same argument as in the proof of the claim it follows that f_1^ν is a positional winning strategy on W_1^ν that keeps

the play in W_1^ν.
Now let $\xi+1$ be a successor ordinal. Define $X^\xi = \mathrm{Attr}_1(W_1^\xi)$. Then obviously X^ξ is a 1-paradise, since $\mathrm{Attr}_1(W_1^\xi)$ is a 0-trap by definition and the positional attractor strategy on $\mathrm{Attr}_1(W_1^\xi)\backslash W_1^\xi$ combined with f_1^ξ on W_1^ξ yields the desired positional winning strategy that stays in X^ξ. This strategy that we will call g_1^ξ extends f_1^ξ. By Remark 1.3.16, $Y = V\backslash X^\xi$ is a 1-trap.
Now define $M = \{v \in Y\colon\ \mathrm{val}(v) = m\}$ and $Z_1^\xi = Y\backslash \tilde{\mathrm{Attr}}_0(M)$, where $\tilde{\mathrm{Attr}}_0(M)$ denotes the 0-attractor of M in the subgame of $\mathcal{G}$ induced by Y. Since Z_1^ξ is the complement of a 0-attractor, it is a 0-trap in this subgame. Additionally, Z_1^ξ does not contain any nodes v with $\mathrm{val}(v) = m$ and therefore, by induction hypothesis, Z_1^ξ can be partitioned into a 0-paradise $Z_{1,0}^\xi$ and a 1-paradise $Z_{1,1}^\xi$ with positional winning strategies z_1^ξ and z_2^ξ, respectively, that stay in the respective sets. Since $Z_{1,1}^\xi$ is a 0-trap in the game restricted to Z_1^ξ and Z_1^ξ is a 0-trap in the game restricted to Y, $W_1^{\xi+1} := X^\xi \cup Z_{1,1}^\xi$ is a 0-trap in $\mathcal{G}$.
Let z_1^ξ denote a positional winning strategy for Player 1 that stays in $Z_{1,1}^\xi$ in the subgame induced by Y. Then define $f_1^{\xi+1}\colon W_1^{\xi+1} \cap V_1 \to W_1^{\xi+1}$ by $f_1^{\xi+1}(v) = g_1^\xi(v)$ for all $v \in X^\xi$ and $f_1^{\xi+1}(v) = z_1^\xi(v)$ for all $v \in Z_{1,1}^\xi$. It follows that any play π beginning in $W_1^{\xi+1}$ consistent with $f_1^{\xi+1}$ will stay in $W_1^{\xi+1}$ and is winning for Player 1: Either the play stays forever in $Z_{1,1}^\xi \subseteq W_1^{\xi+1}$ and π is from some point onwards consistent with z_1^ξ and therefore winning for Player 1, the play starts in X^ξ, in which case it will stay there forever by definition of g_1^ξ and is winning for Player 1, or the play starts in $Z_{1,1}^\xi$ and then moves out of it. But since $Z_{1,1}^\xi$ is a trap in the game induced by $Y = V\backslash X^\xi$ and z_1^ξ is the according strategy, the game then moves to X^ξ and stays there forever, again by definition of g_1^ξ, and is therefore winning for Player 1. It follows that $W_1^{\xi+1}$ is a 1-paradise with strategy $f_1^{\xi+1}$.
Let σ be the smallest ordinal such that $W_1^\sigma = W_1^{\sigma+1}$. Define $W_1 = W_1^\sigma$. We have just shown that W_1 is a 1-paradise. What remains to be shown is that $W_0 := V\backslash W_1$ is a 0-paradise. To show this, note first that since

$$W_1 \subseteq X^\sigma = \mathrm{Attr}_1(W_1^\sigma) \subseteq W_1^{\sigma+1} = W_1,$$

$W_1 = \mathrm{Attr}_1(W_1^\sigma)$ is an i-attractor and therefore W_0 is an i-trap. Additionally, we can employ the exact same reasoning as above by replacing X^ξ with W_1 and Y with W_0 to see that $W_1 \cup Z_{1,1}^\sigma = W_1^{\sigma+1} = W_1$ is a 1-paradise. Since $W_1 \cap Z_{1,1}^\sigma = \emptyset$, $Z_{1,1}^\sigma = \emptyset$. From this we can show that W_0 is a 0-paradise in the following way.
Let z_0^σ denote a positional winning strategy for Player 0 that stays in $Z_{1,0}^\xi$ in the subgame induced by Z_1^σ and let g_0^σ be an attractor strategy for Player 0 on $\tilde{\mathrm{Attr}}_0(M)$ in the subgame induced by W_0. Define a strategy $f_0\colon W_0 \cap V_0 \to W_0$ on W_0 by

$$f_0(v) = \begin{cases} z_0^\sigma(v), & v \in Z_1^\sigma \\ g_0^\sigma(v), & v \in \tilde{\mathrm{Attr}}_0(M)\backslash M \\ v', & v \in M \text{ and } v' \in vE \cap W_0 \end{cases}$$

for all $v \in W_0 \cap V_0$. By definition of Z_1^σ as $W_0 \setminus \tilde{\text{Attr}}_0(M)$, this definition covers all possible cases and for the last case, such a node v' always exists, since W_0 is a 1-trap.
Let π be a play in $\mathcal{G}$ consistent with f_0. If from some point on, π stays forever in $Z_{1,0}^\sigma$, Player 0 wins since $Z_{1,0}^\sigma$ is a 0-paradise in $Z_{1,0}^\sigma$. If the play moves infinitely often out of $Z_{1,0}^\sigma$, then there are infinitely many nodes in $\tilde{\text{Attr}}_0(M)$ visited since W_0 is a 1-trap and therefore infinitely many nodes in M. It follows that the largest value occurring infinitely often in π is m, which is even, and therefore Player 0 wins π. □

As already explained, this implies the positional determinacy of parity games.

1.3.2 Complexity

As mentioned in the beginning of this section, the complexity of solving parity games is a very interesting question. We will see later on that parity games are decidable in quasipolynomial time. Whether they are solvable in polynomial time is still unknown.
In this subsection we prove that parity games are in NP ∩ Co-NP, following [Grä16], and mention the complexity bounds that were known before parity games were shown to be decidable in quasipolynomial time. Let us begin with the former.

1.3.19 Theorem: The problem of deciding whether Player 0 wins a given parity game $\mathcal{G}$ from a starting node v is in NP ∩ Co-NP.

Proof. We begin by proving that deciding whether Player 0 wins $\mathcal{G} = (V, V_0, V_1, E, \text{val})$ from v is in NP. In order to do so, we guess a positional strategy f for Player 0 from v and check, whether f is a winning strategy for Player 0. All that needs to be shown is that we can do so in polynomial time.
Let $G' = (V, V_0, V_1, E')$ be the subgraph of $G = (V, V_0, V_1, E)$ obtained by deleting any edge (v, w) from E, where $v \in V_0$ and $w \neq f(v)$. Now we need to check if there is a node v' reachable from v for Player 1 such that $\text{val}(v')$ is odd and v' lies on a cycle where $\text{val}(v')$ is the largest value. This can be done in polynomial time.
In order to show that the problem also lies in Co-NP it suffices to realise that showing that Player 0 does not have a winning strategy for $\mathcal{G}$ from v can be done by showing that Player 1 has such a strategy. By the same argument as above, this problem is in NP and therefore, the original problem is in Co-NP. □

It was proven by Jurdziński [Jur98] that parity games are in UP ∩ Co-UP. UP denotes unambiguous non-deterministic polynomial time. It contains P and is itself contained in NP, making the bound possibly better than NP ∩ Co-NP.
Also, it was previously known that parity games can be solved in time $n^{m/3+\mathcal{O}(1)}$ [Sch07], where

n denotes the number of nodes and m the largest occurring value. However, since m can be as large as n, this can still be exponential time.
In this thesis we will present two different approaches that improve this runtime to quasipolynomial time, i.e. to time $2^{\mathcal{O}(\log(n)^c)}$ for some constant c. In their paper *Deciding Parity Games in Quasipolynomial Time* [Cal17], Calude, Jain, Khoussainov, Li and Stephan showed that parity games can be solved in time $\mathcal{O}(n^{\log(m)+6})$. Using a different method, Jurdziński and Lazić presented an algorithm running in time $\mathcal{O}(kn^{\log(m)-\log(\log(n))+4,03})$, where k is the number of edges in the graph, in their paper *Succinct progress measures for solving parity games* [JL17]. We will discuss both methods in detail in the next chapter.

1.4 Muller Games

In this section we will get to know Muller games as preparation for the next section, were we will consider particular Muller games, namely Streett-Rabin games. Here we are only interested in coloured Muller games, meaning that all games we consider will have a value function, just as the parity games. Another similarity to parity games is that in Muller games, too, we consider the set of values occurring infinitely often, only now we look at the whole set, not just the largest value in it.
The definitions and results in this section are based on [Grä16].

1.4.1 Definition: A *Muller game* $\mathcal{G} = (V, V_0, V_1, E, \mathrm{val}\colon V \to \mathbb{N}_{>0})$ is defined as follows: V is a finite set of $n < \infty$ nodes and $V_0 \cup V_1 = V$, $V_0 \cap V_1 = \emptyset$. $E \subseteq V \times V$ is the set of edges of the game graph and val is the value function with $|\mathrm{val}(V)| = m < \infty$.
The winning condition is given by the sets $\mathcal{F}_0, \mathcal{F}_1 \subseteq \mathcal{P}(\mathrm{val}(V))$, where $\mathcal{F}_0 \cup \mathcal{F}_1 = \mathcal{P}(\mathrm{val}(V))$ and $\mathcal{F}_0 \cap \mathcal{F}_1 = \emptyset$.
The moves are the same as in parity games, meaning that from a node $v \in V_i$, $i \in \{0,1\}$, Player i moves to a node $w \in V$ with $(v,w) \in E$.
Player $i \in \{0,1\}$ wins a play $\pi = v_1\ v_2\ v_3\ \dots$ in $\mathcal{G}$ if the set

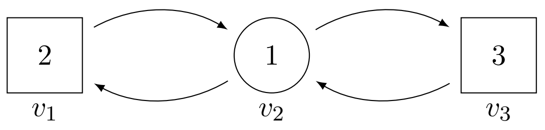
$$\inf(\pi) = \{j \in \mathbb{N}_{>0} \colon \text{there are infinitely many } v_k \text{ in } \pi \text{ with } \mathrm{val}(v_k) = j\}$$

is in $\mathcal{F}_i$.

1.4.2 Remark: Without loss of generality we can assume in the context above, that $\mathrm{val}(V) = \{1, \dots, m\}$, since we can simply replace each different value by a distinct number between 1 and m, both in the definition of val and in the definitions of $\mathcal{F}_0$ and $\mathcal{F}_1$.

Let us consider the example from the previous chapter. We had three nodes with values 1, 2, and 3 and Player 0 was the winner if the largest value occurring infinitely often was 2. We can reformulate this parity game into a Muller game as follows.

1.4.3 Example: Consider the game graph below with $V_0 = \{v_2\}$, $V_1 = \{v_1, v_3\}$, $\mathrm{val}(v_1) = 2$, $\mathrm{val}(v_2) = 1$ and $\mathrm{val}(v_3) = 3$.



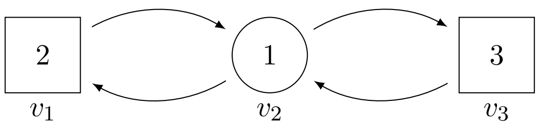
We define the winning conditions as follows:

$$\mathcal{F}_0 = \{\emptyset, \{2\}, \{1,2\}\}, \quad \mathcal{F}_1 = \{\{1\}, \{3\}, \{1,3\}, \{2,3\}, \{1,2,3\}\}.$$

Note that $\mathcal{F}_0$ contains precisely the subsets of $\{1, 2, 3\}$ where the largest number is even (and the empty set). Therefore Player 0 can win the game by always moving from v_2 to v_1, forcing for any play π played according to that strategy that $\inf(\pi) = \{1, 2\} \in \mathcal{F}_0$.

In fact any parity game can be stated as a Muller game by including in $\mathcal{F}_0$ precisely those subsets of the set of values $\{1, \ldots, m\}$ where the largest value is even.
The reverse however is not true. Not every Muller game can be stated as a parity game. Consider again the graph from the previous example, but this time with different winning conditions.

1.4.4 Example: Again $V_0 = \{v_2\}$, $V_1 = \{v_1, v_3\}$, $\mathrm{val}(v_1) = 2$, $\mathrm{val}(v_2) = 1$ and $\mathrm{val}(v_3) = 3$.



We now define the winning conditions as

$$\mathcal{F}_0 = \{\emptyset, \{1\}, \{2\}, \{1, 2, 3\}\}, \quad \mathcal{F}_1 = \{\{3\}, \{1, 3\}, \{2, 3\}, \{1, 2\}\}.$$

Player 0 can win this Muller game, too, from any position. A winning strategy is to alternate between moving to v_1 and to v_3 from v_2. Any play π consistent with this strategy will satisfy $\inf(\pi) = \{1, 2, 3\} \in \mathcal{F}_0$. This however is not a positional winning strategy. In fact there does not exist a positional winning strategy for Player 0: If she always moves from v_2 to v_1, we will have $\inf(\pi) = \{1, 2\} \in \mathcal{F}_1$ for every π played with this strategy. Similarly for each π where she always moved from v_2 to v_3 we have $\inf(\pi) = \{1, 3\} \in \mathcal{F}_1$.

Just as for parity games, it follows from the result by Martin (see [Mar75]) that Muller games are determined.
We have seen in the previous chapter that if a player in a parity game has a winning strategy, she has a positional winning strategy. Therefore the Muller game in the example cannot be a parity game.
If we cannot guarantee that the winning player in a Muller game (from a certain starting node) has a positional winning strategy, we can at least ask ourselves, how far she has to look back in the play in order to decide where to move from the current node. Or, put differently, what space is needed for her winning strategy. We will see that that space is indeed finite and we can give an upper bound for it.
To see this, we will translate Muller games into parity games. Note that this does not mean a direct translation of the winning conditions as we have shown in the example that this is impossible. Instead we will construct a parity game from a given Muller game on a different game graph, using the so called *latest appearance record.*

1.4.5 Definition: Let $\mathcal{G} = (V, V_0, V_1, E, \mathrm{val}\colon V \to \{1, \ldots, m\})$ be a Muller game. We define the *latest appearance record* (LAR) as a triple $(\mathrm{LAR}(m), \mathrm{init}, \mathrm{update})$ with

$$\mathrm{LAR}(m) = \{c_1 \cdots c_k \# c_{k+1} \cdots c_l \in (\{1, \ldots, m\} \cup \{\#\})^l \colon l \leq m,\ \text{each } c_i \in \{1, \ldots, m\}$$
$$\text{appears at most once and } \# \text{ exactly once}\}$$

and

- $\mathrm{init}(v) = \#\,\mathrm{val}(v)$
- $\mathrm{update}(c_1 \cdots c_k \# c_{k+1} \cdots c_l, v) = \begin{cases} c_1 \cdots c_k \# c_{k+1} \cdots c_l\, \mathrm{val}(v), & \mathrm{val}(v) \notin \{c_1, \ldots, c_l\} \\ c_1 \cdots c_{j-1} \# c_{j+1} \cdots c_l c_j, & \mathrm{val}(v) = c_j \end{cases}.$

For $m' = c_1 \cdots c_k \# c_{k+1} \cdots c_l$ we define the *hitset* $\mathrm{hit}(m')\colon = \{c_{k+1}, \ldots, c_l\}$.

The LAR keeps record of values that occurred recently and the hitset is the set of values that occurred since the last appearance of the current value. We will see that the hitset helps us identify the nodes that occur infinitely often. But first let us get more familiar with the definition by considering the following example.

1.4.6 Example: We can apply the LAR to an arbitrary (initial part of a) play. Let

$$\pi = \ v_4\ v_1\ v_3\ v_3\ v_2\ v_3\ v_4\ v_2\ v_4\ \cdots$$

To simplify matters let us suppose that $\mathrm{val}(v_i) = i$ for each $1 \leq i \leq 4$. We can see the respective state of the LAR at each position in π in the following table:

Position	LAR
v_4	# 4
v_1	# 4 1
v_3	# 4 1 3
v_3	4 1 # 3
v_2	4 1 # 3 2
v_3	4 1 # 2 3
v_4	# 1 2 3 4
v_2	1 # 3 4 2
v_4	1 3 # 2 4

Every time a new value is introduced, it is appended to the previous state of the LAR. Whenever a value occurs that was already there, it is moved from its place in the previous state to the end and to its previous position we move #.

We already mentioned that the hitset will help us identify the set of values occurring infinitely often in a play. The following lemma will make clearer how.

1.4.7 Lemma: Let $\mathcal{G}$ be a Muller game with latest appearance record LAR and $\pi = v_1 v_2 v_3 \cdots$ a play in $\mathcal{G}$. Let m_n denote the state of the update function at position v_n in the play. Then there exists n_0 such that for all $n > n_0$

1. $\mathrm{hit}(m_n) \subseteq \inf(\pi)$ and
2. $\mathrm{hit}(m_n) = \inf(\pi)$ infinitely often.

Proof.

1. There exists n_0' such that for all $n > n_0'$ we have $\mathrm{val}(v_n) \in \inf(\pi)$. From that point on, no value outside $\inf(\pi)$ will occur in π and therefore, no such value will be appended to the hitset by the update function. Additionally, by the time a value $a \in \inf(\pi)$ is visited for the second time after n_0', say at v_{n_0}, there cannot be any value outside of $\inf(\pi)$ in the hitset, because it then contains only values that were visited between the first and second appearance of a after n_0'. It follows that from n_0 on, we have $\mathrm{hit}(m_n) \subseteq \inf(\pi)$ for all $n > n_0$. Also, once every $a \in \inf(\pi)$ has been visited at least once after n_0, for all n from this point onwards m_n will be of the form $c_1 \cdots c_j c_{j+1} \cdots c_k \# c_{k+1} \cdots c_l$, where $\{c_{j+1}, \ldots, c_l\} = \inf(\pi)$ and $c_1 \cdots c_j$ remains unchanged.
2. Let n_0 be large enough such that all m_n for $n > n_0$ are of the form just described and let $m_n = c_1 \cdots c_j c_{j+1} \cdots c_k \# c_{k+1} \cdots c_l$, where $\{c_{j+1}, \ldots, c_l\} = \inf(\pi)$, for some $n > n_0$. Then $c_1 \cdots c_j c_{j+1}$ remains unchanged after v_n until a node with value c_{j+1} is visited at some node v_{n+r}. At that point we have $m_{n+r} = c_1 \cdots c_j \# c_{j+1} \cdots c_l$ and therefore $\mathrm{hit}(m_{n+r}) = \inf(\pi)$. It follows that for each $n \in \mathbb{N}_{>0}$ there is $m_{n'}$ with $n' > n$ such that $\mathrm{hit}(n') = \inf(\pi)$ and therefore this happens infinitely often.

□

Now we have all the tools necessary to construct a parity game for a given Muller game such that Player 0 wins the parity game if and only if she wins the Muller game.

1.4.8 Theorem: Every Muller game can be reduced to a parity game, using the latest appearance record.

Proof. Let $\mathcal{G} = (V, V_0, V_1, E, \mathrm{val}\colon V \to \{1, \ldots, m\})$ be a Muller game with winning condition $(\mathcal{F}_0, \mathcal{F}_1)$ and let $\mathrm{LAR} = (\mathrm{LAR}(m), \mathrm{init}, \mathrm{update})$ be its latest appearance record. We will show that there is a parity game $\mathcal{G}' = (V', V_0', V_1', E', \mathrm{val}'\colon V' \to \{1, \ldots, m\})$, where

- $V' = V \times \mathrm{LAR}(m)$, $V_0' = V_0 \times \mathrm{LAR}(m)$ and $V_1' = V_1 \times \mathrm{LAR}(m)$
- $E' = \{((v, m), (v', m'))\colon (v, v') \in E \text{ and } m' = \mathrm{update}(m, v')\}$

- $\text{val}'((v, c_1 \cdots c_k \# c_{k+1} \cdots c_l)) = \begin{cases} 2m+1-(2k+1), & \{c_{k+1}, \ldots, c_l\} \in \mathcal{F}_0 \\ 2m+1-2k, & \{c_{k+1}, \ldots, c_l\} \in \mathcal{F}_1 \end{cases}$

and Player 0 wins a play $\pi = v_1 v_2 v_3 \cdots$ in $\mathcal{G}$ if and only if Player 0 wins the projected play $\pi' = (v_1, m_1 = \text{init}(v_1))(v_2, m_2 = \text{update}(m_1))(v_3, m_3 = \text{update}(m_2)) \cdots$ in $\mathcal{G}'$.
To prove this, let $\pi = v_1 v_2 v_3 \cdots$ be a play in $\mathcal{G}$ and let n_0 be a point in π such that any value that occurs at or after v_{n_0} in π occurs infinitely often and the state of the update function at v_n for $n > n_0$ is of the form $c_1 \cdots c_j c_{j+1} \cdots c_k \# c_{k+1} \cdots c_l$, where $\{c_{j+1}, \ldots, c_l\} = \inf(\pi)$. It follows that in $\pi' = (v_1, m_1 = \text{init}(v_1))(v_2, m_2 = \text{update}(m_1))(v_3, m_3 = \text{update}(m_2)) \cdots$ all nodes (v_n, m_n) for $n > n_0$ in $\mathcal{G}'$ have values $\text{val}((v_n, m_n)) \geq 2j$. If Player i wins π then $\inf(\pi) \in \mathcal{F}_i$ and $\text{val}((v_n, m_n))$ with $\text{hit}(m_n) = \inf(\pi)$ is even. Additionally, for any node $(v_n, m_n) \in V'$ with $m_n = c_1 \cdots c_j c_{j+1} \cdots c_k \# c_{k+1} \cdots c_l$ and $\{c_{k+1} \cdots c_l\} \subseteq \inf(\pi)$ we have $\text{val}((v_n, m_n)) = 2m+1-(2k+\sigma) \leq 2m+1-(2j+i) = \text{val}((v'_n, c_1 \cdots c_j \# c_{j+1} \cdots c_l))$, where $\sigma \in \{0, 1\}$ and $\{c_{j+1} \cdots c_l\} = \inf(\pi)$. Since by Lemma 1.4.7 for all $n > n_0$ we have $\text{hit}(m_n) \subseteq \inf(\pi)$ and $\text{hit}(m_n) = \inf(\pi)$ infinitely often, it follows that Player i wins π'.
If on the other hand Player i wins π', then the largest value in □

This reduction to parity games also helps us to specify the space needed for a winning strategy. In particular we are interested in the number of different moves consistent with her winning strategy that a player can make from a fixed node, dependent on the state of the play at that node. This will become very relevant in later chapters.

1.4.9 Corollary: Muller games are determined with finite space. The size of the memory is bounded by $(m+1)!$ with m the number of different values.

Proof. Let $\mathcal{G}$ be a Muller game and $\mathcal{G}'$ the corresponding parity game defined as above. The parity game is positionally determined, but the positions are of the form (v, m'), with $m' \in \text{LAR}(m)$. There size of $\text{LAR}(m)$ is at most $(m+1)!$ since each of the $m+1$ symbols $1, \ldots, m, \#$ can occur at most once in any element of LAR. □

The last notion presented in this chapter is that of a Zielonka tree. The Zielonka tree is a more succinct representation of the sets $\mathcal{F}_0$ and $\mathcal{F}_1$ in one tree. Each node of the Zielonka tree represents a set in $\mathcal{F}_0$ or in $\mathcal{F}_1$ with each level (meaning all nodes with a set distance from the root) being either entirely in $\mathcal{F}_0$ or entirely in $\mathcal{F}_1$. Additionally, the size of sets at the nodes decreases along each path from the root. What makes the representation more succinct is that not all sets in $\mathcal{F}_0 \cup \mathcal{F}_1$ have to appear in the tree. Formally, we have the following definition.

1.4.10 Definition: Let $\mathcal{G}$ be a Muller game with winning condition $(\mathcal{F}_0, \mathcal{F}_1)$. The *Zielonka tree* $Z(\mathcal{F}_0, \mathcal{F}_1)$ for $(\mathcal{F}_0, \mathcal{F}_1)$ is defined inductively in the following way:
Let $C \in \mathcal{F}_i$, $i \in \{0, 1\}$, where $C = \text{val}(V)$, and let $C_0, \ldots, C_{k-1}$ be the maximal elements in the

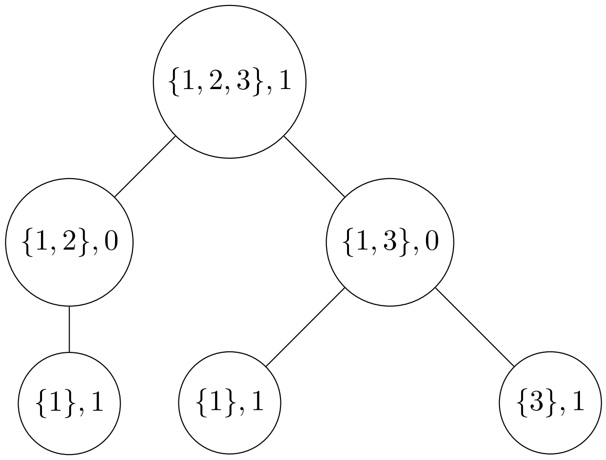
set $\{X \subseteq C \colon X \in \mathcal{F}_{1-i}\}$. Then $Z(\mathcal{F}_0, \mathcal{F}_1)$ has as its root the node $(\{1, \dots, m\}, i)$ with subtrees $Z(\mathcal{F}_0 \cap \mathcal{P}(C_0), \mathcal{F}_1 \cap \mathcal{P}(C_0)), \dots, Z(\mathcal{F}_0 \cap \mathcal{P}(C_{k-1}), \mathcal{F}_1 \cap \mathcal{P}(C_{k-1}))$ attached to it.

As an example let us construct a Zielonka tree for a winning condition $(\mathcal{F}_0, \mathcal{F}_1)$.

1.4.11 Example: Let the set C of values be $C = \{1, 2, 3\}$ and define

$$\mathcal{F}_0 = \{\{2\}, \{1, 2\}, \{1, 3\}\}, \quad \mathcal{F}_1 = \{\emptyset, \{1\}, \{3\}, \{2, 3\}, \{1, 2, 3\}\}.$$

This gives us the following Zielonka tree:



One advantage of presenting the winning conditions in a Zielonka tree is that using the following remark we can immediately see if the Muller game with these winning conditions is a parity game.

1.4.12 Remark: If $(\mathcal{F}_0, \mathcal{F}_1)$ is a parity condition, then the Zielonka tree is a single path.

Proof. For each set of values $\{1, \dots, m\}$ in $\mathcal{F}_i$ for $i = 0$ or $i = 1$ we have only one maximal subset that is in $\mathcal{F}_{1-i}$, namely $\{1, \dots, m-1\}$. Therefore each node has a unique successor in $Z(\mathcal{F}_0, \mathcal{F}_1)$. $\square$

In the next section we will use Zielonka trees in the context of Streett-Rabin games.

1.5 Streett-Rabin Games

1.5.1 Two Characterizations

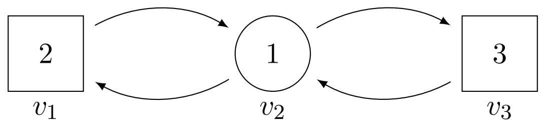
In this thesis we are particularly interested in a certain subclass of Muller games, namely the Streett-Rabin games. In a way, they are in between' Muller games and parity games in that one of the two players has a positional winning strategy on her winning region, but not necessarily the other.
The definitions and results in the first part of this section are again based on [Grä16]. Let us begin with the formal definition of a Streett-Rabin game.

1.5.1 Definition: A *Streett-Rabin game* is a Muller game $\mathcal{G} = (V, V_0, V_1, E; \mathrm{val}\colon V \to \{1, \dots, m\})$ with winning condition $(\mathcal{F}_0, \mathcal{F}_1)$, such that additionally $\mathcal{F}_0$ is closed under union, meaning that

$$X, X' \in \mathcal{F}_0 \quad \Rightarrow \quad X \cup X' \in \mathcal{F}_0.$$

Let us consider the examples from the previous section. We had the same game graph with different winning conditions.

1.5.2 Example: Again let $V_0 = \{v_2\}$, $V_1 = \{v_1, v_3\}$, $\mathrm{val}(v_1) = 2$, $\mathrm{val}(v_2) = 1$ and $\mathrm{val}(v_3) = 3$ and consider the following game graph.



First the winning conditions were

$$\mathcal{F}_0 = \{\emptyset, \{2\}, \{1, 2\}\}, \quad \mathcal{F}_1 = \{\{1\}, \{3\}, \{1, 3\}, \{2, 3\}, \{1, 2, 3\}\},$$

representing a parity game. As we can easily see this is also a Streett-Rabin game since $\mathcal{F}_0$ is closed under union. For the second example we had the winning conditions

$$\mathcal{F}_0 = \{\emptyset, \{1\}, \{2\}, \{1, 2, 3\}\}, \quad \mathcal{F}_1 = \{\{3\}, \{1, 3\}, \{2, 3\}, \{1, 2\}\}.$$

This game is not a Streett-Rabin game, since $\{1\} \cup \{2\} = \{1, 2\} \notin \mathcal{F}_0$. Now consider the winning conditions

$$\mathcal{F}_0 = \{\emptyset, \{2\}, \{3\}, \{2, 3\}, \{1, 2, 3\}\}, \quad \mathcal{F}_1 = \{\{1\}, \{1, 3\}, \{1, 2\}\}.$$

This is a Streett-Rabin game but not a parity game, since the only way for Player 0 to win is as in the previous example to alternate between going to v_1 and v_3 from v_2. So she does not have a positional winning strategy.

Not just the parity game in our example is a Streett-Rabin game, but every parity game is. Consider a parity game whose winning condition is given as a Muller condition. Then $\mathcal{F}_0$ contains precisely those subsets of the set C of occurring values, whose maximal value is even. But then the maximal value in the union of two such sets is also even. In fact we can give an analogous argument for $\mathcal{F}_1$ and obtain that in the case of parity games, both $\mathcal{F}_0$ and $\mathcal{F}_1$ are closed under union.
But now let us come back to arbitrary Streett-Rabin games. We have already mentioned that one of the players has a positional winning strategy from her winning region. Now let us prove it.

1.5.3 Proposition: Let $\mathcal{G}$ be a Streett-Rabin game with m different values in the set C of occurring values and winning condition $(\mathcal{F}_0, \mathcal{F}_1)$. Then

- $\mathcal{G}$ is determined.
- Player 1 has a positional winning strategy on his winning region W_1.
- Player 0 has a winning strategy using finite space of size bounded by $(m+1)!$ on her winning region W_0.

Proof. Because $\mathcal{G}$ is also a Muller game, $\mathcal{G}$ is determined and Player 0 has a winning strategy using finite space of size $(m+1)!$ on W_0 by Corollary 1.4.9. What remains to be shown is that Player 1 has a positional winning strategy on W_1.
We will prove this by induction over the height of the Zielonka tree.

Case 1: $C \in \mathcal{F}_1$

We know that $W_1 = V \backslash W_0$ is a trap for Player 0, so there exists a positional strategy f for Player 1 to remain in W_1 from W_1 (always choose a successor in W_1, which exists by definition). Let C' be the unique largest subset of C which is in $\mathcal{F}_0$. This exists, because if there were two distinct largest subsets $C_1, C_2 \subseteq C$ in $\mathcal{F}_0$ then because $\mathcal{F}_0$ is closed under union, $C_1 \cup C_2 \in \mathcal{F}_0$, which is larger than either C_1 or C_2. Define

$$Y = W_1 \cap \operatorname{val}^{-1}(C \backslash C') \quad \text{and} \quad Z = \operatorname{Attr}_1(Y) \backslash Y.$$

Then Player 1 has a positional attractor strategy a to move from Z to Y. Define a game $\mathcal{G}' = (V', V_0', V_1', E', \operatorname{val}|_{V'})$ by $V' = W_1 \backslash (Y \cup Z)$, $V_0' = V' \cap V_0$, $V_1' = V' \cap V_1$ and $(v, w) \in E'$ if and only if $(v, w) \in E$ and $v, w \in V'$. The set of values occurring in $\mathcal{G}'$ is now a subset of C' and therefore the Zielonka tree of $\mathcal{G}'$ has a height reduced by at least 1 compared to that of $\mathcal{G}$. By induction hypothesis, there exists a partition of V' into W_0' and W_1' such that Player 0 has a winning strategy on W_0' and Player 1 has a positional winning strategy g' on W_1'. Since however V' is a subset of W_1, $W_0' = \emptyset$ and therefore $W_1' = V'$. The sets V', Z and Y are a partition of W_1, so we can define a positional strategy $g\colon W_1 \cap V_1 \to W_1$

for Player 1 on W_1 by

$$g(v) = \begin{cases} g'(v), & v \in V' \\ a(v), & v \in Z \\ f(v), & v \in Y. \end{cases}$$

for all $v \in W_1$. Let π be a play in $\mathcal{G}$ consistent with g. If infinitely many positions of π are in Y then there is a value in $C\backslash C'$ that occurs infinitely often in π. Since C' is the unique maximal subset of C in $\mathcal{F}_0$, the set $\inf(\pi)$ of values occurring infinitely often in π cannot be in $\mathcal{F}_0$. Therefore, Player 1 wins π. If on the other hand only finitely many positions of π are in Y then Z is also visited only finitely many times, because it is the attractor of Y and every visit of a node in Z leads to a visit of Y according to f. Thus at some point π never leaves V' and is from then on consistent with g' which is a winning strategy on V'. Therefore Player 1 wins π in this case as well.

Case 2: $C \in \mathcal{F}_0$

Let $X_1 = \{v \in V\colon$ Player 1 has a positional winning strategy from $v\}$ and $X_0 = V\backslash X_1$. We need to show that Player 0 has a winning strategy on X_0 in order to show that $X_1 = W_1$.

Let $C_0, \ldots, C_{k-1}$ be the maximal subsets of C that are in $\mathcal{F}_1$. Note that if we have a set $D \subseteq C$ such that $D \cap (C\backslash C_i) \neq \emptyset$ for all $0 \leq i \leq k-1$ then $D \in \mathcal{F}_0$ since otherwise D would also be a maximal subset in $\mathcal{F}_1$.

For all $0 \leq i \leq k-1$ define

$$Y_i = X_0 \cap \mathrm{val}^{-1}(C\backslash C_i) \quad \text{ and } \quad Z_i = \mathrm{Attr}_0(Y_i)\backslash Y_i.$$

For all $i < k$ let a_i be an attractor strategy on Z_i. Define games $\mathcal{G}_i = (V_i, V_{0_i}, V_{1_i}, E_i, \mathrm{val}\,|_{V_i})$ for all $0 \leq i \leq k-1$ by $V_i = X_0\backslash(Y_i \cup Z_i)$, $V_{0_i} = V_i \cap V_0$, $V_{1_i} = V_i \cap V_1$ and $(v, w) \in E_i$ if and only if $(v, w) \in E$ and $v, w \in V_i$. Then the set of values occurring in $\mathcal{G}_i$ is a subset of C_i for all $i < k$. Therefore the Zielonka tree has a height of at least one less than the original Zielonke Tree. By induction hypothesis for all $i < k$ there is a partition of V_i in W_{0_i} and W_{1_i} auch that Player 0 has a winning strategy on W_{0_i} and Player 1 has a positional winning strategy on W_{1_i}. This implies $W_{1_i} = \emptyset$ since $W_{1_i} \subseteq X_0 = V\backslash X_1$. Therefore $V_i = W_{0_i}$ for all $i < k$. For all $i < k$ let f_i be a winning strategy for Player 0 on V_i. Define $f_i'\colon V_i \cup Z_i \to X_0$ by

$$f_i'(v) = \begin{cases} f_i(v), & v \in V_i \\ a_i(v), & v \in Z_i \end{cases}$$

for all $v \in V_i \cup Z_i$. We now define a strategy f for Player 0 on X_0 in the following way: In $\bigcap_{i=0}^{k-1} Y_i$ play according to a trap strategy t to keep the play in X_0. The first time the play enters $X_0\backslash\bigcap_{i=0}^{k-1} Y_i$ at some node v, take the smallest $i < k$ such that $v \notin Y_i$ and play according to f'_i until entering $Y_i\backslash\left(\bigcap_{i=0}^{k-1} Y_i\right)$ at some node w. Now start playing according to $f_{i+j(\text{mod } k)}$, where j is the smallest natural number such that $w \notin Y_{i+j}$. Repeat this upon entering $Y_{i+j}\backslash\left(\bigcap_{i=0}^{k-1} Y_i\right)$ and so on.

What remains to be seen is that Player 0 wins any play π consistent with f. Let π be such a play. We claim that either π stays in V_i for some $i < k$ from some point onwards, in which case the play will from then on be consistent with f_i and therefore winning for Player 0, or there are infinitely many positions of π in each Y_i, meaning that $\inf(\pi) \cap (C\backslash C_i) \neq \emptyset$ for all $i < k$ which by the remark in the beginning of the proof implies $\inf(\pi) \in \mathcal{F}_0$.

To prove the claim, assume first that Player 0 plays only according to t and f'_i for a fixed $i < k$ from some point onwards. Then either the play stays in V_i forever or there are infinitely many positions outside V_i. If it is not in V_i, it will enter Y_i either immediately or as a result of the attractor strategy on Z_i. But if at this point it left $\bigcap_{i=0}^{k-1} Y_i$, Player 0 would switch to f'_j for some $j \neq i$, which contradicts the assumption. Therefore $\bigcap_{i=0}^{k-1} Y_i$ is visited infinitely often.

If on the other hand Player 0 switches between the f'_i infinitely often, define

$$I = \{i < k\colon \text{ Player 0 switches infinitely often to } f'_i\}$$

and $J = \{0, \dots, k-1\}\backslash I$. In particular, Player 0 also switches infinitely often away from f'_i for all $i \in I$. Since this only happens at $v \in Y_i$, Y_i is visited infinitely often for all $i \in I$. If Y_j was visited only finitely many times for some $j \in J$, then at some point π would always be outside Y_j. But since j lies between i_1 and i_2 modulo k for some $i_1, i_2 \in I$ which have no $i_3 \in I$ between them modulo k, Player 0 would then switch to f'_j upon leaving Y_{i_1} each time, which is a contradiction. Therefore, all Y_i are visited infinitely many times.

□

The previous result is the main result about Streett-Rabin games that we will use in later chapters. However, the way we have defined Streett-Rabin games in this section is only one of two possible ways of characterising them, with the other option being the more common one. Of course we will not neglect this second characterisation.

All the premises concerning the game graph, the players and the moves are the same as before, but the winning condition will be stated in a different way.

1.5.4 Definition: Let $G = (V, V_0, V_1, E, \text{val})$ be a game graph with $V_0 \cup V_1 = V$, $V_0 \cap V_1 = \emptyset$ and $\text{val}\colon V \to \{1, \dots, m\}$. Let $k \in \mathbb{N}_{>0}$ and consider the *pair condition* $(G_i, F_i)_{1\leq i\leq k}$ with $G_i, F_i \subseteq \{1, \dots, m\}$ for all $i \leq k$. Player 0 wins a play π on G if $\inf(\pi) \cap G_i \neq \emptyset$ implies $\inf(\pi) \cap F_i \neq \emptyset$ for all $i \in \{1, \dots, k\}$ and Player 1 wins otherwise. This defines a Muller condition $(\mathcal{F}_0, \mathcal{F}_1)$ for $\mathcal{G} = (V, V_0, V_1, E, \text{val}\colon V \to \{1, \dots, m\})$ in the following way:

$$\mathcal{F}_0 = \{F \subseteq \{1, \dots, m\} \colon (F \cap G_i \neq \emptyset \;\Rightarrow\; F \cap F_i \neq \emptyset) \text{ for all } i = 1, \dots, k\}$$
$$\mathcal{F}_1 = \mathcal{P}(\{1, \dots, m\}) \backslash \mathcal{F}_0$$

At first glance, this winning condition looks very different from the one previously defined for Streett-Rabin games. So let us make sure, they are indeed interchangeable. The proof of the following result is based on [Zie98].

1.5.5 Theorem: A Muller game with winning conditions as in Definition 1.5.4 is a Streett-Rabin game and every Streett-Rabin game is equivalent to a pair condition as in Definition 1.5.4.

Proof. To prove the first part of the statement, let $(G_i, F_i)_{1 \leq i \leq k}$ be a pair condition. We need to show that $\mathcal{F}_0 = \{F \subseteq \{1, \dots, m\} \colon (F \cap G_i \neq \emptyset \;\Rightarrow\; F \cap F_i \neq \emptyset) \text{ for all } i = 1, \dots, k\}$ is closed under union. So let $F, F' \in \mathcal{F}_0$ and let $(F \cup F') \cap G_i \neq \emptyset$ for some $i \leq k$. Then $F \cap G_i \neq \emptyset$ or $F' \cap G_i \neq \emptyset$. In the first case $F \cap F_i \neq \emptyset$ since $F \in \mathcal{F}_0$ and in the second case $F' \cap F_i \neq \emptyset$ since $F' \in \mathcal{F}_0$. In either case we have $(F \cup F') \cap F_i \neq \emptyset$ and since i was arbitrary, $F \cup F' \in \mathcal{F}_0$.
It follows that $\mathcal{F}_0$ is closed under union and $(\mathcal{F}_0, \mathcal{F}_1)$ is a Streett-Rabin condition. Obviously $\inf(\pi) \in \mathcal{F}_0$ if and only if $\inf(\pi) \cap G_i \neq \emptyset \;\Rightarrow\; \inf(\pi) \cap F_i \neq \emptyset$ for all $i \in \{1, \dots, k\}$ and therefore Player 0 wins π according to the winning conditions $(\mathcal{F}_0, \mathcal{F}_1)$ if and only if she wins π according to the pair condition.
Now we show the second part of the statement. Let $(\mathcal{F}_0, \mathcal{F}_1)$ be a Streett-Rabin condition, i.e. a Muller condition where $\mathcal{F}_0$ is closed under union. If $\mathcal{F}_1 = \emptyset$ we can define a pair condition with no pairs. Then any play is won by Player 0 according to $(\mathcal{F}_0, \mathcal{F}_1)$ and for any play π we have $\inf(\pi) \cap G_i \neq \emptyset \;\Rightarrow\; \inf(\pi) \cap F_i \neq \emptyset$ for all $i \in \{1, \dots, k\}$ since there are no G_i. Therefore Player 0 also wins any play according to the pair condition.
So let us suppose that $\mathcal{F}_1 \neq \emptyset$. Let $H_1, \dots, H_l \in \mathcal{F}_1$ be all the sets such that there are nodes in the Zielonka tree $Z(\mathcal{F}_0, \mathcal{F}_1)$ labeled with $(H_i, 1)$ for each $1 \leq i \leq l$. Now define $Q_i \in \mathcal{F}_0$ as the set labelling the unique child of H_i for each $1 \leq i \leq l$ or set $Q_i = \emptyset$ if H_i has no child. Note that $Y \in \mathcal{F}_1$ if and only if $Y \subseteq H_i$ and $Y \not\subseteq Q_i$ for some $1 \leq i \leq k$: Let $Y \in \mathcal{F}_1$. If $Y \not\subseteq H_i$ for all $1 \leq i \leq k$, then Y would itself be the set C of all occurring values, if $C \in \mathcal{F}_1$, or a maximal subset of C, if $C \in \mathcal{F}_0$. But then Y would be one of the H_i which is a contradiction. Now consider all H_i such that $Y \subseteq H_i$. If $Y \subseteq Q_i$ for all such i, Then at some point Y would be a maximal subset of a Q_i, since the Q_i become ever smaller, the further we descend in the tree. But then again Y would be one of the H_i, which is a contradiction. On the other hand assume that $Y \in \mathcal{F}_0$ and that $Y \subseteq H_i$ and $Y \not\subseteq Q_i$ for some $1 \leq i \leq k$. Since Q_i is the only child of H_i and $Y \not\subseteq Q_i$, Q_i cannot be a maximal subset of H_i in $\mathcal{F}_0$, a contradiction.
But $Y \subseteq H_i$ is equivalent to $Y \cap (C \backslash H_i) = \emptyset$ and $Y \not\subseteq Q_i$ is equivalent to $Y \cap (C \backslash Q_i) \neq \emptyset$. Thus define $G_i = C \backslash Q_i$ and $F_i = C \backslash H_i$ for all $1 \leq i \leq k$ to obtain a pair condition $(G_i, F_i)_{1 \leq i \leq k}$

corresponding to the Muller condition $(\mathcal{F}_0, \mathcal{F}_1)$. □

We have already considered the winning strategies for both players in Streett-Rabin games given with a Muller condition. However if we present a Streett-Rabin game with a pair condition, we have another parameter, namely the number k of pairs. Naturally, we are interested in how this parameter influences the winning strategies, or rather the size of the winning strategies of Player 0. For the Muller condition we got a statement depending on the parameter m using the latest appearance record. We will now define a similar record for Streett-Rabin games given with a pair condition, based on [Hor05] and [BLV96].

1.5.6 Definition: Let $\mathcal{G}$ be a Streett-Rabin game with pair condition $(G_i, F_i)_{1\leq i\leq k}$. An *index of appearance record (IAR)* is a tuple (S, e, f) with

- $S \in S_k$, which denotes the set of all permutations over $\{1, \dots, k\}$ and
- $e, f \in \{1, \dots, k\}$.

Now we have the tools to prove the following statement.

1.5.7 Proposition: Let $\mathcal{G} = (V, V_0, V_1, E, \text{val}\colon V \to \{1, \dots, m\})$ be a Streett-Rabin game with m different colours and winning condition $(G_i, F_i)_{1\leq i\leq k}$. Then

- $\mathcal{G}$ is determined.
- Player 1 has a positional winning strategy on his winning region W_1.
- Player 0 has a winning strategy using space of size bounded by $(k+2)!$ on her winning region W_0.

Proof. That $\mathcal{G}$ is determined and that Player 1 has a positional winning strategy on W_1 follow directly from Theorem 1.5.5 and Proposition 1.5.3.
In order to prove the third statement, we show that we can reduce $\mathcal{G}$ to a parity game $\mathcal{G}'$ using the IAR, just as we have done with the LAR and the Muller conditions before. Define the parity game $\mathcal{G}' = (V', V_0', V_1', E', \text{val}')$ as follows:

- $V' = V \times S_k \times \{1, \dots, k\} \times \{1, \dots, k\}$,
- $V_0' = \{(v, S, e, f) \in V' \colon v \in V_0\}$,
- $V_1' = \{(v, S, e, f) \in V' \colon v \in V_1\}$,
- for $(v, S, e, f), (v', S', e', f') \in V'$ we have $((v, S, e, f), (v', S', e', f')) \in E'$ if all of the following hold
  - $(v, v') \in E$.

- Let $I = \{i \leq k\colon \operatorname{val}(v') \in F_i\}$. Then S' is obtained from S by shifting all $i \in I$ to the beginning of the permutation while keeping their order nad the order of the remaining elements the same.
- e is the maximal position of an $i \in I$ in S'.
- f is the maximal position in S' of an $i < k$ such that $\operatorname{val}(v') \in G_i$.

- $\operatorname{val}'\colon V' \to \{1, \dots, 2m\},\ (v, S, e, f) \mapsto \max(2e, 2f - 1)$.

We need to show that Player 0 wins a play $\pi = v_1v_2v_3\cdots$ in $\mathcal{G}$ if and only if Player 0 wins the corresponding play $\pi' = (v_1, S_1, e_1, f_1)(v_2, S_2, e_2, f_2)(v_3, S_3, e_3, f_3)\cdots$ in $\mathcal{G}'$.
Suppose first that Player 0 wins π in $\mathcal{G}$. Then $\inf(\pi) \cap G_i \neq \emptyset$ implies $\inf(\pi) \cap F_i \neq \emptyset$ for all $i \in \{1, \dots, k\}$. Let $n_0 \in \mathbb{N}$ be such that $\operatorname{val}(v_n) \in \inf(\pi)$ for all $n > n_0$. Note that $2e > 2f - 1$ if and only if $e \geq f$. Let a be the largest element in $\{1, \dots, k\}$ such that a is the maximal position in infinitely many S_n, $n > n_0$, such that $\operatorname{val}(v_n) \in F_i$. Let b be an element in $\{1, \dots, k\}$ such that b is the maximal position in infinitely many S_n, $n > n_0$, of an $i < k$ such that $\operatorname{val}(v_n) \in G_i$. Since Player 0 wins π, there also have to be infinitely many $v_{n'}$, $n' > 0$, such that $\operatorname{val}(v_{n'}) \in F_i$ and therefore $b \leq a$. It follows that the largest e_n that occurs infinitely often is larger than or equal to the largest f_n that occurs infinitely often and therefore the largest value of val' occurring infinitely often is of the form $2e$ for some e, thus even. It follows that Player 0 wins π'.
Now suppose that Player 1 wins π. Then there is $j \leq k$ and a value c occurring infinitely often such that $c \in G_j$ but after n_0 defined as above there are no such values in F_i. Since S_n for $n > 1$ is obtained from S_{n-1} by shifting all $i \leq k$ such that $v_n \in F_i$ to the beginning, all i such that there is a value $a \in F_i$ that occurs infinitely often in π will forever be at lower positions in the permutations S_n for all n large enough such that all values occurring infinitely often have been visited at least once after n_0. Let us call this point $v_{n_0'}$. At each of the infinitely many occurrences of c at nodes v_n after $v_{n_0'}$ we have $f_n \geq j$ and j is larger than any i such that $\operatorname{val}(v_l) \in F_i$ for every v_l with $l > n_0'$. Thus, $f_n > e_l$ for this f_n occurring infinitely often and each e_l occurring after $v_{n_0'}$ and therefore the largest value occurring infinitely often in π' is of the form $2f - 1$ for some f, hence odd. It follows that Player 1 wins π'.
We have shown that $\mathcal{G}$ can be reduced to the parity game $\mathcal{G}'$ which is positionally determined, but with respect to positions in $V \times S_k \times \{1, \dots, k\} \times \{1, \dots, k\}$. Thus for each $v \in V_0$ there are at most $k! \cdot k^2 \leq (k + 2)!$ different positions in $\mathcal{G}'$ that correspond to v in $\mathcal{G}$. Therefore, Player 0 has a winning strategy using space of size bounded by $(k + 2)!$ on her winning region W_0. $\square$

1.5.2 Complexity

In the next chapter we will see that parity games can be solved in quasipolynomial time. In many ways Streett-Rabin games are very similar to parity games but they are also a strictly larger class of games. So we have to curb our enthusiasm a little bit when trying to find similar results as for parity games. In fact solving Streett-Rabing games is NP-complete or Co-NP-complete, depending on the player considered, as we can see in the following proposition, which can also be found in [Grä16].

1.5.8 Proposition: Deciding whether Player 0 wins a Street-Rabin game $\mathcal{G}$ from starting position v is Co-NP-complete. Determining whether Player 1 wins is NP-complete.

Proof. We prove the statement for Player 1 using a reduction from SAT. Let $\psi = \bigwedge_{i\in I} C_i$, $C_i = \bigvee_{j\in J} Y_{ij}$, Y_{ij} literals, be a formula in conjunctive normal form with variables $X_1, \dots, X_k$. We define a Streett-Rabin game $\mathcal{G}_\psi = (V, V_0, V_1, E, \text{val})$ with winning conditions $(\mathcal{F}_0, \mathcal{F}_1)$ as follows:

- V is the set of clauses and literals in ψ.
- V_0 is the set of literals $X_1, \dots, X_k, \neg X_1, \dots, \neg X_k$ in V.
- V_1 is $V \backslash V_0$.
- $E = \{(\phi, \phi') \colon \phi = C_i \text{ and } \phi' = Y_{ij} \text{ for some } i \in I,\ j \in J\} \cup \{(\phi, \phi') \colon \phi = Y_{ij} \text{ and } \phi' = C_l \text{ for some } i, l \in I,\ j \in J\}$.
- val assigns to each literal and clause its own distinct value in $\{1, \dots, |I| \cdot |J|\}$ (and is not relevant here).
- $\mathcal{F}_0 = \{Z \subseteq \{1, \dots, |I| \cdot |J|\} \colon \{\text{val}(X_l), \text{val}(\neg X_l)\} \in Z \text{ for some } l \in \{1, \dots, k\}\}$.
- $\mathcal{F}_1 = \mathcal{P}(\{1, \dots, |I| \cdot |J|\}) \backslash \mathcal{F}_0$.

Obviously, $\mathcal{F}_0$ is closed under union, so $(\mathcal{F}_0, \mathcal{F}_1)$ is a Streett-Rabin condition.
What remains to be shown is that ψ is satisfiable if and only if Player 1 wins from any starting position.
Suppose that ψ is satisfiable. Then there exists an interpretation $\text{Int} \colon \{X_1, \dots, X_k\} \to \{0, 1\}$ such that if we substitute $\text{Int}(X_i)$ for each X_i in ψ we obtain 1. We define a strategy f for Player 1, where $f(C)$ for a clause C is a literal Y in C with $\text{Int}(Y) = 1$. Such a Y exists, because of the conjunctive normal form of ψ and the definition of Int. But then all literals Y (except possibly the starting position) visited in any play consistent with f satisfy $\text{Int}(Y) = 1$ and therefore no two literals of the form $X, \neg X$ can be visited infinitely often. It follows that Player 1 wins the play in $\mathcal{G}_\psi$.
Now suppose that ψ is unsatisfiable. Assume that Player 1 has a positional winning strategy f from any position in $\mathcal{G}_\psi$. Since ψ is unsatisfiable, there exist clauses C, C' such that $f(C) = X$ and $f(C') = \neg X$ for some variable X: If this were not the case, define an interpretation $\text{Int} \colon \{X_1, \dots, X_k\} \to \{0, 1\}$ by defining $\text{Int}(X) = 1$ if $f(C) = X$ for some clause C and $\text{Int}(X) = 0$ if $f(C) = \neg X$ for some clause C, which would then be well defined. It follows that in each

clause there is at least some literal Y with $\mathrm{Int}(Y) = 1$ and therefore if we substitute $\mathrm{Int}(X_i)$ for each X_i in ψ we obtain 1. This contradicts the unsatisfiability of ψ.
Let C, C' and X be such that $f(C) = X$ and $f(C') = \neg X$. Define a strategy g for Player 0 by $g(\neg X) = C$ and $g(Y) = C'$ for every literal $Y \neq \neg X$. Because Player 1 always moves from C to X and from C' to $\neg X$, in any play consistent with f and g both X and $\neg X$ will be visited infinitely often and Player 0 wins.
This means that Player 1 cannot have a positional winning strategy for every position in $\mathcal{G}_\psi$ and by Proposition 1.5.3 does not have any winning strategy for every position in $\mathcal{G}_\psi$.
Since SAT is an NP-complete problem it follows that deciding whether Player 1 wins a Streett-Rabin game from a given starting proition is NP-complete. By the determinacy of Streett-Rabin games as Muller games we obtain the Co-NP-completeness for Player 0. $\square$

This does not mean, however, that there are no possible interesting complexity improvements for Streett-Rabin games. In fact, using the methods for parity games presented in the next chapter and applying them to Streett-Rabin games, we will see that they are decidable in FPT, a result which was not known before the paper [Cal17] by Calude et al. was published.

1.6 Modal μ-Calculus

In this section we will introduce the modal μ-calculus. It is a fixed point logic based on modal logic and we are particularly interested in it, because its model checking games are parity games. The last chapter of this thesis will be dedicated to how the methods for solving parity games in quasipolynomial time can best be implemented to solve model checking games for the modal μ-calculus. But first, let us get to know this logic.
Most readers will be familiar with modal logic, but we will still include its definition for convenience. Let us start with the syntax of modal logic ML and base upon it the modal μ-calculus L_μ. This section is largely based on [**?**] and [Zap02].

1.6.1 Definition: Let $(P_i)_{i\in I}$ be atomic propositions and A a set of agents. The syntax of *modal logic* ML is defined as follows.

- For all $i \in I$ we have $P_i \in \mathrm{ML}$ and $\neg P_i \in \mathrm{ML}$,
- if $\phi_1, \phi_2 \in \mathrm{ML}$ then $\phi_1 \wedge \phi_2 \in \mathrm{ML}$,
- if $\phi_1, \phi_2 \in \mathrm{ML}$ then $\phi_1 \vee \phi_2 \in \mathrm{ML}$,
- for all $a \in A$ if $\phi \in \mathrm{ML}$ then $\langle a\rangle\phi \in \mathrm{ML}$,
- for all $a \in A$ if $\phi \in \mathrm{ML}$ then $[a]\phi \in \mathrm{ML}$.

If we allow the following additions to the syntax of ML, we obtain the *modal μ-calculus* L_μ:

- For all $i \in I$ we have $P_i \in \mathrm{L}_\mu$ and $\neg P_i \in \mathrm{L}_\mu$,
- if $\phi_1, \phi_2 \in \mathrm{L}_\mu$ then $\phi_1 \wedge \phi_2 \in \mathrm{L}_\mu$,
- if $\phi_1, \phi_2 \in \mathrm{L}_\mu$ then $\phi_1 \vee \phi_2 \in \mathrm{L}_\mu$,
- for all $a \in A$ if $\phi \in \mathrm{L}_\mu$ then $\langle a\rangle\phi \in \mathrm{L}_\mu$,
- for all $a \in A$ if $\phi \in \mathrm{L}_\mu$ then $[a]\phi \in \mathrm{L}_\mu$.
- if $\phi \in \mathrm{L}_\mu$ with X appearing only positively then $\mu X.\phi \in \mathrm{L}_\mu$ and
- if $\phi \in \mathrm{L}_\mu$ with X appearing only positively then $\nu X.\phi \in \mathrm{L}_\mu$.

1.6.2 Remark: If $|A| = 1$, we write $\diamond\phi$ and $\Box\phi$ instead of $\langle a\rangle\phi$ and $[a]\phi$.

The following well-known theorem by Knaster and Tarski gives us what we need in order to give well-defined semantics fro the modal μ-calculus. A proof can be found in [Tar55].

1.6.3 Theorem (Knaster and Tarski)**:** Let $(L, \preceq)$ be a complete lattice and $g\colon L \to L$ a monotonic function with respect to $\preceq$. Then g has a *least fixed point* μg and a *greatest fixed point* νg satisfying

$$\mu g = \bigcap\{V \in L\colon g(V) \preceq V\}, \text{ and}$$
$$\nu g = \bigcup\{V \in L\colon V \preceq g(V)\}.$$

Before we give the semantics for L_μ let us briefly give the semantics for modal logic.

1.6.4 Definition: Let A and I be finite sets. A *Kripke structure* is $\mathcal{K} = (W, (E_a)_{a\in A}, (P_i)_{i\in I})$ with W a set of worlds, $E_a \subseteq W \times W$ for each $a \in A$ and $P_i \subseteq W$ an atomic proposition for all $i \in I$. The elements in A are called *agents*.

1.6.5 Definition: Let $\mathcal{K} = (W, (E_a)_{a\in A}, (P_i)_{i\in I})$ be a Kripke structure. We define for $v \in W$, $\phi_1, \phi_2 \in \mathrm{ML}$, $i \in I$ and $a \in A$

- $\mathcal{K}, v \vDash P_i \;:\Leftrightarrow\; v \in P_i$,
- $\mathcal{K}, v \vDash \neg P_i \;:\Leftrightarrow\; v \notin P_i$,
- $\mathcal{K}, v \vDash \phi_1 \vee \phi_2 \;:\Leftrightarrow\; \mathcal{K}, v \vDash \phi_1$ or $\mathcal{K}, v \vDash \phi_2$,
- $\mathcal{K}, v \vDash \phi_1 \wedge \phi_2 \;:\Leftrightarrow\; \mathcal{K}, v \vDash \phi_1$ and $\mathcal{K}, v \vDash \phi_2$,
- $\mathcal{K}, v \vDash \langle a\rangle\phi_1 \;:\Leftrightarrow\;$ there is $(v, w) \in E_a$ such that $\mathcal{K}, w \vDash \phi_1$,
- $\mathcal{K}, v \vDash [a]\phi_1 \;:\Leftrightarrow\;$ for all $(v, w) \in E_a$ we have $\mathcal{K}, w \vDash \phi_1$.

All that remains for the semantics of L_μ are the fixed point formulas $\mu X.\phi \in \mathrm{L}_\mu$ and $\nu X.\phi \in \mathrm{L}_\mu$. Calling them *fixed point formulas* already implies, that they are satisfied by some set of worlds in the underlying Kripke structure that lies in a fixed point. But a fixed point of what? The following definition hints at the answer to that question.

1.6.6 Definition: Let $\phi \in \mathrm{ML}$ with X appearing only positively in ϕ and let a Kripke structure $\mathcal{K} = (W, (E_a)_{a\in A}, (P_i)_{i\in I})$ be given. Then we define

$$g_\phi^{\mathcal{K}}\colon \mathcal{P}(W) \to \mathcal{P}(W),\; X \mapsto \{w \in W\colon (\mathcal{K}, X), w \vDash \phi\}.$$

To apply the Knaster-Tarski Theorem we need monotony, but this is not a problem.

1.6.7 Remark: The function $g_\phi^{\mathcal{K}}$ is monotone, since X appears only positively.

Now we can finally give the full semantics for L_μ.

1.6.8 Definition: Let $\mathcal{K} = (W, (E_a)_{a\in A}, (P_i)_{i\in I})$ be a Kripke structure. We define for $i \in I$, $a \in A$, $\phi_1, \phi_2 \in \mathrm{L}_\mu$ and $v \in W$

- $\mathcal{K}, v \vDash P_i \;:\Leftrightarrow\; v \in P_i$,
- $\mathcal{K}, v \vDash \neg P_i \;:\Leftrightarrow\; v \notin P_i$,
- $\mathcal{K}, v \vDash \phi_1 \vee \phi_2 \;:\Leftrightarrow\; \mathcal{K}, v \vDash \phi_1$ or $\mathcal{K}, v \vDash \phi_2$,
- $\mathcal{K}, v \vDash \phi_1 \wedge \phi_2 \;:\Leftrightarrow\; \mathcal{K}, v \vDash \phi_1$ and $\mathcal{K}, v \vDash \phi_2$,
- $\mathcal{K}, v \vDash \langle a\rangle\phi_1 \;:\Leftrightarrow\;$ there is $(v, w) \in E_a$ such that $\mathcal{K}, w \vDash \phi_1$,
- $\mathcal{K}, v \vDash [a]\phi_1 \;:\Leftrightarrow\;$ for all $(v, w) \in E_a$ we have $\mathcal{K}, w \vDash \phi_1$,

just as in ML. Now let $\phi \in \mathrm{L}_\mu$ with X appearing only positively. Then

- $\mathcal{K}, v \vDash \mu X.\phi \;:\Leftrightarrow v \in \mu g^{\mathcal{K}}_\phi$,
- $\mathcal{K}, v \vDash \nu X.\phi \;:\Leftrightarrow v \in \nu g^{\mathcal{K}}_\phi$

We can determine the least and the greatest fixed point of $g^{\mathcal{K}}_\phi$ in the following way: For $\mu g^{\mathcal{K}}_\phi$ define $X^0 = \emptyset$, $X^{\alpha+1} = g^{\mathcal{K}}_\phi(X^\alpha)$ for successor ordinals and $X^\lambda = \bigcup_{\alpha<\lambda} X^\alpha$ for limit ordinals. Since $g^{\mathcal{K}}_\phi$ is monotone, we have $X^\alpha \subseteq X^{\alpha+1}$ by induction. It follows that for some ordinal α we have $X^\beta = X^{\beta+1}$ for all $\beta \geq \alpha$. For this α we have $X^\alpha = \mu g^{\mathcal{K}}_\phi$ since obviously

$$\mu g^{\mathcal{K}}_\phi = \bigcap\{V \in L \colon g^{\mathcal{K}}_\phi(V) \subseteq V\} \subseteq X^\alpha.$$

On the other hand induction shows that $X^\beta \subseteq \mu g^{\mathcal{K}}_\phi$ for all β.
For $\nu g^{\mathcal{K}}_\phi$ define $X^0 = W$, $X^{\alpha+1} = g^{\mathcal{K}}_\phi(X^\alpha)$ for successor ordinals and $X^\lambda = \bigcap_{\alpha<\lambda} X^\alpha$ for limit ordinals. By analogous arguments as for $\nu g^{\mathcal{K}}_\phi$ we obtain that $X^\alpha = \nu g^{\mathcal{K}}_\phi$ for any α such that $X^\beta = X^{\beta+1}$ for all $\beta \geq \alpha$.

1.6.9 Example: Let $\mathcal{K}$ be a Kripke structure, $a \in A$ and consider the L_μ-formula $\mu X.(X \vee \langle a\rangle P)$. We have

$$\begin{aligned}
&X^0 = \emptyset\\
&X^1 = \{w \in W \colon \mathcal{K}, w \vDash X^0 \vee \langle a\rangle P\} = \{w \in W \colon v \in P \text{ for some } (w, v) \in E_a\}\\
&X^2 = \{w \in W \colon \text{a world in P is reachable in at most 2 steps from } w\}\\
&X^\alpha = \{w \in W \colon \text{a world in is P reachable in at most } \alpha \text{ steps from } w\}.
\end{aligned}$$

It follows that $\mu X.(X \vee \langle a\rangle P)$ is satisfied by precisely those $w \in W$ from which a world in P is reachable.

Let us now come to the model checking game for L_μ, which was after all our main motivation for looking at the modal μ-calculus. To obtain well-defined games, let us first make sure that there can be no confusion in the naming of variables by introducing a normal form.

1.6.10 Definition: Let $\phi \in L_\mu$. We say that ϕ is in *normal form* if every variable X in ϕ occurs at most once in a fixed point operator μX or νX and all occurrences of X are in the scope of this quantification. Every L_μ-formula can be translated to an equivalent formula in normal form by renaming variables.

This allows us to consider a new parameter for fixed point formulas, describing how many alternations of fixed point operators occur.

1.6.11 Definition: Let $\phi \in L_\mu$ be in normal form. The *alternation depth ad* of ϕ is defined by

- $ad(P_i) = ad(\neg P_i) = ad(X) = 0$ for all $i \in I$ and variables X,
- $ad(\psi_1 \vee \psi_2) = ad(\psi_1 \wedge \psi_2) = \max\{ad(\psi_1), ad(\psi_2)\}$,
- $ad(\langle a\rangle\psi) = ad([a]\psi) = ad(\psi)$,
- $ad(\mu X.\psi) = \max\{\{1, ad(\psi)\} \cup \{ad(\nu Y.\psi') + 1\colon \nu Y.\psi' \text{ subformula of } \psi, X \text{ free in } \nu Y.\psi'\}\}$,
- $ad(\nu X.\psi) = \max\{\{1, ad(\psi)\} \cup \{ad(\mu Y.\psi') + 1\colon \mu Y.\psi' \text{ subformula of } \psi, X \text{ free in } \mu Y.\psi'\}\}$

Before we come to the model checking game for the modal μ-calculus, let us remind ourselves of the model checking game for modal logic without fixed points.

1.6.12 Definition: Let $\mathcal{K} = (W, (E_a)_{a\in A}, (P_i)_{i\in I})$ be a Kripke structure, $\phi \in$ML. We define the *model checking game* $\mathcal{G}_{\mathrm{ML}}(\mathcal{K}, \phi)$ in the following way.
The players 0 and 1 play on positions of the form (ψ, v), where $v \in W$ and ψ is a subformula of ϕ. The moves are the following.

- Player 0 moves from $(\psi_1 \vee \psi_2, v)$ to (ψ_1, v) or to (ψ_2, v).
- Player 1 moves from $(\psi_1 \wedge \psi_2, v)$ to (ψ_1, v) or to (ψ_2, v).
- Player 0 moves from $(\langle a\rangle\psi, v)$ to (ψ, w) for some $(v, w) \in E_a$.
- Player 1 moves from $([a]\psi, v)$ to (ψ, w) for some $(v, w) \in E_a$.

The terminal positions are positions of the form $((\neg)P_i, v)$ for some $v \in W$. Player 0 wins if $\mathcal{K}, v \models (\neg)P_i$ and Player 1 wins otherwise. Additionally, a player who cannot move when it is her turn loses.

1.6.13 Proposition: Let $\mathcal{K} = (W, (E_a)_{a\in A}, (P_i)_{i\in I})$ be a Kripke structure, $\phi \in$ML and $v \in W$. Then $\mathcal{K}, v \vDash \phi$ if and only if Player 0 has a winning strategy in $\mathcal{G}_{\mathrm{ML}}(\mathcal{K}, v)$ from (ϕ, v).

Now let us extend the model checking for ML to model checking for L_μ. To do so, we need to introduce a value function on the game graph.

1.6.14 Definition: Let $\mathcal{K} = (W, (E_a)_{a\in A}, (P_i)_{i\in I})$ be a Kripke structure, $\phi \in \mathrm{L}_\mu$ in normal form. We define the game $\mathcal{G}(\mathcal{K}, \phi)$ as the game for ML with some additional moves and a value function:
The players are again 0 and 1 and they play on positions of the form (ψ, v), where $v \in W$ and $\psi \in \mathrm{L}_\mu$ is a subformula of ϕ. The moves are those from $\mathcal{G}_{\mathrm{ML}}$ with some additions.

- Player 0 moves from $(\psi_1 \vee \psi_2, v)$ to (ψ_1, v) or to (ψ_2, v).
- Player 1 moves from $(\psi_1 \wedge \psi_2, v)$ to (ψ_1, v) or to (ψ_2, v).
- Player 0 moves from $(\langle a\rangle\psi, v)$ to (ψ, w) for some $(v, w) \in E_a$.
- Player 1 moves from $([a]\psi, v)$ to (ψ, w) for some $(v, w) \in E_a$.
- From $(\mu X.\psi, v)$ and from $(\nu X.\psi, v)$ the game moves to (ψ, v) (deterministically).
- From (X, v) the game moves to $(\psi(X), v)$ (deterministically).

Let V be the set of nodes in $\mathcal{G}(\mathcal{K}, \phi)$. We define a value function $\mathrm{val}\colon V \to \mathbb{N}$ such that

- $\mathrm{val}(X, v)$ is even if X is a ν-variable,
- $\mathrm{val}(X, v)$ is odd if X is a μ-variable,
- $\mathrm{val}(X, v) \geq \mathrm{val}(Y, w)$ if X appears as a free variable in $\mu Y.\psi(X, Y)$ or $\nu Y.\psi(X, Y)$ and
- $\mathrm{val}(\psi) = 1$, if ψ has any other form.

Positions of the form $((\neg)P_i, v)$ for some $v \in W$ are again terminal and the play is won by Player 0 if $\mathcal{K}, v \vDash (\neg)P_i$ and by Player 1 otherwise.
If $\pi = v_1v_2v_3\cdots$ is an infinite play in $\mathcal{G}(\mathcal{K}, \phi)$ then Player 0 wins π if the largest value occurring infinitely often in π is even and Player 1 wins if it is odd. This makes $\mathcal{G}(\mathcal{K}, v)$ a parity game.

We can make the definition of the value function constructive in the following way.

1.6.15 Remark: Let m denote the alternation depth of fixed points in ϕ. The value function can be easily defined by defining $\mathrm{val}(X, v) \in \{m, m+1\}$ for all $v \in W$, depending on whether m is even and wether X appears in $\nu X.\psi$ or $\mu X.\psi$, for the outermost fixed point variables, meaning the ones such that Y does not occur freely in $\nu X.\psi$ or $\mu X.\psi$ for any fixed point variable Y.
Now let the values of fixed point variables $X_1, X_2, \ldots, X_{k-1}$ be already defined. Let X_k be

a fixed point variable such that only $X_1, \dots, X_{k-1}$ occur freely in $\nu X_k.\psi$ or $\mu X_k.\psi$. Define $\mathrm{val}(X_k, v)$ as the maximal $k \leq \min\{\mathrm{val}(X_i, v) \colon 1 \leq i < k\}$ such that the parity of k satisfies the previous definition for all $v \in W$.
For all subformulas ψ not of the form X for some fixed point variable X define $\mathrm{val}(\psi, v) = 1$ for all $v \in W$.

All that remains to be shown is that the parity game we just defined is actually the correct model checking game. First we introduce so called *unfoldings*.

1.6.16 Definition: Let $\mathcal{G} = (V, V_0, V_1, E, \mathrm{val})$ be a parity game such that each $v \in V$ with $\mathrm{val}(V)$ maximal has a unique successor $s(v)$ in $\mathcal{G}$. Without loss of generality let the largest value be odd. Otherwise just switch the players. Define $T = \{v \in V \colon \mathrm{val}(v) \text{ maximal}\}$ and define a new game $\mathcal{G}' = (V, V_0, V_1, E', \mathrm{val})$ with $E' = E \backslash (T \times V)$. Then define for each ordinal α a game $\mathcal{G}^\alpha$ as follows:
For each α define a partition (T_0^α, T_1^α) of T inductively:

- $T_1^0 = T$,
- $T_1^{\alpha+1} = \{v \in T \colon s(v) \in W_1^\alpha\}$ for α ordinal,
- $T_1^\lambda = \bigcup_{\alpha<\lambda} T_1^\alpha$ for λ limit ordinal,
- $T_1^\alpha = T \backslash T_1^\alpha$ for any ordinal α.

Player 1 wins at positions in T_1^α in $\mathcal{G}^\alpha$ and Player 0 wins at positions T_0^α in $\mathcal{G}^\alpha$. Otherwise, $\mathcal{G}^\alpha$ is identical to $\mathcal{G}'$.
This gives us $W_1^0 \supseteq W_1^1 \supseteq W_1^2 \supseteq \dots \supseteq W_1^\alpha \supseteq W_1^{\alpha+1} \supseteq \dots$ and $W_0^0 \subseteq W_0^1 \subseteq W_0^2 \subseteq \dots \subseteq W_0^\alpha \subseteq W_0^{\alpha+1} \subseteq \dots$. Hence we have $W_0^\alpha = W_0^{\alpha+1}$ and $W_1^\alpha = W_1^{\alpha+1}$ for some $\alpha \leq |V|$. Call these sets W_0^∞ and W_1^∞.

1.6.17 Lemma: In the above context we have $W_0 = W_0^\infty$ and $W_1 = W_1^\infty$.

Proof. Again suppose the largest value is odd. Let $W_1^\infty = W_1^\alpha$ with f^α a positional winning strategy for Player 1 from W_1^α in $\mathcal{G}^\alpha$. We define a strategy f for Player 1 in $\mathcal{G}$ as follows:

$$f \colon V_1 \to V, \; v \mapsto \begin{cases} f^\alpha(v), & v \in V_1 \backslash T \\ s(v), & v \in V_1 \cap T. \end{cases}$$

Any play consistent with f never leaves W_1^∞, once it is in it, since f^α is a winning strategy and all $s(v)$ that are visited are in $W_1^\alpha = W_1^\infty$. Thus, Player 1 wins any play π in $\mathcal{G}$ consistent with f: Either there are only finitely many positions of π in T, in which case the play will become entirely consistent with f^α from some point onwards, or T is visited infinitely often, but then the largest value that occurs infinitely often in π is the largest value overall, which is odd, so

Player 1 wins.
On the other hand, let $v \in W_0^\infty$. Define $\rho(v) = \min\{\beta\colon v \in W_0^\beta\}$ and let g^β be a positional winning strategy for Player 0 on W_0^β in $\mathcal{G}^\beta$. Define a strategy for Player 0 in $\mathcal{G}$ as follows:

$$g\colon V_0 \to V,\ v \mapsto \begin{cases} g^{\rho(v)}(v), & v \in (W_0^\infty \backslash T) \cap V_0 \\ s(v), & v \in T \cap V_0 \\ \text{arbitrary}, & \text{otherwise.} \end{cases}$$

Let $\pi = v_1 v_2 v_3 \cdots$ be a play in $\mathcal{G}$ that is consistent with g and $v_0 \in W_0^\infty$. If $v_i \in W_0^\infty$ it follows that $v_{i+1} \in W_0^\infty$ for all i. Additionally, we have $\rho(v_{i+1}) \le \rho(v_i)$ for all i and if $v_i \in T$, then $\rho(v_{i+1}) < \rho(v_i)$.
Since there does not exist an infinite descending chain of ordinals, there exists a β satisfying $\rho(v_i) = \rho(v_k) = \beta$ for all $i \ge k$, which means that from some point on, π has to be consistent with g^β and thus winning for Player 0.
Since $V = W_0^\infty \cup W_1^\infty$, this proves the statement. □

1.6.18 Theorem: Let $\mathcal{K} = (W, (E_a)_{a\in A}, (P_i)_{i\in I})$ be a Kripke structure, $\phi \in \mathrm{L}_\mu$ in normal form. Player 0 has a winning strategy for the parity game $\mathcal{G}(\mathcal{K}, \phi)$ from (ϕ, v) if and only if $\mathcal{K}, v \vDash \phi$.

Proof. The proof goes by induction over the form of ϕ. It is easily checked that the statement holds for all ϕ of the form P_i, $\neg P_i$, $\psi_1 \wedge \psi_2$, $\psi_1 \vee \psi_2$, $\langle a \rangle \psi_1$ and $[a]\psi_1$, $a \in A$, $i \in I$, if it holds for ψ_1 and ψ_2. So suppose that the statement holds for ψ and let $\phi = \nu X.\psi$. The positions (X, v) for $v \in W$ have the maximal value in the parity game and they have the uniquely determined successor (ψ, v). Thus, Lemma 1.6.17 can be applied. Consider the fixed point induction defined earlier in the section: $X^0 \subseteq X^1 \subseteq X^2 \subseteq \cdots \subseteq X^\alpha \subseteq X^{\alpha+1} \subseteq \ldots$. Then for $v \in W$ we have

$$\begin{aligned} (\mathcal{K}, v) \vDash \phi \quad &\Leftrightarrow \quad v \in \nu g_\phi^{\mathcal{K}} \\ &\Leftrightarrow \quad v \in X^\alpha \text{ for all } \alpha \\ &\Leftrightarrow \quad v \in X^{\alpha+1} \text{ for all } \alpha \\ &\Leftrightarrow \quad (\mathcal{K}, v, X^\alpha) \vDash \psi \text{ for all } \alpha. \end{aligned}$$

By induction hypothesis Player 0 has a winning strategy from (ψ, v) in $\mathcal{G}(\mathcal{K}, X, \psi)$. By Lemma 1.6.17 Player 0 has a winning strategy for $\mathcal{G}^\alpha(\mathcal{K}, \phi)$ if and only if Player 0 has a winning strategy in $\mathcal{G}(\mathcal{K}, \phi)$ from (ψ, v). Since the only successor of (ϕ, v) is (ψ, v) this is the case if and only if Player 0 has a winning strategy in $\mathcal{G}(\mathcal{K}, \phi)$ from (ϕ, v).
We claim that $v \in X^\alpha$ if and only if $(X, v) \in T_0^\alpha$ and prove it by induction over α:
If $\alpha = 0$ we have $X^0 = W$ and $T_0^0 = T = \{(Y, w)\colon w \in W\}$.
Now suppose the statement is true for β. We have $v \in X^{\beta+1}$ if and only if $(\mathcal{K}, v, X^\beta) \vDash \psi$, which holds if Player 0 wins $\mathcal{G}((\mathcal{K}, X^\beta), \psi)$ from (ψ, v). But by induction hypothesis this is the case if and only if Player 0 wins $\mathcal{G}^\beta(\mathcal{K}, \phi)$ from (ψ, v), which is the case if and only if $(X, v) \in T_0^{\beta+1}$.

If we have a limit ordinal λ, it holds that $v \in X^{\lambda}$ if and only if $v \in X^{\beta}$ for all $\beta < \lambda$. By induction hypothesis this is the case if and only if $(X, v) \in T_0^{\beta}$ for all $\beta < \lambda$, so $(X, v) \in T_0^{\lambda}$.
We have that Player 0 wins $\mathcal{G}((\mathcal{K}, X), \psi)$ from (X, v) if and only if $v \in X$ and Player 0 wins $\mathcal{G}^{\alpha}(\mathcal{K}, \phi)$ from (X, v) if and only if $(X, v) \in T_0^{\alpha}$. Hence, Player 0 wins $\mathcal{G}((\mathcal{K}, X), \psi)$ from (X, v) if and only if Player 0 wins $\mathcal{G}^{\alpha}(\mathcal{K}, \phi)$ from (X, v). It follows that Player 0 wins $\mathcal{G}(\mathcal{K}, \phi)$ form (ϕ, v).
Analogously, Player 1 wins $\mathcal{G}(\mathcal{K}, \mu X.\psi)$ from $(\mu X.\psi, v)$ if and only if $\mathcal{K}, v \vDash \mu X.\psi$. $\square$

Chapter 2

Two Quasi-Polynomial Time Algorithms for Solving Parity Games

In this chapter we will get to know two new methods for solving parity games. Both of them allow deciding the winner of a parity game in quasipolynomial time, which is a better time bound than any previous results for parity games were able to achieve.
The first method was presented by Calude et al. in their paper *Deciding Parity Games in Quasipolynomial Time* [Cal17]. Here, we will call it the *succinct counting* method, for reasons which will become clear when we discuss it in detail in the first section of this chapter. The second method was published a few months later by Jurdziński and Lazić in their paper *Succinct progress measures for solving parity games* [JL17]. As the title suggests, it uses progress measures in order to decide the winner of a parity game. This approach was already taken by Jurdziński in an earlier paper called *Small Progress Measures for solving Parity games*[Jur00], but the introduction of a succinct way for tree coding in the more recent paper allowed for an improvement in the runtime of the algorithm devised in [JL17] to quasipolynomial time.

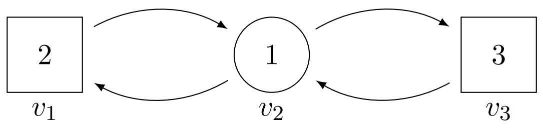
2.1 Succinct Counting

The algorithm presented in this section, which I will call the *succinct counting* method, was developed by Calude, Jain, Khoussainov, Li and Stephan in their 2017 paper *Deciding Parity Games in Quasipolynomial Time* [Cal17]. The idea is to consider a play π of the parity game in question and decide the winner of π with an alternating algorithm that uses a succinct way of counting needing only polylogarithmic space. Then one can use a reachability game to solve the actual parity game.

2.1.1 The Alternating Polylogarithmic Space Algorithm

In this subsection we will see the alternating polylogarithmic space algorithm which constitutes the main advancement made by Calude et al. in order to solve parity games in faster time. The algorithm is based on the following ideas: Since we can assume that the players follow a positional winning strategy, there will certainly be a loop in the play π that we are considering after $n + 1$ moves. We can therefore maintain a *winning statistics* for one of the players, say Player 0, that builds up favourable' sequences (we will see what is meant by that shortly) whose lengths are powers of two until reaching a length longer than n. If the latter is achieved, we will know that Player 0 has won the play π. Let us illustrate this in an example:

2.1.1 Example: Consider the parity game from Example 1.3.3 with $V_0 = \{v_2\}$, $V_1 = \{v_1, v_3\}$, $\text{val}(v_1) = 2$, $\text{val}(v_2) = 1$ and $\text{val}(v_3) = 3$:



Let $\pi = v_3\ v_2\ v_1\ v_2\ v_1\ v_2\ v_1\ v_2\ v_1\ v_2\ v_1\ v_2\ v_1\ v_2\ v_1\ v_2\ v_1$ ... be the play we consider. Note, that both players' strategies are positional, i.e. memoryless. Let $\text{val}(\pi) = 3\ 1\ 2\ 1\ 2\ 1\ 2\ 1\ 2\ 1\ 2\ 1\ 2\ 1\ 2\ 1\ 2$ ... be the sequence of values of the nodes in π. The algorithm we will get to know in this section will run through $\text{val}(\pi)$ from left to right and collect and build up sequences that are 'favourable' to Player 0:

Position	s_2	s_1	s_0
v_3			
v_2			
v_1			2
v_2			2
v_1		2 2	
v_2		2 2	
v_1		2 2	2
v_2		2 2	2
v_1	2 2 2 2		
...			

We will se shortly, how these sequences are chosen and built up. But note, that their lengths are powers of two and that sequences of lengths $2^0, 2^1, 2^2, \dots, 2^k$ together with one additional element are connected to form a new sequence of length 2^{k+1}.

Now let us see how it works exactly:
Let $\mathcal{G}$ be a parity game with n nodes and m values and let π be a play in $\mathcal{G}$. The algorithm tracks sequences $s_0, s_1, \ldots, s_{\lceil\log(n)\rceil+1}$ of positions in the play with $|s_i| = 2^i$ for all i with the following properties:

1. The positions $a_1, \ldots, a_{2^i}$ of π in a sequence s_i are in order, i.e. a_j occurs in π before a_{j+1} for all $j < 2^i$, but not necessarily consecutive.
2. If sequences $s_i = (a_1, \ldots, a_{2^i})$ and $s_j = (\hat{a}_1, \ldots, \hat{a}_{2^j})$, $i < j$, are tracked, then $\hat{a}_{2^j}$ appears before a_1 in π.
3. For all $1 \leq k \leq 2^i - 1$ the largest value of a node in π occurring between the nodes a_k and a_{k+1} of a sequence s_i (including a_k and a_{k+1}) is of even parity.

To see why we call sequences of this form favourable', consider the following remark.

2.1.2 Remark: If there exists a sequence as described above of length $2^{\lceil\log(n)\rceil+1} \geq n+1$, then Player 0 wins the parity game.

Proof. If we have a sequence $s_{\lceil\log(n)\rceil+1} = (a_1, \ldots, a_{2^{\lceil\log(n)\rceil+1}})$ of more than n pairwise distinct points in the play π appearing in order, then there have to be two points with the same corresponding nodes and since we are assuming that both players follow a positional winning strategy, the sequence has to contain a loop. Since for all $1 \leq k \leq 2^{\lceil\log(n)\rceil+1} - 1$ the largest value of a node in π occurring between a_k and a_{k+1} is even, the largest value occurring in the loop is even. The values occurring infinitely often in π are precisely the nodes occurring in a loop in π and therefore Player 0 wins the parity game. $\square$

The reader might have noticed that storing a sequence of values which has length $2^{\lceil\log(n)\rceil+1} \geq n+1$ takes more than polylogarithmic space. However, the succinct counting method of Calude et al. avoids this problem by only ever storing for each sequence $s_i = (a_1, \ldots, a_{2^i})$ the largest value b_i that occurred since a_{2^i} in the play (including a_{2^i}).
We will now see the update rules which allow the algorithm to start with short sequences and, when possible, put together sequences of lengths 1, 2, 4, ..., 2^{i-1} with one new node from π to form a sequence of length 2^i that satisfies the above criteria, all while only storing a single value per sequence, as mentioned.
The algorithm tracks the values $b_0, \ldots, b_{\lceil\log(n)\rceil+1} \in \{0, \ldots, m\}$ which are all initially set to 0. Now let b be the value of the current node in the play.

1. If b is even or $b > b_i > 0$ for some i, then one selects the largest i such that

   a) either b is even and b_i is odd or 0 but all b_j with $j < i$ are even and non-zero

   b) or $0 < b_i < b$

   and then one updates $b_i = b$ and $b_j = 0$ for all $j < i$.

2. If $b_{\lceil \log(n) \rceil+1} > 0$, Player 0 is declared the winner.

If both a) and b) are applicable according to the rules, we use a).

First we note an important property of any tuple $(b_0, \ldots, b_{\lceil \log(n) \rceil+1})$ obtained by these update rules.

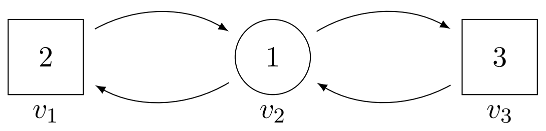
2.1.3 Remark: It follows from the update rules above, that if $i < j$ and $b_i, b_j > 0$, then $b_i \leq b_j$.

Proof. Let $i < j$ and $b_i, b_j > 0$. Since only one b_k is updated from 0 to a nonzero value at a time, there has to have been a point at which b_i was updated from 0 to a nonzero value while b_j was already nonzero. The other way around is not relevant because whenever a b_k is updated to $b > 0$, all $b_{k'}$ with $k' < k$ are updated to zero. Let b be the value of the node at that point in π. If b were larger than b_j, than the largest i such that either b is even and b_i is odd or 0 but all b_j with $j < i$ are even and non-zero or $0 < b_i < b$ would be at least j, since $0 < b_j < b$ by assumption. Thus, b_i would not be updated at this point. It follows that $b \leq b_j$ and since b_i is updated to b, $b_i \leq b_j$ at any point where they are both nonzero and were not both nonzero before.
If $b_i, b_j > 0$ was already the case before and an update changed the value of one of them to another $b > 0$, then either b_j is updated, which sets b_i to 0, or b_i is updated to $b > 0$ which again means $b_j \geq b$ for the same reason as above. □

2.1.4 Definition: Let π be a play of a parity game $\mathcal{G}$. We call the tuple $(b_0, \ldots, b_{\lceil \log(n) \rceil+1})$ generated by the algorithm at each step of π the *winning statistics* of Player 0. We say that her winning statistics *matures* if $b_{\lceil \log(n) \rceil+1} > 0$.

Now that we have the full update rules, let us consider two examples. First, let us take a new look at Example 2.1.1.

2.1.5 Example: Again, let $V_0 = \{v_2\}$, $V_1 = \{v_1, v_3\}$, $\text{val}(v_1) = 2$, $\text{val}(v_2) = 1$ and $\text{val}(v_3) = 3$:



Let $\pi = v_3\ v_2\ v_1\ v_2\ v_1\ v_2\ v_1\ v_2\ v_1\ v_2\ v_1\ v_2\ v_1\ v_2\ v_1\ v_2\ v_1$ ... be the play we consider. This time, let us see how the algorithm tracks the b_i. The s_i are still added for convenience, but they are not tracked by the actual algorithm.

Position	b_2	b_1	b_0	s_2	s_1	s_0
v_3	0	0	0			
v_2	0	0	0			
v_1	0	0	2			2
v_2	0	0	2			2
v_1	0	2	0		2 2	
v_2	0	2	0		2 2	
v_1	0	2	2		2 2	2
v_2	0	2	2		2 2	2
v_1	2	0	0	2 2 2 2		
...						

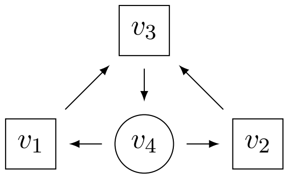
Of course this example is very simple, so let us consider an a little more complex one.

2.1.6 Example: The following table for the b_i in the winning statistics for Player 0 is the result of analysing the play and game graph on the next page.

Position	b_3	b_2	b_1	b_0
v_4	0	0	0	4
v_2	0	0	2	0
v_3	0	0	3	0
v_4	0	0	4	0
v_2	0	0	4	2
v_3	0	0	4	3
v_4	0	0	4	4
v_2	0	2	0	0
v_3	0	3	0	0
v_4	0	4	0	0
v_2	0	4	0	2
v_3	0	4	0	3
v_4	0	4	0	4
v_2	0	4	2	0
v_3	0	4	3	0
v_4	0	4	4	0
v_2	0	4	4	2
v_3	0	4	4	3
v_4	0	4	4	4
v_2	2	0	0	0

The play and game graph are:

$$\pi = v_4\ v_2\ v_3\ v_4\ v_2\ v_3\ v_4\ v_2\ v_3\ v_4\ v_2\ v_3\ v_4\ v_2\ v_3\ v_4\ v_2\ v_3\ v_4\ v_2\ v_3\ v_4\ \ldots$$



Again, $\mathrm{val}(v_i) = i$. Player 0 moves at node v_2 and Player 1 at the other nodes.

Now that we have an idea of how the algorithm works, let us see that it is actually correct.

2.1.7 Proposition: If Player 0 is declared the winner by the rules given above, then Player 0 is the winner of the play π of the parity game.

Proof. To prove this it suffices to show that if Player 0's winning statistics matures then there exists a sequence $s_{\lceil\log(n)\rceil+1} = (a_1, \ldots, a_{2^{\lceil\log(n)\rceil+1}})$ such that the a_i appear in order in π and for all $1 \leq k \leq 2^{\lceil\log(n)\rceil+1} - 1$ the largest value of a node in π occurring between the nodes a_k and a_{k+1} (including a_k and a_{k+1}) is of even parity. To prove this, we prove by induction, that if $b_i > 0$, then there exists a sequence $s_i = (a_1, \ldots, a_{2^i})$ such that the a_k appear in order in π, for all $1 \leq k \leq 2^i - 1$ the largest value of a node in π occurring between the nodes a_k and a_{k+1} (including a_k and a_{k+1}) is of even parity and b_i is the largest value that occurred in π since (including) a_{2^i}. Additionally, if $b_i, b_j > 0$, $i < j$, then the first entry of the sequence s_i occurs after the last entry of the sequence s_j.
For $i = 0$, define $s_0 = (a)$ where a is the point in π at which b_1 is updated according to rule a). When b_0 is updated according to rule b), keep s_0 the same as before. Assume that there is a value $\mathrm{val}(v) > b_0 > 0$ for some v that occurred in π after a. If $\mathrm{val}(v)$ is even, b_0 and therefore s_0 would have been updated according to rule a). If $\mathrm{val}(v)$ is odd, b_0 would have been updated to $\mathrm{val}(v)$ or 0 according to rule b).
Let $i > 1$ and consider a point a in the algorithm at which b_i is updated from 0 to a value $b = \mathrm{val}(a) > 0$. This occurs only if at that point $b_j > 0$ is even for all $j < i$ and b is even. By induction hypothesis, there exist sequences $s_0 = (a_1^1), \ldots, s_{i-1} = (a_1^{i-1}, \ldots, a_{2^{i-1}}^{i-1})$ with the above criteria. Define $s_i = (a_1^1, \ldots, a_1^{i-1}, \ldots, a_{2^{i-1}}^{i-1}, a)$. Since all b_j for $j < i$ are even, the largest values that occurred since the end of each of the respective sequences is even. Additionally, none of the b_j were updated after the sequences s_k, $k < j$ as they appear here were tracked, since otherwise b_k would have been set to 0. Hence for each $1 \leq k \leq i-2$ the largest value occurring in π between $a_{2^k}^k$ and a_1^{k+1} is even.
Assume that there is a point v between $a_{2^{i-1}}^{i-1}$ and a in π with $\mathrm{val}(v) > \max\{b, b_{i-1}\}$. Without loss of generality let $\mathrm{val}(v)$ be maximal with these properties. If $\mathrm{val}(v)$ is even, b_i would have been updated at v. If $\mathrm{val}(v)$ is odd, b_{i-1} would have been updated at v and b_i would not have

been updated at a since b_{i-1} would have been odd.
Since the sequences $s_1, \dots, s_{i-1}$ satisfy that if $b_k, b_j > 0$, $k < j$, then the first entry of the sequence s_k occurs after the last entry of the sequence s_j, this continues to hold by construction.
Lastly, assume that b_i was already nonzero and is updated again at a point a' in π. If this happens according to rule b), keep s_i the same. If this happens according to rule a), change the last entry in s_i to a'. In the first case assume that b_i is not the largest value occurring in π since the end of s_i. Then there would be a point c after a with $\mathrm{val}(c) > b_i$. Assume that c was a point after a such that $\mathrm{val}(c) > b_i$ is maximal. Then b_i would have been updated at c according to rule b) and after c only according to rule a), if at all, so c would not appear after the end of s_i. In the second case assume that the largest value between $a^{i-1}_{2^{i-1}}$ and a' in π is odd. Then at point a', b_{i-1} would be odd and b_i would not be updated.
Hence, we can apply Remark 2.1.2 and therefore prove the statement. □

Now for the opposite direction.

2.1.8 Proposition: If Player 0 wins the play π of the parity game, she is declared the winner by the algorithm using the update rules above.

Proof. Suppose that Player 0 wins the play π. Then the largest value occurring infinitely often in π is even. Say this value is c. We will prove the statement by introducing a certain counter to track the occurrences of c in relation to the occurrences of other values in the play. First, number the steps of the algorithm, meaning the moments after each update, starting from the first update of the winning statistics in π. For $0 \leq k \leq \lceil \log(n) \rceil + 1$ and $t \in \mathbb{N}$ we define $b_k(t)$ to mean the value of b_k at the t-th step of the algorithm. Define

$$B_c(t) = \{0 \leq k \leq \lceil \log(n) \rceil + 1 \colon b_k(t) \text{ is of even parity and } b_k(t) \geq c\}$$

and

$$\mathrm{count}(c,t) = \sum_{k \in B_c(t)} 2^k.$$

Claim: Let $t < t'$ be steps at which a move to a node v with $\mathrm{val}(v) = c$ was made and there is no move at any step $t < t'' < t'$ to a node with a value larger than or equal to c. Then $\mathrm{count}(c,t) < \mathrm{count}(c,t')$.
To prove this, suppose that $t < t'$ are as described in the claim and consider the largest $0 \leq i \leq \lceil \log(n) \rceil + 1$ such that b_i is updated at some $t < t'' \leq t'$ and let t'' be maximal with that property. If $b_i(t)$ were even and larger than or equal to c, b_i would not have been updated between t and t' according to either rule a) or b), so $b_i(t) < c$ or $b_i(t)$ is even. Additionally, the maximal value b_i can be updated to between t and t' is c and since i is maximal such that b_i is updated, b_i is not updated to zero, so $0 < b_i(t'') \leq c$.

Assume that $b_i(t'') < c$. Then $t'' \neq t'$. But since at step t' we move to a node with value c we have $0 < b_i(t'') < c$ and b_i is updated to c or to 0 at t' which is a contradiction to the maximality of t''. It follows that $b_i(t'') = b_i(t') = c$. Therefore $i \in B_c(t') \backslash B_c(t)$ and since b_j is not updated between t and t' for $j > i$

$$\text{count}(c, t') \geq \sum_{j \in B_c(t),\ j > i} 2^j + 2^i > \sum_{j \in B_c(t),\ j > i} 2^j + \sum_{k \in B_c(t),\ k < i} 2^k = \text{count}(c.t).$$

This proves the claim.
Since c is the maximal value in π occurring infinitely often, there will be a point x in π after which no value larger than c occurs but c still occurs infinitely often. Thus, the conditions of the claim are met infinitely many times after x and therefore we can always find a step t such that count(c, t) is arbitrarily large. So at some point we will have $b_i(t) \geq c$ and $2^i > 2n$, so the algorithm will eventually terminate and declare Player 0 the winner. □

Let us sum up this subsection in one theorem.

2.1.9 Theorem: It is possible to decide the winner of a parity game $\mathcal{G}$ from starting node s in polylogarithmic space.

Proof. The previous two propositions have shown that the algorithm can decide the winner of the parity game. To store the winning statistics it needs space $\mathcal{O}(\log(n) \cdot \log(m))$, namely $\lceil \log(n) \rceil + 2$ slots for $b_i \in \{1, \dots, m\}$ which can be represented with $\lceil \log(m) \rceil$ bits. □

It was shown by Chandra, Kozen and Stockmeyer [CKS81] that an alternating polylogarithmic space algorithm can be translated into a deterministic algorithm with runtime $\mathcal{O}(n^{c \cdot \log(m)})$ for some constant c, i.e. quasipolynomial time. In the next two subsections, still following [Cal17], we will show how to achieve this runtime in this particular case and what complexity we get exactly.

2.1.2 The Reachability Game

In order to obtain an algorithm for solving parity games, it is not enough to consider a single play π. It does suffice however, to check whether Player 0 can or cannot choose her moves in such a way that a play occurs that is found to be winning for her by the algorithm in the previous subsection. This can be done via a reachability game.

2.1.10 Definition: Given a parity game $\mathcal{G}$ we define the reachability game $\tilde{\mathcal{G}}$ in the following way: The nodes are of the form $(a, \tilde{b})$ with

- a a node of the parity game and

- $\tilde{b}$ the winning statistics for Player 0.

The starting node is $(s, (0, \dots, 0))$ with s the starting node of the parity game.
Player 0 can move from $(a, \tilde{b})$ to $(a', \tilde{b}')$ if $a \in V_0$, if she can move from a to a' in the parity game and this causes her winning statistics to update from $\tilde{b}$ to $\tilde{b}'$. Player 1 can move from $(a, \tilde{b})$ to $(a', \tilde{b}')$ if $a \in V_1$, if he can move from a to a' in the parity game and this causes Player 0's winning statistics to update from $\tilde{b}$ to $\tilde{b}'$.
Player 0 is declared the winner of a play $\tilde{\pi}$ in the reachability game, if a node with a matured winning statistic is reached.

2.1.11 Theorem: Player 0 wins the parity game $\mathcal{G}$ from starting node s if and only if she wins the reachability game $\tilde{\mathcal{G}}$ defined above from starting node $(s, (0, \dots, 0))$.

Proof. First suppose that Player 0 wins $\mathcal{G}$ from s. Then she has a positional winning strategy $f \colon V_0 \to V$ from s in $\mathcal{G}$. Since she can move from each $(a, \tilde{b})$ for each $(a, a') \in E$ to $(a', \tilde{b}')$ with $\tilde{b}'$ the winning statistics updated accordingly, this gives us a strategy on $\tilde{\mathcal{G}}$: From $(a, \tilde{b})$ move to $(f(a), \tilde{b}')$, with $\tilde{b}'$ the accordingly updated winning statistics.
Let $\tilde{\pi}$ be a play in $\tilde{\mathcal{G}}$ played according to this strategy. This gives us a play π played according to f in $\mathcal{G}$ by considering the first component of each tuple in $\tilde{\pi}$. By Proposition 2.1.8, Player 0's winning statistics matures on π. Therefore at some point in $\tilde{\pi}$ a point $(a, \tilde{b})$ will be reached, where $\tilde{b}$ is a matured winning statistics.
Now suppose that Player 1 wins $\mathcal{G}$ from s. Then he has a positional winning strategy $g \colon V_1 \to V$ from s in $\mathcal{G}$. Again this gives us a strategy on $\tilde{\mathcal{G}}$ the same way as for Player 0. Consider a play π consistent with g and the corresponding play $\tilde{\pi}$ in $\tilde{\mathcal{G}}$. Since Player 0 cannot win π by assumption, it follows from Proposition 2.1.7 that her winning statistics does not mature. Hence, Player 1 wins $\tilde{\pi}$. □

2.1.3 Complexity

What remains to be seen is that this reachability game actually yields quasipolynomial time complexity. But before we do that, let us show another time bound which was also unknown to hold for parity games before the paper by Calude et al., namely that they can be solved in FPT.

Let $\mathcal{G}$ be a parity game and $\mathcal{G}'$ the corresponding reachability game. Let Q be the set of nodes in the reachability game and F the set of edges. Then the game can be decided in time $O(|Q|+|F|)$ (see for instance [KI16]). In this case, the number of edges is bounded by $|Q| \cdot n$, since every node in the reachability game has at most n successors. Thus, the reachability game can be solved in time $\mathcal{O}(|Q| \cdot n)$. What needs to be determined is the number of elements in Q. Let $(b_0, \ldots, b_{\lceil \log(n) \rceil + 1})$ be a winning statistics. Define $(b'_0, \ldots, b'_{\lceil \log(n) \rceil + 1})$ by

- $b'_0 = \max\{1, b_0\}$
- $b'_{k+1} = \max\{b'_k, b_{k+1}\}$

Why we introduce these b'_i will become obvious with the following remarks.

2.1.12 Remark: We have $b'_k \neq b_k$ if and only if $b_k = 0$. Thus, we can use $\lceil \log(n) \rceil + 2$ additional bits to reconstruct the b_k from the b'_k..

2.1.13 Remark: A winning statistics $(b_0, \ldots, b_{\lceil \log(n) \rceil + 1})$ can be uniquely represented by the set $\{b'_k + k \colon 0 \leq k \leq \lceil \log(n) \rceil + 1\} \subseteq \{1, \ldots, m + \lceil \log(n) \rceil + 1\}$.

Combining these results leads to the following lemma.

2.1.14 Lemma: There are

- $m + 2\lceil \log(n) \rceil + 3$ bits needed to represent the winning statistics and
- $\lceil \log(n) \rceil$ bits needed to represent the node of the parity game.

Proof. Follows from the previous two remarks. □

Now we can conclude, that we can decide parity games in FPT.

2.1.15 Theorem: The reachability game can be decided in time $\mathcal{O}(2^m \cdot n^4)$

Proof. Each node in Q can be represented by $m + 3\lceil \log(n) \rceil + 3$ bits. Hence

$$|Q| \leq 2^{m+3\lceil \log(n) \rceil + 3} = 2^{m+3} \cdot 2^{3\lceil \log(n) \rceil} \in \mathcal{O}(2^m \cdot n^4).$$

□

Additionally, we get a corollary which is very interesting for practical applications. In most cases, the number of values will be substantially smaller than the number of nodes in the graph, which might allow us to solve the game in polynomial time.

2.1.16 Corollary: If $m < \log(n)$, the parity game can be decided in time $O(n^5)$.

Now let us look at the complexity from a different angle, by bounding $|Q|$ differently.

2.1.17 Lemma: In $\mathcal{G}'$ there are

- $\lceil \log(n) \rceil$ bits needed to represent the node of the parity game and
- the winning statistics consists of
  - $\lceil \log(n) \rceil + 2 \leq \log(n) + 3$ numbers
  - represented by $\lceil \log(m) \rceil \leq \log(m) + 1$ bits each.

This gives us the desired quasipolynomial runtime.

2.1.18 Theorem: The reachability game can be decided in time $O(n^{\log(m)+6})$.

Proof. We have from the previous lemma and from $\log(m) \leq \log(n)$

$$\begin{aligned} |Q| &\leq 2^{(\log(n)+3)\cdot(\log(m)+1)+\log(n)+1} \\ &\leq 2^{\log(n)\cdot\log(m)+5\log(n)+4} \\ &= 2^4 \cdot n^{\log(m)+5}. \end{aligned}$$

□

2.2 Succinct Progress Measures

In this section we will get to know a second method for solving parity games in quasipolynomial time, published shortly after the Calude et al. paper [Cal17] by Jurdziński and Lazić in their paper *Succinct progress measures for solving parity games* [JL17]. Their idea is based upon an earlier paper by Jurdziński, titled *Small Progress Measures for Solving Parity Games* [Jur00], which showed that one can define a progress measure on the game graph of the parity game in order to determine its winner. We will start by presenting the results of this earlier paper, before going into the refinements made in the more recent publication.

2.2.1 The Lifting Algorithm

In this section, we will get to know progress measures for parity games as introduced by Jurdziński in the paper *Small Progress Measures for Solving Parity Games* [Jur00]. To start, let us give names to two concepts we have already come upon in the last section.

2.2.1 Definition: A *parity graph* is a tuple (V, E, val) consisting of a directed graph (V, E) and a value function $\mathrm{val}\colon V \to \{1, \dots, m\}$ for some $m \in \mathbb{N}$ and $|\,\mathrm{val}(V)| = m$.

2.2.2 Definition: Let G be a parity graph. A cycle in G is an *i-cycle* if i is the largest value of a node in the cycle. It is called an *even cycle* if i is even and an *odd cycle* if i is odd.

Progress measures are based on the idea of assigning to each node a label from an ordered set such that it is possible for Player 0 to increase this label monotonely along any play, until reaching a top element. We will begin with labels in $\mathbb{N}^m$, where m is, as usual, the number of values in the parity game, as well as the largest value. Then, following the paper this section is based on, we will introduce 'small progress measures', which are a little more succinct than arbitrary tuples in $\mathbb{N}^m$. As we will see, this does not yield quasipolynomial time. However, we will see in the next subsection, based on the more recent paper [JL17] by Jurdziński and Lazić, that it is possible to give even more succinct labels.

2.2.3 Definition: Let $\leq$ denote the lexicographical ordering from left to right on $\mathbb{N}^m$. We define $\leq_i$ for $i \in \{1, \dots, m\}$ as the lexicographical ordering on the tuple without the rightmost $i-1$ components, meaning that

$$(\alpha_m, \dots, \alpha_1) \leq_i (\beta_m, \dots, \beta_1) \quad \text{if and only if} \quad (\alpha_m, \dots, \alpha_i) \leq (\beta_m, \dots, \beta_i)$$

where $\leq$ is the lexicographical ordering on $\mathbb{N}^{m-i+1}$.

2.2.4 Remark: Note that $(\alpha_m, \dots, \alpha_1) \leq_i (\beta_m, \dots, \beta_1)$ implies that $(\alpha_m, \dots, \alpha_1) \leq_j (\beta_m, \dots, \beta_1)$ for all $j > i$.

Now let us define parity progress measures formally.

2.2.5 Definition: A function $\rho\colon V \to \mathbb{N}^m$ on a parity graph $G = (V, E, \mathrm{val})$ is called a *parity progress measure* for G if

- $\rho(v) \geq_{\mathrm{val}(v)} \rho(w)$ for all $(v, w) \in E$
- $\rho(v) >_{\mathrm{val}(v)} \rho(w)$ for all $(v, w) \in E$, $\mathrm{val}(v)$ odd.

The first question to answer is in what way a progress measure helps us to determine the winner of a parity game. Part of the answer is given by the next proposition.

2.2.6 Proposition: If there is a parity progress measure for a parity graph G, then all cycles in G are even.

Proof. Assume that G has an i-cycle $v_1 v_2 v_3 \cdots v_{k-1} v_k v_1$ where i is odd and a parity progress measure $\rho\colon V \to \mathbb{N}^m$. Without loss of generality let $\mathrm{val}(v_1) = i$ (otherwise shift the enumeration of the cycle). Then we have

$$\rho(v_1) >_i \rho(v_2) \geq_i \rho(v_3) \geq_i \cdots \geq_i \rho(v_{k-1}) \geq_i \rho(v_k) \geq_i \rho(v_1),$$

a contradiction. $\square$

As mentioned, the crucial point in achieving a good runtime for the algorithm that will be presented later in this section is to give the labels in a form as succinct as possible. The first step in this direction is made by the following definition.

2.2.7 Definition: Let G be a parity graph with values $1, \dots, m$. We define

$$M_G = \{r_m 0 r_{m-2} 0 \cdots 0 r_1 \colon r_i \in \mathbb{N} \text{ and } 0 \leq r_i \leq |\{v \in V\colon \mathrm{val}(v) = i\}| \text{ for all } i\}$$

if m is odd and

$$M_G = \{0 r_{m-1} 0 r_{m-3} 0 \cdots 0 r_1 \colon r_i \in \mathbb{N} \text{ and } 0 \leq r_i \leq |\{v \in V\colon \mathrm{val}(v) = i\}| \text{ for all } i\}$$

if m is even.
A *small progress measure* for G is a progress measure $\rho\colon V \to M_G$.

From this definition it is not yet clear, what the use of such a progress measure might be. To begin with, we do not even know if and when such progress measures exist. Let us start by clarifying this in a theorem.

2.2.8 Theorem: Let G be a parity graph. If all cycles in G are even, then there exists a small progress measure $\rho \colon V \to M_G$.

Proof. We prove the theorem by induction over the number of nodes in G. Let m be the largest value of a node in G. It is useful for the proof to add to the statement the additional condition that $\rho(v) >_{\mathrm{val}(v)} (0, \ldots, 0)$ for all $v \in V$ for which $\mathrm{val}(v)$ is odd.
If G consists only of a single vertex v, there are no edges and thus we can define a progress measure by setting $\rho(v) = (1, 0, \ldots, 0)$ if $\mathrm{val}(v) = m$ is odd or $\rho(v) = (0, \ldots, 0)$ if $\mathrm{val}(v) = m$ is even.
Now assume that the statement holds for all parity graphs with less than n nodes and let G be a parity graph with n nodes. Let m be the largest value occurring in G. Suppose first that m is even and let M denote the set of vertices in G with value m. Then by induction hypothesis we can define a progress measure $\rho'' \colon (V \backslash M) \to M_{G \upharpoonright (V \backslash M)}$ on $G \upharpoonright (V \backslash M)$, denoting the induced subgraph over $(V \backslash M)$, and extend it to a progress measure $\rho' \colon (V \backslash M) \to M_G$ by adding a 0 at the beginning of each tuple $\rho''(v)$, $v \in V \backslash M$. Defining ρ by setting $\rho(v) = \rho'(v)$ for all $v \in V \backslash M$ and $\rho(v) = (0, \ldots, 0)$ for all $v \in M$ then gives us a progress measure on G.
Now suppose that m is odd and let u be a node with $\mathrm{val}(u) = m$. We claim that we can provide a partition of V into two disjoint sets $W_1 \neq \emptyset$ and $W_2 \neq \emptyset$ such that there is no edge from a node in W_1 to a node in W_2. If u has no successors in G then we can simply define $W_1 = \{u\}$ and $W_2 = V \backslash \{u\}$. Otherwise let $U \neq \emptyset$ be the set of nodes nontrivially reachable in G from u. Define $W_1 = V \backslash U$ and $W_2 = U$. Obviously $W_2 \neq \emptyset$. But $W_1 \neq \emptyset$ as well, since $u \in W_1$: If $u \in U$, then there is a cycle in G containing u and since $\mathrm{val}(u) = m$ is the largest value occurring in G and also odd, this would be an odd cycle which is a contradiction. Hence we have proven the claim.
By induction hypothesis there exist progress measures $\rho'_1 \colon W_1 \to M_{G \upharpoonright W_1}$ and $\rho'_2 \colon W_2 \to M_{G \upharpoonright W_2}$ on $G \upharpoonright W_1$ and $G \upharpoonright W_2$, respectively. Again, by adding zeros to the beginning of the tuples if necessary, we obtain progress measures $\rho_1 \colon W_1 \to M_G$ and $\rho_2 \colon W_1 \to M_G$ on $G \upharpoonright W_1$ and $G \upharpoonright W_2$. For all $i \leq m$ we define $n'_i = |\{v \in W_1 \colon \mathrm{val}(v) = i\}|$. Since there are no edges from W_1 to W_2 in G and since $\rho_1(v) >_{\mathrm{val}(v)} (0, \ldots, 0)$ for all $v \in W_1$ with $\mathrm{val}(v)$ odd and $\rho_2(v) >_{\mathrm{val}(v)} (0, \ldots, 0)$ for all $v \in W_2$ with $\mathrm{val}(v)$ odd, we can define a parity progress measure ρ on G by setting $\rho(v) = \rho_1(v)$ for all $v \in W_1$ and $\rho(v) = \rho_2(v) + (\ldots, 0, n'_3, 0, n'_1)$ for all $v \in W_2$. □

From this and Lemma 2.2.6 we obtain that a small progress measure on a parity graph G exists if and only if all cycles in G are even. Now let us make the transition from parity graphs to parity games. Progress measures are defined in such a way that Player 0 can increase the labels along any play. The aim will be to reach a fixed point before reaching a top element which is introduced in the following definition.

2.2.9 Definition: Let $\mathcal{G} = (G, V_0, V_1)$ with $G = (V, E, \mathrm{val})$ be a parity game with parity graph G. Then we define $M_G^\top = M_G \cup \{\top\}$ and extend the lexicographical ordering on M_G to $M_G^\top$ by setting $\top$ as being strictly larger than any other element. Additionally we prescribe that $a <_i \top$ and $\top =_i \top$ for all $a \in M_G$ and $i \in \{1, \dots, m\}$, where m is, as usual, the largest value that occurs in G.
For a function $\rho\colon V \to M_G^\top$ and $v, w \in V$ we define $\mathrm{Prog}(\rho, v, w)$ as the smallest $a \in M_G^\top$ such that $a \geq_{\mathrm{val}(v)} \rho(w)$ and if $\mathrm{val}(v)$ is odd then either $a >_{\mathrm{val}(v)} \rho(w)$ or $a = \rho(w) = \top$.

This gives us the tools to define a progress measure for a parity game, rather than only a parity graph.

2.2.10 Definition: Let $\mathcal{G} = (G, V_0, V_1)$ be a parity game with parity graph $G = (V, E, \mathrm{val})$ and let $\rho\colon V \to M_G^\top$ be a function. ρ is called a *game parity progress measure* if for any $v \in V$

1. if $v \in V_0$ then $\rho(v) \geq_{\mathrm{val}(v)} \mathrm{Prog}(\rho, v, w)$ for some $w \in V$ with $(v, w) \in E$ and
2. if $v \in V_1$ then $\rho(v) \geq_{\mathrm{val}(v)} \mathrm{Prog}(\rho, v, w)$ for all $w \in V$ with $(v, w) \in E$.

2.2.11 Definition: Let $\mathcal{G}$ be a parity game and $\rho\colon V \to M_G^\top$ a game parity progress measure for $\mathcal{G}$. Then we define

$$||\rho|| = \{v \in V : \rho \neq \top\}.$$

Not only does the existence of a game parity progress measure give us a winning strategy for Player 0 for certain starting nodes, but we can even find a progress measure such that we can see the winning region for Player 0 immediately from its definition.

2.2.12 Proposition: Let $\mathcal{G}$ be a parity game with parity graph $G = (V, E)$.

1. If $\rho\colon V \to M_G^\top$ is a a game parity progress measure for $\mathcal{G}$, then Player 0 has a winning strategy from all $v \in ||\rho||$.
2. There exists a game parity progress measure $\rho\colon V \to M_G^\top$ such that

$$\rho(v) \neq \top \quad \Leftrightarrow \quad \text{player 0 has a winning strategy from } v$$

Proof. 1. Let $\rho\colon V \to M_G^\top$ be a game parity progress measure for $\mathcal{G}$ and let $v \in ||\rho||$. Consider the parity graph $G' = (V', E')$ derived from G by deleting for each $w \in V_0$ every outgoing edge $(w, u) \in E$ but one that satisfies $\rho(w) \geq_{\mathrm{val}(w)} \mathrm{Prog}(\rho, w, u)$, which exists by assumption. Then delete all nodes that are not reachable from v.
Now G' is a parity graph such that $\rho(w) \geq_{\mathrm{val}(w)} \rho(u)$ for all $w, u \in G'$, $(w, u) \in E'$, and therefore $\rho(w) < \top$ for all $w \in G'$ since $\rho(v) < \top$ and all $v \neq w \in G'$ are reachable

from v. Additionally, $\rho(w) >_{\mathrm{val}(w)} \rho(u)$ for all $w, u \in G'$ with $\mathrm{val}(w)$ odd. It follows that $\rho' \colon V' \to M_G,\ w \mapsto \rho(w)$ is a parity progress measure for G' and by Proposition 2.2.6 all cycles in G' are even. Hence, Player 0 wins any parity game with parity graph G' and since we deleted only edges and nodes from G that Player 0 can avoid, picking from each node $w \in V_0$ the one edge $(w, u) \in E'$ is a winning strategy for Player 0.

2. Let W_0 be the winning region for Player 0. Then Player 0 has a winning strategy f on all $v \in W_0$. Define a parity graph $G' = (W_0, E')$ by deleting from $G \restriction W_0$ all edges (v, w) such that $v \in V_0$ and $w \neq f(v)$.
   We claim that all cycles in G' are even. To prove this, assume that $v_0 v_1 \cdots v_k v_0$ is an odd cycle in G'. Then Player 1 can win a play π consistent with f starting from v_0 in $\mathcal{G}$ by choosing for $v_i \in V_1$ the edge (v, v_{i+1}) (or (v_k, v_0) if $i = k$). This is a contradiction to f being a winning strategy from $v_0 \in W_0$.
   Knowing that all cycles in G' are even, we can conclude using Theorem 2.2.8 that there exists a parity progress measure $\rho' \colon W_0 \to M_{G'}$. Extending this to $\rho \colon V \to M_G^{\top}$ by setting $\rho(v) = \top$ for all $v \in V \backslash W_0$ and $\rho(v) = \rho'(v)$ for all $v \in W_0$ yields the desired game parity progress measure for $\mathcal{G}$.

□

The question that remains to be answered is how to find this progress measure from which we can immediately determine the winning regions. This is done by a lifting algorithm which makes use of the Knaster-Tarski Theorem. The following lifting operator forms the core of the algorithm.

2.2.13 Definition: Let $\mathcal{G}$ be a parity game, $\rho \colon V \to M_G^{\top}$ a function and $v \in V$. Then we define $\mathrm{Lift}(\rho, v) \colon V \to M_G^{\top}$ by

$$\mathrm{Lift}(\rho, v)(u) = \begin{cases} \rho(u) & \text{if } u \neq v \\ \min_{(v,w)\in E} \mathrm{Prog}(\rho, v, w) & \text{if } u = v \in V_0 \\ \max_{(v,w)\in E} \mathrm{Prog}(\rho, v, w) & \text{if } u = v \in V_1 \end{cases} .$$

To be able to apply the Knaster-Tarski Theorem, we need an ordering on the functions the lifting operator is applied to.

2.2.14 Definition: Let $\mathcal{G}$ be a parity game. We define a partial ordering on the functions $V \to M_G^{\top}$ in the following way: Let $\rho_1 \colon V \to M_G^{\top}$ and $\rho_2 \colon V \to M_G^{\top}$. We write $\rho_1 \sqsubseteq \rho_2$ if $\rho_1(v) \leq \rho_2(v)$ for all $v \in V$ and $\rho_1 \sqsubset \rho_2$ if $\rho_1 \sqsubseteq \rho_2$ and $\rho_1 \neq \rho_2$.

What remains to be shown is that the operator is monotone with respect to the ordering given in the previous definition.

2.2.15 Proposition: Let $\mathcal{G}$ be a parity game. For all $v \in V$ the lifting operator $\mathrm{Lift}(\cdot, v)$ is $\sqsubseteq$-monotone.

Proof. Let $\rho_1, \rho_2 \colon V \to M_G^\top$ be functions, $\rho_1 \sqsubseteq \rho_2$, and $v \in V$. We need to show that $\mathrm{Lift}(\rho_1, v) \sqsubseteq \mathrm{Lift}(\rho_2, v)$. Since $\mathrm{Lift}(\rho_1, v)(u) = \rho_1(u)$ and $\mathrm{Lift}(\rho_2, v)(u) = \rho_2(u)$ for all $v \neq u$ it suffices to show that $\mathrm{Lift}(\rho_1, v)(v) \leq \mathrm{Lift}(\rho_2, v)(v)$.
Let $w \in V$ with $(v, w) \in E$. Then for $\rho \colon V \to M_G^\top$, $\mathrm{Prog}(\rho, v, w)$ is defined as the smallest $a \in M_G^\top$ such that $a \geq_{\mathrm{val}(v)} \rho(w)$ and $a >_{\mathrm{val}(v)} \rho(w)$ or $a = \rho(w) = \top$ if $\mathrm{val}(v)$ is odd. We claim that $\mathrm{Prog}(\rho_1, v, w) \leq \mathrm{Prog}(\rho_2, v, w)$. Since $\rho_1 \sqsubseteq \rho_2$ we know that $\rho_1(w) \leq \rho_2(w)$. By definition of the lexicographical ordering it follows that $\rho_1(w) \leq_i \rho_2(w)$ for all $i \in \{1, \ldots, m\}$. Hence, $\mathrm{Prog}(\rho_2, v, w) \geq_{\mathrm{val}(v)} \rho_2(w) \geq_{\mathrm{val}(v)} \rho_1(w)$ and $\mathrm{Prog}(\rho_2, v, w) > \rho_2(w) \geq_{\mathrm{val}(v)} \rho_1(w)$ or $\mathrm{Prog}(\rho_2, v, w) = \rho_2(w) = \top \geq_{\mathrm{val}(v)} \rho_1(v)$ if $\mathrm{val}(v)$ is odd. In either case, since $\mathrm{Prog}(\rho_1, v, w)$ is minimal with these properties, $\mathrm{Prog}(\rho_1, v, w) \leq \mathrm{Prog}(\rho_2, v, w)$.
By definition of the Lift-operator via Prog it follows that $\mathrm{Lift}(\rho_1, v)(v) \leq \mathrm{Lift}(\rho_2, v)(v)$ and therefore $\mathrm{Lift}(\rho_1, v) \sqsubseteq \mathrm{Lift}(\rho_2, v)$. □

The Knaster-Tarski Theorem guarantees the existence of a fixed point for the lifting operator. The following proposition shows why this is desirable.

2.2.16 Proposition: Let $\mathcal{G}$ be a parity game and $\rho \colon V \to M_G^\top$ a function. Then ρ is a game parity progress measure if and only if $\mathrm{Lift}(\rho, v) \sqsubseteq \rho(v)$ for all $v \in V$.

Proof. Let ρ be a game parity progress measure and $v \in V$. Since $\mathrm{Lift}(\rho, v)(u) = \rho(u)$ for all $u \neq v$ it suffices to show that $\mathrm{Lift}(\rho, v)(v) \leq \rho(v)$. Suppose first that $v \in V_0$. Because ρ is a game parity progress measure we know that $\rho(v) \geq_{\mathrm{val}(v)} \mathrm{Prog}(\rho, v, w)$ for some $(v, w) \in E$. Since $\mathrm{Prog}(\rho, v, w)$ is the minimal $a \in M_G^\top$ such that $a \geq_{\mathrm{val}(v)} \rho(w)$ it follows that $\mathrm{Prog}(\rho, v, w)$ only contains zeros after its $\mathrm{val}(v)$'th entry or $\mathrm{Prog}(\rho, v, w) = \top$ and therefore $\rho(v) \geq \min_{(v,w) \in E} \mathrm{Prog}(\rho, v, w) = \mathrm{Lift}(\rho, v)(v)$.
If $v \in V_1$ then $\rho(v) \geq_{\mathrm{val}(v)} \mathrm{Prog}(\rho, v, w)$ for all $(v, w) \in E$. Since $\mathrm{Prog}(\rho, v, w)$ is the minimal $a \in M_G^\top$ such that $a >_{\mathrm{val}(v)} \rho(w)$ or $a = \rho(w) = \top$ it follows that for all $(v, w) \in E$ $\mathrm{Prog}(\rho, v, w)$ only contains zeros after its $\mathrm{val}(v)$'th entry or $\rho(v) \geq_{\mathrm{val}(v)} \top$ and therefore we have $\rho(v) \geq \max_{(v,w) \in E} \mathrm{Prog}(\rho, v, w) = \mathrm{Lift}(\rho, v)(v)$ since $\rho(v) \geq_{\mathrm{val}(v)} \top$ implies $\rho(v) = \top$.
Conversely, let $\mathrm{Lift}(\rho, v) \sqsubseteq \rho(v)$ for all $v \in V$. Let a fixed node $v \in V$ be given. In particular, $\mathrm{Lift}(\rho, v) \sqsubseteq \rho$ implies that $\mathrm{Lift}(\rho, v)(v) \leq \rho(v)$. If $v \in V_0$, this means $\min_{(v,w) \in E} \mathrm{Prog}(\rho, v, w) \leq \rho(v)$ and therefore there exists an edge $(v, w) \in E$ with $\mathrm{Prog}(\rho, v, w) \leq_{\mathrm{val}(v)} \rho(v)$. If $v \in V_1$ we have $\max_{(v,w) \in E} \mathrm{Prog}(\rho, v, w) \leq \rho(v)$ and therefore $\mathrm{Prog}(\rho, v, w) \leq_{\mathrm{val}(v)} \rho(v)$ for all $w \in V$ with $(v, w) \in E$. Hence ρ satisfies the defining conditions of a game parity progress measure. □

Applying the Knaster-Tarski Theorem to the lifting operator for some parity game $\mathcal{G}$ gives us the $\sqsubseteq$-least game parity progress measure by the following simple algorithm.

2.2.17 Algorithm (ProgressMeasureLifting)**:**
Initialize $\mu\colon V \to M_G^\top,\ v \mapsto (0,\dots,0)$
while $\mu \sqsubset \mathrm{Lift}(\mu, v)$ for some $v \in V$ do $\mu := \mathrm{Lift}(\mu, v)$.
Return $W_0 = \{v \in V\colon \mu(v) \neq \top\}$, $W_1 = V \backslash W_0$.

Let us illustrate this algorithm in a simple example.

2.2.18 Example: Consider the following game graph.

$$\begin{tikzcd}
 & v_3 \arrow[d] & \\
\boxed{v_1} \arrow[ur] & v_4 \arrow[l] \arrow[r] & \boxed{v_2} \arrow[ul]
\end{tikzcd}$$

The value of v_i is i for each $i \in \{1,\dots,4\}$ and $V_0 = \{v_3, v_4\}$, $V_1 = \{v_1, v_2\}$. Let us see how ProgressMeasureLifting works in a table.

$\mu(v_1)$	$\mu(v_2)$	$\mu(v_3)$	$\mu(v_4)$	$\mathrm{Lift}(\mu, v_1)(v_1)$	$\mathrm{Lift}(\mu, v_2)(v_2)$	$\mathrm{Lift}(\mu, v_3)(v_3)$	$\mathrm{Lift}(\mu, v_4)(v_4)$
0000	0000	0000	0000	0001			
0001	0000	0000	0000	0001	0000	0100	
0001	0000	0100	0000	0001	0100		
0001	0100	0100	0000	0001	0100	0100	0000

To calculate $\mathrm{Lift}(\mu, v_1)(v_1)$ in the first step we need to calculate

$$\max_{(v_1,w)\in E} \mathrm{Prog}((\mu_0\colon V \to \{0x0x\colon x \in \{0,1\}\}, v \mapsto 0), v_1, w),$$

hence in this case $\mathrm{Prog}(\mu_0, v_1, v_3)$ which is defined as the smallest a in $\{0x0x\colon x \in \{0,1\}\}$ such that $a >_{\mathrm{val}(v_1)} \mu_0(v_3)$ or $a = \mu_0(v_3) = \top$, since $\mathrm{val}(v_1) = 1$ is odd. Thus, $\mathrm{Prog}(\mu_0, v_1, v_3)$ is 0001, since $0001 > 0000$ and $x \leq 0000$ for each $x < 0001$, $x \in \{0,1\}$. The other instances of the lifting operation can be calculated in the same way. Once we have found an instance that differs from the current value of μ at that node, we can stop and update μ. The algorithm terminates when we cannot update any more.
It follows that Player 0 wins from any position according to the algorithm.

To see that this algorithm indeed gives us the winning regions of both players, we need to combine the previous lemmata and propositions in the proof of the following theorem, which also gives us the runtime for this particular algorithm.

2.2.19 Theorem: Let $\mathcal{G}$ be a parity game. Then ProgressMeasureLifting computes the winning sets for players 0 and 1 and a winning strategy for Player 0 from her winning set. The space complexity is $\mathcal{O}(mn)$, where m is the largest occurring value and n the number of nodes in V. Let k be the number of edges in the parity graph. Then the algorithm runs in time

$$\mathcal{O}\left(mk \cdot \left(\frac{n}{\lfloor m/2 \rfloor}\right)^{\lfloor m/2 \rfloor}\right).$$

Proof. By the Knaster-Tarski Theorem ProgressMeasureLifting computes the least simultaneous fixed point μ of the Lift-operators $\mathrm{Lift}(\cdot, v)$ for all $v \in V$. By Proposition 2.2.16 μ is a game parity progress measure. Proposition 2.2.12 gives us a winning strategy for the set $||\mu||$ which is therefore a subset of her winning region. However the same proposition gives us equality here, because if $v \in W_0 \backslash ||\mu||$, then there exists a progress measure $\rho \colon V \to M_G^\top$ such that $\rho(v) \neq \top$ which contradicts μ being a $\sqsubseteq$-least fixed point.
The space complexity stems from ProgressMeasureLifting only having to store an m-tuple of integers or the top element, namely $\mu(v) \in M_G^\top$ for each $v \in V$.
The Lift-operator works in time $\mathcal{O}(d \cdot \mathrm{outdeg}(v))$ and every node in v can be lifted at most $|M_G|$ many times. Hence the running time is in

$$\mathcal{O}\left(\sum_{v \in V} d \cdot \mathrm{outdeg}(v) \cdot |M_G|\right) = \mathcal{O}(kn \cdot |M_G|).$$

Since we can assume as usual that the values in $\mathcal{G}$ lie in the set $\{1, \ldots, m\}$ we have that $\sum_{i=1}^{\lfloor m/2 \rfloor}(|\{v \in V \colon \mathrm{val}(v) = i\}| + 1) \leq n$. It follows that

$$|M_G| = \prod_{i=1}^{\lfloor m/2 \rfloor} (|\{v \in V \colon \mathrm{val}(v) = i\}| + 1) \leq \left(\frac{n}{\lfloor m/2 \rfloor}\right)^{\lfloor m/2 \rfloor}.$$

□

This runtime is not quasipolynomial, but the next section will show how we can improve the methods used, to achieve this time bound.

2.2.2 Improving the runtime to quasi-polynomial

This subsection is based on the paper *Succinct progress measures for solving parity games* by Jurdziński and Lazić [JL17]. It will largely employ the same methods as the previous section with one crucial refinement: The labels used by the progress measures will be significantly more succinct. Jurdziński and Lazić call this refinement *succinct tree coding*. It takes an ordered tree and encodes it using binary strings.
Let us begin by defining an ordering on such binary strings.

2.2.20 Definition: Consider the set of finite binary strings, i.e. finite strings containing only the symbols 0, 1. We denote the empty word by ϵ. We define a linear ordering $<$ on this set in the following way. For all finite binary strings s and s' and all $b \in \{0,1\}$ or $b = \epsilon$

$$0s < \epsilon, \qquad \epsilon < 1s, \qquad bs < bs' \Leftrightarrow s < s'.$$

The idea of the succinct tree coding is to have for each leaf a code in the form of a tuple containing as few 0s and 1s as possible. Formally, these tuples have the following form.

2.2.21 Definition: The set $B_{l,h}$ of *l-bounded adaptive h-counters* is the set of h-tuples of finite binary strings where the tuple contains at most l zeros and ones in total.
We extend the ordering $<$ to $B_{l,h}$ lexicographically.

Let us introduce a few new terms to describe precisely, what we mean by an ordered tree and by a tree coding.

2.2.22 Definition: An *ordered tree* is a prefix-closed set of sequences of elements of a linearly ordered set. The sequences are called *nodes* and the maximal nodes with respect to the given ordering are called *leaves*. The elements of the linearly ordered set are called *branching directions* and the sequences of branching directions that uniquely identify the nodes are referred to as *navigation paths*.
An *ordered tree coding* is an order preserving relabelling of the branching directions. The word 'adaptive', which appeared in the previous definition, stems from the fact that we allow the labels of the same branching direction for different nodes to differ from one another.

Note that the sequence of branching directions for node v codes which edges lead to v from the root: The branching direction from the root, meaning the first branching direction in the navigation path, indicates the unique child of the root which lies on the path to v, the second branching direction indicates the unique child of that node on the path to v and so on.
The aim is to take any ordered tree and rename the branching directions such that they still fulfil the definition of branching directions, but the navigation paths are much more succinct. For this, we use the adaptive counters defined earlier.

2.2.23 Lemma: Let T be an ordered tree of height h with at most n leaves. Then there is a tree coding in which each navigation path is a $\lceil \log(n) \rceil$-bounded adaptive h-counter.

Proof. The proof is an induction over n and h.
If $n = 1$ and $h = 0$ take as branching direction the empty tuple $()$.
Now let T be an ordered tree of height h with at most n leaves and assume that the statement of the lemma hold for all ordered trees T' of height $< h$ with at most n leaves or of height h with less than n leaves.
Let M be a branching direction from the root such that the sets of leaves $L_<$ whose branching directions from the root are strictly smaller than M and $L_>$ whose branching directions from the root are strictly larger both are of size at most $n/2$. The set of leaves whose branching direction from the root is M is referred to as $L_=$. Then we obtain the desired coding in the following way:

- For $L_<$, if it is nonempty, consider the subtree of T that has as its leaves the elements from $L_<$. This subtree has height at most h and at most $n/2 < n$ leaves and we can therefore apply the induction hypothesis. Now add a 0 to the beginning of all the binary strings that code the branching directions from the root.

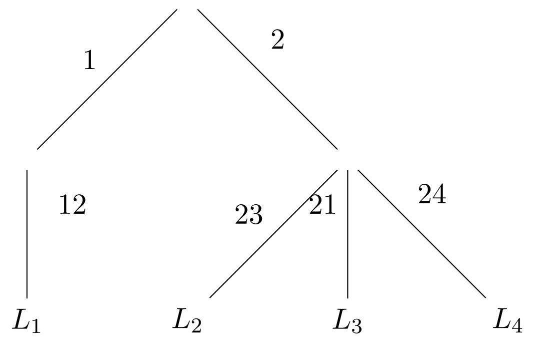
- For $L_>$, if it is nonempty, consider the subtree of T that has as its leaves the elements from $L_>$. This subtree has height at most h and at most $n/2 < n$ leaves and we can therefore apply the induction hypothesis. Now add a 1 to the beginning of all the binary strings that code the branching directions from the root.

- The code of the branching direction M from the root is set to ϵ. The subtree of T induced by the child of the root with branching direction M as the new root has precisely the nodes in $L_=$ as leaves and has height $n - 1$. Thus we can apply the induction hypothesis to this subtree.

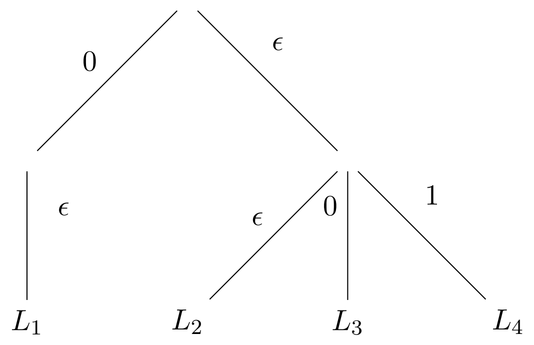
□

Let us illustrate this in an example.

2.2.24 Example: Consider the following ordered tree.



Application of the previous lemma yields:



The navigation path for L_1 is now $(0, \epsilon)$, for L_2 it is (ϵ, ϵ), for L_3 it is $(\epsilon, 0)$ and for L_4 it is $(\epsilon, 1)$.

2.2.25 Definition: A *succinctly coded tree* is an ordered tree T such that the branching directions are given as finite binary strings with the ordering from Definition 2.2.20 and the sum of lengths of binary strings in a navigation path is bounded by $\lceil \log(k) \rceil$, where k is the number of leaves in T.

Note that by Lemma 2.2.23 every ordered tree can be presented as a succinctly coded tree. Let us see how we can apply this succinct tree coding to progress measures. To do so, we need to translate progress measures into ordered trees.

2.2.26 Remark: Let $\rho \colon V \to M_G$ be a parity progress measure for a parity graph G. Then ρ can be considered as an ordered tree of height $\lceil m/2 \rceil$ with n leaves by taking the nodes $v \in V$ as the leaves with branching direction $r_m r_{m-2} r_{m-4} \cdots r_1$ if m is odd or $r_{m-1} r_{m-3} \cdots r_1$ if m is even, where $\rho(v) = r_m 0 r_{m-2} 0 r_{m-4} 0 \cdots r_1$ or $\rho(v) = 0 r_{m-1} 0 r_{m-3} 0 \cdots r_1$, respectively.
For simplicity, from now on we will present all arguments only for odd m since the case for even m is completely analogous.

Similarly to the use of $\leq_i$ in the previous section, we need to define an ordering on subsequences as well. The subsequences we need to consider are the truncations defined in the following definition. The ordering on these truncations is simply the lexicographical ordering from left to right.

2.2.27 Definition: Let $s = r_m r_{m-2} \cdots r_1$ be a sequence as described in the remark above and let $p \in \mathbb{N}$. We define the *p-truncation* of s as

$$r_m r_{m-2} \cdots r_1|_p = r_m r_{m-2} \cdots r_{k+2} r_k,$$

where k is the smallest odd number such that $k \geq p$.

Now we can define succinct game progress measures using the succinct tree coding for the labels.

2.2.28 Definition: Let $\mathcal{G}$ be a parity game. A *succinct game progress measure* for $\mathcal{G}$ is a function $\mu\colon V \to T^\top$, where $T^\top$ is a succinctly coded tree T with an additional top element $\top$. $\top$ is defined to be larger than any $t \in T$ and to satisfy $\top|_p > t$, $\top|_p = \top$, for all $p \in \mathbb{N}$, $t \in T$. We require that

1. if $v \in V_0$ then for some $(v, w) \in E$ it holds that $\mu(v)|_{\text{val}(v)} \geq \mu(w)|_{\text{val}(v)}$ and $\mu(v)|_{\text{val}(v)} > \mu(w)|_{\text{val}(v)}$ or $\mu(v) = \mu(w) = \top$ if val(v) is odd and
2. if $v \in V_1$ then for all $(v, w) \in E$ it holds that $\mu(v)|_{\text{val}(v)} \geq \mu(w)|_{\text{val}(v)}$ and $\mu(v)|_{\text{val}(v)} > \mu(w)|_{\text{val}(v)}$ or $\mu(v) = \mu(w) = \top$ if val(v) is odd.

We define $||\mu||$ as the set of $v \in V$ such that $\mu(v) \neq \top$.

To show that we can use these succinct game progress measures in the same way as we used the parity progress measures in the previous subsection, we will have to prove very similar propositions anew.

2.2.29 Proposition: Let $G = (V, E, \text{val})$ be a parity graph and let $\mu\colon V \to T$ be such that for all $(v, w) \in E$ we have $\mu(v)|_{\text{val}(v)} \geq \mu(w)|_{\text{val}(v)}$ and the inequality is strict if val(v) is odd. Then every cycle in G is even.

Proof. Assume that $v_1 v_2 v_3 \cdots v_{k-1} v_k v_1$ is an odd cycle in G and without loss of generality let v_1 have the largest value in the cycle. Then

$$\mu(v_1)|_{\text{val}(v)} > \mu(v_2)|_{\text{val}(v)} \geq \mu(v_3)|_{\text{val}(v)} \geq \cdots \geq \mu(v_{k-1})|_{\text{val}(v)} \geq \mu(v_k)|_{\text{val}(v)} \geq \mu(v_1)|_{\text{val}(v)}.$$

This is a contradiction. □

2.2.30 Proposition: Let $\mathcal{G}$ be a parity game.

1. If $\mu\colon V \to T^\top$ is a a succinct game progress measure for $\mathcal{G}$, then Player 0 has a winning strategy from all $v \in ||\mu||$.
2. There exists a succinct game progress measure $\mu\colon V \to T^\top$ such that

$$\mu(v) \neq \top \quad \Leftrightarrow \quad \text{player 0 has a winning strategy from } v$$

Proof. 1. Let $\mu\colon V \to T^\top$ be a succinct game progress measure for $\mathcal{G}$ and let $v \in ||\mu||$. Consider the parity graph $G' = (V', E')$ derived from G by deleting for each $w \in V_0$ every outgoing edge $(w, u) \in E$ but one that satisfies $\mu(v)|_{\text{val}(v)} \geq \mu(w)|_{\text{val}(v)}$ and $\mu(v)|_{\text{val}(v)} > \mu(w)|_{\text{val}(v)}$ or $\mu(v) = \mu(w) = \top$ if val(v) is odd, which exists by assumption. Then delete all nodes that are not reachable from v.
Now G' is a parity graph such that $\mu(w)|_{\text{val}(v)} \geq \mu(u)_{\text{val}(v)}$ for all $w, u \in G'$, $(w, u) \in$

E', and therefore $\mu(w) < \top$ for all $w \in G'$ since $\mu(v) < \top$ and all $v \neq w \in G'$ are reachable from v. Additionally, $\mu(w)|_{\mathrm{val}(v)} > \mu(u)|_{\mathrm{val}(v)}$ for all $w, u \in G'$ with $\mathrm{val}(w)$ odd. It follows that $\mu' \colon V' \to T,\ w \mapsto \mu(w)$ is a succinct game progress measure for $\mathcal{G}' = (V', E', \mathrm{val}|_{V'}, V_0 \cap V', V_1 \cap V')$ and by Proposition 2.2.29 all cycles in G' are even. Hence, Player 0 wins $\mathcal{G}'$ and since we deleted only edges and nodes from G that Player 0 can avoid, picking from each node $w \in V_0$ the one edge $(w, u) \in E'$ is a winning strategy for Player 0.

2. By Proposition 2.2.12 there exists a game parity progress measure $\rho \colon V \to M_G^{\top}$ such that

$$\rho(v) \neq \top \quad \Leftrightarrow \quad \text{player 0 has a winning strategy from } v.$$

Let $G' = (V', E')$ be the subgraph of G induced by the $v \in ||\rho||$. Using the coding of ρ into an ordered tree T as described in Remark 2.2.26 and applying Lemma 2.2.23 we obtain a function $\mu' \colon V' \to T$ by mapping each $v \in V'$ to the corresponding leaf of T in the translation. Now extend μ' to a function $\mu \colon V \to T^{\top}$ by defining $\mu(v) = \mu'(v)$ for all $v \in V'$ and $\mu(v) = \top$ for all $v \notin V'$. Since the translation and the application of the lemma are both order preserving, it follows immediately from the definitions of parity progress measures and succinct game progress measures, that μ is a succinct game progress measure. Additionally, by Proposition 2.2.12

$$\mu(v) \neq \top \quad \Leftrightarrow \quad \text{player 0 has a winning strategy from } v.$$

□

Obviously, we do not want to consider just any ordered tree T in the progress measures, but apply the succinct tree coding.

2.2.31 Remark: Since in Proposition 2.2.30 2. the succinct game progress measure maps every $v \in V$ to either a leaf of the succinctly coded tree T or to $\top$ we can consider ρ as a function $\rho \colon V \to B^{\top}_{\lceil \log n \rceil, \lceil m/2 \rceil}$, where $B^{\top}_{\lceil \log n \rceil, \lceil m/2 \rceil}$ is defined as $B_{\lceil \log n \rceil, \lceil m/2 \rceil} \cup \{\top\}$ with $\top$ defined as above.

Again we can retrace the steps from the previous section and define a lifting operator via the operator Prog.

2.2.32 Definition: Let $\mathcal{G}$ be a parity game and $\mu \colon V \to B^{\top}_{\lceil \log n \rceil, \lceil m/2 \rceil}$ a function. Let $(v, w) \in E$. Then we define $\mathrm{Prog}(\mu, v, w)$ as the smallest $a \in B^{\top}_{\lceil \log n \rceil, \lceil m/2 \rceil}$ such that $a|_{\mathrm{val}(v)} \geq \mu(w)|_{\mathrm{val}(v)}$ and if $\mathrm{val}(v)$ is odd then either $a|_{\mathrm{val}(v)} > \mu(w)|_{\mathrm{val}(v)}$ or $a = \mu(w) = \top$.

2.2.33 Definition: Let $\mathcal{G}$ be a parity game, $\mu\colon V \to B^{\top}_{\lceil \log n \rceil, \lceil m/2 \rceil}$ a function and $v \in V$. Then we define a function $\operatorname{Lift}(\mu, v)\colon V \to B^{\top}_{\lceil \log n \rceil, \lceil m/2 \rceil}$ by

$$\operatorname{Lift}(\mu, v)(u) = \begin{cases} \mu(u) & \text{if } u \neq v \\ \min_{(v,w)\in E} \operatorname{Prog}(\mu, v, w) & \text{if } u = v \in V_0 \\ \max_{(v,w)\in E} \operatorname{Prog}(\mu, v, w) & \text{if } u = v \in V_1 \end{cases}.$$

Here, too, we want to apply the Knaster-Tarski Theorem, so we need an ordering on the functions and we need to show monotony of the Lift operator with respect to this ordering.

2.2.34 Definition: Let $\mathcal{G}$ be a parity game. We define a partial ordering on the functions $V \to B^{\top}_{\lceil \log n \rceil, \lceil m/2 \rceil}$ in the following way: Let $\mu_1\colon V \to B^{\top}_{\lceil \log n \rceil, \lceil m/2 \rceil}$ and $\mu_2\colon V \to B^{\top}_{\lceil \log n \rceil, \lceil m/2 \rceil}$. We write $\mu_1 \sqsubseteq \mu_2$ if $\mu_1(v) \leq \mu_2(v)$ for all $v \in V$ and $\mu_1 \sqsubset \mu_2$ if $\mu_1 \sqsubseteq \mu_2$ and $\mu_1 \neq \mu_2$.

2.2.35 Proposition: Let $\mathcal{G}$ be a parity game. For all $v \in V$ the Lift-operator $\operatorname{Lift}(\cdot, v)$ is $\sqsubseteq$-monotone.

Proof. Let $\mu_1, \mu_2\colon V \to B^{\top}_{\lceil \log n \rceil, \lceil m/2 \rceil}$ be functions, $\mu_1 \sqsubseteq \mu_2$, and $v \in V$. We need to show that $\operatorname{Lift}(\mu_1, v) \sqsubseteq \operatorname{Lift}(\mu_2, v)$. Since $\operatorname{Lift}(\mu_1, v)(u) = \mu_1(u)$ and $\operatorname{Lift}(\mu_2, v)(u) = \mu_2(u)$ for all $v \neq u$ it suffices to show that $\operatorname{Lift}(\mu_1, v)(v) \leq \operatorname{Lift}(\mu_2, v)(v)$.
Let $w \in V$ with $(v, w) \in E$. Then for $\mu\colon V \to B^{\top}_{\lceil \log n \rceil, \lceil m/2 \rceil}$ $\operatorname{Prog}(\mu, v, w)$ is defined as the smallest $a \in B^{\top}_{\lceil \log n \rceil, \lceil m/2 \rceil}$ such that $a|_{\operatorname{val}(v)} \geq \mu(w)|_{\operatorname{val}(v)}$ and $a|_{\operatorname{val}(v)} > \mu(w)|_{\operatorname{val}(v)}$ or $a = \mu(w) = \top$ if $\operatorname{val}(v)$ is odd. We claim that $\operatorname{Prog}(\mu_1, v, w) \leq \operatorname{Prog}(\mu_2, v, w)$. Since $\mu_1 \sqsubseteq \mu_2$ we know that $\mu_1(w) \leq \mu_2(w)$. By definition of the lexicographical ordering it follows that $\mu_1(w)|_i \leq \mu_2(w)|_i$ for all $i \in \{1, \ldots, m\}$. Hence, $\operatorname{Prog}(\mu_2, v, w)|_{\operatorname{val}(v)} \geq \mu_2(w)|_{\operatorname{val}(v)} \geq \mu_1(w)|_{\operatorname{val}(v)}$ and $\operatorname{Prog}(\mu_2, v, w) > \mu_2(w)$ and $\mu_2(w)|_{\operatorname{val}(v)} \geq \mu_1(w)|_{\operatorname{val}(v)}$ or $\operatorname{Prog}(\mu_2, v, w) = \mu_2(w) = \top = \top|_{\operatorname{val}(v)} \geq \mu_1(v)|_{\operatorname{val}(v)}$ if $\operatorname{val}(v)$ is odd. In either case, since $\operatorname{Prog}(\mu_1, v, w)$ is minimal with these properties, $\operatorname{Prog}(\mu_1, v, w) \leq \operatorname{Prog}(\mu_2, v, w)$.
By definition of the Lift-operator via Prog it follows that $\operatorname{Lift}(\mu_1, v)(v) \leq \operatorname{Lift}(\mu_2, v)(v)$ and therefore $\operatorname{Lift}(\mu_1, v) \sqsubseteq \operatorname{Lift}(\mu_2, v)$. □

The fixed point the Knaster-Tarski Theorem will give us is as useful here as it was previously.

2.2.36 Proposition: Let $\mathcal{G}$ be a parity game and $\mu\colon V \to B^{\top}_{\lceil \log n \rceil, \lceil m/2 \rceil}$ a function. Then μ is a succinct game progress measure if and only if $\operatorname{Lift}(\mu, v) \sqsubseteq \mu(v)$ for all $v \in V$.

Proof. Let μ be a succinct game progress measure and $v \in V$. Since $\operatorname{Lift}(\mu, v)(u) = \mu(u)$ for all $u \neq v$ it suffices to show that $\operatorname{Lift}(\mu, v)(v) \leq \mu(v)$. Suppose first that $v \in V_0$. Because μ is a succinct game progress measure we know that $\mu(v)|_{\operatorname{val}(v)} \geq \operatorname{Prog}(\mu, v, w)|_{\operatorname{val}(v)}$ for some $(v, w) \in E$. Since $\operatorname{Prog}(\mu, v, w)$ is the minimal $a \in B^{\top}_{\lceil \log n \rceil, \lceil m/2 \rceil}$ such that $a|_{\operatorname{val}(v)} \geq \mu(w)|_{\operatorname{val}(v)}$

it follows that $\mu(v) \geq \min_{(v,w)\in E} \operatorname{Prog}(\mu, v, w) = \operatorname{Lift}(\mu, v)(v)$.
If $v \in V_1$ then $\mu(v)|_{\operatorname{val}(v)} \geq \operatorname{Prog}(\mu, v, w)|_{\operatorname{val}(v)}$ for all $(v, w) \in E$. Since $\operatorname{Prog}(\mu, v, w)$ is the minimal $a \in B^{\top}_{\lceil \log n \rceil, \lceil m/2 \rceil}$ such that $a|_{\operatorname{val}(v)} > \mu(w)|_{\operatorname{val}(v)}$ or $a = \mu(w) = \top$ it follows that $\mu(v) \geq \max_{(v,w)\in E} \operatorname{Prog}(\mu, v, w) = \operatorname{Lift}(\mu, v)(v)$ since $\mu(v)|_{\operatorname{val}(v)} \geq \top$ implies $\mu(v) = \top$.
Conversely, let $\operatorname{Lift}(\mu, v) \sqsubseteq \mu(v)$ for all $v \in V$. Let a fixed node $v \in V$ be given. Since $\operatorname{Lift}(\mu, v) \sqsubseteq \mu$ we have in particular $\operatorname{Lift}(\mu, v)(v) \leq \mu(v)$. If $v \in V_0$ this means $\min_{(v,w)\in E} \operatorname{Prog}(\mu, v, w) \leq \mu(v)$ and therefore there exists an edge $(v, w) \in E$ with $\operatorname{Prog}(\mu, v, w)|_{\operatorname{val}(v)} \leq \mu(v)|_{\operatorname{val}(v)}$. If $v \in V_1$ we have $\max_{(v,w)\in E} \operatorname{Prog}(\mu, v, w) \leq \mu(v)$ and therefore $\operatorname{Prog}(\mu, v, w)|_{\operatorname{val}(v)} \leq \mu(v)|_{\operatorname{val}(v)}$ for all $w \in V$ with $(v, w) \in E$. Hence μ satisfies the defining conditions of a game parity progress measure. □

Applying the Knaster-Tarski Theorem to the Lift-operator for some parity game $\mathcal{G}$ gives us the $\sqsubseteq$-least succinct game progress measure by the following simple algorithm, which is very similar to the old one, with the exception of the images of the progress measures.

2.2.37 Algorithm (SuccinctProgressMeasureLifting)**:**

```
Initialize $\mu\colon V \to B^{\top}_{\lceil \log n \rceil, \lceil m/2 \rceil}$, $v \mapsto (0 \cdots 0, \epsilon, \dots, \epsilon)$
while $\mu \sqsubset \operatorname{Lift}(\mu, v)$ for some $v \in V$ do $\mu := \operatorname{Lift}(\mu, v)$.
Return $W_0 = \{v \in V : \mu(v) \neq \top\}$, $W_1 = V \backslash W_0$.
```

2.2.38 Example: Consider the game graph below with $\operatorname{val}(v_i) = i$ for all i and $V_0 = \{v_3, v_4\}$, $V_1 = \{v_1, v_2\}$.

$$\begin{tikzcd}
 & v_3 \arrow[d] & \\
\boxed{v_1} \arrow[ur] & v_4 \arrow[l] \arrow[r] & \boxed{v_2} \arrow[ul]
\end{tikzcd}$$

The value of v_i is i for each $i \in \{1, \dots, 4\}$ and $V_0 = \{v_3, v_4\}$, $V_1 = \{v_1, v_2\}$. Let us see how SuccinctProgressMeasureLifting works in a table.
Note that $\lceil \log n \rceil = 2$ and $m/2 = 2$.

$\mu(v_1)$	$\mu(v_2)$	$\mu(v_3)$	$\mu(v_4)$	$\operatorname{Lift}(\mu, v_1)(v_1)$	$\operatorname{Lift}(\mu, v_2)(v_2)$	$\operatorname{Lift}(\mu, v_3)(v_3)$	$\operatorname{Lift}(\mu, v_4)(v_4)$
00,ϵ	00,ϵ	00,ϵ	00,ϵ	0,0			
0,0	00,ϵ	00,ϵ	00,ϵ	0,0	00,ϵ	0,0	
0,0	00,ϵ	0,0	00,ϵ	01,ϵ			
01,ϵ	00,ϵ	0,0	00,ϵ	01,ϵ	00,ϵ	0,0	0,0
01,ϵ	00,ϵ	0,0	0,0	01,ϵ	0,0		
01,ϵ	0,0	0,0	0,0	01,ϵ	0,0	0,1	
01,ϵ	0,0	0,1	0,0	01,ϵ	0,0	0,1	0,1
01,ϵ	0,0	0,1	0,1	01,ϵ	0,0	0,1	0,1

To calculate $\mathrm{Lift}(\mu, v_1)(v_1)$ in the first step we need to calculate

$$\max_{(v_1,w)\in E} \mathrm{Prog}((\mu_0\colon V \to B^{\top}_{\lceil \log n\rceil,\lceil m/2\rceil},\ v \mapsto (0\cdots 0, \epsilon, \dots, \epsilon)), v_1, w),$$

hence in this case $\mathrm{Prog}(\mu_0, v_1, v_3)$ which is defined as the smallest a in $B^{\top}_{\lceil \log n\rceil,\lceil m/2\rceil}$ such that $a|_{\mathrm{val}(v_1)} > \mu_0(v_3)|_{\mathrm{val}(v_1)}$ or $a = \mu_0(v_3) = \top$, since $\mathrm{val}(v_1) = 1$ is odd. Thus, $\mathrm{Prog}(\mu_0, v_1, v_3)$ is 0,0, since $0 > 00$ and $x \leq 00$ for each $x < 0$, which is only 00. The other instances of the lifting operation can be calculated in the same way. Once we have found an instance that differs from the current value of μ at that node we can stop and update μ. The algorithm terminates when we cannot update any more. It follows that Player 0 wins from any position according to the algorithm.

The correctness of the algorithm follows immediately from the previous considerations:

2.2.39 Theorem: Let $\mathcal{G}$ be a parity game. Then SuccinctProgressMeasureLifting computes the winning sets for players 0 and 1 and a winning strategy for Player 0 from her winning set.

Proof. By the Knaster-Tarski Theorem SuccinctProgressMeasureLifting computes the least simultaneous fixed point μ of the Lift-operators $\mathrm{Lift}(\cdot, v)$ for all $v \in V$. By Proposition 2.2.36 μ is a game parity progress measure. Proposition 2.2.30 gives us a winning strategy for the set $||\mu||$ which is therefore a subset of her winning region. However the same proposition gives us equality here, because if $v \in W_0 \backslash ||\mu||$, then there exists a progress measure $\mu'\colon V \to B^{\top}_{\lceil \log n\rceil,\lceil m/2\rceil}$ such that $\mu'(v) \neq \top$ which contradicts μ being a $\sqsubseteq$-least fixed point.

□

Now let us take a look at the big difference to the previous subsection, the complexity. It is largely dependent on the size of $B_{\lceil \log n\rceil,\lceil m/2\rceil}$. So let us take a look at this size first.

2.2.40 Lemma: Let $n, m \in \mathbb{N}$. Then

1. $|B_{\lceil \log n\rceil,\lceil m/2\rceil}| \leq 2^{\lceil \log n\rceil}\binom{\lceil \log n\rceil + m/2}{m/2}$.
2. If $m = \mathcal{O}(\log n)$ then $|B_{\lceil \log n\rceil,\lceil m/2\rceil}|$ is bounded by a polynomial in n.
3. If $m \geq \log(n)$ then $|B_{\lceil \log n\rceil,\lceil m/2\rceil}| = \mathcal{O}(n^{\log m - \log\log n + 4{,}03})$.

Proof. 1. The set $B_{\lceil \log n\rceil,\lceil m/2\rceil}$ is the set of $\lceil \log n\rceil$-bounded adaptive $\lceil m/2\rceil$-counters, meaning the set of $\lceil m/2\rceil$-tuples of binary sequences with a total number of at most $\lceil \log n\rceil$ zeros and ones. For the sequences of zeros and ones of length exactly $\lceil \log n\rceil$ there are $2^{\lceil \log n\rceil}$ possibilities. For a $\lceil m/2\rceil$-tuple with at most $\lceil \log n\rceil$ zeros and ones in total there are

$$\binom{\lceil \log n \rceil + m/2}{m/2} = \binom{\lceil \log n \rceil + m/2}{\lceil \log n \rceil}$$

possibilities of distributing $\lceil \log n \rceil$ numbers to $m/2+1$ components, one for the unused of the $\lceil \log n \rceil$ bits.

2. Let $m = 2\delta \log n$. Then one can obtain

$$\binom{\log n + m/2}{\log n} = \binom{\log n + \delta \log n}{\delta \log n} = \binom{(\delta+1)\log n}{\delta \log n}.$$

This yields

$$|B_{\lceil \log n \rceil, \lceil m/2 \rceil}| = \Theta\left(\frac{n^{(\delta+1)H\left(\frac{\delta}{\delta+1}\right)+1}}{\sqrt{\log n}}\right),$$

where $H(p) = -p \log p - (1-p)\log(1-p)$ for $p \in [0,1]$. It is known that

$$(\delta+1)H\left(\frac{\delta}{\delta+1}\right) = \log(\delta+1) + \log(e_\delta).$$

Since this second expression does not depend on n, we obtain the desired complexity result.

3. It holds that

$$\begin{aligned}\log\left(\binom{\lceil \log n \rceil + m/2}{\lceil \log n \rceil}\right) &\leq \lceil \log n \rceil \cdot (\log(\lceil \log n \rceil + m/2) - \log(\log n) + \log e) \\ &\leq \lceil \log n \rceil \cdot (\log n - \log m - \log(\log n) + 2.03).\end{aligned}$$

The first inequality is obtained by applying log to both sides of $\binom{l}{k} \leq ((el)/k)^k$ with the binomial coefficient above substituted. The second inequality follows from $m \geq \log(n)$ and $\log(3/2) + \log(e) \approx 2.0277 < 2.03$. It follows by $m \leq n$ that

$$\begin{aligned}|B_{\lceil \log n \rceil, \lceil m/2 \rceil}| \leq 2^{\lceil \log n \rceil \cdot (\log m - \log(\log n) + 3.03)} &\leq 2^{\log n \cdot (1 + 1/\log(n)) \cdot (\log m - \log(\log n) + 3.03)} \\ &= \mathcal{O}\left(n^{\log m - \log(\log n) + 4.03}\right).\end{aligned}$$

□

To see how exactly the runtime depends on $|B_{\lceil \log n \rceil, \lceil m/2 \rceil}|$, consider the following lemma.

2.2.41 Lemma: Let $\mathcal{G}$ be a parity game with n nodes and k edges in the parity graph and m

the largest value occurring. The running time of SuccinctProgressMeasureLifting is bounded by

$$\mathcal{O}\left(\sum_{v\in V}\operatorname{outdeg}(v)\cdot\log n\cdot\log m\cdot|B_{\lceil\log n\rceil,\lceil m/2\rceil}|\right)=\mathcal{O}(k\log n\cdot\log m\cdot|B_{\lceil\log n\rceil,\lceil m/2\rceil}|).$$

Proof. Note first that every node can be updated by the Lift operator at most $|B_{\lceil\log n\rceil,\lceil m/2\rceil}|$ times. For each node $v\in V$ there are at most $\operatorname{outdeg}(v)$ computations of Prog necessary to compute $\operatorname{Lift}(\mu,v)(v)$. Thus, it suffices to show that $\operatorname{Prog}(\mu,v,w)$ can be computed in time $\mathcal{O}(\log n\cdot\log m)$ for all $\mu\colon V\to B_{\lceil\log n\rceil,\lceil m/2\rceil}$, $(v,w)\in E$.
Obviously the computation of $\operatorname{Prog}(\mu,v,w)$ requires the most steps if $\operatorname{val}(v)$ is odd and

$$\mu(v)|_{\operatorname{val}(v)}<\mu(w)|_{\operatorname{val}(v)}\neq\top,$$

since this requires finding the smallest $a\in B^{\top}_{\lceil\log n\rceil,\lceil m/2\rceil}$ such that $a|_{\operatorname{val}(v)}>\mu(w)|_{\operatorname{val}(v)}$. Let $\mu(w)=(s_m,s_{m-2},\dots,s_1)$ and consider the following cases. (With the length of a string or multiple strings we mean in this proof the number of zeros and ones occurring in it or all of them concatenated, respectively.)

Case 1: The total length of the s_i with $i\geq\operatorname{val}(v)$ is less than $\lceil\log n\rceil$.
In this case set $a=(s_m,s_{m-2},\dots,s_{\operatorname{val}(v)+2},s_{\operatorname{val}(v)}10\cdots0,\epsilon,\dots,\epsilon)$, using as many zeros as needed to obtain length $\lceil\log n\rceil$.

Case 2: The total length of the s_i with $i\geq\operatorname{val}(v)$ is equal to $\lceil\log n\rceil$, $j\geq\operatorname{val}(v)$ is the smallest odd value such that $s_j\neq\epsilon$ and $s_j=s'01\cdots1$ with $k\geq0$ ones and s' some binary string.
In this case set $a=(s_m,s_{m-2},\dots,s_{j+2},s',0\dots0,\epsilon,\dots,\epsilon)$ with the number of zeros in the entry after s' being $k+1$.

Case 3: The total length of the s_i with $i\geq\operatorname{val}(v)$ is equal to $\lceil\log n\rceil$, $m>j\geq\operatorname{val}(v)$ is the smallest odd value such that $s_j\neq\epsilon$ and $s_j=1\cdots1$ with $k\geq0$ ones.
In this case set $a=(s_m,s_{m-2},\dots,,s_{j+4},s_{j+2}10\dots0,\epsilon,\dots,\epsilon)$ with the number of zeros after $s_{j+2}1$ being $k-1$.

Case 4: None of the cases 1-3 apply.
Then set $a=\top$.

Note that each $b\in B_{\lceil\log n\rceil,\lceil m/2\rceil}$ can be represented using space $\mathcal{O}(\log n\cdot\log m)$: indicate with at most $\lceil\log n\rceil$ bits each zero and one in $B_{\lceil\log n\rceil,\lceil m/2\rceil}$, each followed by $\log(\lceil m/2\rceil)$ bits to denote the respective position in the tuple. Thus, every case needs running time at most $\mathcal{O}(\log n\cdot\log m)$ for computing $\operatorname{Prog}(\mu,v,w)$. □

Now we can combine the previous two lemmas to obtain the overall complexity, which is, as promised, quasipolynomial.

2.2.42 Theorem: Let $\mathcal{G}$ be a parity game with n nodes and k edges in the parity graph and m the largest value occurring. The algorithm SuccinctProgressMeasureLifting has the following running times.

1. If $m \leq \lceil \log n \rceil$ then the algorithm runs in time $\mathcal{O}(kn^{2.38})$.
2. If $m \geq \lceil \log n \rceil$ then the algorithm runs in time $\mathcal{O}(kn^{\log m - \log\log n + 4{,}03})$.

The algorithm needs space $\mathcal{O}(n \log n \cdot \log m)$.

Proof. The space complexity stems from the fact that at most $n \cdot \lceil \log n \rceil \cdot (\lceil \log(m/2) \rceil + 1)$ bits are needed to store a function $\mu \colon V \to |B_{\lceil \log n \rceil, \lceil m/2 \rceil}|$.
For the first statement of the theorem note that $\delta < 1/2$ (δ as in Lemma 2.2.40) implies that if $m \leq \lceil \log n \rceil$ then $m/2 \leq \lceil \delta \log n \rceil$. Hence

$$\log(\delta + 1) + \log(e_\delta) + 1 = 3/2 \log 3 \approx 2.3775 < 2.38.$$

The rest follows also from combining Lemma 2.2.40 and Lemma 2.2.41. □

Just as for the succinct counting method by Calude et al, this method, too, gives us a runtime in FPT.

2.2.43 Corollary: Solving parity games is in FPT with the number of values as the parameter.

Proof. The algorithm SuccintProgressMeasureLifting can solve parity games in time

$$2^{\mathcal{O}(m \log m)} \geq \mathcal{O}(kn^{\log m - \log\log n + 4{,}03})$$

if $m \geq \lceil \log n \rceil$ and in time $\mathcal{O}(kn^{2.38})$ otherwise, where m is the number of values, n the number of nodes and $k \leq n^2$ the number of edges in the parity game. □

2.3 Comparing the two methods

At first glance, the succinct counting method and the method using succinct progress measures look very different from one another. The first ist based upon analysing individual plays and then constructing a reachability game from that, while the second updates a function again and again until it is a progress measure which shows the winning regions of both players. The similarity of both methods of course is the use of a very succinct encoding, of a winning statistics for the succinct counting method and of functions as succinctly coded trees in the method by Jurdziński and Lazić.
This second method has one large advantage in comparison to the method by Calude et al., namely the space complexity. For the latter, one has to construct the entire reachability game, so the space complexity like the time complexity is quasipolynomial. Using succinct progress measures, the space complexity is, as we have seen, $\mathcal{O}(n \log n \cdot \log m)$, which is quasi-linear.

2.3.1 Progress Measure for the Succinct Counting Method

In this section we will present a method, developed by Fearnley et al. in their paper *An Ordered Approach to Solving Parity Games in Quasi Polynomial Time and Quasi Linear Space* [Fea17] which shows that the succinct counting method can be adapted to incorporate progress measures as well, making the construction of the reachability game obsolete. As the title of the paper suggests, this also resolves the difference in space complexity between the two methods discussed in the previous sections.
The first relatively minor change to the method of Calude et al. is that we will analyse the play backwards, when building up the winning statistics.

2.3.1 Definition: Let $\mathcal{G}$ be a parity game and $\pi = v_1 v_2 v_3 \cdots v_k$ a partial play. We define the *backwards play* $\overleftarrow{\pi}$ as

$$\overleftarrow{\pi} = v_k \cdots v_3 v_2 v_1.$$

2.3.2 Definition: A *backwards statistics* for a partial play π for Player 0 is a winning statistics for $\overleftarrow{\pi}$ (or for an initial sequence of $\overleftarrow{\pi}$).

We now formulate the winning statistics as its own game, giving Player 0 an initial move in which she my choose how long the partial play π should be that the backwards winning statistics is build for.

2.3.3 Definition: We define the *backwards update game* using the algorithm defined in the first section of this chapter:

The algorithm tracks the values $b_0, \ldots, b_{\lceil \log(n) \rceil+1} \in \{0, \ldots, m\}$ which are all initially set to 0. Now let b be the value of the current node in the play.

1. If b is even or $b > b_i > 0$ for some i, then one selects the largest i such that

   a) either b is even and b_i is odd or 0 but all b_j with $j < i$ are even and non-zero

   b) or $0 < b_i < b$

   and then one updates $b_i = b$ and $b_j = 0$ for all $j < i$.

The only difference to the previous update rules is that this time, for a play $\pi = v_1 v_2 v_3 \cdots$, instead of evaluating in each step the forwards partial play $\pi = v_1 v_2 v_3 \cdots v_k$, where v_k is the current node in the play, Player 0 chooses some $l \in \mathbb{N}$ in the beginning of the game and we then evaluate the backwards play $v_l \cdots v_3 v_2 v_1$ from left to right. Player 0 wins the update game if $b_{\lceil \log(n) \rceil+1} > 0$.

2.3.4 Corollary: Player 0 wins a play π of a parity game $\mathcal{G}$ if and only if she wins the backwards update game.

Proof. In a play π of a parity game played according to positional strategies for both players it does not matter in which order we go through the nodes to find a loop. Therefore the exact same arguments as in the first section of this chapter hold here. $\square$

Based on this backwards update game we will define a so called *antagonistic update game.* The purpose of this game is to build up winning statistics such that the update rules are order preserving according to the ordering defined below, while still guaranteeing that Player 0 wins the play if and only if her winning statistics matures. This will be necessary to define a progress measure later.

2.3.5 Definition: We define an ordering $\sqsubseteq$ on the set

$$B = \{(b_0, \ldots, b_{\lceil \log(n) \rceil+1}) \colon b_i \in \{0, \ldots, m\} \text{ for all } i\}$$

by first defining an ordering $\preceq$ on $\{0, \ldots, m\}$ as follows:
Let $b, c \in \{0, \ldots, m\}$. Then $c \preceq b$ if one of the following holds:

- $c = 0$,
- b and c are both odd and $b \leq c$,
- c is odd and b is even or
- b and c are both even and $b \geq c$.

Then we define $\sqsubseteq$ on B, comparing $x, y \in B$ lexicographically according to $\preceq$ from right to left.

The ordering $\preceq$ simply prefers everything over 0, even numbers over odd numbers, larger even numbers over smaller ones and smaller odd numbers over larger ones. For instance the set $\{0, 1, 2, 3, 4, 5\}$ is ordered as $0, 5, 3, 1, 2, 4$, where 0 is the smallest element with respect to $\preceq$ and 4 the largest.

2.3.6 Definition: Let x be the winning statistics at node v, where v is a node in the backwards play π that we are tracking the winning statistics for (according to the algorithm just defined). Then we define the *antagonistic update*

$$\mathrm{au}(x, v) = \min_{\sqsubseteq}\{z \in B\colon z \text{ is the update of some } y \sqsupseteq x \text{ at } v\},$$

where by update we mean the normal update rules used in both algorithms defined so far in this chapter.

The term *antagonistic* stems from the fact that in this game the 'antagonist', Player 1, basically receives an additional move, being able to choose the smallest possible update (with respect to $\sqsubseteq$).
Note that the definition of au guarantees that if $x \sqsubseteq y$, then $\mathrm{au}(x, v) \sqsubseteq \mathrm{au}(y, v)$ for all nodes v. Thus, the antagonistic update is order preserving.

2.3.7 Definition: Let $\mathcal{G}$ be a parity game and $\pi = v_1 v_2 v_3 \cdots$ a play in $\mathcal{G}$. For the *antagonistic update game*, Player 0 chooses $l \in \mathbb{N}$ and evaluates $v_l \cdots v_3 v_2 v_1$ with initial winning statistics $x_l = (0, \ldots, 0)$ and update $x_{i-1} = \mathrm{au}(x_i, v_{i-1})$ for all $l \geq i > 1$. Player 0 wins the antagonistic update game on π if x_j is matured for some $1 \leq j \leq l$.

Let us illustrate this with an example.

2.3.8 Example: Let us suppose that Player 0 has chosen a point in a play π with values $\{1, \ldots, 4\}$ and we get the following sequence of values in the backwards initial play:

$$4\;2\;3\;1\;2\;3\;4\;2\;3\;1\;2\;3\;4\;2\;3\;1\;2\;3\;4\;2\;3\;1\;2\;3\;4\;2\;3\;1\;2\;3\;\cdots\;4\;2\;3\;1\;2\;3.$$

We can track the updates according to the rules for the antagonistic update game in a table and compare it to the backwards update game. The antagonistic update game is on the left and the backwards update game on the right (note that more b_i may be necessary to determine the winner but are omitted here because we only look at the first few updates):

Current value	b_2	b_1	b_0	b_2	b_1	b_0
4	0	0	4	0	0	4
2	0	3	2	0	2	0
3	0	3	3	0	3	0
1	0	3	3	0	3	0
2	0	3	2	0	3	2
3	0	3	3	0	3	3
4	0	4	3	0	4	3
2	0	4	2	0	4	2
3	0	4	3	0	4	3
1	0	4	3	0	4	3
2	0	4	2	0	4	2
3	0	4	3	0	4	3
4	0	4	4	0	4	4
2	3	0	2	2	0	0
...						

The antagonistic updates differ slightly from the normal backwards update game. For instance in the second step, $(0, 3, 0) \sqsupseteq (0, 0, 4)$ and $(0, 3, 0)$ is updated to $(0, 3, 2) \sqsubset (0, 2, 0)$, which is the update of $(0, 0, 4)$. So the antagonistic update is $(0, 3, 2)$.

Now we need to show that the antagonistic update game preserves the properties of our original update game, which guaranteed that Player 0 wins a play if and only if her winning statistics matures.

2.3.9 Theorem: Let $\mathcal{G}$ be a parity game and $\pi = v_1 v_2 v_3 \cdots$ a play in $\mathcal{G}$. If Player 0 has a winning strategy for the antagonistic update game on π, she wins π.

Proof. Let $l \in \mathbb{N}$ be the point that Player 0 chooses and consider $v_l \cdots v_3 v_2 v_1$. The backwards update game gives us a sequence $x_0 = (0, \ldots, 0), x_1, x_2, \ldots, x_l$ of winning statistics in B such that x_i is the update of x_{i-1} for all $1 \leq i \leq l$ and the antagonistic update game yields another such sequence $y_0 = (0, \ldots, 0), y_1, y_2, \ldots, y_l$. We claim that $x_i \sqsupseteq y_i$ for all $0 \leq i \leq l$ and prove it by induction over i.
Obviously, $x_0 \sqsupseteq y_0$, since the two are equal.
Now suppose that $x_i \sqsupseteq y_i$ for some $i < l$. Let x define the update of y_i at v_{i+1} according to the update rules in the backwards update game. Then

$$\begin{aligned} y_{i+1} &= \mathrm{au}(y_i, v_{l-i}) \\ &= \min_{\sqsubseteq}\{z \in B\colon z \text{ is the update in the backwards update game of some } y \sqsupseteq y_i \text{ at } v_{l-i}\} \\ &\sqsubseteq x_{i+1}. \end{aligned}$$

The last inequality is by induction hypothesis, since $x_i \sqsupseteq y_i$ and we take a minimum.
Since a matured winning statistics is larger with respect to $\sqsubseteq$ than a non-matured one, Player 0 also wins the backwards update game with the same strategy as for the antagonistic update game. Therefore, she wins π in $\mathcal{G}$. $\square$

To prove the opposite direction, we need a few preliminary considerations, which are a bit more technical.

2.3.10 Definition: Let $\mathcal{G}$ be a parity game, $x = (b_0, \ldots, b_{\lceil \log(n) \rceil + 1}) \in B$ a winning statistics and $q \leq m$ an even number. We define $x \downarrow_q$ in the following way:

- If for all $i \leq \lceil \log(n) \rceil + 1$ we have $b_i = 0$ or $b_i \geq q$ then $x \downarrow_q = x$.
- Otherwise let j be the largest index in $\{0, \ldots, \lceil \log(n) \rceil + 1\}$ such that $b_j \neq 0$ and $b_j < q$. Define $b'_i = b_i$ for all $i > j$, $b'_j = q - 1$ and $b'_i = 0$ for all $i < j$. Then $x \downarrow_q = (b'_0, \ldots, b'_{\lceil \log(n) \rceil + 1})$.

This gives us some useful properties.

2.3.11 Lemma: Let $\mathcal{G}$ be a parity game, $x = (b_0, \ldots, b_{\lceil \log(n) \rceil + 1}), y = (b'_0, \ldots, b'_{\lceil \log(n) \rceil + 1}) \in B$ not matured, $q \leq m$ an even number and v a node in $\mathcal{G}$.

1. If $x \sqsupseteq y$ then $x \downarrow_q \sqsupseteq y \downarrow_q$.
2. If $\mathrm{val}(v) < q$ and z denotes the update of x at node v in the backwards update game then $z \downarrow_q \sqsupseteq x \downarrow_q$.
3. If $\mathrm{val}(v) < q$ and z denotes the update of x at node v in the antagonistic update game then $z \downarrow_q \sqsupseteq x \downarrow_q$.
4. If $\mathrm{val}(v) = q$ and z denotes the update of x at node v in the backwards update game then $z \downarrow_q \sqsupset x \downarrow_q$.
5. If $\mathrm{val}(v) = q$ and z denotes the update of x at node v in the antagonistic update game then $z \downarrow_q \sqsupset x \downarrow_q$.

Proof. 1. If $x = y$ the claim is obvious. Suppose therefore $x \sqsupset y$ and let $i \leq \lceil \log(n) \rceil + 1$ be the maximal index such that $b_i \neq b'_i$. By definition of $\sqsubseteq$ it follows that $b'_i \prec b_i$. If there is an index $k > i$ such that $b'_k = b_k \neq 0$ and $b'_k = b_k < q$ then

$$\begin{aligned} x \downarrow_q &= (0, \ldots, 0, q-1, b_{k+1}, \ldots, b_{\lceil \log(n) \rceil + 1}) \\ &= (0, \ldots, 0, q-1, b'_{k+1}, \ldots, b'_{\lceil \log(n) \rceil + 1}) \\ &= y \downarrow_q . \end{aligned}$$

So from now on suppose $b'_j = b_j = 0$ or $b'_j = b_j \geq q$ for all $j \geq i$.
If $q \preceq b_i$ then $q \leq b_i$ and b_i is even since q is even. Then we have $(x \downarrow_q)_j = b_j$ and $(y \downarrow_q)_j = b'_j = b_j$ for all $j > i$ and $(y \downarrow_q)_i \in \{y_i, q-1\}$, so $(y \downarrow_q)_i \prec b_i$, since b_i is even, and therefore $x \downarrow_q \sqsupset x \downarrow_q$.
If $b'_i \preceq q+1$ then b'_i is an odd number $\geq q+1$. Thus, $(y \downarrow_q)_j = b'_j = b_j = (x \downarrow_q)_j$ for all $j > i$ and $(y \downarrow_q)_i = b'_i$. The only two possibilities for $x \downarrow_q$ are b_i and $q-1$. If $(x \downarrow_q)_i = b_i$ then $x \downarrow_q \sqsupset y \downarrow_q$. If $(x \downarrow_q)_i = q-1$ then $y \downarrow_{q_i} = b'_i > q-1$. Thus, $(y \downarrow_q)_i \prec (x \downarrow_q)_i$ and $x \downarrow_q \sqsupset y \downarrow_q$.
So the only remaining option is $q+1 \prec b_i \prec b'_i \prec q$. In this case it is impossible that $b_i \geq q$ or $b'_i \geq q$ by definition of $\preceq$, since q is even, and therefore $(x \downarrow_q)_i = q-1 = (y \downarrow_q)_i$, $(x \downarrow_q)_i = b_j = b'_j = (y \downarrow_q)_i$ for all $j > i$ and $(x \downarrow_q)_i = 0 = (y \downarrow_q)_i$ for all $j < i$.

2. If $z = x$ the claim is obvious. So assume $z \neq x$ and let $z = (a_0, \ldots, a_{\lceil \log(n) \rceil + 1})$ and i be the largest index such that $b_i \neq a_i$. As in 1. if there is an index $k > i$ such that $a_k = b_k \neq 0$ and $a_k = b_k < q$, then $x \downarrow_q = (0, \ldots, 0, q-1, b_{k+1}, \ldots, b_{\lceil \log(n) \rceil + 1}) = (0, \ldots, 0, q-1, b'_{k+1}, \ldots, b'_{\lceil \log(n) \rceil + 1}) = z \downarrow_q$. So suppose this is not the case.
Since $a_i = \text{val}(v)$ we have $a_i < q$ and since b_i was updated in the backwards update game at v we have $b_i \prec q$ by definition of $\prec$. Assume $b_i \preceq q+1$. Then b_i is a larger odd number than $q+1$ or equal to $q+1$. Hence $(x \downarrow_q)_i = b_i \preceq q+1 \prec q-1 \leq (z \downarrow_q)_i$ and thus $z \downarrow_q \sqsupset x \downarrow_q$. If on the other hand $q+1 \prec b_i$ then b_i is an odd number smaller than $q+1$ or an even number smaller than q, since $b_i \prec q$. In either case $b_i < q$ and therefore $x \downarrow_q = (0, \ldots, 0, q-1, b_{i+1}, \ldots, b_{\lceil \log(n) \rceil + 1}) = (0, \ldots, 0, q-1, b'_{i+1}, \ldots, b'_{\lceil \log(n) \rceil + 1}) = z \downarrow_q$.

3. This follows from 1. and 2..

4. Since q is even and x is not matured we have $z \neq x$. Let $z = (a_0, \ldots, a_{\lceil \log(n) \rceil + 1})$ and i be the largest index such that $b_i \neq a_i$. By definition of the update rules in the backwards update game we have $b_i \prec a_i = q$. If there was a maximal index $k > i$ such that $a_k = b_k \neq 0$ and $a_k = b_k < q$ then b_k would have been updated to q. It follows that $(z \downarrow_q)_i = a_i = q$. In both possible cases, namely $(x \downarrow_q)_i = q-1$ and $(x \downarrow_q)_i = b_i$, we have $(x \downarrow_q)_i \prec (z \downarrow_q)_i$ and therefore $z \downarrow_q \sqsupset x \downarrow_q$.

5. This follows from 1. and 4..

□

Now we can easily prove the missing direction of the equivalence.

2.3.12 Theorem: Let $\mathcal{G}$ be a parity game and π a play in $\mathcal{G}$. If Player 0 wins π then she has a winning strategy for the antagonistic update game on π.

Proof. Since the highest value m occurring infinitely often in π is even, there will be a point in π at which an even value q has occurred more often than the size of the image of $\downarrow_q \colon B \to B$ after the last value $c > q$ has occurred. Application of the previous lemma shows that Player 0 can then win the antagonistic update game since the fifth case occurs $|B|$ times with only the third case in between. Thus, the winning statistics in the antagonistic update game matures. □

Until now, we have only considered single plays and not the whole parity game. Calude et al. constructed a reachability game to get from single plays to the parity game. But Fearnley et al. now have all the tools necessary to give a lifting operator on functions of the from $\iota\colon V \to B$.

2.3.13 Definition: Let $\mathcal{G} = (V, V_0, V_1, E, \mathrm{val})$ be a parity game. For $v \in V$ and $\iota\colon V \to B$ we define

$$\mathrm{Lift}(\iota, v)(v') = \begin{cases} \iota(v'), & v' \neq v \\ \max_{\sqsubseteq}\{\mathrm{au}(\iota(w), v)\colon (v,w) \in E\}, & v = v' \in V_0 \\ \min_{\sqsubseteq}\{\mathrm{au}(\iota(w), v)\colon (v,w) \in E\}, & v = v' \in V_1. \end{cases}$$

To apply the Knaster-Tarski Theorem and get a fixed point, we need an ordering on the functions such that the lifting operator is monotone.

2.3.14 Remark: Let $\mathcal{G}$ be a parity game. Define an ordering $\sqsubseteq^*$ on the set of functions $V \to B$ by $\iota_1 \sqsubseteq^* \iota_2$ if and only if $\iota_1(v) \sqsubseteq \iota_2(v)$ for all $v \in V$ and $\iota_1 \sqsubset^* \iota_2$ if $\iota_1 \sqsubseteq^* \iota_2$ and $\iota_1 \neq \iota_2$. Then the operator $\mathrm{Lift}(\cdot, v)$ is $\sqsubseteq^*$-monotone for all $v \in V$.

Proof. Let $\iota_1 \sqsubseteq^* \iota_2$. We need to show that $\mathrm{Lift}(\iota_1, v) \sqsubseteq^* \mathrm{Lift}(\iota_2, v)$ or, since

$$\mathrm{Lift}(\iota_1, v)(v') = \iota_1(v') \sqsubseteq \iota_2(v') = \mathrm{Lift}(\iota_2, v)(v')$$

for all $v' \neq v$, that $\mathrm{Lift}(\iota_1, v)(v) \sqsubseteq^* \mathrm{Lift}(\iota_2, v)(v)$. From the proof of Theorem 2.3.9 and from the definition of $\sqsubseteq$ via $\preceq$ it follows that $\mathrm{au}(\iota_1(w), v) \sqsubseteq \mathrm{au}(\iota_2(w), v)$ for all $(v, w) \in E$. In particular this holds for the maximum and the minimum over such (v, w). □

By the Knaster-Tarski Theorem the following algorithm yields a least fixed point with respect to $\sqsubseteq^*$:

2.3.15 Algorithm (SuccinctCountingLifting)**:**

```
Initialize $\iota\colon V \to B,\ v \mapsto (0, \dots, 0, )$.
While $\iota(v') \sqsubset \mathrm{Lift}(\iota, v)(v')$ for some $v, v' \in V$ do $\iota := \mathrm{Lift}(\iota, v)$.
Return $W_0 = \{v \in V : \iota(v) \text{ is matured}\}$, $W_1 = V \backslash W_0$.
```

What remains to be shown is the correctness of the algorithm.

2.3.16 Theorem: Let $\mathcal{G}$ be a parity game and $v \in V$. Let $\iota\colon V \to B$ be the function resulting from SuccinctCountingLifting. If $\iota(v)$ is matured, then Player 0 has a winning strategy from v in $\mathcal{G}$.

Proof. Let ι_i denote the state of the function in SuccinctCountingLifting after i applications of the Lift-operator. Let k denote the least index at which $\iota_k = \iota$. Define $v_k = v$ and for each $v_j \in V_0$ let Player 0 select a neighbour v_{j-1} such that $\iota_j(v_j) \sqsubseteq \mathrm{au}(\iota_{j-1}(v_{j-1}), v_j)$. Such a successor exists because otherwise ι_{j-1} would not have been lifted. Similarly for all $v_j \in V_1$ we have $\iota_j(v_j) \sqsubseteq \mathrm{au}(\iota_{j-1}(w), v_j)$ for all $(v_j, w) \in E$. Let $\pi = v_k \cdots v_0$ be the initial part of a play played according to this strategy for Player 0.
Consider the antagonistic update game on π. Player 0 chooses to start at v_0 and we evaluate the play backwards. For each $0 \leq i \leq k$ let $x_i \in B$ be the winning statistic just before position v_i in the play. In particular $x_0 = (0, \ldots, 0)$. By definition of π we have $\iota_j(v_j) \sqsubseteq \mathrm{au}(\iota_{j-1}(v_{j-1}), v_j)$ for all $1 \leq j \leq k$. We will show by induction over j that $\iota_j(v_j) \sqsubseteq x_j$ for all $0 \leq j \leq k$.
Since $\iota_0(w) = (0 \cdots 0)$ for all $w \in V$ we have $\iota_0(v_0) = x_0$. Now let $\iota_j(v_j) \sqsubseteq x_j$ for some $j < k$. Then

$$\iota_{j+1}(v_{j+1}) \sqsubseteq \mathrm{au}(\iota_j(v_j), v_{j+1}) \sqsubseteq \mathrm{au}(x_j, v_{j+1}) = x_{j+1}.$$

Since $\iota(v) = \iota_k(v)$ is matured, so is x_k and Player 0 wins the antagonistic update game. It follows from Theorem 2.3.9 that Player 0 wins π. Since the strategy for Player 1 on π was unspecified, Player 0 wins $\mathcal{G}$ from v. $\square$

For the opposite direction, consider the following theorem.

2.3.17 Theorem: Let $\mathcal{G}$ be a parity game and $v \in V$. Let $\iota\colon V \to B$ be the function resulting from SuccinctCountingLifting. If $\iota(v)$ is not matured, Player 1 has a winning strategy from v in $\mathcal{G}$.

Proof. For each $z \in V_1$ let Player 1 choose $w \in V$ with $(z, w) \in E$ and $\iota(z) \sqsubseteq \mathrm{au}(\iota(w), z)$. This exists by assumption since otherwise ι would not be a fixed point.
Let $v_k = v$ and let $\pi = v_k v_{k-1} \cdots v_0$ be the initial part of a play played according to this strategy by Player 1. The length k is arbitrarily chosen by Player 0 at the beginning of the antagonistic update game. If $v_j \in V_0$ for some $1 \leq j \leq k$ we have

$$\iota(v_{j+1}) \sqsupseteq \max_{\sqsubseteq}\{\mathrm{au}(\iota(w), v_{j+1})\colon (v_{j+1}, w) \in E\} \sqsupseteq \mathrm{au}(\iota(v_j), v_{j+1}),$$

since otherwise ι would not be a least fixed point. Since $\iota(v_k) = \iota(v)$ is not matured it follows that neither is $\iota(v_j)$ for any $j < k$.
Now consider the antagonistic update game on π. Player 0 chooses to start at v_0 and we evaluate the play backwards. For each $0 \leq i \leq k$ let $x_i \in B$ be the winning statistic just before position v_i in the play. In particular $x_0 = (0, \ldots, 0)$. Then $x_0 \sqsubseteq \iota(v_0)$ and since au is monotonous in

the first element we have $x_j \sqsubseteq \iota(v_j)$ for all $0 \leq j \leq k$. Therefore, Player 0 does not win the antagonistic update game and by Theorem 2.3.12 does not win π. Since the strategy for Player 0 on π was unspecified, Player 1 wins $\mathcal{G}$ from v. □

Now that we have seen the correctness of the algorithm, let us come to the complexity.

2.3.18 Theorem: Let $\mathcal{G}$ be a parity game with k edges and n nodes. Then the algorithm SuccinctCountingLifting runs in time $\mathcal{O}(k \cdot |B|)$ and space $\mathcal{O}(n \cdot \log(|B|) + k \cdot \log(\log(|B|)))$.

Proof. For the space complexity we need $n \cdot \log(|B|)$ bits to store the current state of the function ι. Additionally we store for each node v for each incoming edge the position and bit whose increase would result in a lifting operation on this node. This gives us the stated space complexity.
For each vertex we can have at most $|B|$ lifting operations. The same goes for every edge: For every edge $(v, w) \in E$, the maximal number of lifting operations that take place on v or w is at most $|B|$. This gives us the time complexity. □

We have already calculated $|B|$ in the first section of this chapter, since B for the antagonistic update game is no different than the set of possible winning statistics for the original algorithm by Calude et al.:
We have

$$|B| \leq 2^{(\log(n)+3)\cdot(\log(m)+1)}$$

and therefore a time complexity of

$$\mathcal{O}(k \cdot 2^{\log(n)\cdot\log(m)+3\log(m)+\log(n)}) \text{ or } \mathcal{O}(k \cdot n^{\log(m)+4}),$$

i.e. quasipolynomial time, and a space complexity of

$$\mathcal{O}(n \cdot (\log(n) \cdot \log(m) + 3\log(m) + \log(n)) + k \cdot \log(\log(n) \cdot \log(m) + 3\log(m) + \log(n)).$$

The time bound is very similar to the bound $\mathcal{O}(kn^{\log m - \log\log n + 4{,}03})$ we get for the method by Jurdziński and Lazić, but the latter might be slightly faster for very large n.

Chapter 3

Solving Streett-Rabin Games in FPT

It is known that if parity games are in FPT, then so are Streett-Rabin games (see for instance [BSV03]). Since Calude et al. and Jurdziński and Lazić have shown that parity games are fixed parameter tractable, it follows that so are Streett-Rabin games.
In this chapter we will modify the succinct counting method to apply it to Streett-Rabin games in order to give FPT-algorithms for solving them. A major difference between parity games and Streett-Rabin games is that the latter are not positionally determined, so for the succinct counting method it does not suffice to track $n+1$ positions in a play to find a loop. Instead, we have to track $(m+1)!n$ positions, where n is the number of nodes and m the number of values, since at each node in V_0, there are at most $(m+1)!$ different states of the game, on which her strategy can depend (see LAR). If we define the Streett-Rabin game with pair conditions, we have to track $(k+2)!n$ positions, where k is the number of pairs (see IAR).

3.1 Succinct Counting for Streett Rabin Games with Muller Conditions

First, we will look at Streett-Rabin games defined by a Muller condition and try to adapt the succinct counting method to them.
Consider a Streett-Rabin game with n nodes and m colours and a winning condition given as $\mathcal{F}_0 \subseteq \mathcal{P}(\{1,\ldots,m\})$, $\mathcal{F}_1 = \mathcal{P}(\{1,\ldots,m\})\backslash\mathcal{F}_0$, where $\mathcal{F}_0$ is closed under union. Define $\mathcal{F}_{\max}$ as the maximal set in $\mathcal{F}_0$ with respect to inclusion. $\mathcal{F}_{\max}$ exists, because $\mathcal{F}_0$ is finite, and is uniquely defined, because $\mathcal{F}_0$ is closed under union.
Similarly to the succinct counting method for parity games, we are going to maintain winning statistics for Player 0 for a single play π and then use a reachability game to determine the winner of the Streett-Rabin game from a given starting position. The winning statistics has the

form

$$(b_0, \ldots, b_{\lceil \log((m+1)!n) \rceil + 1}, c_1, \ldots, c_m, c),$$

where $b_0, \ldots, b_{\lceil \log((m+1)!n) \rceil + 1}, c_1, \ldots, c_m \in \{0, 1\}$ and $c \in (m+1)!n + 1$.

We introduce the following update rules for a winning statistics that analyses a play π.

3.1.1 Update Rules: Let π now be a play of a given Streett-Rabin game $\mathcal{G}$. Then updating the winning statistics works as follows:

Initialization: $(b_0, \ldots, b_{\lceil \log((m+1)!n) \rceil + 1}, c_1, \ldots, c_m, c) = (0, \ldots, 0, 0, \ldots, 0, 0)$

Update Rules: Let v_i be the current node in the play with colour m_i.

- If $m_i \notin \mathcal{F}_{\max}$ or $c = (m+1)!n + 1$ then set

  $$(b_0, \ldots, b_{\lceil \log((m+1)!n) \rceil + 1}, c_1, \ldots, c_m, c) = (0, \ldots, 0, 0, \ldots, 0, 0),$$

  move on to node v_{i+1} and follow the update rules.
- Else, if $c_{m_i} = 1$ then $c = c + 1$ and if $c_{m_i} = 0$ then $c_{m_i} = 1$.
  - If $\{j \colon c_j = 1\} \notin \mathcal{F}_0$ then move on to node v_{i+1} and follow the update rules.
  - If $\{j \colon c_j = 1\} \in \mathcal{F}_0$ then set the b_j with the smallest index j such that $b_j = 0$ to 1 and all b_l with $l < j$ to 0. Additionally, update all $c_1, \ldots, c_{i-1}, c_{i+1}, \ldots, c_m$ and c to 0. If $b_{\lceil \log((m+1)!n) \rceil + 1} = 1$ we say that the winning statistics *matures* and stop, else move on to node v_{i+1} and follow the update rules.

The idea of these winning statistics is to find sequences of positions in π in order, but not necessarily consecutive, such that the set of values between every two nodes of the sequence (including these nodes) is in $\mathcal{F}_0$. Since $\mathcal{F}_0$ is closed under union, it suffices to guarantee this for any two consecutive elements of the sequence. Then we build up sequences of lengths powers of two, until we find one which is longer than $(m+1)!n$. The b_i mark the length of the sequences we are currently tracking and the c_i code which colours, i.e. which sets in $\mathcal{F}_0$, currently appear in that sequence. The counter c is necessary so we can discard a set $F \in \mathcal{F}_0$ if it turns out that it does not appear infinitely often.

3.1.2 Example: Consider the following very simple play in a Streett-Rabin game with three nodes v_1, v_2, v_3 with $\mathrm{val}(v_i) = i$ for all $i \in \{1, 2, 3\}$:

$$\pi = v_3\ v_2\ v_2\ v_2\ v_2\ v_2\ v_3\ v_3\ v_2\ v_2\ v_2\ v_2\ v_3\ v_2\ v_2\ v_2\ v_2\ v_2\ v_2\ v_2\ v_2\ v_2\ v_2\ v_2\ v_2\ v_2\ v_2\ v_2\ v_2\ \cdots .$$

The winning conditions are $(\mathcal{F}_0, \mathcal{F}_1)$ with

$$\mathcal{F}_0 = \{\{2\}, \{1,2,3\}\}, \qquad \mathcal{F}_1 = \mathcal{P}(\{1,2,3\})\backslash\mathcal{F}_0.$$

We have $(m+1)!n+1 = 4! \cdot 3 + 1 = 73$. Let us see the first few updates in a table (note that more b_i are necessary for the whole algorithm, but we do not need them for the first few steps we are going to look at):

Value	b_0	b_1	b_2	b_3	b_4	b_5	c_1	c_2	c_3	c
3	0	0	0	0	0	0	0	1	0	0
2	0	0	0	0	0	0	0	1	1	0
2	0	0	0	0	0	0	0	1	1	1
2	0	0	0	0	0	0	0	1	1	2
2	0	0	0	0	0	0	0	1	1	3
2	0	0	0	0	0	0	0	1	1	4
3	0	0	0	0	0	0	0	1	1	5
...										
2	0	0	0	0	0	0	0	1	1	73
	0	0	0	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	0	1	0
2	0	1	0	0	0	0	0	0	1	0
2	1	1	0	0	0	0	0	0	1	0
2	0	0	1	0	0	0	0	0	1	0
2	1	0	1	0	0	0	0	0	1	0
2	0	1	1	0	0	0	0	0	1	0
2	1	1	1	0	0	0	0	0	1	0
2	0	0	0	1	0	0	0	0	1	0
2	1	0	0	1	0	0	0	0	1	0
2	0	1	0	1	0	0	0	0	1	0
2	1	1	0	1	0	0	0	0	1	0
2	0	0	1	1	0	0	0	0	1	0
...										

Note that at first, the algorithm tries to track occurrences of all elements in the set $\{1,2,3\}$, but since 1 never appears, it has to discard this set after the counter c reaches its maximum. Then it switches over to tracking the occurrences of elements in the set $\{2\}$ successfully.

3.1.3 Theorem: Let π be played according to a positional winning strategy for Player 1 and according to a winning strategy using LAR for Player 0. Player 0's winning statistics matures if and only if Player 0 wins π.

Proof. We first show that if Player 0's winning statistics matures then Player 0 wins π. To prove this, we claim as for the winning statistics for parity games that for each $b_i = 1$ there is a sequence $(a_1, \ldots, a_{2^i})$ of length i, beginning after those of the $b_j = 1$, $j > i$, end, of nodes of π in order, but not necessarily consecutive, such that for each section $v_k = a_l, v_{k+1}, \ldots, v_{k+j-1}, v_{k+j} = a_{l+1}$ of π we have $\{\mathrm{val}(v_k), \ldots, \mathrm{val}(v_{k+j})\} \in \mathcal{F}_0$. It follows from the closure under union that for any a_i, a_j, $i < j$, the set of values occurring between a_i and a_j, including those of a_i and a_j, is in $\mathcal{F}_0$. The claim is true because we can choose as a_k those nodes of the play, at which $\{c_j \colon c_j = 1\} \in \mathcal{F}_0$. This means that the set of values occurring since the last a_j is in $\mathcal{F}_0$. Additionally, for any b_j there are exactly 2^j such nodes from the last point where all b_i, $i < j$, were 0 to the point where b_j is set to 1.
If the winning statistics matures, then $b_{\lceil \log((m+1)!n)\rceil+1} = 1$. It follows that there is a sequence $(a_1, \ldots, a_{2^{\lceil \log((m+1)!n)\rceil+1}})$ with the properties described above. Thus, there have to be a_i, a_j, $i < j$, such that a_i and a_j denote the same node and if $a_i \in V_0$ then the LAR at a_i is the same as at a_j. It follows, that there is a loop in π, infinitely repeating the sequence of nodes between (including) a_i and a_j. Thus, Player 0 wins π by our considerations above.
Now assume that Player 0 wins π. Let $\{m_{i_1}, \ldots, m_{i_k}\} \in \mathcal{F}_0$ be the set of values occurring infinitely often in π. Then there is a node v_j in π such that for any $l \geq j$, $\mathrm{val}(v_l) \in \{m_{i_1}, \ldots, m_{i_k}\}$ and all values $m_{i_1}, \ldots, m_{i_k}$ occur infinitely often after v_j.

Case 1: $(c_1, \ldots, c_m) = (0, \ldots, 0)$ or $\{j \colon c_j = 1\} \subseteq \{m_{i_1}, \ldots, m_{i_k}\}$ at v_j.

In this case the winning statistics will collect only values from the set $\{m_{i_1}, \ldots, m_{i_k}\}$ and thus only update the values $c_{i_1}, \ldots, c_{i_k}$. First, we note that the case $c = (m+1)!n + 1$ will never occur, since if it did, it would mean that there is a loop containing not all the values $\{m_{i_1}, \ldots, m_{i_k}\}$ in π which is a contradiction to the assumption. Thus the rules for the winning statistics keep updating $c_{i_1}, \ldots, c_{i_k}$ until $\{j \colon c_j = 1\} \subseteq \{m_{i_1}, \ldots, m_{i_k}\}$ is in $\mathcal{F}_0$ and the b_i are updated positively, meaning that the number of updates minimally necessary to achieve $b_{\lceil \log((m+1)!n)\rceil+1} = 1$ has shrunk by one. This will happen at some point, because in the worst case, we can update the $c_{i_1}, \ldots, c_{i_k}$ until they are all 1, which is possible, since c will never turn $(m+1)!n + 1$.

Let $v_{j'}$ be the node of π at which this update happens. Then again we are in a case where for any $l \geq j'$, $\mathrm{val}(v_l) \in \{m_{i_1}, \ldots, m_{i_k}\}$, all values $m_{i_1}, \ldots, m_{i_k}$ occur infinitely often after $v_{j'}$ and $\{j \colon c_j = 1\} \subseteq \{m_{i_1}, \ldots, m_{i_k}\}$. Thus, the considerations above apply again and again and therefore we will reach a point, where $b_{\lceil \log((m+1)!n)\rceil+1}$ is updated to 1.

Case 2: $\{m_j \colon c_j = 1\} \nsubseteq \{m_{i_1}, \ldots, m_{i_k}\}$ at v_j.

Let $\{i \colon c_i = 1\} = \{m_{j_1}, \ldots, m_{j_l}\}$. In this case there are two possibilities. The first is, that there is an $a \geq 0$ such that $\{m_{j_1}, \ldots, m_{j_l}\} \cup \{\mathrm{val}(v_j), \ldots, \mathrm{val}(v_{j+a})\} \in \mathcal{F}_0$. Without loss of generality, assume that a is minimal such that this is the case. In this case, $c = (m+1)!n+1$ will not occur before or at v_{j+a}, because otherwise there would be a loop in π missing at least one of the values in $\{m_{i_1}, \ldots, m_{i_k}\}$, which is a contradiction. Thus, at v_{j+a} the rules for the winning statistics update the b_i positively (in the sense described above). At the

node v_{j+a} we are in a situation, where for any $i \geq j+a$, $\text{val}(v_i) \in \{m_{i_1}, \dots, m_{i_k}\}$, all values $m_{i_1}, \dots, m_{i_k}$ occur infinitely often after v_{j+a} and $\{j \colon c_j = 1\} \subseteq \{m_{i_1}, \dots, m_{i_k}\}$. Thus, we can apply the consideration of Case 1 and arrive at a point, where $b_{\lceil \log((m+1)!n) \rceil + 1}$ is updated to 1.
So let us assume that there is no $a \geq 0$ such that $\{m_{j_1}, \dots, m_{j_l}\} \cup \{\text{val}(v_j), \dots, \text{val}(v_{j+a})\} \in \mathcal{F}_0$. In this case, the winning statistics will reach a point where

$$\{i \colon c_i = 1\} = \{m_{j_1}, \dots, m_{j_l}\} \cup \{m_{i_1}, \dots, m_{i_k}\}.$$

($c = (m+1)!n + 1$ cannot occur before this occurs for the first time, for the same reason as above.) From that point on, at each value $m_i \in \{m_{i_1}, \dots, m_{i_k}\}$, we have that $c_i = 1$ already. Thus, c is increased by one. It follows that c will reach $(m+1)!n + 1$ eventually, say at $v_{j'}$, and after following the update rule for this case, we are at a point, where for any $l \geq j'$, $\text{val}(v_l) \in \{m_{i_1}, \dots, m_{i_k}\}$, all values $m_{i_1}, \dots, m_{i_k}$ occur infinitely often after $v_{j'}$ and $\{j \colon c_j = 1\} \subseteq \{m_{i_1}, \dots, m_{i_k}\}$. Thus, again, we can apply our considerations from case one and arrive at a point, where $b_{\lceil \log((m+1)!n) \rceil + 1}$ is updated to 1.

□

As with parity games, we can now define a reachability game

3.1.4 Definition: Given a Streett-Rabin game $\mathcal{G}$ we define the reachability game $\tilde{\mathcal{G}}$ in the following way: The nodes are of the form $(a, \tilde{b})$ with

- a a node of the Streett-Rabin game and
- $\tilde{b}$ the winning statistics for Player 0.

The starting node is $(s, (0, \dots, 0))$ with s the starting node of the Streett-Rabin game.
Player 0 can move from $(a, \tilde{b})$ to $(a', \tilde{b}')$ if $a \in V_0$, if she can move from a to a' in the Streett-Rabin game and this causes her winning statistics to update from $\tilde{b}$ to $\tilde{b}'$. Player 1 can move from $(a, \tilde{b})$ to $(a', \tilde{b}')$ if $a \in V_1$, if he can move from a to a' in the Streett-Rabin game and this causes Player 0's winning statistics to update from $\tilde{b}$ to $\tilde{b}'$.
Player 0 is declared the winner of a play $\tilde{\pi}$ in the reachability game, if a node with a matured winning statistics is reached.

The correctness follows the same way as it did for parity games.

3.1.5 Theorem: Player 0 wins the Streett-Rabin game $\mathcal{G}$ if and only if she wins the reachability game $\tilde{\mathcal{G}}$ defined above.

Let us now take a look at the complexity of the algorithm. The reachability game can be solved in time $\mathcal{O}(|Q| \cdot n)$, where Q is the set of nodes in the reachability game, since this is the maximal

number of edges (and a higher number than nodes) it can have. Hence, we need to calculate $|Q|$:
Let $v = (a, \tilde{b})$ be a node of the reachability game, where $\tilde{b} = (b_0, \ldots, b_{\lceil \log((m+1)!n) \rceil + 1}, c_1, \ldots, c_m, c)$. There are n possible values for a, if we number the nodes from 1 to n. Additionally, there are two possible values for each of the $b_0, \ldots, b_{\lceil \log((m+1)!n) \rceil + 1}$ and each of the $c_1, \ldots, c_m$ and $(m+1)!n+1$ possible values for c. Hence

$$|Q| \leq n \cdot 2^{\lceil \log((m+1)!n) \rceil + 2} \cdot 2^m \cdot ((m+1)!n + 1)$$

It follows that the complexity of solving the reachability game is in $\mathcal{O}(n^4 \cdot (2m+2)!)$, which is in FPT.

3.2 Succinct Counting for Pair Conditions

Now we consider a Streett-Rabin game given with the pair condition $(G_i, F_i)_{1\leq i\leq k}$ with $G_i, F_i \subseteq \{1, \ldots, m\}$ for all $i \leq k$. The additional difficulty in defining a winning statistics in this scenario, compared to the above one, is that the pairs (G_i, F_i) do not tell us exactly what the set of values appearing infinitely often should look like. Therefore, the winning statistics will be defined in a different way. The first difference is that we are tracking the winning statistics for Player 1. Secondly, part of the winning statistics will be a permutation, allowing us to circle through the pairs.

The winning statistics has the form

$$(b_0, \ldots, b_{\lceil\log((k+2)!n)\rceil+1}, b, c_1, \ldots, c_k, c),$$

where

- $b_0, \ldots, b_{\lceil\log((k+2)!n)\rceil+1} \in \{0, 1\}$,
- $b \in \{0, \ldots, k\}$,
- $c_1, \ldots, c_k \in \{1, \ldots, k\}$,
- $c \in \{0, \ldots, (k+2)!n + 1\}$.

Consider the following update rules.

3.2.1 Update Rules: Let π be a play of a Streett-Rabin game $\mathcal{G}$ with winning condition $(G_i, F_i)_{1\leq i\leq k}$. Then updating the winning statistics works as follows:

Initialization: $(b_0, \ldots, b_{\lceil\log((k+2)!n)\rceil+1}) = (0, \ldots, 0)$, $b = 0$, $(c_1, \ldots, c_k) = (1, \ldots, k)$, $c = 0$.

Update Rules: Let v_i be the current node in the play with value m_i.

Case 1: There exists $b_j > 0$, $b = l$ and $m_i \in G_l \backslash (F_l \cup \bigcup_{a\leq k:\, c_a<c_l}(G_a\backslash F_a))$.

- Set the b_a with the smallest index such that $b_a = 0$ to 1 and all b_j with $j < a$ to 0.
- Set $c = 0$.
- If $b_{\lceil\log((k+2)!n)\rceil+1} > 0$ we say that the winning statistics *matures* and stop, otherwise move on to v_{i+1}.

Case 2: There exists $b_j > 0$, $b = l$ and $m_i \in F_l \backslash \bigcup_{a\leq k:\, c_a\leq c_l}(G_a\backslash F_a)$.

- Set all b_a to 0 and b to 0.
- Set $c = 0$.
- Let $c_l = x$. Set $c_l = k$ and $c_y = c_y - 1$ for all $y > x$.
- Move on to v_{i+1}.

Case 3: $b_j = 0$ for all j.

- Let l be the smallest index such that $m_i \in G_l \backslash F_l$. If this does not exist, go to v_{i+1}, otherwise set $b_0 = 1$, $b = l$ and go to v_{i+1}.

Case 4: There exists $b_j > 0$, $b = l$ and $m_i \in G_a \backslash F_a$ for some $c_a < c_l$.

- Set $b_x = 0$ for all $x > 0$ and $b_0 = 1$, $b = a$.
- Set $c = 0$.
- Move on to v_{i+1}.

Case 5: There exists $b_j > 0$, $b = l$ and $m_i \notin G_l \cup F_l \cup \bigcup_{a \leq k:\, c_a < c_l} (G_a \backslash F_a))$ or ($b_j = 0$ for all j and $m_i \notin G_a \backslash F_a$ for all $a \in \{1, \dots, k\}$).

- Set $c = c + 1$.
- If $c = (k+2)!n + 1$ set $b_a = 0$ for all a, $b = 0$ and $c = 0$. Let $c_l = x$. Set $c_l = k$ and $c_y = c_y - 1$ for all $y > x$.
- Move on to v_{i+1}.

The idea is the following: Again we look for sequences of positions in π, in order but not necessarily consecutive. This time, we want that each value of a node in the sequence is in a fixed G_i and there is no value occurring in π between the first and the last element of the sequence which is in F_i. Then we add up sequences of lengths powers of two.
The b_i mark the lengths of the sequences currently tracked and b marks the pair currently considered. The counter c again makes sure we can discard a pair (G_i, F_i) if we realise, that no value in G_i will occur infinitely often. The permutation $(c_1, \dots, c_k)$ allows us to rotate through the pairs once we have discarded one pair, so that on the one hand, we get to consider each pair and on the other hand, we do not discard a pair for good, if a value in F_i occurs only in an initial sequence of π.

3.2.2 Example: Consider the same play as in the previous example, again in a Streett-Rabin game with three nodes v_1, v_2, v_3 with $\text{val}(v_i) = i$ for all $i \in \{1, 2, 3\}$:

$$\pi = v_3\ v_2\ v_2\ v_2\ v_2\ v_2\ v_3\ v_3\ v_2\ v_2\ v_2\ v_2\ v_3\ v_2\ v_2\ v_2\ v_2\ v_2\ v_2\ v_2\ v_2\ v_2\ v_2\ v_2\ v_2\ v_2\ v_2\ v_2\ v_2\ \cdots .$$

The winning conditions this time are $((\{1,3\}, \{2\}), (\{2,3\}, \{1\}))$. Let us see the first few updates in a table (note that more b_i are necessary for the whole algorithm, but we do not need them for the first few steps we are going to look at):

Value	b_0	b_1	b_2	b_3	b_4	b_5	b	c_1	c_2	c
3	1	0	0	0	0	0	1	1	2	0
2	0	0	0	0	0	0	0	2	1	0
2	1	0	0	0	0	0	2	2	1	0
2	0	1	0	0	0	0	2	2	1	0
2	1	1	0	0	0	0	2	2	1	0
2	0	0	1	0	0	0	2	2	1	0
3	1	0	1	0	0	0	2	2	1	0
3	0	1	1	0	0	0	2	2	1	0
2	1	1	1	0	0	0	2	2	1	0
2	0	0	0	1	0	0	2	2	1	0
2	1	0	0	1	0	0	2	2	1	0
2	0	1	0	1	0	0	2	2	1	0
3	1	1	0	1	0	0	2	2	1	0
2	0	0	1	1	0	0	2	2	1	0
2	1	0	1	1	0	0	2	2	1	0
2	0	1	1	1	0	0	2	2	1	0
2	1	1	1	1	0	0	2	2	1	0
2	0	0	0	0	1	0	2	2	1	0
...										

The algorithm starts by considering the pair $(\{1,3\},\{2\})$ but in the second step the value 2 occurs. Then the permutation is changed so that the second pair is prioritised over the first one and we can start building up favourable sequences.

3.2.3 Theorem: Let π be a play of a Streett-Rabin game $\mathcal{G}$ with winning condition $(G_i, F_i)_{1\leq i\leq k}$ played with a positional winning strategy for Player 1 and a winning strategy using IAR for Player 0. Then Player 1's winning statistics matures if and only if Player 1 wins π.

Proof. We show first that if the winnings statistics matures, then Player 1 wins the Streett-Rabin game. We note that b is only set to 0 when all b_i are, so let $b = l$. We claim that whenever $b_i = 1$ for some i, there is a sequence $(a_1, \ldots, a_{2^i})$ of length i, beginning after those of the $b_j = 1$, $j > i$, end, of nodes of π in order, but not necessarily consecutive such that for each section $v_x = a_l, v_{x+1}, \ldots, v_{x+j-1}, v_{x+j} = a_{l+1}$ of π we have $\{\text{val}(v_x), \ldots, \text{val}(v_{x+j})\} \cap G_l \neq \emptyset$ and $\{\text{val}(v_x), \ldots, \text{val}(v_{x+j})\} \cap F_l = \emptyset$.
The claim is true because we can choose as a_x those nodes of the play at which the value m_i is in $G_l \backslash (F_l \cup \bigcup_{a\leq k:\, c_a<c_l}(G_a\backslash F_a))$. This means that the set of values occurring since the last a_j (or the value of a_x, if $x = 0$) has a nonempty intersection with G_l and an empty intersection with F_l. The first is true because $\text{val}(a_x) \in G_l$ and the latter because otherwise Case 2 or 4 would have occurred before a_x. Additionally, for any b_j there are exactly 2^j such nodes from

the last point where all b_i, $i < j$, where 0 to the point where b_j is set to 1.
If $b_{\lceil\log((k+2)!n)\rceil+1} = 1$, $b = l$, there is a sequence $(a_1, \ldots, a_{2^{\lceil\log((k+2)!n)\rceil+1}})$ such that for each section $v_x = a_l, v_{x+1}, \ldots, v_{x+j-1}, v_{x+j} = a_{l+1}$ of π we have $\{\mathrm{val}(v_x), \ldots, \mathrm{val}(v_{x+j})\} \cap G_l \neq \emptyset$ and $\{\mathrm{val}(v_x), \ldots, \mathrm{val}(v_{x+j})\} \cap F_l = \emptyset$. Since Player 1 is following a positional winning strategy and Player 0 a strategy with a memory of $(k+2)!$ possible configurations, there is a loop in π forever repeating the sequence of nodes between some a_x and a_j. This means that the set of values that occur between a_x and a_j, including theirs, is the set of values occurring infinitely often and it has nonempty intersection with G_l and empty intersection with F_l. Thus, by definition of the pair condition as winning condition, Player 1 wins π.
Now we show that if Player 1 wins π, the winning statistics matures. If Player 1 wins π, there exist $l \in \{1, \ldots, k\}$, $v_j \in \pi$ with $\mathrm{val}(v_j) \in G_l$ such that all values occurring after v_j, including $\mathrm{val}(v_j)$, occur infinitely often and no values from F_l occur after v_j in π.

Case a): $b = l$ or $b = 0$ at v_j.

Let $\{G_{i_1}, \ldots, G_{i_x}\}$ be the set of first elements in the pair condition such that after v_j there are nodes $v_{j_1}, \ldots, v_{j_x}$ in $G_{i_1}, \ldots, G_{i_x}$, respectively, in π. Let us first assume, that c_l is minimal among the $c_{i_1}, \ldots, c_{i_x}$. Note that $b = 0$ implies $b_i = 0$ for all i and therefore at v_j, the update in Case 3 will be triggered, which leads to $b = l$. Under the assumption, cases 2 and 4 for updating the winning statistics will never occur after v_j. Additionally, c will never be $(k+2)!n+1$, since every occurrence of $\mathrm{val}(v_j)$ after v_j sets c to 0 by triggering the update in Case 1 and if $c = (k+2)!n+1$ were to be the case, then the update in Case 5 would have to have been triggered $(k+2)!n+1$ times without an occurrence of $\mathrm{val}(v_j)$, which means there is a loop without $\mathrm{val}(v_j)$ in it, a contradiction.
Thus, Case 1 will be applied at every node after v_j with value $\mathrm{val}(v_j)$, without the b_i being all set to 0 in between. Hence, at some point $b_{\lceil\log((k+2)!n)\rceil+1}$ will be set to 1.
Now assume that there is $l \neq a \in \{i_1, \ldots, i_x\}$ such that $c_a < c_l$. Let $v_{j'}$ be the first node after (including) v_j such that $\mathrm{val}(v_{j'}) \in G_{i_{j'}}$ and $c_{i_{j'}} < c_l$. Then the update in Case 4 will be triggered. If no value in $F_{i_{j'}}$ occurs after $v_{j'}$, then we can just replace v_j in our considerations with $v_{j'}$. (We might have to repeat such a replacement a few times, whenever Case 4 is triggered, but since $\{1, \ldots, k\}$ is finite, eventually the assumption in the previous consideration will be the case.) So let us assume that $\mathrm{val}(v_{i'}) \in F_{i_{j'}}$ for some $v_{i'}$ occurring after $v_{j'}$. If a node with a value in G_a occurs, such that $c_a < c_{i_{j'}}$, just repeat the considerations in this paragraph for that node. So without loss of generality (since $\{1, \ldots, k\}$ is finite and the order is only changed in cases 2 and 5), assume that this does not occur until the first node $v_{i'}$ with $\mathrm{val}(v_{i'}) \in F_{i_{j'}}$ is reached. Note that $c = (k+2)!n+1$ cannot occur before this, since otherwise there would be a loop without $\mathrm{val}(v_{i'})$ after $v_{j'}$. Now at $v_{i'}$, the update in Case 2 is triggered. This means that the b_i, b and c are set to zero and $i_{j'}$ is moved to the last spot in the ordering by setting $c_{i_{j'}}$ to k, while the order within the other $k-1$ indices of pairs stays the same. At some point after this, another node with value $\mathrm{val}(v_j)$ will occur, putting us back at the beginning of Case a) or b) by

replacing v_j with that node, but this time, l is at least one spot lower in the ordering than before. Hence, the considerations in this paragraph are only applied finitely many times.

Case b): $b = a$ at v_j and $c_a > c_l$
Here, Case 4 for the updates applies and b will be updated to l. The considerations of Case a) then apply in the same way as described above.

Case c): $b = a$ at v_j and $c_a < c_l$
Assume first that there is a node v_i after v_j in π such that $\mathrm{val}(v_i) \in F_a$. If c reaches $(k+2)!n + 1$ before the first such node is reached the updates in Case 5 are triggered, meaning that the b_i, b and c are set to 0 and c_a is set to k, while the order within the other c_i stays the same. Then the update rules move through the nodes, according to Case 5, until Case 3 applies for the first time at some node $v_{i'}$. c will not reach $k!kn+1$ until then, since otherwise there would be a loop after v_i not containing $\mathrm{val}(v_j)$, to which Case 3 applies, a contradiction. Let a' be the index with the smallest $c_{a'}$ such that $\mathrm{val}(v_{i'}) \in G_{a'} \backslash F_{a'}$. If the set of values occurring after $v_{i'}$ has nonempty intersection with $G_{a'}$ and empty intersection with $F_{a'}$, we can apply the considerations of Case a) with $v_j = v_{i'}$. Otherwise, $\mathrm{val}(v_j)$ will be reached under the conditions of Case a), b) or c) only with the value of c_l being lowered by at least one in comparison to the previous application of the considerations in these cases. Hence, we will have to repeat these considerations only a finite amount of times until we can apply the first paragraph of Case a).

□

Again we define a reachability game.

3.2.4 Definition: Given a Streett-Rabin game $\mathcal{G}$ we define the reachability game $\tilde{\mathcal{G}}$ in the following way: The nodes are of the form $(a, \tilde{b})$ with

- a a node of the Streett-Rabin game and
- $\tilde{b}$ the winning statistics for Player 1.

The starting node is $(s, (0, \dots, 0, 1, \dots, k, 0))$ with s the starting node of the Streett-Rabin game. Player 0 can move from $(a, \tilde{b})$ to $(a', \tilde{b}')$ if $a \in V_0$, if she can move from a to a' in the Streett-Rabin game and this causes Player 1's winning statistics to update from $\tilde{b}$ to $\tilde{b}'$. Player 1 can move from $(a, \tilde{b})$ to $(a', \tilde{b}')$ if $a \in V_1$, if he can move from a to a' in the Streett-Rabin game and this causes his winning statistics to update from $\tilde{b}$ to $\tilde{b}'$.
Player 1 is declared the winner of a play $\tilde{\pi}$ in the reachability game, if a node with a matured winning statistics is reached.

3.2.5 Theorem: Player 1 wins the Streett-Rabin game $\mathcal{G}$ if and only if he wins the reachability game $\tilde{\mathcal{G}}$ defined above.

The reachability game can be solved in time $\mathcal{O}(|Q| \cdot n)$, where Q is the set of nodes in the reachability game, since this is the maximal number of edges (and a higher number than nodes) it can have. Hence, we need to calculate $|Q|$:
Let $v = (a, \tilde{b})$ be a node of the reachability game, where $\tilde{b} = (b_0, \ldots, b_{\lceil \log((k+2)!n) \rceil + 1}, b, c_1, \ldots, c_k, c)$. There are n possible values for a, if we number the nodes from 1 to n. Additionally, there are two possible values for each of the $b_0, \ldots, b_{\lceil \log((k+2)!n) \rceil + 1}$, k possible values for b and each of the $c_1, \ldots, c_k$ and $(k+2)!n + 1$ possible values for c. Hence

$$|Q| \leq n \cdot 2^{\lceil \log((k+2)!n) \rceil + 2} \cdot k^{k+1} \cdot ((k+2)!n + 1)$$

It follows that the complexity of solving the reachability game is in $\mathcal{O}(n^4 \cdot (3k)!)$, which is in FPT.

Chapter 4

Applications to the Modal μ-Calculus

In this chapter we will apply the quasipolynomial time algorithms from Chapter 2 to model checking games of the modal μ-calculus. These games are parity games, as we have seen, and we can use some properties of the game graphs to our advantage, namely that only at positions with modal operators we have more than two outgoing edges. Before we begin, we modify the model checking game slightly by adding for each node of the form (P_i, w) an edge $((P_i, w), (P_i, w))$ and change the value of (P_i, w) to the largest even value occurring, if $w \in P_i$, and to the largest odd value occurring, otherwise. This guarantees that all plays are infinite and obviously does not change the winner of the model checking game.

4.1 Solving the Model Checking Game with Succinct Counting

We start by applying the succinct counting methods to a model checking game.
Let $\mathcal{K} = (W, (E_a)_{a \in A}, (P_i)_{i \in I})$ be a Kripke structure and let $n_{\mathcal{K}} = |W|$ be the number of worlds. Let $\psi \in \mathrm{L}_\mu$ be a formula with alternation depth m and with k modal operators $\langle a \rangle$ or $[a]$ for some $a \in A$. Then in the model checking game $\mathcal{G}$ a node (ϕ, w) with ϕ a subformula of ψ and $w \in W$ has at most $n_{\mathcal{K}}$ successors if it is of the form $\phi = \langle a \rangle \phi'$ or $\phi = [a]\phi'$ and at most two successors otherwise. This allows us to make the following complexity considerations.

4.1.1 Lemma: Let $\mathcal{K}$ and ψ be as described above, n_ψ the length of ψ and $\mathcal{G}$ the model checking game. Denote by $\mathcal{G}' = (Q, E)$ the reachability game for the Succinct Counting method for the parity game $\mathcal{G}$. Then

$$\begin{aligned}|Q| &\leq 2^{\lceil \log(n_{\mathcal{K}} \cdot n_\psi) \rceil + (\log(n_{\mathcal{K}} \cdot n_\psi)+3)\cdot(\log(m)+1)} \leq 2^4 n_{\mathcal{K}} \cdot n_\psi \cdot m^3 \cdot (n_{\mathcal{K}} n_\psi)^{\log(m)+1} \\ &\leq 2^4 \cdot n_{\mathcal{K}}^{\log(m)+2} \cdot n_\psi^{\log(m)+5}.\end{aligned}$$

Proof. This follows from the proof of Theorem 2.1.18 and the fact that $\mathcal{G}$ has at most $n_{\mathcal{K}} \cdot n_\psi$

nodes and m different values, where $m \leq n_\psi$. □

4.1.2 Theorem: Let $\mathcal{K}$ and ψ be as described above, n_ψ the length of ψ and $\mathcal{G}$ the model checking game. Again, k denotes the number modal operators in ψ, i.e. the number of subformula of the form $\phi = \langle a \rangle \phi'$ or $\phi = [a]\phi'$. The model checking game can be decided in time

$$\mathcal{O}(n_{\mathcal{K}}^{\log(m)+2} \cdot n_{\psi}^{\log(m)+4} \cdot (k + n_\psi)).$$

Proof. Denote by $\mathcal{G}' = (Q, E)$ the reachability game for the succinct counting method for the parity game $\mathcal{G}$. The reachability game can be solved in time $\mathcal{O}(|Q| + |E|)$. Since each node has an outgoing edge, it can be solved in time $\mathcal{O}(|E|)$. Since the winning statistics consists of $\lceil \log(n_{\mathcal{K}} \cdot n_\psi) \rceil + 2 \leq \log(n_{\mathcal{K}} \cdot n_\psi) + 3$ numbers represented by $\lceil \log(m) \rceil \leq \log(m) + 1$ bits we have

$$\lceil \log(k) \rceil + (\log(n_{\mathcal{K}} \cdot n_\psi) + 3) \cdot (\log(m) + 1)$$

bits to represent the nodes of the form $((\langle a \rangle \phi, w), b)$ or $(([a]\phi, w), b)$. Thus there are at most

$$2^{\lceil \log(k) \rceil + (\log(n_{\mathcal{K}} \cdot n_\psi) + 3) \cdot (\log(m)+1)} \leq 2^4 k \cdot m^3 \cdot (n_{\mathcal{K}} n_\psi)^{\log(m)+1} \leq 2^4 k \cdot n_{\mathcal{K}}^{\log(m)+1} \cdot n_\psi^{\log(m)+4}$$

nodes of this form in Q. Thus the number of edges is

$$\begin{aligned}
|E| &\leq (2^4 k \cdot n_{\mathcal{K}}^{\log(m)+1} \cdot n_\psi^{\log(m)+4}) \cdot n_{\mathcal{K}} + 2(|Q| - 2^4 k \cdot n_{\mathcal{K}}^{\log(m)+1} \cdot n_\psi^{\log(m)+4}) \\
&\leq (2^4 k \cdot n_{\mathcal{K}}^{\log(m)+1} \cdot n_\psi^{\log(m)+4}) \cdot n_{\mathcal{K}} + 2(2^4 \cdot n_{\mathcal{K}}^{\log(m)+2} \cdot n_\psi^{\log(m)+5} - 2^4 k \cdot n_{\mathcal{K}}^{\log(m)+1} \cdot n_\psi^{\log(m)+4}) \\
&= (2^4 k \cdot n_{\mathcal{K}}^{\log(m)+1} \cdot n_\psi^{\log(m)+4}) \cdot (n_{\mathcal{K}} - 2) + 2^5 \cdot n_{\mathcal{K}}^{\log(m)+2} \cdot n_\psi^{\log(m)+5} \\
&\leq 2^5 k \cdot n_{\mathcal{K}}^{\log(m)+2} \cdot n_\psi^{\log(m)+4} + 2^5 \cdot n_{\mathcal{K}}^{\log(m)+2} \cdot n_\psi^{\log(m)+5} \\
&= 2^5 \cdot n_{\mathcal{K}}^{\log(m)+2} \cdot n_\psi^{\log(m)+4} \cdot (k + n_\psi).
\end{aligned}$$

It follows that the complexity of solving $\mathcal{G}'$ and therefore of solving $\mathcal{G}$ is in

$$\mathcal{O}(n_{\mathcal{K}}^{\log(m)+2} \cdot n_{\psi}^{\log(m)+4} \cdot (k + n_\psi)).$$

□

4.2 Solving the Model Checking Game with Succinct Progress Measures

Using succinct progress measure, we can give an algorithm for solving model checking games for the modal μ-calculus by making the algorithm for solving parity games more precise.
Recall the algorithm SuccinctProgressMeasureLifting:

Initialise $\mu\colon V \to B^{\top}_{\lceil \log n \rceil, \lceil m/2 \rceil}$, $v \mapsto (0, \ldots, 0, \epsilon, \ldots, \epsilon)$
while $\mu \sqsubset \mathrm{Lift}(\mu, v)$ for some $v \in V$ do $\mu := \mathrm{Lift}(\mu, v)$.

The algorithm does not specify, which $v \in V$ to choose for each lifting operation. We will devise a protocol which $v \in V$ to use in order to obtain an algorithm for solving the model checking game.

As in the previous section let $\mathcal{K} = (W, (E_a)_{a\in A}, (P_i)_{i\in I})$ be a Kripke structure and let $n_{\mathcal{K}} = |W|$ be the number of worlds. Let $v \in W$ and let $\psi \in \mathrm{L}_\mu$ be a formula with alternation depth m and with k modal operators $\langle a \rangle$ or $[a]$ for some $a \in A$. Let $\mathcal{G} = (V, V_0, V_1, E, \mathrm{val})$ be the model checking game for $\mathcal{K}$, v and ψ as defined in Definition 1.6.14 in Chapter 1.
Again, we modify the model checking game slightly by adding for each node of the form (P_i, w) an edge $((P_i, w), (P_i, w))$ and change the value of (P_i, w) to the largest even value occurring if $w \in P_i$, and to the largest odd value occurring otherwise.

4.2.1 Algorithm:

```
Initialize $\mu\colon V \to B^{\top}_{\lceil \log n \rceil, \lceil m/2 \rceil}$, $(\phi, w) \mapsto (0, \ldots, 0, \epsilon, \ldots, \epsilon)$;
while $\mu \sqsubset \mathrm{Lift}(\mu, (\phi, w))$ for some $(\phi, w) \in V$ do
while $\mu \sqsubset \mathrm{Lift}(\mu, (\phi', w))$ for some $(\phi', w) \in V$ and $\phi'$ not of the form $\phi' =< a > \phi''$ or $\phi' = [a]\phi''$
do $\mu := \mathrm{Lift}(\mu, (\phi', w))$;
$\mu := \mathrm{Lift}(\mu, (\phi, w))$;
If $\mu(\psi, v) \neq \top$, return true;
Else return false;
```

This means that we lift only nodes $(\phi', w) \in V$ with ϕ' not of the form $\phi' = \langle a \rangle \phi''$ or $\phi' = [a]\phi''$ until this lifting does not change anything anymore. Only then, we lift a node of the form $\langle a \rangle \phi''$ or $[a]\phi''$, for which the lifting makes a difference. Then again we lift all liftable nodes $(\phi', w) \in V$ with ϕ' not of the form $\phi' = \langle a \rangle \phi''$ or $\phi' = [a]\phi''$ and so on.
If ϕ is not of the form $\phi' = \langle a \rangle \phi''$ or $\phi' = [a]\phi''$, all edges from (ϕ, w) in E are edges to nodes of the form (ϕ', w). Thus in most of the updates it suffices to look at the values for the formula at a fixed world w. Only at nodes $(\phi, w) \in V$ with ϕ of the form $\phi = \langle a \rangle \phi'$ or $\phi = [a]\phi'$ we have successors of the form (ϕ', z), where $(w, z) \in E_a$. In this case we have to consider values of ϕ' at different worlds, but within those worlds the values of the subformulas are already lifted as far as possible.

Note that this algorithm does not change the complexity of the original algorithm, but it gives us a nice systematic approach to model checking games, which allows us to consider only the formula for most lifting operations. Let us illustrate this with an example. For the sake of easier understanding, we will use the method from [Jur00], not applying succinct tree coding, in the example. The algorithm works exactly the same with the succinct tree coding.

4.2.2 Example: Consider the formula $\psi = \nu X.\mu Y. \diamond ((P \wedge X) \vee Y)$ over the simple Kripke structure $\mathcal{K} = (\{a, b, c\}, E, P\}$ with one agent, $P = \{a\}$ and E defined by the diagram below:

$$\begin{tikzcd}
& c \arrow[dr] & \\
b \arrow[ur] & & a \arrow[ll] \\
& & P
\end{tikzcd}$$

Note that only (X, w) for $w \in \{a, b, c\}$ and (P, a) have value 2 and all other nodes of the model checking game have value 1. We illustrate the algorithm in a table. The subformulas of ψ can be uniquely identified with the point in ψ where they begin (including paranthesis). Instead of writing all nodes (ϕ, w) next to one another, we will label the top of the table with the formula, indicating where a subformula begins, and rotate through the $w \in W$ in the rows of the table. For the later steps we will do a few steps at a time to save rows.
In the first step we update $\mu(Y, a)$, because $(\diamond((P \wedge X) \vee Y), a)$ is the only successor of (Y, a) and $\text{val}(Y, a) = 1$ is odd, so $\text{Prog}(\mu((Y, a), \diamond((P \wedge X) \vee Y), a)) = 01 >_{\text{val}(X,a)} 00$. In the second step, we take the minimum of $\text{Prog}(\mu, ((P \wedge X) \vee Y, a), (P \wedge X, a))$ and $\text{Prog}(\mu, ((P \wedge X) \vee Y, a), (Y, a))$, because $((P \wedge X) \vee Y, a) \in V_0$. In the third step we update $\mu((P \wedge X, a), a)$, which is in V_1 and therefore we take the maximum of $\text{Prog}(\mu, (P \wedge X, a), (P, a))$ and $\text{Prog}(\mu, (P \wedge X, a), (X, a))$, which is 02. In the fourth and fifth step we update $\mu(\psi, a)$ and $\mu(\mu Y. \diamond ((P \wedge X) \vee Y), a)$.
That is all we can update for now, without considering the modal operators, so we change the node in the Kripke structure and update there without considering the modal operators as far as we can. Note, that we can update $\mu(P, b)$ and $\mu(P, c)$ to $\top$, because we have to update them again and again until we reach the top element.
After the seventh line of the table there are no more possible updates without considering the modal operators, so that is what we do: we update $\mu(\diamond((P \wedge X) \vee Y), a)$, $\mu(\diamond((P \wedge X) \vee Y), b)$ and $\mu(\diamond((P \wedge X) \vee Y), c)$ and then update again as far as possible without considering the modal operator and so on until in the end no more update is possible. Since (ψ, a), (ψ, b) and (ψ, c) never reach $\top$, ψ holds on all $w \in W$.

	1	1	1	1	1			2			1	
$w \in W$	$\nu X.$	$\mu Y.$	$\diamond$	(	(	P	$\wedge$	X	)	$\vee$	Y	)
a	0 0	0 0	0 0	0 0	0 0	0 0		0 0			0 1	
a	0 0	0 0	0 0	0 0	0 1	0 0		0 0			0 1	
a	0 0	0 0	0 0	0 2	0 1	0 0		0 0			0 1	
a	0 0	0 1	0 0	0 2	0 1	0 0		0 0			0 1	
a	0 2	0 1	0 0	0 2	0 1	0 0		0 0			0 1	
b	0 2	0 1	0 0	0 3	$\top$	$\top$		0 0			0 1	
c	0 2	0 1	0 0	0 3	$\top$	$\top$		0 0			0 1	
a	0 6	0 5	0 4	0 2	0 1	0 0		0 0			0 5	
b	0 6	0 5	0 4	0 6	$\top$	$\top$		0 0			0 5	
c	0 5	0 4	0 3	0 5	$\top$	$\top$		0 0			0 4	
a	0 9	0 8	0 7	0 2	0 1	0 0		0 0			0 8	
b	0 8	0 7	0 6	0 8	$\top$	$\top$		0 0			0 7	
c	0 5	0 4	0 3	0 5	$\top$	$\top$		0 0			0 4	
a	0 11	0 10	0 9	0 2	0 1	0 0		0 0			0 10	
b	0 8	0 7	0 6	0 8	$\top$	$\top$		0 0			0 7	
c	0 5	0 4	0 3	0 5	$\top$	$\top$		0 0			0 4	

4.2.3 Theorem: The algorithm described above is correct.

Proof. This follows immediately from the correctness of the algorithm SuccinctProgressMeasureLifting. □

Conclusion

In this thesis we have gotten to know two methods for solving parity games in quasipolynomial time. The development of these methods by Calude et al. and Jurdziński and Lazić has been a major breakthrough in the study of parity games, but a lot of research is still to be done. For instance we still do not know whether parity games can indeed be solved in polynomial time or whether quasipolynomial time is as good as it gets.
This thesis has given a few examples of how the two methods can be used in other areas to find more efficient algorithms than where known before parity games where shown to be decidable in quasipolynomial time. In the third chapter, we have seen algorithms for solving Streett-Rabin games in FPT, both for Muller conditions and for pair conditions. One improvement to these algorithms could be to try and construct a lifting algorithm based on an antagonistic update game, like Fearnley et al. did for parity games, as presented in Section 2.3. Additionally, Piterman and Pnueli presented the concept of *Rabin rankings* and *Streett rankings* in 2006 [PP06]. Both rankings result in lifting algorithms for Streett-Rabin games, very similar to the lifting algorithm using progress measures for parity games presented by Jurdziński in [Jur00]. Adaptation of their lifting algorithms for Streett-Rabin games to include succinct tree coding could yield interesting results.
We have also seen that we can apply succinct counting as well as succinct progress measures to the model checking games of the modal μ-calculus. Whereas in theory, this still leaves us with quasipolynomial time complexity, in practice most instances of such model checking games will have a very small alternation depth in comparison to the size $n \leq |W| \cdot |\psi|$ of the game graph, where $|\psi|$ is the length of the formula. And we have seen both for the method by Calude et al. and for the method by Jurdziński and Lazić, that if m, which here is the alternation depth, is at most $\log(n)$, then the game can be decided in polynomial time. Therefore for most relevant instances of model checking games for L_μ, the algorithms considered in Chapter 4 run in polynomial time.

Bibliography

[BSV03] H. Björklund, S. Sandberg, S. Vorobyov, *On fixed-parameter complexity of infinite games*, The Nordic Workshop on References 733 Programming Theory (NWPT03), Åbo Akademi University, Turku, Finland, pp. 6264, 2003, Åbo Akademi, Department of Computer Science, 2003. 530

[BLV96] N. Buhrke, H. Lescow, J. Vöge, *Strategy construction in infinite games with Streett and Rabin chain winning conditions*, Tools and Algorithms for the Construction and Analysis of Systems, TACAS 1996. Lecture Notes in Computer Science, vol 1055. Springer Verlag, 1996.

[Cal17] C. S. Calude, S. Jain, B. Khoussainov, W. Li, F. Stephan. 2017. *Deciding Parity Games in Quasipolynomial Time*, Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC 2017), ACM, pp. 252-263, 2017

[CKS81] A. K. Chandra, D. C. Kozen, L. J. Stockmeyer, *Alternation*, Journal of the ACM 28(1), pp. 114-133, 1981.

[Fea17] J. Fearnley, S. Jain, S. Schewe, F. Stephan, D. Wojtczak, *An ordered approach to solving parity games in quasi polynomial time and quasi linear space*, March 2017. arXiv/1703.01296v1.

[Grä16] E. Grädel, *Logic and Games, WS 2015/2016*, Lecture Notes, Mathematische Grundlagen der Informatik, RWTH Aachen, 2016.

[Hor05] F. Horn, *Streett games on finite graphs*, Proc. 2nd Workshop on Games in Design and Verification, 2005.

[HR97] L. A. Hemaspaandra, J. Rothe, *Unambiguous Computation: Boolean Hierarchies and Sparse Turing-Complete Sets*, SIAM J. Comput. 26(3), pp. 634653, 1997

[Jur98] M. Jurdziński, *Deciding the winner in parity games is in* **UP∩co-UP**, Information Processing Letters 68(3), pp. 119-124, 1998.

[Jur00] M. Jurdziński, *Small Progress Measures for Solving Parity Games*, STACS 2000, LNCS 1770, pp. 290-301, Springer Verlag, 2000.

[JL17] M. Jurdziński, R. Lazić, *Succinct progress measures for solving parity games*, 2017 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), vol. 00, pp. 1-9, 2017.

[KI16] I. Khaliq, G. Imran, *Reachability games revisited*, Second Interna- tional Conference on Advances and Trends in Software Engineering, SOFTENG 2016, 21-25 February 2016, Lisbon, Portugal, Proceedings, International Academy, Research and Industry Association (IARIA), pp. 129-133, 2016.

[Küs02] R. Küster, *Memoryless Determinacy of Parity Games*, Automata, Logics and Infinite Games, LNCS 2500, pp. 95-106, Springer Verlag, 2002.

[Mar75] D. A. Martin, *Borel determinancy*, Anals of Mathematics 102, pp. 363-371, 1975.

[PP06] N. Piterman, A. Pnueli, *Faster solution of Rabin and Streett games*, Proc. 21st Symposium on Logic in Computer Science, pp. 275284, IEEE, IEEE press, 2006.

[Sch07] S. Schewe, *Solving parity games in big steps*, FSTTCS 2007: Foundations of Software Technology and Theoretical Computer Science, Lecture Notes in Computer Science, vol 4855. Springer Verlag.

[Tar55] A. Tarski, *A lattice-theoretical fixpoint theorem and its application*, Pacific Journal of Mathematics 5, pp. 285309, 1955.

[Zap02] J. Zappe, *Modal μ.Calculus and Alternating Tree Automata*, Automata, Logics and Infinite Games, LNCS 2500, pp. 95-106, Springer Verlag, 2002.

[Zie98] W. Zielonka, *Infinite games on finitely coloured graphs with applications to automata on infinite trees*, Theoretical Computer Science 200, pp. 135-183, 1998.