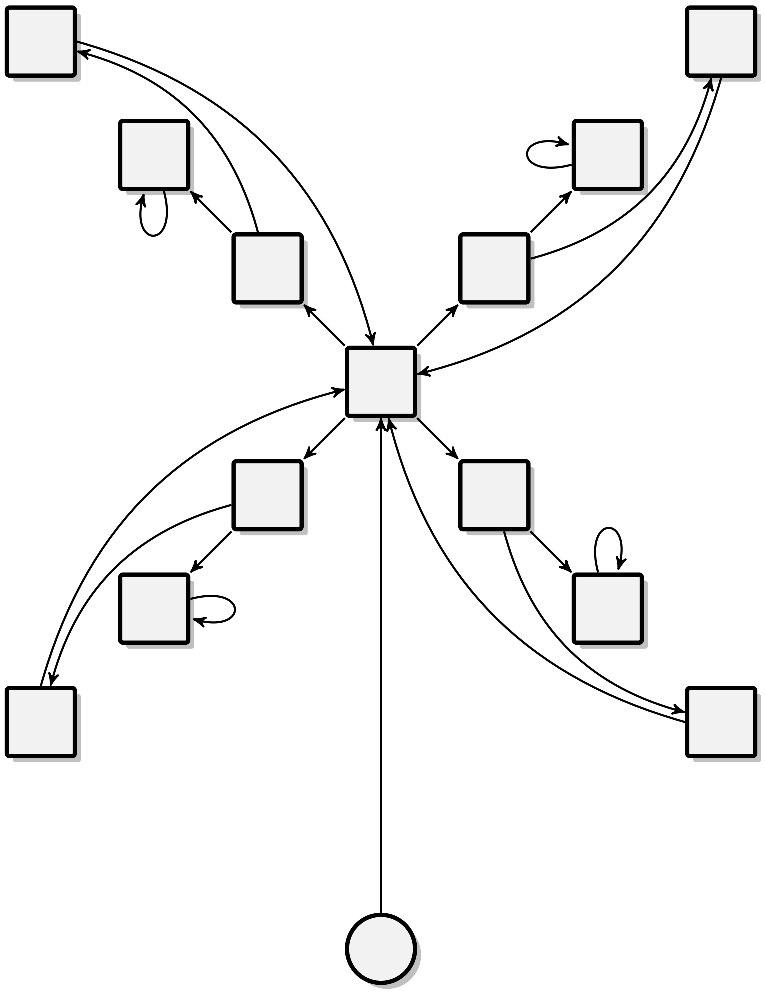
Infinite Games

Lecture Notes



by **Martin Zimmermann**,
Felix Klein, and **Alexander Weinert**

In case of questions, errors, or suggestions for improvements
please send an email to: *mzi@cs.aau.dk*

Aalborg, October 14, 2022

Contents

1 Introduction

Many of today's problems in computer science are no longer concerned with programs that transform data and then terminate, but with non-terminating systems that have to interact with a possibly antagonistic environment. An example is the controller regulating the anti-lock brake in a car. It receives a constant input stream of sensor readings like the wheel speed of each wheel and selectively applies the brakes on each wheel to maintain a uniform wheel speed. In such a setting, we are not interested in a controller terminates that and generates some output, but we need the controller to run forever (which over-approximates an arbitrary long car ride) and control the wheel speed in a manner that avoids locking brakes, amongst other requirements.

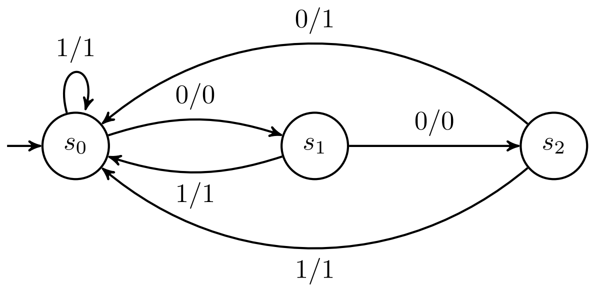
The emergence of so-called reactive systems requires new approaches to verification and synthesis. Over the course of the last fifty years it turned out to be very fruitful to model and analyze reactive systems in a game-theoretic framework, which captures the antagonistic and strategic nature of the interaction between the system and its environment.

This approach can be traced back to work on the synthesis problem for boolean circuits, nowadays known as Church's problem: given a requirement on the input-output behavior of circuits expressed in some suitable formalism, find a circuit that satisfies the given requirement (or determine that there is no such circuit). This problem can be interpreted as a game between two agents: an environment generating an infinite stream of input bits, each of which is answered by an output bit generated by the circuit. The requirement on the input-output behavior determines the winner of each execution: if the pair of bitstreams satisfies the requirement, then the circuit wins, otherwise the environment wins. In this view, Church's problem boils down to finding a finitely represented rule which prescribes for every finite sequence of input bits an output bit such that every input stream is answered by an output stream in a way that the pair of streams satisfies the given requirement.

As an example, consider the conjunction of the following three requirements:

1. Whenever the input bit is 1, then the output bit is 1, too.
2. At least one out of every three consecutive output bits is a 1.
3. If there are infinitely many 0's in the input stream, then there are infinitely many 0's in the output stream.

Note that the first two requirements are satisfied by always outputting a 1, a strategy that is spoiled by the third requirement. However, the following strategy satisfies all three requirements: if the input bit is a 0, answer with a 0, unless the last two output bits were already 0, in this case output a 1. On the other hand, every 1 in the input stream is answered by a 1. This strategy is implemented by the following automaton with output where the label 1/1 stands for "process a 1 and output a 1".



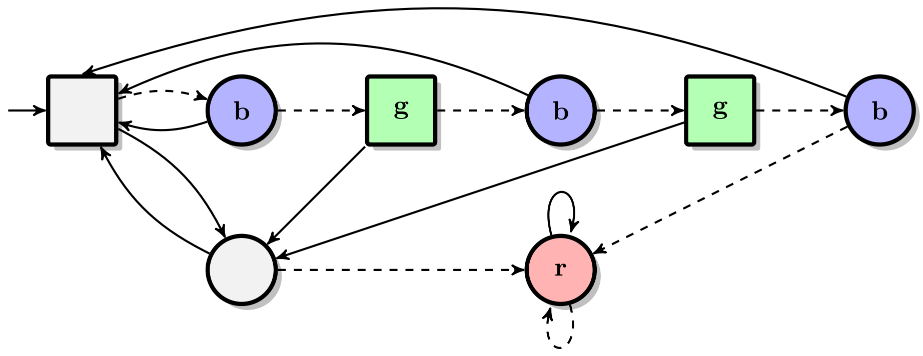
An example run of the automaton is given in the following.

state	s_0	s_1	s_2	s_0	s_1	s_0	s_0	s_1	s_2	s_0	s_1	s_2	$\cdots$
input	0	0	0	0	1	1	0	0	0	0	0		$\cdots$
output	0	0	1	0	1	1	0	0	1	0	0		$\cdots$

In this course we model such a synthesis problem as a two-player game and learn how to compute a finite-state representation of a strategy satisfying the requirements, if one exists. To model the general synthesis problem for reactive systems, another level of abstraction is added to the game described above:

an infinite, graph-based, two-player game is played in a graph without dead ends whose set of vertices is partitioned into the positions of Player 0 and the positions of Player 1. The players construct a play, an infinite path through the graph, according to the following rule: a token is placed at an initial vertex and whenever the token is at a position of Player i, she has to move the token to some successor. After ω moves, the winning condition of the game, a subset of the plays of the graph, determines the winner of the play. A strategy for Player i in such a game is a mapping prescribing a legal move for every play prefix ending in a position of Player i. A strategy is winning from a given vertex, if every play that starts in this vertex and is played according to the strategy is won by Player i. A game is determined, if from each vertex, one of the players has a winning strategy.

The following game models the example from above: in the graphical representation, Player 0's vertices are drawn as cycles and Player 1's as rectangle. Furthermore, dashed edges model picking a 0 while solid edges model picking a 1. Finally, the winning condition is the set of paths that never visit the red vertex (marked by r) and if they visit infinitely many blue vertices (marked by b), then also infinitely many green ones (marked by g). Note that the third conjunct is captured by the winning condition while the first two conjuncts of the requirement are captured by the structure of the graph. In particular, the upper chain of vertices counts the number of zeros picked by Player 0 while ever answering a 1 with a 0 leads to the red sink vertex.



Player 0 wins this game from the initial vertex by never moving the token to the red sink vertex and by always moving from a blue vertex to a green one whenever possible. If infinitely many blue vertices are visited, then this strategy visits infinitely many green ones too, as the only blue vertex without a green successor is only reachable via a green vertex. Furthermore, Player 1 is never able to force the token to the red sink vertex, as only edges from Player 0 vertices lead to the sink. From all these vertices, she has the choice to move the token to the initial vertex.

Note that the input-output behavior encoded by this strategy is the one given by the automaton above. Even more so, one can derive the automaton from the game graph and the strategy described above by merging vertices. Thus, by determining a winning strategy for this game, we obtain a solution to our instance of Church's problem.

In the framework of infinite games, the seminal Büchi-Landweber Theorem, which solves Church's problem as special case, reads as follows: every infinite game in a finite graph with ω-regular winning condition is determined and finite-state strategies – strategies implemented by finite automata with output – suffice to win these games and can be computed effectively. Ever since, this result was extended along different dimensions, e.g., the number of players, the type of graph the game is played in, the type of winning condition, the nature of the interaction between the players (alternation or concurrency), the presence or absence of probabilistic influences, and complete or incomplete information for the players about the evolution of the play.

The synthesis problem for reactive systems can be solved as follows: we model the system and its environment by a finite graph whose edge relation describes the interaction between the environment and the system; the requirement on the system is expressed as ω-regular winning condition. Applying the Büchi-Landweber Theorem yields an automaton with output so that every execution that is controlled by the automaton satisfies the requirement (or it yields a strategy for the environment witnessing that the requirement cannot be satisfied).

1.1 Course Overview

This course is divided into five chapters, each dealing with one aspect of infinite games.

In the first chapter, we give all necessary definitions to reason about infinite games, i.e., arenas, the graphs infinite games are played in and strategies.

In the second chapter, we introduce basic winning conditions derived on acceptance conditions for automata on infinite objects: reachability and safety, Büchi and co-Büchi, and parity. The results proved for these games are the building blocks for the advanced results we prove in the latter chapters.

In the third chapter, we consider games with finite-state strategies, strategies implemented by finite-automata with output. The Büchi-Landweber Theorem states that such strategies suffice for all games with ω-regular winning conditions. Furthermore, we discuss the tight relation between finite-state strategies and game reductions, the classical way to compute finite-state strategies.

In the fourth chapter, we consider infinite games played on infinite graphs. Since we still want to solve them algorithmically, we need infinite graphs that can be represented finitely: we show how to determine the winner and winning strategies for games played on the configuration graphs of pushdown automata. These automata can model recursive programs with finite data domains, using the pushdown stack to model the stack of function calls.

In the fifth chapter, we consider an application of infinite games to logics: Rabin's theorem states that monadic second-order logic over the binary tree is decidable, i.e., there is an algorithm that given a sentence φ decides whether it is satisfiable or not. We introduce monadic second-order logic and parity tree automata, which we show to be equally expressive. Infinite games are used to prove that such automata are closed under negation (which is one step we need to take when proving the equivalence of automata and logic) and to solve the emptiness problem for tree automata. Thus, given a formula φ, one translates it into an equivalent automaton, which is non-empty if, and only if, φ is satisfiable.

2 The Foundations

In this introductory chapter, we introduce infinite games and all necessary concepts to work with them. Then, we consider games with basic winning conditions derived from acceptance conditions for automata on infinite words.

In these notes, we only consider two-player games between Player 0 and Player 1. To avoid writing "Player 0" and "Player 1" again and again, we assume Player 0 to be female and Player 1 to be male and refer to them by their personal pronouns. Oftentimes, we present definitions or arguments that hold for both players. Here, we use the indeterminate player "Player i" and the opponent "Player $1-i$", for $i \in \{0,1\}$. Here, we agree that Player i is female and Player 1 is male.

2.1 Arenas, Games, and Strategies

First we define the main component of a game: its arena. It describes the rules the two players have to follow and the order in which the players make their moves. Figure 2.1 depicts an arena. In drawings, the round vertices are owned by Player 0, the rectangular ones by Player 1. The edges in between describe the moves the players can perform.

In general, an arena is a directed graph consisting of vertices and edges. Each vertex is assigned to one of the players. A play in such an arena proceeds as follows: A token is placed at some arbitrary initial vertex and then the players move it through the arena. If the vertex is owned by Player 0, she has to move the token, otherwise Player 1 has to move it. The next vertex the token can be moved to, is then given by the edges, where, if a vertex has multiple outgoing edges, the corresponding player owning that vertex must choose one of them. Finally, since we want to play infinitely long, we only consider arenas that have at least one outgoing edge for each vertex. This way, we ensure that the token does not end up in a deadlock.

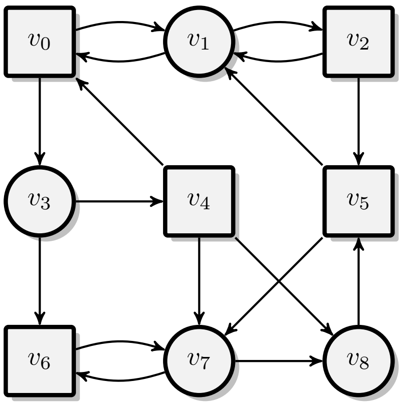


Figure 2.1: An arena.

Definition 2.1 (Arena)**.** *An* arena $\mathcal{A} = (V, V_0, V_1, E)$ *consists of*

- *a finite set V of vertices,*
- *disjoint subsets $V_0, V_1 \subseteq V$ with $V = V_0 \cup V_1$ denoting the vertices of Player 0 and Player 1 respectively, and*
- *a set $E \subseteq V \times V$ of (directed) edges such that every vertex has at least one outgoing edge, i.e., $\{v' \mid (v, v') \in E\}$ is non-empty for every $v \in V$.*

The size of $\mathcal{A}$, denoted by $|\mathcal{A}|$, is defined to be $|V|$.

Sometimes, we need to restrict our attention to some part of an arena, e.g., in Figure 2.1 on the part consisting of the vertices v_4, v_5, v_7, and v_8, and the edges between these vertices (see Figure 2.2). Such a structure is a sub-arena of the original arena. As in the example, a sub-arena is always induced by a subset of the vertices and contains all edges between them.

Note that we have to ensure that the resulting sub-arena is valid, i.e., that it has an outgoing edge for each vertex. The sub-arena in Figure 2.2 satisfies this property, while the set of vertices $\{v_7, v_8\}$ does not induce a valid sub-arena of the one depicted in Figure 2.1, since v_8 has no successor in this set. Thus, v_8 would be a terminal vertex, which is not allowed by our definition of an arena.

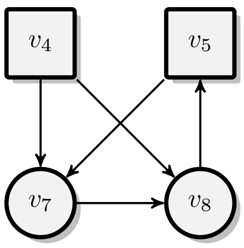


Figure 2.2: A sub-arena of the arena from Figure 2.1.

Definition 2.2 (Sub-Arena)**.** *Let $\mathcal{A} = (V, V_0, V_1, E)$ be an arena and $V' \subseteq V$ be such that every vertex in V' has a successor vertex in V'. The* sub-arena of $\mathcal{A}$ induced by V', *denoted by $\mathcal{A}\restriction V'$, is defined as*

$$\mathcal{A}\restriction V' = (V \cap V', V_0 \cap V', V_1 \cap V', E \cap (V' \times V')).$$

We already explained that the players play in such an arena by moving a token through it. After infinitely many moves, the players have produced an infinite path. We call such a path a play in the arena. An example for a play in the example arena is given in Figure 2.3.

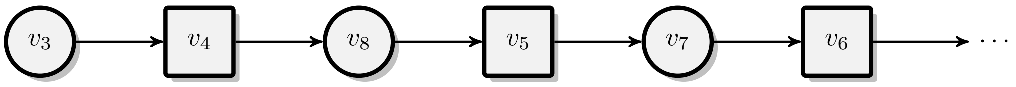


Figure 2.3: A play prefix in the arena of Figure 2.1 starting in v_3.

Definition 2.3 (Play)**.** *A* play *in an arena $\mathcal{A} = (V, V_0, V_1, E)$ is an infinite sequence $\rho = \rho_0\rho_1\rho_2 \cdots \in V^\omega$ such that $(\rho_n, \rho_{n+1}) \in E$ holds for every $n \in \mathbb{N}$. We say ρ starts in the vertex ρ_0. The set of plays in $\mathcal{A}$ is denoted by* $\mathrm{Plays}(\mathcal{A})$*, the set of all plays starting in v by* $\mathrm{Plays}(\mathcal{A}, v)$*, and we define* $\mathrm{Plays}(\mathcal{A}, V') = \bigcup_{v \in V'} \mathrm{Plays}(\mathcal{A}, v)$ *for every $V' \subseteq V$.*

The most important aspect of a game are its strategies. In an infinite game, the strategy prescribes how to move the token through the arena. On the one hand, this decision depends on the structure of the arena, which describes the possible moves. On the other hand, it may also depend on the decisions made in the past, i.e., on the vertices already visited. Thus, the choice of the next move may depend on the play prefix produced thus far. Formally, a strategy is a function mapping play prefixes to a successor of their last vertex.

Definition 2.4 (Strategy)**.** *A* strategy *for Player $i \in \{0, 1\}$ in an arena (V, V_0, V_1, E) is a function $\sigma\colon V^*V_i \to V$ such that $\sigma(wv) = v'$ implies $(v, v') \in E$ for every $w \in V^*$ and every $v \in V_i$.*

By convention, we denote strategies for Player 0 by σ and strategies for Player 1 by τ. Oftentimes, we reason about an arbitrary player and its opponent, i.e., about Player $i \in \{0, 1\}$ and its opponent Player $1 - i$. In this situation, we denote strategies for Player i by σ and strategies for Player $1 - i$ by τ.

If a play results from applying the strategy, then it is consistent with the strategy.

Definition 2.5 (Consistent Play)**.** *A play $\rho_0\rho_1\rho_2 \cdots$ in an arena $\mathcal{A} = (V, V_0, V_1, E)$ is* consistent *with a strategy σ for Player i in $\mathcal{A}$ if, $\rho_{n+1} = \sigma(\rho_0 \cdots \rho_n)$ for every $n \in \mathbb{N}$ with $\rho_n \in V_i$. Given a vertex v, we denote the set of plays that are consistent with σ and start in v with* $\mathrm{Plays}(\mathcal{A}, v, \sigma)$*. Finally, we define* $\mathrm{Plays}(\mathcal{A}, V', \sigma)$ *for $V' \subseteq V$ by* $\mathrm{Plays}(\mathcal{A}, V', \sigma) \bigcup_{v \in V'} \mathrm{Plays}(\mathcal{A}, v, \sigma)$

Consistency of a finite play prefix with a strategy is defined similarly and the following property holds: a play ρ is consistent with a strategy σ if, and only if, every prefix of σ is consistent with σ.

Consider a strategy σ for Player 0 with $\sigma(v_3) = v_4$, $\sigma(v_3v_4v_8) = v_5$, and $\sigma(v_3v_4v_8v_5v_7) = v_6$. Then, the play prefix depicted in Figure 2.3 is consistent with σ.

In general, multiple plays are consistent with a strategy for one player, as it only determines the choices of this player. However, if we fix an initial vertex v and strategies σ and τ for Player 0 and Player 1, then there is a unique play that is consistent with σ and τ and starts in v. We denote this play by $\rho(v, \sigma, \tau)$, which is formally the unique element in $\mathrm{Plays}(\mathcal{A}, v, \sigma) \cap \mathrm{Plays}(\mathcal{A}, v, \tau)$. Alternatively, we can define $\rho(v, \sigma, \tau) = \rho_0\rho_1\rho_2 \cdots$ inductively as $\rho_0 = v$ and

$$\rho_{n+1} = \begin{cases} \sigma(\rho_0 \cdots \rho_n) & \text{if } \rho_n \in V_0, \\ \tau(\rho_0 \cdots \rho_n) & \text{if } \rho_n \in V_1. \end{cases}$$

Our definition of strategy is very general and not suitable for practical use, since it is an infinite object. Accordingly, we are interested in some weaker representations. One of them is the class of positional strategies. Here, the choice of the strategy is not allowed to depend on the whole play prefix, but only on the vertex the token is currently at. Thus, every time the token is at some vertex, the strategy prescribes the same successor. Later, we will introduce a more complex representation that allows to store some finite information about the history.

Definition 2.6 (Positional Strategy)**.** *A strategy σ for Player i in an arena (V, V_0, V_1, E) is* positional *if $\sigma(wv) = \sigma(v)$ for all $w \in V^*$ and $v \in V_i$.*

As a positional strategy for a Player i is equivalent to a function $\sigma\colon V_i \to V$ we usually denote them as such functions. Nevertheless, technically we still mean a strategy in the original sense, since otherwise notions like consistency are not properly defined anymore.

After introducing strategies we now explain how to determine the winner of a play. Similarly to the definition of strategies, we start with a very general definition and restrict it later on: a winning condition is a subset of the arenas plays. If a play is in the winning condition, then Player 0 wins, otherwise Player 1 wins. Combining an arena with a winning condition yields a game.

Definition 2.7 (Game)**.** *A* game $\mathcal{G} = (\mathcal{A}, \mathrm{Win})$ *consists of an arena $\mathcal{A}$ with vertex set V and a set of* winning sequences $\mathrm{Win} \subseteq V^\omega$*. We call a sequence ρ winning for Player 0 if, and only if, $\rho \in \mathrm{Win}$ and winning for Player 1 otherwise.*

Note that Win is defined as a subset of V^ω, not of $\mathrm{Plays}(\mathcal{A})$. Thus, sequences that are not plays may be winning as well.

As an example consider the game $\mathcal{G}_e = (\mathcal{A}, \mathrm{Win})$ where $\mathcal{A} = (V, V_0, V_1, E)$ is the arena depicted in Figure 2.1 and the winning condition is defined as $\mathrm{Win} = \{\rho \in V^\omega \mid \mathrm{Occ}(\rho) \neq V\}$. [1] A play is winning for Player 0 in this game if, and only if, at least one of the vertices of the arena is not visited. Hence, the play $(v_0v_1)^\omega$ obtained by alternating between v_0 and v_1 is winning for Player 0 while a play with the prefix $v_4v_7v_8v_5v_1v_2v_1v_0v_3v_6$ is winning for Player 1

When playing a game, both players are trying to guarantee a win against any possible moves of their opponent. This is formalized by the notion of a winning strategy.

Definition 2.8 (Winning Strategy)**.** *Let $\mathcal{G} = (\mathcal{A}, \mathrm{Win})$ be a game with $\mathcal{A} = (V, V_0, V_1, E)$. A strategy σ for Player i in $\mathcal{A}$ is a* winning strategy *from a vertex $v \in V$ if every play that is consistent with σ and starts in v is winning for Player i, i.e., if $\mathrm{Plays}(\mathcal{A}, v, \sigma) \subseteq \mathrm{Win}$ for $i = 0$ and $\mathrm{Plays}(\mathcal{A}, v, \sigma) \subseteq V^\omega \setminus \mathrm{Win}$ for $i = 1$.*

We collect the vertices from which Player i can win a game $\mathcal{G}$ in her winning region.

Definition 2.9 (Winning Region)**.** *The* winning region *$W_i(\mathcal{G})$ of Player i in a game $\mathcal{G}$ is the set of vertices from which Player i has a winning strategy.*

If $\mathcal{G}$ is clear from the context we just write W_i instead of $W_i(\mathcal{G})$. Regarding our example game $\mathcal{G}_e$ given above, Player 0 has a winning strategy from all vertices. From vertex v_1 she always moves to v_0, from v_3 always to v_4, and from v_7 always to v_8. This way she avoids visiting the vertices v_2 and v_6 when starting from any other vertex. When starting at one of these vertices, at least the other one is never reached, when playing consistently with this strategy. Consequently, $\mathcal{G}_e$ has the following winning regions: $W_0 = V$ and $W_1 = \emptyset$. Furthermore, note that the winning strategy for Player 0 described above is positional and winning from every vertex.

Lemma 2.1. *We have $W_0(\mathcal{G}) \cap W_1(\mathcal{G}) = \emptyset$ for every game $\mathcal{G}$.*

Proof. Let $\mathcal{G} = (\mathcal{A}, \mathrm{Win})$. Towards a contradiction, assume there exists a vertex $v \in W_0(\mathcal{G}) \cap W_1(\mathcal{G})$. Then, both players have a winning strategy from v; call them σ and τ. Let $\rho = \rho(v, \sigma, \tau)$, i.e., we let the players both use their winning strategy against each other.

Then, $\rho \in \mathrm{Win}$, as σ is a winning strategy for Player 0, and $\rho \notin \mathrm{Win}$, as τ is a winning strategy for Player 1. Hence, we have derived the desired contradiction. □

Thus, the winning regions of the two players are always disjoint. Another interesting question is whether it is possible that a vertex is in neither of the two winning regions, i.e., are there vertices from which no player has a winning strategy? We return to this question later on. If there is no such vertex, then every vertex is in one of the winning regions and we say the game is determined.

Definition 2.10 (Determinacy, Positional Determinacy)**.** *Let $\mathcal{G}$ be a game with vertex set V. We say that $\mathcal{G}$ is* determined *if $W_0(\mathcal{G}) \cup W_1(\mathcal{G}) = V$. Furthermore, we say that $\mathcal{G}$ is* positionally determined *if, from every vertex $v \in V$ one of the players has a positional winning strategy.*

Note that in the definition of positional determinacy, we allow the positional winning strategies to depend on the initial vertex v, i.e., Player i may use different positional strategies for different initial vertices. In contrast, a uniform positional winning strategy has to be winning from every vertex in the winning region.

Definition 2.11 (Uniform Positional Winning Strategy)**.** *Let the game $\mathcal{G} = (\mathcal{A}, \mathrm{Win})$. A strategy σ for Player i is a* uniform positional winning strategy *if it is positional and winning from every vertex in $W_i(\mathcal{G})$.*

[1] $\mathrm{Occ}(\rho)$ denotes the set of vertices occurring in ρ. See the appendix for a formal definition.

The strategy for Player 0 in $\mathcal{G}_e$ described above is a uniform positional winning strategy.

To conclude, we introduce some general results about infinite games that are applied in proofs throughout these lecture notes.

First, we introduce the notion of a trap for Player i, which is a region of an arena which Player i cannot leave without the help of Player $1-i$. For example, consider the set $T = \{v_6, v_7\}$ in the arena from Figure 2.1: from v_6, Player 1 has to move to v_7, and from v_7, Player 0 can choose to move back to v_6. Thus, T is a trap for Player 1, i.e., Player 0 has a strategy to keep the token in T, if the play starts in T.

Definition 2.12 (Trap)**.** *Let $\mathcal{A} = (V, V_0, V_1, E)$ be an arena and let $T \subseteq V$. Then, T is a trap for Player i, if*

- *every vertex $v \in T \cap V_i$ of Player i in T has only successors in T, i.e., $(v, v') \in E$ implies $v' \in T$, and*
- *every vertex $v \in T \cap V_{1-i}$ of Player $1-i$ in T there is a successor in T, i.e., there is some $v' \in T$ with $(v, v') \in E$.*

The following remark is a straightforward consequence of the definition of a trap.

Remark 2.1. *Let T be a trap for Player i in an arena with vertex set V. Then, Player $1-i$ has a positional strategy τ that traps the token in T, provided the play starts in T. Formally, τ satisfies $\mathrm{Plays}(\mathcal{A}, T, \tau) \subseteq T^\omega$.*

We call a strategy τ as above a trapping strategy. The following lemma lists several useful properties of traps.

Lemma 2.2. *Let T be a trap for Player i in an arena $\mathcal{A}$.*

1. *The restriction $\mathcal{A}{\upharpoonright}T$ of $\mathcal{A}$ to T is a valid sub-arena.*
2. *If T' is a trap for Player i in $\mathcal{A}{\upharpoonright}T$, then it is also a trap for Player i in $\mathcal{A}$.*

Proof. Exercise 2.3. □

Next, we introduce an important class of winning condition, so-called prefix-independent ones, and show that winning regions in games with such winning conditions are traps. Intuitively, a winning condition is prefix independent, if adding a finite prefix to a play or removing a finite prefix from a play does not change the winner of the play.

Definition 2.13 (Prefix-independence)**.** *A winning condition* $\mathrm{Win} \subseteq V^\omega$ *is* prefix-independent *, if $\rho \in \mathrm{Win} \Leftrightarrow w \cdot \rho \in \mathrm{Win}$ for all $\rho \in V^\omega$ and all $w \in V^*$.*

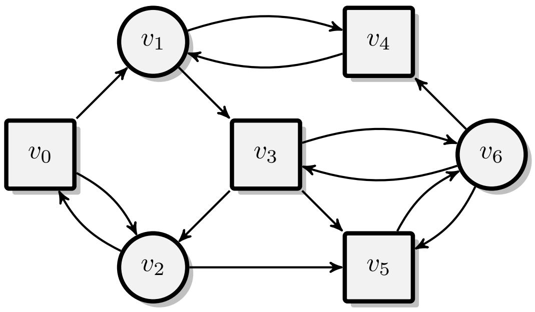
Note that a winning condition is prefix-independent if, and only if, its complement is prefix-independent.

Lemma 2.3. *Let $\mathcal{G} = (\mathcal{A}, \mathrm{Win})$ be a game with prefix-independent winning condition* Win*. Then, $W_i(\mathcal{G})$ is a trap for Player $1-i$.*

Proof. Exercise 2.4. □

2.2 Exercises

Exercise 2.1. Consider the game $\mathcal{G} = (\mathcal{A}, \mathrm{Win})$ with the arena $\mathcal{A}$ depicted below and the winning condition $\mathrm{Win} = \{\rho \in V^\omega \mid \mathrm{Occ}(\rho) = V\}$, i.e., a play ρ is winning for Player 0 in $\mathcal{G}$ if all vertices are visited during ρ.



1. Give at least one winning strategy from some vertex for each player. Argue why they are winning.
2. Determine the winning regions of the game.
3. Is the game positionally determined? Argue formally.

Exercise 2.2. Prove or disprove: If Player i has a positional winning strategy from each vertex $v \in W_i(\mathcal{G})$ for some game $\mathcal{G}$, then Player i has a uniform positional winning strategy for $\mathcal{G}$.

Exercise 2.3. Prove Lemma 2.2.

Exercise 2.4. Prove Lemma 2.3.

Note: The result holds for undetermined games. Make sure that you do not assume determinacy.

Exercise 2.5. Let $\mathcal{G} = (\mathcal{A}, \mathrm{Win})$ be a game with prefix-independent Win.

1. Let σ be a positional winning strategy for Player 0 in $\mathcal{G}$ from some vertex v. Now, let v' be reachable from v via some play $\rho \in \mathrm{Plays}(\mathcal{A}, v, \sigma)$. Show that σ is winning for Player 0 from v' in $\mathcal{G}$.
2. Let V' be a subset of $\mathcal{A}$'s vertices such that Player 0 has a positional winning strategy σ_v in $\mathcal{G}$ from every vertex $v \in V'$. Show that Player 0 has a positional winning strategy σ in $\mathcal{G}$ that is winning from every $v \in V'$.

   Note: The order of quantification changes from $\forall v\, \exists \sigma_v$ to $\exists \sigma\, \forall v$. Compare this to the statement of Exercise 2.2.

3 Reachability, Büchi, and Parity Games

We begin the study of infinite games by considering games with winning conditions derived from acceptance conditions for automata on infinite words. In both situations, one has to represent a language of infinite words in a finite manner. In automata, this is the set of accepting runs while for games it is the set of winning plays Win for Player 0. Our general definition imposes no restrictions on Win, i.e., Win is an arbitrary subset of V^ω. However, this is not practical: for modelling reasons and to be able to represent games as input to algorithms, we need a finite representation.

In this section, we study various formalisms to represent the set of winning plays, namely reachability and safety games, Büchi and co-Büchi games, and parity games.

3.1 Reachability Games

The set of winning plays for Player 0 in a reachability game is induced by a set R of vertices and consists of all plays that visit R at least once. Thus, Player 0's goal is to *reach* R and Player 1 tries to avoid reaching R.

Definition 3.1 (Reachability Game)**.** *Let $\mathcal{A} = (V, V_0, V_1, E)$ be an arena and let $R \subseteq V$ be a subset of $\mathcal{A}$'s vertices. Then, the reachability condition $\textsc{Reach}(R)$ is defined as*

$$\textsc{Reach}(R) := \{\rho \in V^\omega \mid \mathrm{Occ}(\rho) \cap R \neq \emptyset\}.$$

We call a game $\mathcal{G} = (\mathcal{A}, \textsc{Reach}(R))$ a reachability game *with reachability set R.*

An example of a reachability game is given in Figure 3.1, where we depict the reachability set by doubly-framed vertices. In this game, call it $\mathcal{G}_r$, Player 0 has to reach either v_4 or v_5.

Obviously, Player 0 has a winning strategy for $\mathcal{G}_r$ from v_4 or v_5, as the initial vertex of each play starting there is already in R. Now, consider the vertex v_3. From here, Player 0 can move the token to v_4, from where we have already argued that she can enforce a winning play. In general, from every vertex v of Player 0 that has a successor v' from where we know that she can enforce a win, she can enforce a win as well by moving from v to v'. Besides v_3, this condition applies to v_8 and therefore by induction also for v_7, as Player 0 can move from v_7 to v_8 and then to v_5. Note that the case of v_7 requires two applications of the argument.

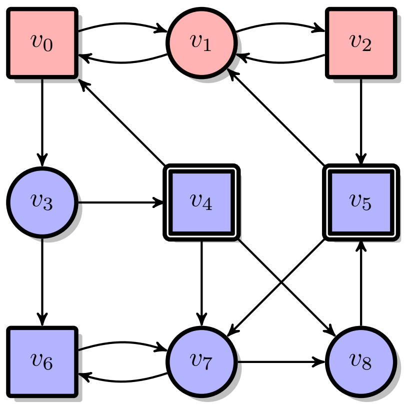


Figure 3.1: Example for a reachability game. We use doubly-framed vertices to denote the reachability set. Player 0's winning region is blue, Player 1's red.

Now, consider the vertex v_6, which is a Player 1 vertex. From v_6, he can only move to v_7, from where we know that Player 0 can enforce a win. Thus, she can enforce a win from v_6 as well. In general, if a vertex v of Player 1 only has successors from which Player 0 can enforce a win, then she can enforce a win from v. In our example, this applies only to v_6. Note that we could only declare the status of v_6 after having determined the status of v_7, which in turn depended on the status of v_8.

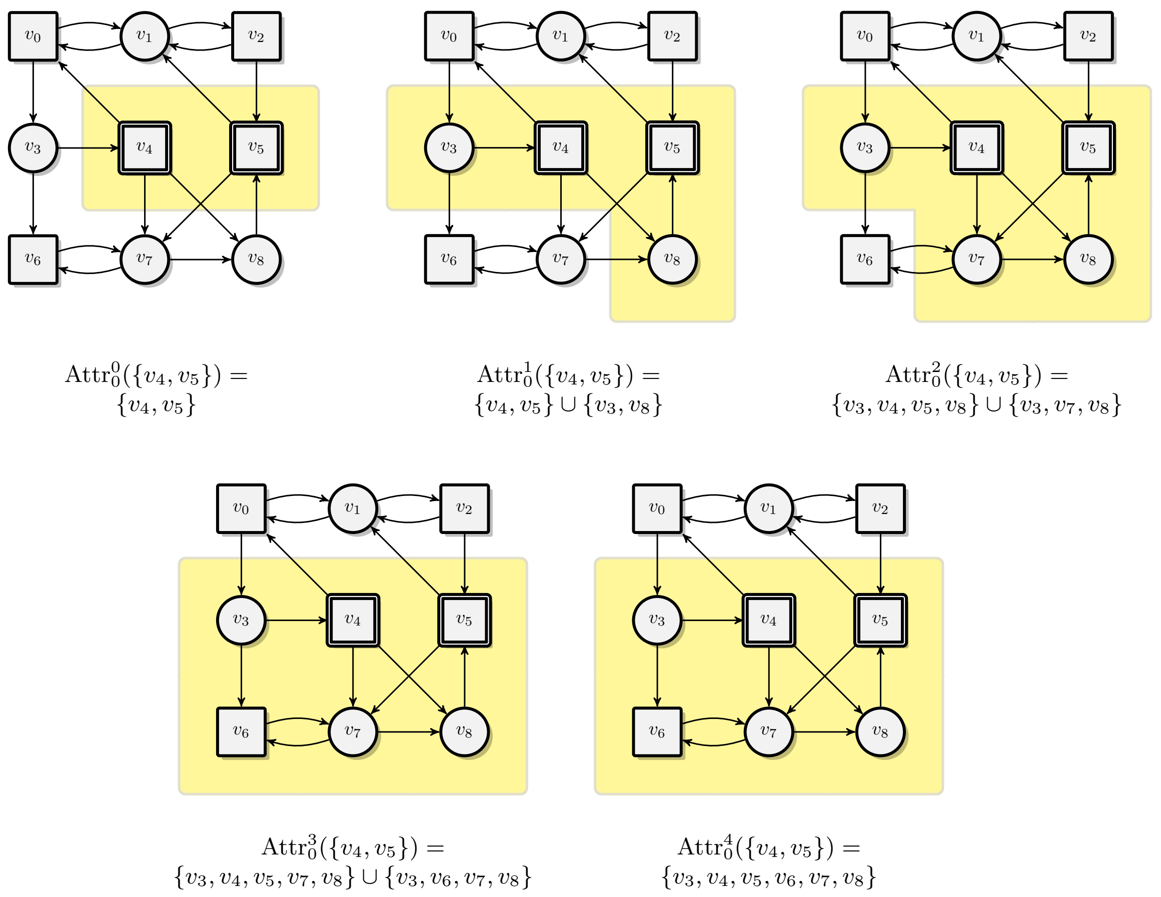
Thus, Player 0 can enforce a win from every vertex in $V' = \{v_3, v_4, v_5, v_6, v_7, v_8\}$. For the remaining vertices, i.e., for those in the set $\{v_0, v_1, v_2\}$, our two rules do not apply: v_1 has no successor in $V \setminus V'$ and both v_0 and v_2 have at least one successor in $V \setminus V'$. Intuitively, Player 1 can use these edges to keep the token in $V \setminus V'$ and Player 0 cannot force the token out of $V \setminus V'$. As R is a subset of V, this means that the token never visits the reachability set, i.e., Player 1 can enforce a win by keeping the token in $V \setminus V'$.

Formalizing this intuition, we define the positional strategies σ for Player 0 and τ for Player 1 as

- $\sigma(v_1) = v_0$,
- $\sigma(v_3) = v_4$,
- $\sigma(v_7) = v_8$,
- $\sigma(v_8) = v_5$,

- $\tau(v_0) = v_1$,
- $\tau(v_2) = v_1$,
- $\tau(v_4) = v_8$,
- $\tau(v_5) = v_7$,
- $\tau(v_6) = v_7$.

Both are uniform positional winning strategies for the winning regions $W_0(\mathcal{G}_r) = V'$ and $W_1(\mathcal{G}_r) = V \setminus V'$ and that the values $\sigma(v_1)$, $\tau(v_4)$, $\tau(v_5)$, and , $\tau(v_6)$ can be defined arbitrarily.



$\mathrm{Attr}_0^0(\{v_4, v_5\}) = \{v_4, v_5\}$

$\mathrm{Attr}_0^1(\{v_4, v_5\}) = \{v_4, v_5\} \cup \{v_3, v_8\}$

$\mathrm{Attr}_0^2(\{v_4, v_5\}) = \{v_3, v_4, v_5, v_8\} \cup \{v_3, v_7, v_8\}$

$\mathrm{Attr}_0^3(\{v_4, v_5\}) = \{v_3, v_4, v_5, v_7, v_8\} \cup \{v_3, v_6, v_7, v_8\}$

$\mathrm{Attr}_0^4(\{v_4, v_5\}) = \{v_3, v_4, v_5, v_6, v_7, v_8\}$

Figure 3.2: Visual representation of the attractor construction for the game of Figure 3.1. The corresponding attractor sets are denoted by the areas marked yellow.

In the following, we show that the intuitive reasoning above is sufficient in general. As V' contains all vertices from where Player 0 can attract the token to R, it is called the 0-*attractor* of R. As already mentioned above, the construction of the attractor is hierarchical, e.g., we have to add v_8 to the attractor before we can add v_7. To keep the formal definition of the attractor as general as possible, we define it for both players.

Construction 3.1 (Attractor)**.** *Let $\mathcal{A} = (V, V_0, V_1, E)$ be an arena, let $R \subseteq V$, and let $i \in \{0, 1\}$ determine a player.*

The controlled predecessor $\mathrm{CPre}_i(R)$ *of R is defined as*

$$\begin{aligned}\mathrm{CPre}_i(R) = \{&v \in V_i \mid v' \in R \textit{ for some successor } v' \textit{ of } v\} \cup \\ \{&v \in V_{1-i} \mid v' \in R \textit{ for all successors } v' \textit{ of } v\}.\end{aligned}$$

The i-attractor $\mathrm{Attr}_i(R)$ *of R in $\mathcal{A}$ is defined by inductively applying the controlled predecessor via*

- $\mathrm{Attr}_i^0(R) = R$*,*
- $\mathrm{Attr}_i^{n+1}(R) = \mathrm{Attr}_i^n(R) \cup \mathrm{CPre}_i(\mathrm{Attr}_i^n(R))$*, and*
- $\mathrm{Attr}_i(R) = \bigcup_{n \in \mathbb{N}} \mathrm{Attr}_i^n(R)$*.*

The computation of the 0-attractor $\mathrm{Attr}_0(R)$ for the game $\mathcal{G}_r$ from Figure 3.1 is depicted in Figure 3.2. We have $\mathrm{Attr}_0^3(R) = \mathrm{Attr}_0^4(R)$, i.e., no new vertex is added at that stage. Thus, it follows that $\mathrm{Attr}_0^n(R) = \mathrm{Attr}_0^3(R)$ for every $n > 3$. Hence, also the union over all stages is equal to the third stage, i.e., $\mathrm{Attr}_0(R) = \mathrm{Attr}_0^3(R)$.

In general, the attractor stages become stationary after at most $|V|$ many steps, as the stages are monotonic in the subset relation. Hence, to compute the attractor $\mathrm{Attr}_i(R)$, we only have to compute $\mathrm{Attr}_i^{|V|}(R)$.

Remark 3.1. *Let $\mathcal{A}$ be an arena with vertex set V, let $R \subseteq V$, and let $i \in \{0,1\}$. Then, there is an $m \leq |V|$ such that*

$$R = \mathrm{Attr}_i^0(R) \subseteq \mathrm{Attr}_i^1(R) \subseteq \cdots \subseteq \mathrm{Attr}_i^m(R) = \mathrm{Attr}_i^{m+1}(R) = \mathrm{Attr}_i(R).$$

Next, we prove that the winning region of Player 0 in a reachability game is the 0-attractor of the reachability set R and that the winning region of Player 1 is the complement of the attractor.

Lemma 3.1. *Let $\mathcal{G} = (\mathcal{A}, \textsc{Reach}(R))$ with $\mathcal{A} = (V, V_0, V_1, E)$. Then, $W_0(\mathcal{G}) = \mathrm{Attr}_0(R)$ and $W_1(\mathcal{G}) = V \setminus \mathrm{Attr}_0(R)$.*

Proof. We show $\mathrm{Attr}_0(R) \subseteq W_0(\mathcal{G})$ and $V \setminus \mathrm{Attr}_0(R) \subseteq W_1(\mathcal{G})$. As $W_0(\mathcal{G})$ and $W_1(\mathcal{G})$ are disjoint and $\mathrm{Attr}_0(R)$ and $V \setminus \mathrm{Attr}_0(R)$ partition V, we conclude $\mathrm{Attr}_0(R) = W_0(\mathcal{G})$ and $V \setminus \mathrm{Attr}_0(R) = W_1(\mathcal{G})$.

We start with Player 0 by constructing a positional strategy σ that is winning from every vertex in $\mathrm{Attr}_0(R)$. Intuitively, the strategy moves the token closer and closer to R. To formalize this, we define the distance between a vertex in $\mathrm{Attr}_0(R)$ and R via

$$\delta(v) = \min\{n \in \mathbb{N} \mid v \in \mathrm{Attr}_0^n(R)\}$$

where $\min\ \emptyset = \infty$.

From the definition of the attractor, we can derive the following properties of the distance function.

1. $\delta(v) = 0$ if, and only if, $v \in R$, which holds by definition of $\mathrm{Attr}_0^0(R)$.

2. If $v \in V_0$ satisfies $0 < \delta(v) < \infty$, then it has a successor v' with $\delta(v') < \delta(v)$: such a v satisfies $v \in \mathrm{Attr}_0^{\delta(v)}(R) \setminus \mathrm{Attr}_0^{\delta(v)-1}(R)$, hence it is in $\mathrm{CPre}_0(\mathrm{Attr}_0^{\delta(v)-1}(R))$. As $v \in V_0$, v is in

$$\{v \in V_0 \mid v' \in \mathrm{Attr}_0^{\delta(v)-1}(R) \text{ for some successor } v' \text{ of } v\},$$

which yields a successor v' with the desired properties.

3. If $v \in V_1$ satisfies $0 < \delta(v) < \infty$, then every successor v' of v satisfies $\delta(v') < \delta(v)$: here, the reasoning is dual to the previous case. We again have $v \in \mathrm{Attr}_0^{\delta(v)}(R) \setminus \mathrm{Attr}_0^{\delta(v)-1}(R)$, hence it is in $\mathrm{CPre}_0(\mathrm{Attr}_0^{\delta(v)-1}(R))$ and now in

$$\{v \in V_1 \mid v' \in \mathrm{Attr}_0^{\delta(v)-1}(R) \text{ for all successors } v' \text{ of } v\}.$$

Thus, every successor v' has the desired properties.

Using these properties, we define a positional strategy and show that it is winning from every $v \in \mathrm{Attr}_0(R)$. If $0 < \delta(v) < \infty$ for some $v \in V_0$, then we define $\sigma(v) = v'$ for some successor v' of v with $\delta(v') < \delta(v)$. Such a successor exists, as argued above. Otherwise, i.e., if $\delta(v) \in \{0, \infty\}$ we define $\sigma(v)$ to be an arbitrary successor of v. This covers the case of vertices in R, where Player 0 has already won and can therefore move arbitrarily, and the case of vertices in $V \setminus \mathrm{Attr}_0(R)$. We will see that such vertices are only encountered after visiting R, when starting in $\mathrm{Attr}_0(R)$. Hence, Player 0 can move arbitrarily from there, too.

It remains to prove that σ is indeed a winning strategy. Let $\rho = \rho_0\rho_1\rho_2 \cdots \in \mathrm{Plays}(\mathcal{A}, \mathrm{Attr}_0(R), \sigma)$ be an arbitrary play that is consistent with σ and starting in $\mathrm{Attr}_0(R)$. We show that there exists an index m with $\rho_m \in R$ by induction on $\delta(\rho_0)$, which is not equal to ∞ by assumption.

For $\delta(\rho_0) = 0$, the claim follows immediately from property (1) above. Thus, assume we have $0 < \delta(\rho_0) < \infty$. Then, applying property 2 and the definition of σ in case $\rho_0 \in V_0$ or applying property 3 in case $\rho_0 \in V_1$, we obtain $\delta(\rho_1) < \delta(\rho_0)$, i.e., $\rho_1 \in \mathrm{Attr}_0(R)$. Furthermore, as σ is positional, the play $\rho_1\rho_2\rho_3 \cdots$ is consistent with σ. Thus, the induction hypothesis is applicable to $\rho_1\rho_2\rho_3 \cdots$ and yields a position m with $\rho_m \in R$. Hence, the position $m+1$ of $\rho_0\rho_1\rho_2 \cdots$ is in R, which concludes the induction step.

Thus, σ is a positional winning strategy for Player 0 from each vertex in $\mathrm{Attr}_0(R)$, i.e., $\mathrm{Attr}_0(R) \subseteq W_0(\mathcal{G})$.

Next, we show $V \setminus \mathrm{Attr}_0(R) \subseteq W_1(\mathcal{G})$. Let $X = V \setminus \mathrm{Attr}_0(R)$. By analyzing Construction 3.1 we obtain the following properties:

4. If $v \in X$, then $v \notin R$, which holds by definition of the attractor.

5. If $v \in X \cap V_0$, then every successor v' of v is in X: if v had a successor v' in $V \setminus X = \mathrm{Attr}_0(R)$, then v would be in $\mathrm{Attr}_0(R)$ as well.

6. If $v \in X \cap V_1$, then there is a successor v' of v in X: if every successor of v were in $V \setminus X = \mathrm{Attr}_0(R)$, then v would be in $\mathrm{Attr}_0(R)$ as well.

Using these properties, we define a positional strategy for Player 1 and show that it is winning from every $v \in X$. For $v \in X \cap V_1$, we define $\tau(v) = v'$ for some successor v' of v with $v' \in X$. Such a vertex always exists due to property 6. In contrast, for $v \in V_1 \setminus X$ we define $\tau(v) = v'$ for some arbitrary successor of v. This covers the vertices in $\mathrm{Attr}_0(R)$, from which Player 0 has a winning strategy. Hence, we only define τ for the sake of completeness on these vertices.

We show that τ is indeed a winning strategy from X, so let $\rho = \rho_0\rho_1\rho_2 \cdots \in \mathrm{Plays}(\mathcal{A}, X, \tau)$ be an arbitrary play starting in some vertex $\rho_0 \in X$ and consistent with τ. We show that all ρ_n are in X by induction on $n \in \mathbb{N}$. Thus, by property 4, $\rho_n \notin R$. Hence, ρ is a winning play for Player 1.

The induction start is trivial, as we have $\rho_0 \in X$ by assumption. Thus, let $n \geq 0$ with $\rho_n \in X$. Applying Property 6 and the definition of τ in case $\rho_n \in V_1$ and Property 5 in case $\rho_n \in V_0$ yields $\rho_{n+1} \in X$ as well. This concludes the induction step.

Hence, we have $X = V \setminus \mathrm{Attr}_0(R) \subseteq W_1(\mathcal{G})$, which concludes the proof as argued above. □

A strategy σ as defined above is an *attractor strategy*, as it attracts to R. Dually, a strategy τ as defined above traps the token in X. Hence, X is called a *trap* for Player 0, as she cannot escape X, if Player 1 plays according to τ. Note that both strategies are uniform positional winning strategies for their respective players.

Relying on Remark 3.1 and suitable data structures one can compute an attractor and corresponding strategies in linear time.

Lemma 3.2. *The attractor $\mathrm{Attr}_i(R)$ in an arena $\mathcal{A}$ with edges E can be computed in linear time in $|E|$. Furthermore, one can simultaneously compute the strategies σ and τ from the proof of Lemma 3.1.*

Proof. Exercise 3.3. □

Solving a game $\mathcal{G}$ amounts to computing the winning regions and winning strategies for both players. Our results on reachability games proved in Lemma 3.1 and Lemma 3.2 are summarized in the following theorem.

Theorem 3.1. *Reachability games are determined with uniform positional winning strategies and can be solved in linear time in the number of edges of the underlying arena.*

In the proof of Lemma 3.1 we have essentially argued that the complement of the 0-attractor of R is a trap for Player 1. This is true in general.

Remark 3.2. *The complement of an i-attractor is a trap for Player $1 - i$.*

To conclude, we state an important consequence of the previous remark, which is useful for constructing algorithms by iteratively applying attractor computations: removing an attractor yields a valid sub-arena, i.e., it does not introduce terminal vertices. The following corollary is obtained by combining Lemma 1 and the previous remark.

Corollary 3.1. *Let $\mathcal{A}$ be an arena with vertex set V, let $R \subseteq V$, and let $i \in \{0, 1\}$. Then, $\mathcal{A}\upharpoonright(V \setminus \mathrm{Attr}_i(R))$ is a valid sub-arena.*

3.2 Safety Games

The next type of winning condition we introduce is the so-called safety condition. Where for the reachability condition Player 0 has to reach a specific region R of the arena, in the safety condition Player 0 is not allowed to leave a specific region S of *safe* vertices.

In this subsection, we introduce an important concept, that of duality. In a reachability game, Player 1's goal is to remain in the region $V \setminus R$ forever, i.e., he has a safety condition. Dually, in a safety game, Player 1's goal is to reach $V \setminus S$. Thus, by swapping the roles of the players (formally, we swap V_0 and V_1) and replacing the set of winning plays by its complement, we turn a reachability game into a safety game and vice versa.

Our analysis of safety games is based solely on this duality, which allows us to transfer all results obtained for reachability games in the previous subsection to safety games.

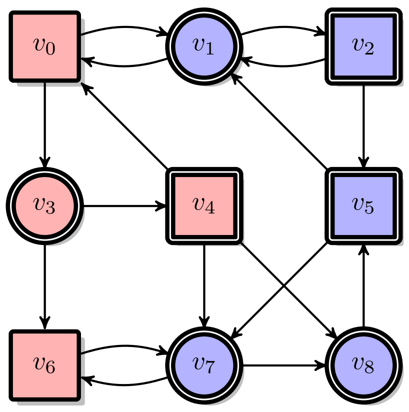


Figure 3.3: Example for a safety game. The doubly-framed vertices mark the safe region. Again, Player 0's winning region is blue, Player 1's red.

Definition 3.2. *Let $\mathcal{A} = (V, V_0, V_1, E)$ be an arena and let $S \subseteq V$ be a subset of $\mathcal{A}$'s vertices. Then, the safety condition $\textsc{Safety}(S)$ is defined as*

$$\textsc{Safety}(S) := \{\rho \in V^\omega \mid \mathrm{Occ}(\rho) \subseteq S\}.$$

We call a game $\mathcal{G} = (\mathcal{A}, \textsc{Safety}(S))$ a safety game *with set S of safe vertices.*

An example for such a game is given in Figure 3.3. Similarly to reachability games we mark the safe region of the game with doubly-framed vertices. At v_4, Player 1 can move to the unsafe vertex v_0. In contrast, from v_3 Player 0 can only move to v_6, which is unsafe, or to v_4 from which we already have seen that Player 1 can move to the unsafe region. Correspondingly, $W_1(\mathcal{G}) \subseteq \{v_0, v_3, v_4, v_6\}$. Note that we have essentially constructed the 1-attractor of $V \setminus S$. Dually, we have $W_0(\mathcal{G}) = \{v_1, v_2, v_5, v_7, v_8\}$, which is witnessed by the positional strategy that moves from v_1 to v_2, from v_7 to v_8, and from v_8 to v_5.

To state the general result connecting safety and reachability games, we introduce the arena obtained by swapping the vertices of the players.

Definition 3.3 (Duality)**.** *Let $\mathcal{A} = (V, V_0, V_1, E)$ be an arena. The dual arena $\overline{\mathcal{A}}$ of $\mathcal{A}$ is defined as $\overline{\mathcal{A}} = (V, V_1, V_0, E)$.*

Let $\mathcal{G} = (\mathcal{A}, \mathrm{Win})$ be a game with vertex set V. Then, $\overline{\mathcal{G}} = (\overline{\mathcal{A}}, V^\omega \setminus \mathrm{Win})$ is the dual game *of $\mathcal{G}$.*

Before we apply duality to solve safety games via dualization to reachability games, we prove a general duality lemma that relates the winning regions in $\mathcal{G}$ and $\overline{\mathcal{G}}$.

Lemma 3.3. *Let $\mathcal{G}$ be a game. Then, $W_i(\mathcal{G}) = W_{1-i}(\overline{\mathcal{G}})$ for $i \in \{0, 1\}$.*

Proof. Follows immediately from Exercise 3.6. □

We obtain our result about safety games as a simple corollary.

Corollary 3.2. *Let $\mathcal{A} = (V, V_0, V_1, E)$ be an arena, let $\mathcal{G} = (\mathcal{A}, \textsc{Safety}(S))$ be a safety game with $S \subseteq V$, and define the reachability game $\mathcal{G}' = (\overline{\mathcal{A}}, \textsc{Reach}(V \setminus S))$, which is the dual game of $\mathcal{G}$. Then, $W_i(\mathcal{G}) = W_{1-i}(\mathcal{G}')$ for each $i \in \{0, 1\}$.*

Thus, our main results about safety games directly follow from Theorem 3.1 and Corollary 3.2.

Theorem 3.2. *Safety games are determined with uniform positional winning strategies and can be solved in linear time in the number of edges of the underlying arena.*

3.3 Büchi Games

In the previous subsections, we studied games where one player has a reachability condition, i.e., reachability games and safety games. A play in such a game is essentially over as soon as the reachability set is visited for the first time: the winner has been determined and all future moves are irrelevant.

In this subsection, we study games in which no play prefix determines the winner, i.e., the winner depends on the complete infinite play. More formally, we require Player 0 to visit a given set F of vertices infinitely often. In the terminology of automata on infinite words, this is the Büchi acceptance condition: a run is accepting if infinitely many accepting states are visited. Consequently, the games we study here are called Büchi games.

Definition 3.4. *Let $\mathcal{A} = (V, V_0, V_1, E)$ be an arena and let $F \subseteq V$ be a subset of $\mathcal{A}$'s vertices. Then, the Büchi condition $\text{BÜCHI}(F)$ is defined as*

$$\text{BÜCHI}(F) := \{\rho \in V^\omega \mid \text{Inf}(\rho) \cap F \neq \emptyset\}.$$

We call a game $\mathcal{G} = (\mathcal{A}, \text{BÜCHI}(F))$ a Büchi game *with recurrence set F.*

As an introductory example, consider the Büchi game $\mathcal{G}$ depicted in Figure 3.4, where the doubly-framed vertices indicate the vertices in F. Hence, the goal of Player 0 is to visit $F = \{v_4, v_6\}$ infinitely often. Clearly, this is only possible from vertices in $\text{Attr}_0(F)$: from all other vertices Player 1 can prevent Player 0 from reaching F even once. Thus, we already conclude $V \setminus \text{Attr}_0(F) = \{v_0, v_1, v_2, v_5, v_8\} \subseteq W_1(\mathcal{G})$.

Now, consider the vertex v_4, which is in F. Thus, it is potentially desirable for Player 0 to reach v_4. However, it is a Player 1 vertex and has a successor that is already determined to be in the winning region of Player 1 (actually, this is the case for both successors). Thus, from v_4, Player 1 can ensure that F is only visited once. Thus, reaching v_4 is not at all desirable for Player 0: we can remove it from F. The situation for v_6, the other vertex in F, is different: it is a Player 1 vertex and all its successors, i.e., the vertex v_7, are not yet known to be winning for Player 1. Hence, we retain v_6.

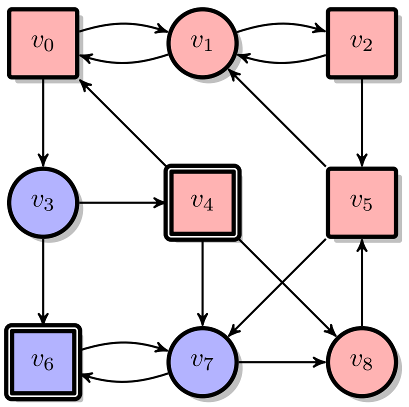


Figure 3.4: A Büchi game. The recurrence set is denoted by doubly-framed vertices.

Now, we repeat our reasoning for the smaller set $F = \{v_6\}$: Player 1 wins from every vertex from which Player 0 cannot attract to F. In particular, due to monotonicity of the attractor, this includes all vertices we already determined to be in $W_1(\mathcal{G})$, but now additionally v_4. Hence, $V \setminus \text{Attr}_0(\{v_6\}) = \{v_0, v_1, v_2, v_4, v_5, v_8\} \subseteq W_1(\mathcal{G})$. But even now, all successors of v_6 are not yet determined to be in Player 1's winning region. Hence, we do not remove it. Thus, we are in the same situation as before, i.e., our reasoning has reached a stationary situation.

We have already argued that Player 1 has a winning strategy from $\{v_0, v_1, v_2, v_4, v_5, v_8\}$, which is even positional: from v_4 move to (say) v_0 and then never visit v_4 and v_6 again. In contrast, Player 0 has a winning strategy from every other vertex, i.e., from the set $\{v_3, v_6, v_7\}$: attract to v_6, which is possible by construction and at v_6 move back to some vertex in $\{v_3, v_6, v_7\}$, which is possible as we have not removed v_6 from F.

The reasoning is summarized in Figure 3.5 and formalized in the following construction, which we prove to solve Büchi games in general. Here, we inductively compute increasing under-approximations W_1^n of Player 1's winning region $W_1(\mathcal{G})$ as follows: F^n contains the vertices from F to which recurring visits are still desirable for Player 0. Initially, we have $F^0 = F$, i.e., every vertex in F is potentially desirable. Then, in each round we determine the vertices from which Player 1 can prevent reaching F^n at all. These form the n-th under-approximation W_1^n. Then, we remove all vertices from F^n from which Player 1 can force to reach W_1^n, as these just turned out to be undesirable: although they are in F, Player 1 can ensure to reach his winning region in one step from there. Formally, we use the controlled predecessor defined in Construction 3.1 here. The removal yields the set F^{n+1} and we repeat the construction described above until the under-approximations become stationary. We show that they yield the winning regions of both players in this case.

Construction 3.2 (Recurrence)**.** *Let $\mathcal{A} = (V, V_0, V_1, E)$ be arena and let $F \subseteq V$. The* recurrence construction *for Player i is defined inductively as*

- $F^0 = F$,
- $W_1^n = V \setminus \text{Attr}_0(F^n)$ *for every $n \geq 0$, and*
- $F^{n+1} = F \setminus \text{CPre}_1(W_1^n)$ *for every $n \geq 0$.*

Figure 3.5 shows the execution of the recurrence construction for the example from Figure 3.4, containing all attractor computations and the increasing under-approximations W_1^n. After one step, the computation becomes stationary, i.e., we have $F^2 = F^1$ and therefore $W_1^2 = W_1^1$. Note that the final under-approximation W_1^1 is equal to the winning region of Player 1 as determined above (see Figure 3.4). We show that this is the case in general.

First, we remark that the computation of the sets F^n gets stationary after a linear number of steps, as they form a descending chain in the subset relation. This implies that the under-approximations W_1^n get stationary, too.

Lemma 3.4. *Let $\mathcal{A}$ be an arena with vertex set V of size n, let $F \subseteq V$, and let $i \in \{0,1\}$. Then, there is an $m \leq n$ such that*

$$F = F^0 \supseteq F^1 \supseteq \cdots \supseteq F^m = F^{m+1}$$

and

$$W_1^0 \subseteq W_1^1 \subseteq \cdots \subseteq W_1^m = W_1^{m+1}.$$

Proof. These sets forming chains in the subset relation follows from an inductive application of the following fact shown in Exercise 3.2: $A \subseteq B$ implies $\mathrm{CPre}_i(A) \subseteq \mathrm{CPre}_i(B)$ (and thus also implies $\mathrm{Attr}_i(A) \subseteq \mathrm{Attr}_i(B)$).

Thus, the existence of an index m as above is guaranteed by the finiteness of the arena. □

Now, we prove that the recurrence construction yields the winning regions in a Büchi game.

Lemma 3.5. *Let $\mathcal{G} = (\mathcal{A}, \text{BÜCHI}(F))$ be a Büchi game. Then, $W_1(\mathcal{G}) = \bigcup_{n\in\mathbb{N}} W_1^n$ and $W_0(\mathcal{G}) = V \setminus W_1(\mathcal{G})$.*

Proof. Let $\mathcal{G} = (\mathcal{A}, \text{BÜCHI}(F))$ be a Büchi game with $\mathcal{A} = (V, V_0, V_1, E)$ and define $X = V \setminus \bigcup_{n\in\mathbb{N}} W_1^n$. We show $X \subseteq W_0(\mathcal{G})$ and $V \setminus X \subseteq W_1(\mathcal{G})$. Then, Lemma 2.1 yields the desired equalities.

We begin with Player 0's winning region. Lemma 3.4 yields an m with $F^m = F^{m+1}$ and $W_1^m = W_1^{m+1}$. Hence, $\bigcup_{n\in\mathbb{N}} W_1^n = W_1^m$ and $X = \mathrm{Attr}_0(F^m)$.

First, we claim $F^m \subseteq \mathrm{CPre}_0(X)$, i.e., from F^m Player 0 can ensure to reach X again. To show this let $v \in F^m$ be arbitrary. Then, by construction, $v \in F \setminus \mathrm{CPre}_1(W_1^m)$ and correspondingly $v \notin \mathrm{CPre}_1(W_1^m)$. We distinguish two cases:

1. If $v \in V_0$, then v not being in $\mathrm{CPre}_1(W_1^m)$ implies the existence of a successor v' of v that is not in W_1^m. Hence, v' is in X and thus $v \in \mathrm{CPre}_0(X)$.
2. If $v \in V_1$, then v not being in $\mathrm{CPre}_1(W_1^m)$ implies that every successor v' of v is not in W_1^m. Hence, every such v' is in X and thus $v \in \mathrm{CPre}_0(X)$.

Now, we use the attractor strategy σ_A associated with the attractor $X = \mathrm{Attr}_0(F^m)$, i.e., from X, σ_A moves the token to $F^m \subseteq F$. Using this strategy, we define a positonal strategy σ for Player 0 as

$$\sigma(v) = \begin{cases} \sigma_A(v) & \text{if } v \in \mathrm{Attr}_0(F^m) \setminus F^m, \\ v' & \text{if } v \in F^m, \text{ for some successor } v' \in X \text{ of } v, \\ v'' & \text{otherwise, for some arbitrary successor } v'' \text{ of } v. \end{cases}$$

The existence of a successor v' as required in the second case is implied by $v \in F^m \subseteq \mathrm{CPre}_0(X)$.

We show that σ is a winning strategy for Player 0 from X. Thus, let $\rho = \rho_0\rho_1\rho_2\cdots \in \mathrm{Plays}(\mathcal{A}, X, \sigma)$ be arbitrary. We first show that ρ never leaves $X = \mathrm{Attr}_0(F^m)$, i.e., $\rho_n \in X$ for all n. The induction start follows from the assumption of ρ starting in X. Now, assume $\rho_n \in X$: if $\rho_n \in \mathrm{Attr}_0(F^m) \setminus F^m$, then $\rho_{n+1} \in \mathrm{Attr}_0(F^m)$: Player 0 uses her attractor strategy in this case, which moves the token to some successor in the attractor, and Player 1 cannot leave the attractor in this case. On the other hand, if $v \in F^m$, then the definition of σ (in case $\rho_n \in V_0$) respectively the fact that v is in $\mathrm{CPre}_0(X)$ (in case $\rho_n \in V_1$) yields the desired result.

We need to show that the set F is visited infinitely often by ρ. Accordingly, let $n \in \mathbb{N}$ be arbitrary and assume $\rho_n \notin F$. Then, $\rho_n \in \mathrm{Attr}_0(F^m) \setminus F$. As σ behaves like an attractor strategy on these vertices, the token reaches $F^m \subseteq F$ eventually. This already suffices, as ρ never leaves X. Hence, σ is indeed a winning strategy for Player 0 from X, i.e., $X \subseteq W_0(\mathcal{G})$.

Now, we consider Player 1, i.e., we show $\bigcup_{n\in\mathbb{N}} W_1^n = W_1^m \subseteq W_1(\mathcal{G})$. To this end, we define a function $\delta\colon W_1^m \to \mathbb{N}$ via

$$\delta(v) = \min\{n \in \mathbb{N} \mid v \in W_1^n\}$$

and show that Player 1 has a strategy that visits at most $\delta(v)$ many vertices in F from every vertex v in W_1^m. To this end, we show the following properties:

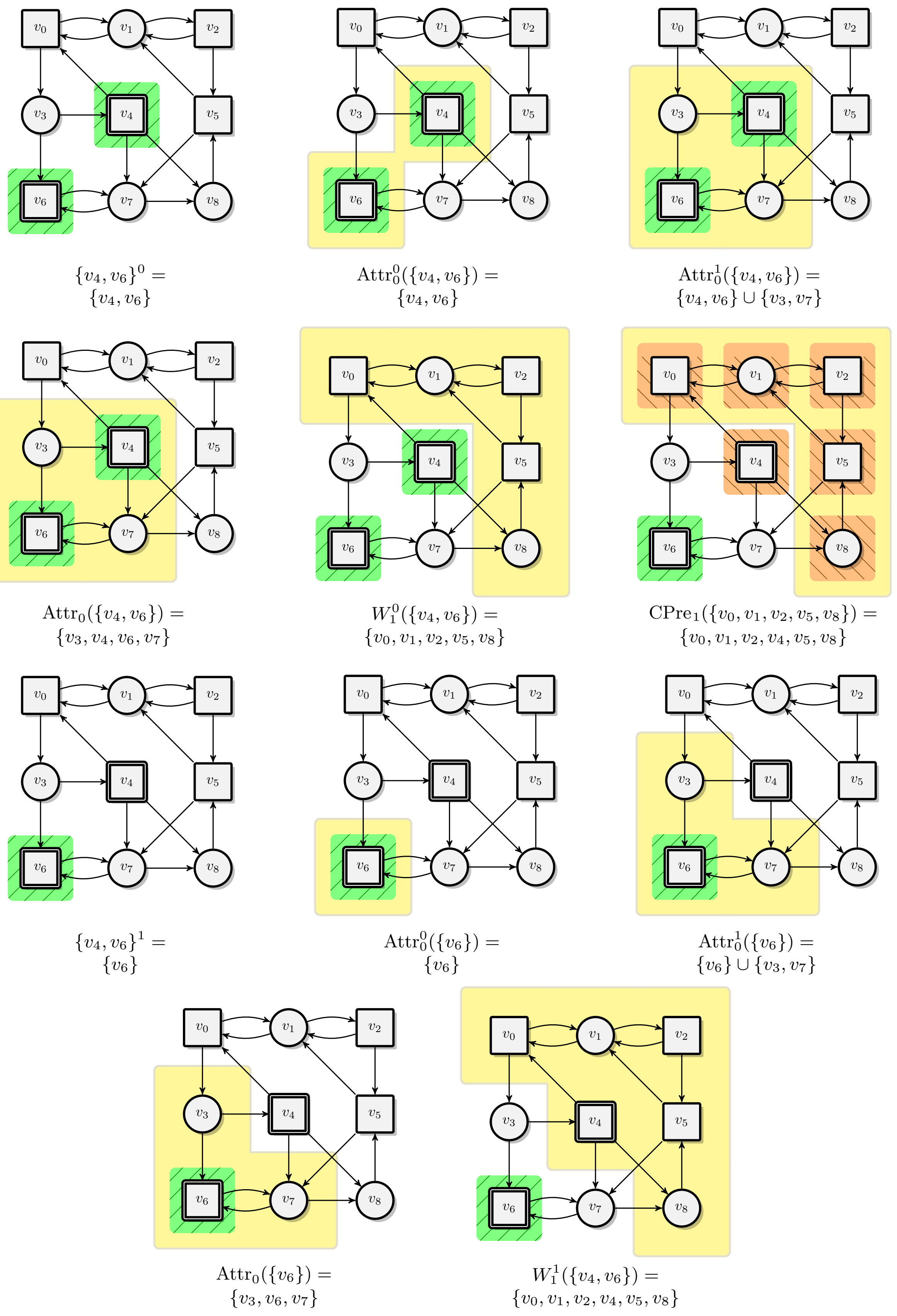

$\{v_4, v_6\}^0 =$ $\{v_4, v_6\}$
$\mathrm{Attr}_0^0(\{v_4, v_6\}) =$ $\{v_4, v_6\}$
$\mathrm{Attr}_0^1(\{v_4, v_6\}) =$ $\{v_4, v_6\} \cup \{v_3, v_7\}$
$\mathrm{Attr}_0(\{v_4, v_6\}) =$ $\{v_3, v_4, v_6, v_7\}$
$W_1^0(\{v_4, v_6\}) =$ $\{v_0, v_1, v_2, v_5, v_8\}$
$\mathrm{CPre}_1(\{v_0, v_1, v_2, v_5, v_8\}) =$ $\{v_0, v_1, v_2, v_4, v_5, v_8\}$
$\{v_4, v_6\}^1 =$ $\{v_6\}$
$\mathrm{Attr}_0^0(\{v_6\}) =$ $\{v_6\}$
$\mathrm{Attr}_0^1(\{v_6\}) =$ $\{v_6\} \cup \{v_3, v_7\}$
$\mathrm{Attr}_0(\{v_6\}) =$ $\{v_3, v_6, v_7\}$
$W_1^1(\{v_4, v_6\}) =$ $\{v_0, v_1, v_2, v_4, v_5, v_8\}$

Figure 3.5: Execution of the recurrence construction on the example from Figure 3.4. The essential parts of the calculations of F^n, $\mathrm{Attr}_i^n(R)$ and $\mathrm{CPre}_i(F)$ are highlighted by 🟩, 🟨 and 🟧, respectively.

1. $\delta(v) > 0$ for all $v \in W_1^m \cap F$: we have $W_1^0 = V \setminus \mathrm{Attr}_0(F)$ by definition, i.e., W_1^0 does not contain a vertex from F, which is a subset of $\mathrm{Attr}_0(F)$. Hence, $v \notin W_1^0$ for every $v \in F$, i.e., $\delta(v) > 0$ for every such v.

2. $\delta(v) \geq \delta(v')$ for all $v \in W_1^m \cap V_1$ and some successor v' of v and furthermore, if $v \in F$, then the inequality is strict: For the sake of contradiction assume that no such v' exists. Then, $\delta(v') > \delta(v)$ for every successor v' and accordingly $v' \notin W_1^{\delta(v)}$. It follows that all successors v' are in $\mathrm{Attr}_0(F^{\delta(v)})$, which implies $v \in \mathrm{CPre}_0(\mathrm{Attr}_0(F^{\delta(v)})) = \mathrm{Attr}_0(F^{\delta(v)})$, as the attractor is closed under the application of the controlled predecessor. But this is a contradiction against $v \in W_1^{\delta(v)} = V \setminus \mathrm{Attr}_0(F^{\delta(v)})$.

   Now, consider the case where v is additionally in F. Then, $\delta(v) > 0$ due to property 1., i.e., $v \in W_1^{\delta(v)}$ and $v \notin W_1^{\delta(v)-1}$. It follows $v \in V \setminus \mathrm{Attr}_0(F^{\delta(v)})$ and accordingly $v \notin \mathrm{Attr}_0(F^{\delta(v)})$. By Construction 3.1 it follows also $v \notin F^{\delta(v)}$ and $v \notin F \setminus \mathrm{CPre}_1(W_1^{\delta(v)-1})$. Hence, $v \in F$ implies $v \in \mathrm{CPre}_1(W_1^{\delta(v)-1})$. Accordingly by definition of CPre it follows that there exists a successor v' of v with $v' \in W_1^{\delta(v)-1}$. Accordingly $\delta(v') \leq \delta(v) - 1 < \delta(v)$.

3. $\delta(v) \geq \delta(v')$ for all $v \in W_1^m \cap V_0$ and all successors v' of v and furthermore, if $v \in F$, then the inequality is strict: the reasoning here is dual to the one for $v \in V_1$ in the previous case.

Intuitively, these properties say that Player 1 can ensure that the δ-value of the current vertex along a play does not increase and actually decreases with every visit to F. Hence, as the value is always non-negative and finite at the initial vertex (provided the play starts in W_1^m), this implies that F is visited only finitely often.

Relying on this intuition, we construct a strategy τ for Player 1 via

$$\tau(v) = \begin{cases} v' & \text{if } v \in W_1^m \cap F, \text{ for some successor } v' \text{ of } v \text{ with } \delta(v) > \delta(v'), \\ v' & \text{if } v \in W_1^m \setminus F, \text{ for some successor } v' \text{ of } v \text{ with } \delta(v) \geq \delta(v'), \\ v'' & \text{otherwise, for some arbitrary successor } v'' \text{ of } v. \end{cases}$$

The existence of the successors v' is guaranteed by the properties proven above.

It remains to show that τ is winning for Player 1 from W_1^m. Let $\rho = \rho_0\rho_1\rho_2 \cdots \in \mathrm{Plays}(\mathcal{A}, W_1^m, \tau)$ be arbitrary. By the definition of τ and by property 3. we have $\delta(\rho_n) \geq \delta(\rho_{n+1})$ for every $n \in \mathbb{N}$. Now assume ρ is not winning for Player 1, so there are infinitely many different positions $n_0, n_1, n_2, \ldots \in \mathbb{N}$ with $\rho_{n_j} \in F$. By definition of τ and by property 3. we obtain $\delta(\rho_0) > \delta(\rho_{n_0+1}) > \delta(\rho_{n_1+1}) > \cdots$ such that there must exist a $j \in \mathbb{N}$ with $\delta(\rho_{n_j}) = 0$. But since $\rho_{n_j} \in F$ this is a contradiction against property 1. Accordingly τ is winning for Player 1 for all plays starting in W_1^m, i.e., τ witnesses $W_1^m \subseteq W_1(\mathcal{G})$. □

Note that both strategies we defined are positional and winning from every vertex in the respective winning region. Our results on Büchi games are summarized in the following lemma.

Theorem 3.3. *Büchi games are determined with uniform positional winning strategies. They can be solved in polynomial time in the number of edges of the underlying arena.*

Proof. Positional determinacy with uniform winning strategies follows directly from the proof of Lemma 3.5. The complexity result follows from turning the recurrence construction into an algorithm: Lemma 3.4 and the linear-time algorithm for computing the attractor (Lemma 3.2) yield the desired polynomial upper bound on the running time. □

3.4 Co-Büchi Games

Similarly to safety games being the dual of reachability games, co-Büchi games are the dual of Büchi games, i.e., Player 0's goal in a co-Büchi game is to only visit vertices from a given set C infinitely often, i.e., vertices in $V \setminus C$ are only visited finitely often. Stated differently, from some point onwards only the play has to be confined to C. Thus, the co-Büchi condition can be seen as a generalization of the safety condition that allows a finite number of visits to unsafe vertices.

Definition 3.5. *Let $\mathcal{A} = (V, V_0, V_1, E)$ be an arena and let $C \subseteq V$ be a subset of $\mathcal{A}$'s vertices. Then, the co-Büchi condition $\textsc{coBüchi}(C)$ is defined as*

$$\textsc{coBüchi}(C) := \{\rho \in V^\omega \mid \mathrm{Inf}(\rho) \subseteq C\}.$$

We call a game $\mathcal{G} = (\mathcal{A}, \textsc{coBüchi}(C))$ a co-Büchi game *with persistence set C.*

Consider the example game in Figure 3.6 where the doubly-framed vertices mark the persistence set. Hence, Player 0 wins if v_0, v_1, and v_3 are visited only finitely often, otherwise Player 1 wins.

Player 0 wins from the vertices v_6 and v_7 by always moving from v_7 to v_6, from where Player 1 has no other choice but to move back to v_7 again. Moreover, Player 0 wins from v_3 by first moving to v_6 and then moving from v_7 back to v_6 ad infinitum. Hence, the token alternates between v_6 and v_7, which are both in the persistence set. Thus, the resulting plays are winning for Player 0. From every other vertex, Player 1 wins by playing according to the attractor strategy for v_4 at the vertices v_3, v_2, v_5, and v_8, and by moving positionally from v_4 to v_0 and from v_0 to v_1. Thus, the vertex v_1, which is not in the persistence set is visited infinitely often, i.e., such a play is winning for Player 1. Note that both winning strategies described above are positional.

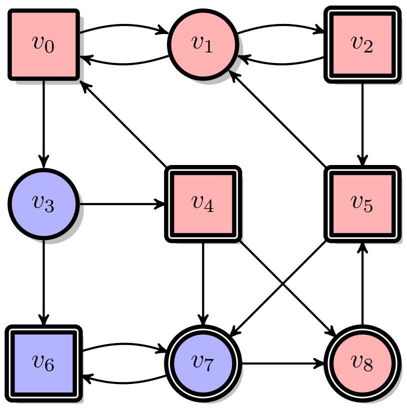


Figure 3.6: Example for a co-Büchi game. The doubly-framed vertices mark the persistence set.

As expected, we solve co-Büchi game via dualization to Büchi games.

Lemma 3.6. *Co-Büchi and Büchi games are dual.*

As a consequence, we obtain the same results for co-Büchi games as for Büchi games (see Theorem 3.3).

Theorem 3.4. *Co-Büchi games are determined with uniform positional winning strategies and can be solved in polynomial time in the number of edges of the underlying arena.*

3.5 Parity Games

After considering games in which Player 0's goal is to reach a fixed set of vertices at least once respectively infinitely often, we now consider parity games, which are a generalization of Büchi games. In a parity game, the vertices of the arena are *colored* by natural numbers and the goal of Player 0 is to ensure that the minimal color seen infinitely often on a play is even. Equivalently, Player i wins a play ρ if, and only if, the minimal color seen infinitely often in ρ has parity i. Thus, the color of a vertex denotes its importance (smaller colors are more important than larger ones) and its value for the players (vertices of even color are desirable for Player 0 but undesirable for Player 1 and vice versa). Hence, the colors are often called *priorities*.

Parity games play a central role in the theory of infinite games and have important applications in logics and automata theory, some of which we explore in the following chapters and in exercises. In particular, model-checking problems for fixed-point logics like the modal mu-calculus are expressible as parity games and in the case of the modal mu-calculus even equivalent in a very strong sense. Even more so, modern proofs of the decidability of monadic second-order logic over infinite trees rely heavily on positional determinacy parity games. This result, known as Rabin's theorem, is presented in the last chapter of these notes. Furthermore, the emptiness of alternating automata and of tree automata can naturally be expressed as a parity game and thereby solved.

Finally, the exact complexity of solving parity games is an intriguing open problem as well: we show that the problem is in NP and Co-NP. Thus, elementary complexity theory results imply that the problem being NP-complete or Co-NP-complete yields NP = Co-NP, which would be a breakthrough result in complexity theory. On the other hand, despite considerable efforts during the last twenty years, no polynomial-time algorithm is known.

Definition 3.6 (Parity Game). *Let $\mathcal{A} = (V, V_0, V_1, E)$ be an arena and let $\Omega \colon V \to \mathbb{N}$ be a* coloring *of $\mathcal{A}$'s vertices. Then, the* parity condition *$\textsc{Parity}(\Omega)$ is defined as*

$$\textsc{Parity}(\Omega) := \{\rho \in V^\omega \mid \min \mathrm{Inf}(\Omega(\rho_0)\Omega(\rho_1)\Omega(\rho_2)\cdots) \text{ is even}\}.$$

We call a game $\mathcal{G} = (\mathcal{A}, \textsc{Parity}(\Omega))$ a parity game.

An example of a parity game together with the winning regions for both players is given in Figure 3.7. The label of a vertex contains its name on the left- and its color on the right-hand side. Player 0 wins this game from the vertices v_3, v_6 and v_7 by cycling between the vertices v_6 and v_7, i.e., Player 0 moves from v_7 to v_6 and Player 1 has to move from v_6 back to v_7. From v_3, Player 0 moves to v_6, i.e., and then enforces the previously described cycle. The minimal color visited infinitely often on this cycle is 2, i.e., even. Thus, every play that is consistent with this strategy is winning for Player 0.

Player 1 wins from the remaining vertices. His winning strategy is to move from v_0 to v_1 from v_2 to v_5, from v_5 to v_1 and from v_4 to v_0. Every play that is consistent with this strategy eventually reaches v_1. From there, either Player 0 moves to v_0, from where Player 1 moves back to v_1, or Player 0 moves to v_2 and the Player 1 moves back to v_1 via v_5. On both of these cycles, the minimal color is odd, i.e., either 3 on the former or 1 on the latter. Thus, every play that is consistent with this strategy is winning for Player 1.

Note that both strategies described above are uniform positional winning strategies. We show that such strategies always exist for both players in parity games.

To start, however, we characterize the parity condition in terms of Büchi (and co-Büchi) conditions.

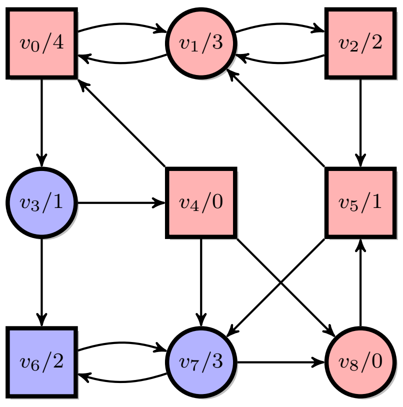


Figure 3.7: A parity game. The label of a vertex denotes its name v and its color $\Omega(v)$.

Lemma 3.7. *Every parity condition* $\textsc{Parity}(\Omega)$ *is a boolean combination of Büchi conditions (equivalently, of co-Büchi conditions).*

Proof. Exercise 3.9. □

Dually, Büchi and co-Büchi games can straightforwardly be expressed as parity games with two colors, e.g., assign color 0 to vertices in F and color 1 to all other vertices in the Büchi case, and assign color 2 to vertices in C and color 1 to all other vertices in the co-Büchi case. The resulting colorings witness the following claim.

Remark 3.3.

1. *Every Büchi condition* $\textsc{Büchi}(F)$ *is equal to a parity condition* $\textsc{Parity}(\Omega)$ *for some suitable coloring* Ω.
2. *Every co-Büchi condition* $\textsc{coBüchi}(C)$ *is equal to a parity condition* $\textsc{Parity}(\Omega)$ *for some suitable coloring* Ω.

Furthermore, the parity condition is quite robust to changes, which turns out to be useful for applications and for designing solution algorithms. As defined above, every vertex of the arena is colored by an arbitrary natural number and the minimal color seen infinitely often determines the winner. Hence, this condition is sometimes called min-parity to distinguish it from the (equivalent) max-parity condition defined below.

- Let $\mathcal{A}$ be an arena and let Ω be a coloring of $\mathcal{A}$'s vertices. The max-parity condition is defined as

$$\textsc{MaxParity}(\Omega) := \{\rho \in V^\omega \mid \max \mathrm{Inf}(\Omega(\rho_0)\Omega(\rho_1)\Omega(\rho_2)\cdots) \text{ is even}\},$$

  i.e., the parity of the maximal color seen infinitely often instead of the parity of the minimal one seen infinitely often determines the winner.

- Let $\mathcal{A}$ be an arena with vertex set V and let $\Omega \colon V \dashrightarrow \mathbb{N}$ be a *partial* coloring such that every cycle of $\mathcal{A}$ contains at least one vertex that is colored by Ω. We call such a partial coloring valid. If Ω is valid, then every infinite play visits infinitely many colored vertices, i.e., $\Omega(\rho_0)\Omega(\rho_1)\Omega(\rho_2)\cdots$ is an infinite sequence. Thus, we can define the parity condition for a partial coloring Ω as before:

$$\textsc{Parity}(\Omega) := \{\rho \in V^\omega \mid \min \mathrm{Inf}(\Omega(\rho_0)\Omega(\rho_1)\Omega(\rho_2)\cdots) \text{ is even}\}.$$

Lemma 3.8.

1. *Every max-parity condition* $\textsc{MaxParity}(\Omega)$ *is equal to a parity condition* $\textsc{Parity}(\Omega')$ *for some suitable coloring* Ω'.

2. *Fix an arena* $\mathcal{A} = (V, V_0, V_1, E)$. *For every valid partial coloring* $\Omega\colon V \dashrightarrow \mathbb{N}$ *there is a complete coloring* $\Omega'\colon V \to V$ *with* $\rho \in \textsc{Parity}(\Omega')$ *if, and only if,* $\rho \in \textsc{Parity}(\Omega)$ *for every* $\rho \in \mathrm{Plays}(\mathcal{A})$.[2]

3. *For every coloring* $\Omega\colon V \to \mathbb{N}$ *there is an injective coloring* $\Omega'\colon V \to \mathbb{N}$ *with* $\textsc{Parity}(\Omega') = \textsc{Parity}(\Omega)$.

4. *For every coloring* $\Omega\colon V \to \mathbb{N}$ *there is a coloring* $\Omega'\colon V \to \mathbb{N}$ *with* $\textsc{Parity}(\Omega') = \textsc{Parity}(\Omega)$ *and* $\Omega'(V) \subseteq \{0, 1, \ldots, |V|\}$.

5. *For every coloring* $\Omega\colon V \to \mathbb{N}$ *there is a coloring* $\Omega'\colon V \to \mathbb{N}$ *with* $\textsc{Parity}(\Omega') = \textsc{Parity}(\Omega)$ *such that* $\Omega'(V) \subseteq \Omega(V)$ *and* $\big|\,|\{c \in \Omega'(V) \mid c \text{ even}\}| - |\{c \in \Omega'(V) \mid c \text{ odd}\}|\,\big| \leq 1$.

Proof. Exercise 3.10. □

Our first result about parity games shows determinacy with uniform positional winning strategies and proceeds via induction over the number of vertices of the arena. Let us remark that there are many other proofs that proceed over the number of colors or over the number of edges.

Theorem 3.5. *Parity games are determined with uniform positional winning strategies.*

Proof. We prove the theorem by induction over the number n of vertices of the arena.

For the induction start $n = 1$, we have to consider a parity game $\mathcal{G} = (\mathcal{A}, \textsc{Parity}(\Omega))$ in an arena $\mathcal{A}$ with a single vertex v, which has a self-loop by assumption. Then, Player $\mathrm{Par}(\Omega(v))$ wins the only play v^ω in $\mathcal{A}$ and thus wins $\mathcal{G}$ from v. Furthermore, every strategy in $\mathcal{G}$ is positional, as there is only a single vertex. Hence, $\mathcal{G}$ is determined with uniform positional winning strategies.

For the induction step, fix a parity game $(\mathcal{A}, \textsc{Parity}(\Omega))$ with $\mathcal{A} = (V, V_0, V_1, E)$ such that $|V| = n > 1$. By induction hypothesis, every parity game with less vertices is determined with uniform positional winning strategies. In particular, it applies to every parity game whose arena is a proper sub-arena of $\mathcal{A}$.

Let $d = \min(\Omega(V))$ be the minimal color occurring in the game and define $i = \mathrm{Par}(d)$ to be the parity of d. Hence, if a play visits vertices of color d infinitely often, then it is winning for Player i, as there is no smaller color of parity $1 - i$. Thus, let $D = \{v \in V \mid \Omega(v) = d\}$ be the set of states colored with d and $A = \mathrm{Attr}_i(D) \neq \emptyset$. Finally, let σ_A be the associated attractor strategy forcing plays from A into D. The situation is depicted in Figure 3.8

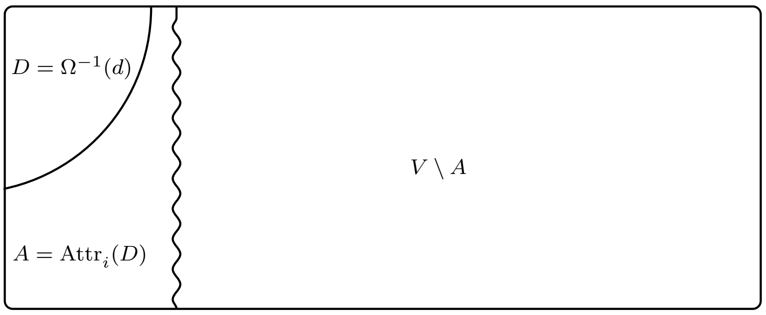


Figure 3.8: The structure of $\mathcal{A}$ in the inductive step: D is the set of vertices labeled by the minimal color d of parity i, and A is the i-attractor of D.

We consider two cases:

Case 1. "$A = V$": In this case, the region right of the wavy line in Figure 3.8 is empty. Thus, Player i can enforce a visit to D from every vertex. This enables her to visit D infinitely often by using her

[2]Note that we do not claim $\textsc{Parity}(\Omega') = \textsc{Parity}(\Omega)$!

attractor strategy again and again. This is winning for her as argued above. Formally, we define the positional strategy σ for Player i via

$$\sigma(v) = \begin{cases} \sigma_A(v) & \text{if } v \in A \setminus D, \\ v' & \text{if } v \in D, \text{ for some arbitrary successor } v' \text{ of } v. \end{cases}$$

To show that σ is a winning strategy from V, let $\rho = \rho_0\rho_1\rho_2 \cdots \in \text{Plays}(\mathcal{A}, V, \sigma)$ be an arbitrary play that is consistent with σ. By $A = V$ it follows that every vertex ρ_j of ρ is in A. If ρ_j is not in D, then ρ is consistent with the attractor strategy and therefore contains a vertex $\rho_{j'} \in D$ with $j' > j$. As this argument applies to every position j, we conclude that ρ visits D infinitely often. As d is the smallest color in the arena, it is the smallest color visited infinitely often along ρ. Thus, ρ is winning for Player i, as i is the parity of d. As ρ is an arbitrary play starting in an arbitrary vertex, we have shown that σ is a winning strategy from every vertex. Thus, Player i has a uniform positional winning strategy, which is winning from every vertex.

Case 2. "$A \neq V$": Consider the sub-arena $\mathcal{A}' = \mathcal{A}\upharpoonright(V \setminus A)$ inducing the game $\mathcal{G}' = (\mathcal{A}', \text{PARITY}(\Omega'))$ with $\Omega'(v) = \Omega(v)$ for all $v \in V \setminus A$. The arena and the game are well-defined due to Corollary 3.1. Also, as $A \supseteq D$ is non-empty, $\mathcal{G}'$ has less vertices than $\mathcal{G}$, i.e., the induction hypothesis is applicable: $\mathcal{G}'$ is positionally determined with uniform winning strategies σ' for Player i and τ' for Player $1-i$. The situation is depicted in Figure 3.9.

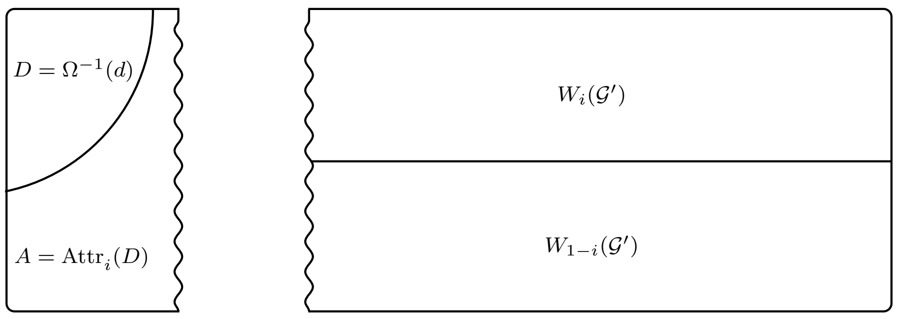


Figure 3.9: The structure of $\mathcal{A}$ in Case 2 of the inductive step: the set $V \setminus A$ induces the parity game $\mathcal{G}'$, which is determined by induction hypothesis.

To continue, we have to distinguish two subcases:

Subcase 1. "$W_{1-i}(\mathcal{G}') = \emptyset$": In this case, Player i wins $\mathcal{G}'$ from every vertex, as $W_{1-i}(\mathcal{G}')$ is empty (see Figure 3.10).

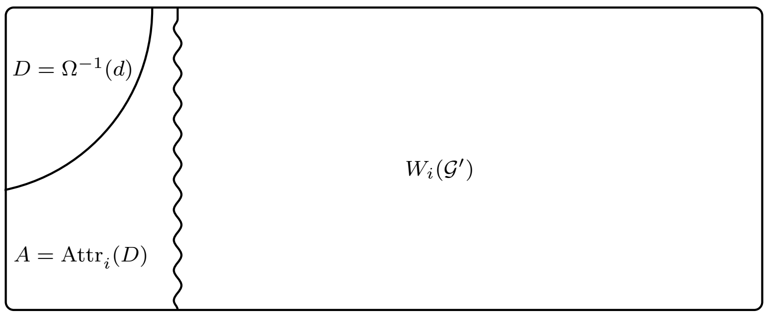


Figure 3.10: The structure of $\mathcal{A}$ in Subcase 1 of the inductive step: Player $1-i$'s winning region in $\mathcal{G}'$ is empty.

We show that $W_i(\mathcal{G}) = V$ by analyzing the following positional strategy σ for Player i.

$$\sigma(v) = \begin{cases} \sigma'(v) & \text{if } v \in W_i(\mathcal{G}'), \\ \sigma_A(v) & \text{if } v \in A \setminus D, \\ v' & \text{if } v \in D \text{ for some arbitrary successor } v' \text{ of } v. \end{cases}$$

To show that σ is a winning strategy from V, $\rho = \rho_0\rho_1\rho_2 \cdots \in \text{Plays}(\mathcal{A}, V, \sigma)$ be an arbitrary play that is consistent with σ. First, assume that there is a $j \in \mathbb{N}$ such that $\rho_{j'} \in V \setminus A$ for every $j' \geq j$. Then, by definition of σ it follows that $\rho' = \rho_j\rho_{j+1}\rho_{j+2} \cdots$ is consistent with σ', i.e., $\rho' \in \text{Plays}(\mathcal{A}', V \setminus A, \sigma')$. Thus, ρ' is winning for Player i in $\mathcal{G}'$. As the parity condition is prefix independent and Ω and Ω' coincide on $V \setminus A$, we conclude that ρ is winning for Player i as well.

Next assume that for all $j \in \mathbb{N}$ there exists a $j' \geq j$ such that $\rho_{j'} \in A$. Then, by definition of σ, there is an infix $\rho_{j'} \cdots \rho_{j''}$ that is consistent with σ_A which ends in a vertex from D. Thus, the color d is visited infinitely often, which again implies that ρ is winning for Player i.

Hence, as ρ is picked arbitrarily, we conclude that σ is indeed a uniform positional winning strategy for Player i from V.

Subcase 2. "$W_{1-i}(\mathcal{G}') \neq \emptyset$": In this subcase, which is the most involved one, we assume that Player $1-i$ wins $\mathcal{G}'$ from at least one vertex. Note that Player i won $\mathcal{G}$ from every vertex in all other cases. In this last case, Player $1-i$ has a non-empty winning region in $\mathcal{G}$ at last.

Let $B = \text{Attr}_{1-i}(W_{1-i}(\mathcal{G}')) \neq \emptyset$ be the attractor of $W_{1-i}(\mathcal{G}')$ in the arena $\mathcal{A}$. Then, the sub-arena $\mathcal{A}'' = \mathcal{A} \upharpoonright (V \setminus B)$ induces the game $\mathcal{G}'' = (\mathcal{A}'', \textsc{Parity}(\Omega''))$ with $\Omega''(v) = \Omega(v)$ for all $v \in V \setminus B$. Again this sub-arena and the game are well-defined due to Corollary 3.1. Furthermore, the induction hypothesis is applicable: $\mathcal{G}''$ is determined with uniform positional winning strategies σ'' for Player i and τ'' for Player $1-i$. The situation is depicted in Figure 3.11.

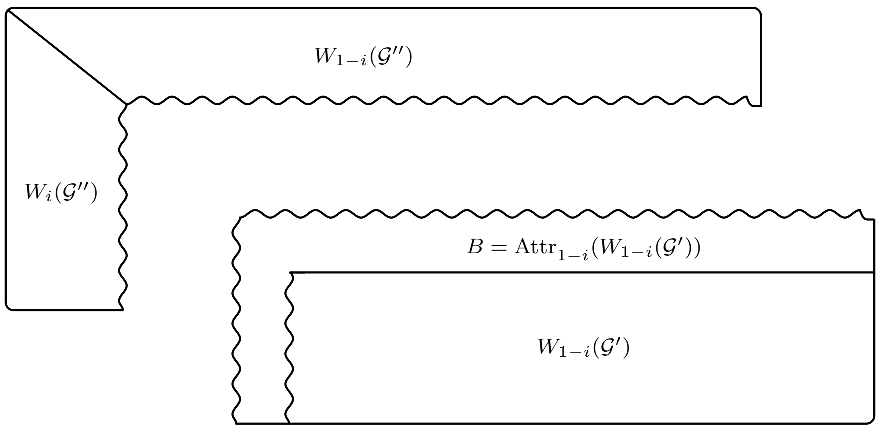


Figure 3.11: The structure of $\mathcal{A}$ in Subcase 2 of the inductive step: the complement $V \setminus B$ of the $(1-i)$-attractor of $W_{1-i}(\mathcal{G}')$ induces a parity game $\mathcal{G}''$, which is determined by induction hypothesis.

We show $W_i(\mathcal{G}) = W_i(\mathcal{G}'')$ and $W_{1-i}(\mathcal{G}) = W_{1-i}(\mathcal{G}'') \cup B$ by showing $W_i(\mathcal{G}'') \subseteq W_i(\mathcal{G})$ and $W_{1-i}(\mathcal{G}'') \cup B \subseteq W_{1-i}(\mathcal{G})$.

First, we show that the uniform positional winning strategy σ'' obtained from the induction hypothesis applied to $\mathcal{G}''$ is a positional winning strategy for Player i from $W_i(\mathcal{G}'')$ in the (larger) game $\mathcal{G}$, which implies $W_i(\mathcal{G}'') \subseteq W_i(\mathcal{G})$.[3]

To this end, let $\rho = \rho_0\rho_1\rho_2 \cdots \in \text{Plays}(\mathcal{A}, W_i(\mathcal{G}''), \sigma'')$ be arbitrary. First, we claim that ρ never leaves $W_i(\mathcal{G}'')$: if ρ_j is in $W_i(\mathcal{G}'')$ and in V_i, then $\rho_{j+1} = \sigma''(\rho_j)$. As σ'' is defined on $\mathcal{G}''$, we already have that ρ_{j+1} is a vertex of $\mathcal{G}''$. Towards a contradiction, assume we have $\rho_{j+1} = \sigma''(\rho_j) \in W_{1-i}(\mathcal{G}'t')$: then, Player $1-i$ has a winning strategy from ρ_{j+1}. Thus, he also wins from ρ_j against σ'', as the parity condition is prefix-independent. This contradicts σ'' being a winning strategy from ρ_j.

For dual reasons, we conclude $\rho_{j+1} \in W_i(\mathcal{G}'')$ in case $\rho_j \in W_i(\mathcal{G}'') \cap V_{1-i}$: Player $1-i$ cannot move to $W_{1-i}(\mathcal{G}'')$, as the winning region $W_i(\mathcal{G}'')$ is a trap for him. Also, he cannot move into B, as B is an attractor and the complement of the attractor is also a trap for him (recall Remark 3.2).

Hence, $\rho \in \textsc{Parity}(\Omega'')$ for $i = 0$ and $\rho \notin \textsc{Parity}(\Omega'')$ for $i = 1$ and therefore $\rho \in \textsc{Parity}(\Omega)$ for $i = 0$ and $\rho \notin \textsc{Parity}(\Omega)$ for $i = 1$, as Ω and Ω'' coincide on $W_i(\mathcal{G}'')$.

[3]Technically, we have to extend σ'' to $\mathcal{A}$ by picking arbitrary moves at vertices in V_i for which σ is undefined.

It remains to show that there is a positional strategy τ for Player $1-i$ that is winning from $W_{1-i}(\mathcal{G}'') \cup B$, which implies $W_{1-i}(\mathcal{G}'') \cup B \subseteq W_{1-i}(\mathcal{G})$. To this end, we define τ via

$$\tau(v) = \begin{cases} \tau''(v) & \text{if } v \in W_{1-i}(\mathcal{G}''), \\ \tau_B(v) & \text{if } v \in B \setminus W_{1-i}(\mathcal{G}'), \\ \tau'(v) & \text{if } v \in W_{1-i}(\mathcal{G}'), \\ v' & \text{otherwise, for some arbitrary successor } v' \text{ of } v. \end{cases}$$

Here, τ_B is an attractor strategy defined on B moving the token from $B \backslash W_{1-i}(\mathcal{G}')$ to $W_{1-i}(\mathcal{G}')$. We show that τ is indeed a winning strategy for Player $1-i$ from $W_{1-1}(\mathcal{G}'') \cup B$.

Let the play $\rho = \rho_0\rho_1\rho_2 \cdots \in \text{Plays}(\mathcal{A}, W_{1-i}(\mathcal{G}'') \cup B, \tau)$ be arbitrary. If there exists a position $j \in \mathbb{N}$ with $\rho_j \in W_{1-i}(\mathcal{G}')$, then $\rho_{j'} \in W_{1-i}(\mathcal{G}')$ for all $j' \geq j$, since $W_{1-i}(\mathcal{G}')$ is a trap for Player i in $\mathcal{A}$. This can be shown similarly to the claims above. Accordingly, the play $\rho' = \rho_j\rho_{j+1}\rho_{j+2} \cdots$ is in $\text{Plays}(\mathcal{A}', W_{1-i}(\mathcal{G}'), \tau')$ and therefore winning for Player $1-i$, i.e., $\rho' \notin \textsc{Parity}(\Omega')$ if $i = 0$ and $\rho' \in \textsc{Parity}(\Omega')$ if $i = 1$. Correspondingly, also $\rho \notin \textsc{Parity}(\Omega)$ if $i = 0$ and $\rho \in \textsc{Parity}(\Omega)$ if $i = 1$, as the parity condition is prefix independent and Ω and Ω' coincide. Hence, ρ is winning for Player $1-i$.

Thus, it remains to consider the case, where ρ starts in $W_{1-i}(\mathcal{G}'')$ and satisfies $\rho_j \notin B$ for every j. By definition of τ, such a play is consistent with τ'' and therefore winning for Player $1-i$ in $\mathcal{G}''$. Thus, it is also winning for Player $1-i$ in $\mathcal{G}$.

Hence, we have shown that $\mathcal{G}$ is indeed determined and that both players have uniform positional winning strategies. □

The inductive argument underlying the previous proof can be turned into a recursive algorithm solving parity games which only needs to compute attractors, which can be done very efficiently. However, as subcase 2 requires the solution of two parity games, i.e., $\mathcal{G}'$ and $\mathcal{G}''$, the algorithm makes (in the worst case) two recursive calls to solve one game. This yields an exponential upper bound on the running time of the algorithm and there are examples where an exponential number of calls are necessary.

Before we introduce the small progress measure algorithm for parity games, we discuss another consequence of positional determinacy of parity games: determining the winner from a given vertex is in NP ∩ Co-NP. Despite considerable effort, no polynomial-time algorithm for this problem is known. In contrast, if the problem is complete for NP or Co-NP, then this implies NP = Co-NP.

Theorem 3.6. *The following problem is in NP ∩ Co-NP: Given a parity game $\mathcal{G}$, a vertex v of $\mathcal{G}$, and $i \in \{0, 1\}$, is $v \in W_i(\mathcal{G})$?*

Proof. We prove the statement by giving a non-deterministic polynomial time algorithm and a universal polynomial-time algorithm.

Given a positional strategy σ for Player i in a game $\mathcal{G} = (\mathcal{A}, \text{Win})$ with $\mathcal{A} = (V, V_0, V_1, E)$, we can hardcode the moves prescribed by σ into the arena to obtain a one-player game, i.e., a game where only one player has non-trivial moves. Formally, we define $\mathcal{A}_\sigma = (V, V_0, V_1, E_\sigma)$ with

$$E_\sigma = \{(v, \sigma(v)) \mid v \in V_i\} \cup \{(v, v') \in E \mid v \in V_{1-i}\},$$

which is a subset of E: for every Player i vertex v, we remove all successors but the one picked by σ, for Player $1-i$ vertices, we retain all successors. Thus, in $\mathcal{A}_\sigma$, every Player i vertex has a unique successor, i.e., only Player $1-i$ has (potentially) several successors available at his vertices. We call a game with such an arena a single-player game. The following algorithm is based on the fact that single-player parity games can be solved in polynomial time and that σ is a winning strategy for Player i from a vertex v in a parity game $\mathcal{G}$ if, and only if, v is in the winning region of Player i in the corresponding single-player game $(\mathcal{A}_\sigma, \textsc{Parity}(\Omega))$ (see Exercise 3.12).

We show that the following algorithm determines the winner of a parity game in non-deterministic polynomial time, which places the problem in NP: Given an input $((\mathcal{A}, \textsc{Parity}(\Omega)), v, i)$, guess a positional strategy σ for Player i in $\mathcal{G}$ and determine whether Player i wins the single-player game $(\mathcal{A}_\sigma, \textsc{Parity}(\Omega))$ from v. As argued above, the running time is indeed polynomial. It remains to prove correctness of the algorithm

If Player i wins $(\mathcal{A}, \textsc{Parity}(\Omega))$ from v, then she also has a positional winning strategy due to Theorem 3.5. Thus, as mentioned above, v is in Player i's winning region in $(\mathcal{A}_\sigma, \textsc{Parity}(\Omega))$, i.e., the algorithm accepts the input. In contrast, if the algorithm accepts the input, then there is a positional strategy σ such that v is in the winning region of Player i in $(\mathcal{A}_\sigma, \textsc{Parity}(\Omega))$. Again, as stated above, this implies that σ is a winning strategy for Player i in $(\mathcal{A}, \textsc{Parity}(\Omega))$ from v.

Now, we show that the problem is in Co-NP by applying the self-duality of parity games and the duality of NP and Co-NP. Due to positonal determinacy of parity games, Player i wins a parity game $\mathcal{G}$ from a vertex v if, and only if Player $1-i$ does not win $\mathcal{G}$ from v, i.e., if, and only if Player $1-i$ does not have a positional winning strategy for $\mathcal{G}$ from v. Thus, the following algorithm places the problem in Co-NP: Given an input $((\mathcal{A}, \textsc{Parity}(\Omega)), v, i)$, verify that all positional strategies τ for Player $1-i$ in $\mathcal{G}$ result in a single-player game $(\mathcal{A}_\tau, \textsc{Parity}(\Omega))$ that is won by Player i from v. The correctness proof is dual to the one above. □

Another way to place the problem in Co-NP is via dualization: Player 1's goal is to ensure that the minimal color seen infinitely often is odd. By increasing every color by one, his goal is again a parity condition as introduced above: ensure that the minimal color seen infinitely often is even. Thus, the dual of the parity condition is again a parity condition: parity games are self-dual.

Remark 3.4. *Let $\mathcal{A}$ be an arena with vertex set V and let Ω be a coloring of V. Then, $V^\omega \setminus \textsc{Parity}(\Omega) = \textsc{Parity}(\Omega')$, where $\Omega'(v) = \Omega(v) + 1$ for every $v \in V$.*

As before for reachability and Büchi games, the determinacy proof for parity games can be turned into an algorithm for solving such games. In the remainder of this section, we present another algorithm to solve parity games, the so-called small progress measure algorithm. The algorithm is essentially a fixed-point computation for a suitable operator and is asymmetric, i.e., it computes the winning region and a uniform positional winning strategy for Player 0. To obtain a winning strategy for Player 1, one has to apply the algorithm to the dual game.

For Player 0, odd colors are undesirable, smaller ones even more than larger ones, as they are harder to counter. A positional winning strategy for Player 0 satisfies the following property: a play that is consistent with σ does not traverse a loop whose minimal color is odd. If it would, then Player 1 could traverse this loop ad infinitum and thereby win. Stated differently, such a play does not have $n_c + 1$ occurrences of an odd color c without a smaller even color in between, where n_c is the number of vertices of color c. If Player 1 can enforce such a situation, the initial vertex is in his winning region.

These observations hint at counting the occurrences of odd colors to compute a witness of a vertex being in Player 0's winning region. Here, such a witness is a vector $(s_1, s_3, \ldots, s_d)$, a so-called score sheet, where d is the largest odd color of the game. Intuitively, s_c represents that Player 1 can enforce at most s_c visits of color c before a smaller even one is encountered. Thus, the entry s_c is bounded by n_c, i.e., the score sheets are *small*. A *progress measure* is a labeling of the vertices by such a score sheet, subject to some requirements that allow to construct a winning strategy for Player 0. The algorithm computes such a small progress measure, which explains its name.

Informally, a progress measure is computed locally as follows: consider a vertex with some odd color c. Then, the score sheet $(s_1, s_3, \ldots, s_d)$ is initialized with $s_c = 1$ and $s_{c'} = 0$ for all $c' \neq c$ as every play starting in v encounters at least one occurrence of c. Vertices of even color are initialized with the zero vector $(0, \ldots, 0)$. Now, consider an arbitrary vertex v. Player 0 tries to minimize the number of odd colors encountered along a play, i.e., she prefers a successor whose score sheet has few occurrences of odd colors, where occurrences of small colors weigh more than occurrences of larger colors (technically, we order the score sheets by the lexicographic ordering). However, we also have to consider the effect of the vertex v has on the play: thus, we take the smallest score sheet of the successors of v and increment the entry with coordinate $\Omega(v)$, if $\Omega(v)$ is odd. If $\Omega(v)$ is even, then all entries s_c with $c > \Omega(v)$ are reset to zero, as $\Omega(v)$ is smaller than these colors. Dually, if it is Player 1's turn at v, we have to consider the largest successor, as he benefits from visiting odd colors.

One issue we have not yet dealt with is the situation where incrementing an entry s_c exceeds the bound b. To be able to apply fixed-point theory smoothly, the update has to be monotonic. This requires us to increment the next incrementable entry, i.e., the entry $s_{c'}$ for the largest $c' < c$ with $s_{c'} < n_{c'}$. If there is no such entry, then we assign the special score sheet $\top$, which is no longer updated. We will see that this represents vertices from which Player 1 can enforce a win.

We begin by introducing score sheets as the data structure tracking the occurrences of odd colors. Recall that $[n]$ denotes the set $\{0, 1, \ldots, n-1\}$. Also, review Subsection A.4, where we introduce the background on fixed-points we need to present the algorithm.

Definition 3.7 (Score Sheets). *Let $\mathcal{G} = (\mathcal{A}, \textsc{Parity}(\Omega))$ be a parity game with vertex set V. Furthermore, let $n_c = |\{v \in V \mid \Omega(v) = c\}|$ denote the number of vertices labeled by color c and let $d = \max\{c \in \Omega(V) \mid c \text{ odd}\}$ denote the largest odd color in $\mathcal{G}$.*

A score sheet *s of $\mathcal{G}$ is either the element $\top$ or a tuple*

$$s = (s_1, s_3, \dots, s_{d-2}, s_d) \in \prod_{c \in \{1,3,\dots,d\}} [n_c + 1].$$

The set of all score sheets of $\mathcal{G}$ is denoted by $\mathrm{Sh}(\mathcal{G})$.

Consider the parity game depicted in Figure 3.7, call it $\mathcal{G}_e$. Here, we have $d = 3$ and $n_1 = n_3 = 2$. Hence, $\mathrm{Sh}(\mathcal{G}_e) = \{0,1,2\} \times \{0,1,2\} \cup \{\top\}$.

Definition 3.8 (Score Sheet Update)**.** *We define the update of a score sheet s by a color c, denoted by $s \oplus c$, as $\top \oplus c = \top$ and for $s = (s_1, s_3, \dots, s_d)$ as*

$$s \oplus c = \begin{cases} (s_1, \dots, s_{c-1}, 0, \dots, 0) & \text{if } c \text{ is even,} \\ (s_1, \dots, s_{c'-2}, s_{c'} + 1, 0, \dots, 0) & \text{if } c \text{ is odd and } c' = \max\{c^* \leq c \mid c^* \text{ odd and } s_{c^*} < n_{c^*}\} \\ & \text{is defined,} \\ \top & \text{otherwise.} \end{cases}$$

In the first case of the definition, all entries with coordinates $c' > c$ are reset to zero. If $s \oplus c$ is obtained by applying the second case with $c' < c$ or by applying the third case, then we say that an overflow occurred.

Continuing the example from above, for $s = (1,1)$ we have $s \oplus 3 = (1,2)$, $(s \oplus 3) \oplus 3 = (2,0)$, i.e., an overflow occurs, and $(s \oplus 3) \oplus 2 = (1,0)$.

Small odd colors are more undesirable for Player 0 than larger ones, as they are harder to counter. Hence, we order score sheets lexicographically[4] and add $\top$ as largest element, i.e., $\top \geq s$ for every score sheet s.

Our first technical result about score sheets shows that the update operation $\oplus$ is monotonic with respect to this order.

Lemma 3.9. *If $s^0 \leq s^1$ for two score sheets s^0 and s^1, then $s^0 \oplus c \leq s^1 \oplus c$ for every color c.*

Proof. We begin by considering some special cases.

- If $s^1 = \top$, then $s^1 \oplus c = \top \geq s^0 \oplus c$, independently of s^0 and c, as $\top$ is maximal.
- If $s^0 = \top$, then also $s^1 = \top$ due to $s^1 \geq s^0 = \top$, i.e., we can reason as in the case above.
- If $s^0 = s^1$, then also $s^0 \oplus c = s^1 \oplus c$ and therefore $s^0 \oplus c \leq s^1 \oplus c$.

Thus, the only case left to consider is the one where we have

$$s^0 = (s^0_1, s^0_3, \dots, s^0_d) < (s^1_1, s^1_3, \dots, s^1_d) = s^1.$$

Then, by definition of the lexicographic ordering, there exists a color c^* such that $s^0_{c^*} < s^1_{c^*}$ and $s^0_{c'} = s^1_{c'}$ for all odd $c' < c^*$.

First, we assume that c is even. Then, we have

$$s^0 \oplus c = (s^0_1, s^0_3, \dots, s^0_{c-1}, 0, \dots, 0) \qquad \text{and} \qquad s^1 \oplus c = (s^1_1, s^1_3, \dots, s^1_{c-1}, 0, \dots, 0).$$

If $c < c^*$, then we have $s^0 \oplus c = s^1 \oplus c$, as we have $s^0_{c'} = s^1_{c'}$ for all $c' < c^*$, which subsumes all entries of $s^0 \oplus c$ and $s^1 \oplus c$ that are not reset. In contrast, if $c > c^*$ then the entries witnessing $s^0 < s^1$ are left unchanged when updating with c. Hence, we also have $s^0 \oplus c < s^1 \oplus c$.

Thus, assume c is odd and let $c_0 = \max\{c' \leq c \mid c' \text{ odd and } s^0_{c'} < n_{c'}\}$ the position where s^0 is incremented to obtain $s^0 \oplus c$ and similarly let $c_1 = \max\{c' \leq c \mid c' \text{ odd and } s^1_{c'} < n_{c'}\}$ the position where s^1 is incremented to obtain $s^1 \oplus c$. Note that both c_0 and c_1 might be undefined, which implies $s^0 \oplus c = \top$ and $s^1 \oplus c = \top$, respectively. Thus, if c_1 is undefined, then we are done, as $s^1 \oplus c = \top$ is the maximal element.

In contrast, if c_0 is undefined, then also c_1 is undefined: c_0 being undefined implies $s^0_{c'} = n_{c'}$ for every $c' \leq c$. As we have $n_{c^*} \geq s^1_{c^*} > s^0_{c^*}$ we conclude $c^* > c$ (otherwise, we have $s^0_{c^*} = n_{c^*}$ and therefore $n_{c^*} > n_{c^*}$). Thus, we have $s^1_{c'} = s^0_{c'} = n_{c'}$ for every $c' \leq c$, i.e., c_1 is indeed undefined. Thus, we are again done in this case, as argued above.

After having ruled out all special cases mentioned above, we now conclude the proof by a final case distinction in case $s_0 < s_1 < \top$, c is odd, and c_0 and c_1 are both defined.

[4]Recall that the lexicographic order $\leq$ on tuples of length n is defined as $(a_0, a_1, \dots, a_{n-1}) \leq (b_0, b_1, \dots, b_{n-1})$ if, and only if, there is a $j < n$ such that $a_j \leq b_j$ and $a_{j'} = b_{j'}$ for all $j' < j$. This order is total, i.e., any two tuples are comparable.

Case 1. "$c < c^*$": In this case, the positions c_0 and c_1 are equal, as c is smaller than c^*. This implies that both c_0 and c_1 are drawn from the common prefix of s^0 and s^1. Thus, we obtain

$$s^0 \oplus c = (s^0_1, s^0_1, \dots, s^0_{c_0-2}, s^0_{c_0} + 1, 0 \dots, 0) = (s^1_1, s^1_1, \dots, s^1_{c_1-2}, s^1_{c_1} + 1, 0 \dots, 0) = s^1 \oplus c.$$

Case 2. "$c = c^*$": In this case, we have $c_0 = c^*$, as $s^0_{c^*}$ is strictly smaller than $s^1_{c^*}$, which in turn is at most n_{c^*}. Also, we have $c_1 \leq c^*$ by definition.

If c_1 is equal to c^*, then we have

$$s^0 \oplus c = (s^0_1, s^0_3, \dots, s^0_{c^*-2}, s^0_{c^*} + 1, 0, \dots, 0) < (s^1_1, s^1_3, \dots, s^1_{c^*-2}, s^1_{c^*} + 1, 0, \dots, 0) = s^1 \oplus c,$$

as $s^0_{c'} = s^1_{c'}$ for all $c' < c^*$ and $s^0_{c^*} < s^1_{c^*}$.

In contrast, if c_1 is strictly smaller than c^*, then we have

$$s^0 \oplus c = (s^0_1, s^0_3, \dots, s^0_{c_1}, \dots, s^0_{c^*-2}, s^0_{c^*} + 1, 0, \dots, 0) < (s^1_1, s^1_3, \dots, s^1_{c_1} + 1, 0, \dots, 0) = s^1 \oplus c,$$

as $s^0_{c'} = s^1_{c'}$ for all $c' < c_1$, as c_1 is at most $c^* - 2$, and as $s^0_{c_1} = s^1_{c_1}$ implies $s^0_{c_1} < s^1_{c_1} + 1$, where the equality is again due to c_1 being strictly smaller than c^*. Thus, the entries with coordinates $1, 3, \dots, c_1$ witness $s^0 \oplus c < s^1 \oplus c$.

Case 3. "$c > c^*$": Here, we have $c_0 \geq c^*$, as $s^0_{c^*}$ is strictly smaller than $s^1_{c^*}$, which in turn is at most n_{c^*}. Thus, $s^0_{c^*}$ is small enough to be incremented, but there might be other larger positions with the same property, which take precedence.

If c_0 is equal to c^*, then we have

$$s^0 \oplus c = (s^0_1, s^0_3, \dots, s^0_{c^*} + 1, 0, \dots, 0).$$

- If $c_1 < c^* = c_0$, then $s^1 \oplus c = (s^1_1, s^1_3, \dots, s^1_{c_1} + 1, 0 \dots, 0)$. Hence, the entries with coordinates $1, 3, \dots, c_1$ witness $s^0 \oplus c < s^1 \oplus c$, as we have $s^0_{c'} = s^1_{c'}$ for every such coordinate c'.
- If $c_1 = c^* = c_0$, then $s^1 \oplus c = (s^1_1, s^1_3, \dots, s^1_{c^*} + 1, 0 \dots, 0)$. Hence, the entries with coordinates $1, 3, \dots, c^*$ witness $s^0 \oplus c < s^1 \oplus c$, as we have $s^0_{c'} = s^1_{c'}$ for every $c' < c^*$ and $s^0_{c^*} < s^1_{c^*}$.
- If $c_1 > c^* = c_0$, then $s^1 \oplus c = (s^1_1, s^1_3, \dots, s^1_{c^*}, \dots, s^1_{c_1} + 1, 0 \dots, 0)$. If the entries with coordinates $1, 3, \dots, c^*$ witness $s^0 \oplus c < s^1 \oplus c$, then we are done. If they do not, then $s^0 \oplus c$ and $s^1 \oplus c$ share a common prefix in these coordinates. As $s^0 \oplus c$ only contain zeros after this prefix, but $s^1 \oplus c$ contains at least one non-zero entry after c^*, we conclude $s^0 \oplus c < s^1 \oplus c$.

In contrast, if c_0 is greater than c^*, then $s^0 \oplus c$ starts with the prefix $(s^0_1, s^0_3, \dots, s^0_{c^*})$.

- If $c_1 \leq c^*$, then $s^1 \oplus c$ starts with the prefix $(s^1_1, s^1_3, \dots, s^1_{c_1} + 1)$. Hence, $s^0_{c'} = s^1_{c'}$ for every coordinate c' of the prefixes implies $s^0 \oplus c < s^1 \oplus c$.
- If $c_1 > c^*$, then $s^0 \oplus c$ starts with the prefix $(s^1_1, s^1_3, \dots, s^1_{c^*})$. Hence, $s^0_{c'} = s^1_{c'}$ for every $c' < c^*$ and $s^0_{c^*} < s^1_{c^*}$ implies $s^0 \oplus c < s^1 \oplus c$. □

Now, we define progress measures as mappings that assign a score sheet to each vertex subject to two constraints that allow to derive a winning strategy from a progress measure.

Definition 3.9 (Progress Measure)**.** *Let $\mathcal{G}$ be a parity game with vertex set V. A function $\wp\colon V \to \mathrm{Sh}(\mathcal{G})$ is a* progress measure *for $\mathcal{G}$, if*

- *every $v \in V_0$ has a successor v' with $\wp(v) \geq \wp(v') \oplus \Omega(v)$ and*
- *for every $v \in V_1$ and every successor v' of v, $\wp(v) \geq \wp(v') \oplus \Omega(v)$ holds.*

The set of all progress measures for $\mathcal{G}$ is denoted by $\mathrm{PM}(\mathcal{G})$. The evaluation $\|\cdot\|\colon \mathrm{PM}(\mathcal{G}) \to 2^V$ of a progress measure $\wp$ is defined as $\|\wp\| = \{v \in V \mid \wp(v) \neq \top\}$.

Note that progress measures consider edges in reverse order: the score sheet of v is compared to that of a successor v' updated by the color of v.

A progress measure $\wp_e$ for our running example game $\mathcal{G}_e$ is given by the following table:

v	v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8
$\wp_e(v)$	$\top$	$\top$	$\top$	$(1,0)$	$\top$	$\top$	$(0,0)$	$(0,1)$	$\top$

For example, we have $\wp_e(v_3) = (1,0) = (0,0) \oplus 1 = \wp_e(v_6) \oplus 1$ for the vertex $v_3 \in V_0$, $\wp_e(v_7) = (0,1) = (0,0) \oplus 3 = \wp_e(v_6) \oplus 3$ for $v_7 \in V_0$, and $\wp_e(v_6) = (0,0) = (0,1) \oplus 2 = \wp_e(v_7) \oplus 2$ for $v_6 \in V_1$, which only has the successor v_7. In contrast, for vertices labeled by $\top$, the requirement is trivially satisfied, as $\top$ is the maximal element.

Note that the evaluation of the progress measure $\wp_e$ is equal to the winning region of Player 0 and that the successors of v_3 and v_7 witnessing that $\wp_e$ is indeed a progress measure, v_6 in both cases, are exactly those picked by a winning strategy for Player 0 that is winning from $\|\wp_e\|$.

The next lemma shows that the evaluation of a progress measure is always a subset of her winning region and that there is a progress measure whose evaluation is her complete winning region. In both cases, we rely on a tight connection between positional winning strategies for Player 0 and progress measures.

A consequence of the following lemma and of positional determinacy is that an evaluation-maximal progress measure yields the winning regions of both players in a parity game. The small progress measure algorithm we present in the following computes such a progress measure.

Lemma 3.10. *Let $\mathcal{G}$ be a parity game.*

1. *$\|\wp\| \subseteq W_0(\mathcal{G})$ for every progress measure $\wp \in \mathrm{PM}(\mathcal{G})$.*
2. *There exists a progress measure $\wp \in \mathrm{PM}(\mathcal{G})$ with $\|\wp\| = W_0(\mathcal{G})$.*

Proof. 1.) This can be proven by constructing a strategy for Player 0 that is winning from $\|\wp\|$ (Exercise 3.13).

2.) We begin by introducing some notation. First, we define the function $sh\colon V^* \to \mathrm{Sh}(\mathcal{G})$ recursively as $sh(\varepsilon) = (0,\ldots,0)$ and $sh(wv) = sh(w) \oplus \Omega(v)$, i.e., we update the empty sheet by the sequence of colors encountered by the play prefix. Furthermore, define $\overleftarrow{sh}\colon V^* \to \mathrm{Sh}(\mathcal{G})$ as $\overleftarrow{sh}(w) = sh(w^R)$. Recall that w^R denotes the reversal of w, i.e., $(w_0 \cdots w_n)^R = w_n \cdots w_0$.

Now, fix a uniform positional winning strategy σ for Player 0 in $\mathcal{G}$ and define $\wp\colon V \to \mathrm{Sh}(\mathcal{G})$ via

$$\wp(v) = \begin{cases} \max\{\overleftarrow{sh}(w) \mid w \in \mathrm{Prefs}(\rho) \text{ for some } \rho \in \mathrm{Plays}(\mathcal{A}, v, \sigma)\} & \text{if } v \in W_0(\mathcal{G}), \\ \top & \text{otherwise.} \end{cases}$$

We claim that $\wp$ is a progress measure whose evaluation is equal to Player 0's winning region. First, we prove the latter claim and then show that $\wp$ is indeed a progress measure.

As we have proven $\|\wp\| \subseteq W_0(\mathcal{G})$ above, we only have to show $W_0(\mathcal{G}) \subseteq \|\wp\|$. Thus, let $v \in W_0(\mathcal{G})$. As $\top$ is the maximal element in $\mathrm{Sh}(\mathcal{G})$, we have to prove that $\overleftarrow{sh}(w) \neq \top$ holds for all $\rho \in \mathrm{Plays}(\mathcal{A}, v, \sigma)$ and all $w \in \mathrm{Prefs}(\rho)$.

First, we claim that every such w does not contain an infix with $n_c + 1$ occurrences of an odd color c and without an occurrence of a smaller even color in between. Towards a contradiction, assume there is such an infix. Then, there is a vertex repetition of some vertex of color c in this infix. Consider the play obtained by reaching this vertex and then traversing the loop induced by the repetition forever, which is consistent with σ. The minimal color in the loop, which is also the minimal color occurring infinitely often, is odd. This contradicts σ being winning from v.

Now, an induction over c from d to 1 shows that $\overleftarrow{sh}(w) = sh(w^R)$ does not have an overflow: c can either overflow if a larger color overflows (which is ruled out by the induction hypothesis) or if $n_c + 1$ occurrences of c without a reset in between are encountered. The latter situation yields an infix of w with $n_c + 1$ occurrences of c and without an occurrence of a smaller even color in between, which does not exist in w as shown above. Thus, $\overleftarrow{sh}(w)$ does not have an overflow. In particular, we conclude $\overleftarrow{sh}(w) \neq \top$.

It remains to show that $\wp$ is indeed a progress measure, i.e., that the conditions of Definition 3.9 are satisfied. Let $v \in V$. We distinguish three cases.

Case 1. "$v \in V_0 \cap W_0(\mathcal{G})$": We use the following property for $w = w_0 w_1 \cdots w_n \in V^*$: vw is consistent with σ if, and only if, $w_0 = \sigma(v)$ and w is consistent with σ.

We show $\wp(v) \geq \wp(\sigma(v)) \oplus \Omega(v)$, i.e., $\sigma(v)$ is the successor required by the definition. We have

$$\wp(v) = \max\{\overleftarrow{sh}(vw) \mid vw \in \mathrm{Prefs}(\rho) \text{ and } \rho \in \mathrm{Plays}(\mathcal{A}, v, \sigma)\}$$

$$\begin{aligned}
&= \max\{sh(w^R v) \mid vw \in \mathrm{Prefs}(\rho) \text{ and } \rho \in \mathrm{Plays}(\mathcal{A}, v, \sigma)\} \\
&= \max\{sh(w^R) \oplus \Omega(v) \mid w \in \mathrm{Prefs}(\rho) \text{ and } \rho \in \mathrm{Plays}(\mathcal{A}, \sigma(v), \sigma)\} \\
&= \max\{sh(w^R) \mid w \in \mathrm{Prefs}(\rho) \text{ and } \rho \in \mathrm{Plays}(\mathcal{A}, \sigma(v), \sigma)\} \oplus \Omega(v) \\
&= \max\{\overleftarrow{sh}(w) \mid w \in \mathrm{Prefs}(\rho) \text{ and } \rho \in \mathrm{Plays}(\mathcal{A}, \sigma(v), \sigma)\} \oplus \Omega(v) \\
&= \wp(\sigma(v)) \oplus \Omega(v).
\end{aligned}$$

Case 2. "$v \in V_1 \cap W_0(\mathcal{G})$": Here, we use the following property for $w = w_0 w_1 \cdots w_n \in V^*$: vw is consistent with σ if, and only if, w is consistent with σ and $(v, w_0) \in E$.

We have to show $\wp(v) \geq \wp(v') \oplus \Omega(v)$ for every successor v' of v. Indeed, we have

$$\begin{aligned}
\wp(v) &= \max\{\overleftarrow{sh}(vw) \mid vw \in \mathrm{Prefs}(\rho) \text{ and } \rho \in \mathrm{Plays}(\mathcal{A}, v, \sigma)\} \\
&= \max\{sh(w^R v) \mid vw \in \mathrm{Prefs}(\rho) \text{ and } \rho \in \mathrm{Plays}(\mathcal{A}, v, \sigma)\} \\
&= \max\{sh(w^R) \oplus \Omega(v) \mid w \in \mathrm{Prefs}(\rho),\ \rho \in \mathrm{Plays}(\mathcal{A}, v', \sigma) \text{ and } (v, v') \in E\} \\
&= \max\{\max\{sh(w^R) \mid w \in \mathrm{Prefs}(\rho) \text{ and } \rho \in \mathrm{Plays}(\mathcal{A}, v', \sigma)\} \oplus \Omega(v) \mid (v, v') \in E\} \\
&= \max\{\max\{\overleftarrow{sh}(w) \mid w \in \mathrm{Prefs}(\rho) \text{ and } \rho \in \mathrm{Plays}(\mathcal{A}, v', \sigma)\} \oplus \Omega(v) \mid (v, v') \in E\} \\
&= \max\{\wp(v') \oplus \Omega(v) \mid (v, v') \in E\} \\
&\geq \wp(v') \oplus \Omega(v) \text{ for all } (v, v') \in E.
\end{aligned}$$

Case 3. "$v \in W_1(\mathcal{G})$": In this case, we have $\wp(v) = \top$, which is the maximal element. Hence, $\wp(v) \geq \wp(v') \oplus \Omega(v)$ for all $(v, v') \in E$. □

To apply the Knaster-Tarski Theorem, we construct a complete lattice and a monotonic operator whose pre-fixed-points are the progress measures. To this end, we first define the ordering.

Definition 3.10 (Progress Measure Ordering). *Let $\mathcal{G} = (\mathcal{A}, \textsc{Parity}(\Omega))$ be a parity game with vertex set V and let $\mathcal{P}_\mathcal{G} = \{\wp \mid \wp\colon V \to \mathrm{Sh}(\mathcal{G})\}$. We define a partial order $\sqsubseteq$ on $\mathcal{P}_\mathcal{G}$ via $\wp \sqsubseteq \wp'$, if $\wp(v) \leq \wp'(v)$ for all $v \in V$.*

As usual, $\sqsubset$ denotes the strict variant of the order, i.e., $\wp \sqsubset \wp'$, if $\wp \sqsubseteq \wp'$ and $\wp \neq \wp'$. Also, note that $\mathcal{P}_\mathcal{G}$ is in general a super set of $\mathrm{PM}(\mathcal{G})$, as the elements of $\mathcal{P}_\mathcal{G}$ do not have to satisfy the requirements in the definition of progress measures.

Lemma 3.11. *Let $\mathcal{G}$ be a parity game. Then, $(\mathcal{P}_\mathcal{G}, \sqsubseteq)$ is a complete lattice.*

Proof. Reflexiveness, antisymmetry, and transitivity are straightforward to verify. Now, let $P \subseteq \mathcal{P}_\mathcal{G}$. Define $\wp_s \in \mathcal{P}_\mathcal{G}$ via

$$\wp_s(v) = \max\{\wp(v) \mid \wp \in P\}.$$

Then, $\wp_u$ is a supremum of P. An infimum of P is obtained by replacing the maximization in the definition of $\wp_s$ by a minimization. □

Next, we introduce the operator whose fixed-points we are interested in. Intuitively, the lift operator implements the local update of the score sheets described above: at Player 0 vertices v, it takes the minimal score sheet of a successor, updates it with the color of v, and then assigns the new score sheet to v. Dually, at Player 1 vertices, the maximal score sheet of the successors is considered.

Definition 3.11 (Lift-operator). *Let $\mathcal{G} = (\mathcal{A}, \textsc{Parity}(\Omega))$ be a parity game with $\mathcal{A} = (V, V_0, V_1, E)$. For every $v \in V$, we define the function $\mathrm{Lift}_v\colon \mathcal{P}_\mathcal{G} \to \mathcal{P}_\mathcal{G}$ via*

$$\mathrm{Lift}_v(\wp)(u) = \begin{cases} \wp(u) & \text{if } u \neq v, \\ \max\{\wp(v), \min\{\wp(v') \oplus \Omega(v) \mid (v, v') \in E\}\} & \text{if } u = v \text{ and } u \in V_0, \\ \max\{\wp(v), \max\{\wp(v') \oplus \Omega(v) \mid (v, v') \in E\}\} & \text{if } u = v \text{ and } u \in V_1. \end{cases}$$

Next, we need to prove that each lift operator is monotonic.

Lemma 3.12. *Let $\mathcal{G} = (\mathcal{A}, \textsc{Parity}(\Omega))$ be a parity game with vertex set V. Then, every lift-operator Lift_v is $\sqsubseteq$-monotonic, i.e., $\wp \sqsubseteq \wp'$ implies $\mathrm{Lift}_v(\wp) \sqsubseteq \mathrm{Lift}_v(\wp')$.*

Proof. We have to show $\wp(u) \leq \wp'(u)$ implies $\mathrm{Lift}_v(\wp)(u) \leq \mathrm{Lift}_v(\wp')(u)$ for all $u, v \in V$ and for all $\wp, \wp' \in \mathcal{P}_\mathcal{G}$. We consider the three cases in the definition of Lift_v:

Case 1. "$u \neq v$": In this case, we have $\mathrm{Lift}_v(\wp)(u) = \wp(u) \leq \wp'(u) = \mathrm{Lift}_v(\wp')(u)$.

Case 2. "$u = v$ and $u \in V_0$": Then,

$$\begin{aligned}
&\mathrm{Lift}_v(\wp)(u) \\
=\;& \max\{\wp(u), \min\{\wp(v') \oplus \Omega(v) \mid (v,v') \in E\}\} \\
\leq\;& \max\{\wp'(u), \min\{\wp(v') \oplus \Omega(v) \mid (v,v') \in E\}\} && (\wp(u) \leq \wp'(u)) \\
\leq\;& \max\{\wp'(u), \min\{\wp'(v') \oplus \Omega(v) \mid (v,v') \in E\}\} && \text{(Lemma 3.9)} \\
=\;& \mathrm{Lift}_v(\wp')(u).
\end{aligned}$$

Case 3. "$u = v$ and $u \in V_1$": Then,

$$\begin{aligned}
&\mathrm{Lift}_v(\wp)(u) \\
=\;& \max\{\wp(u), \max\{\wp(v') \oplus \Omega(v) \mid (v,v') \in E\}\} \\
\leq\;& \max\{\wp'(u), \max\{\wp(v') \oplus \Omega(v) \mid (v,v') \in E\}\} && (\wp(u) \leq \wp'(u)) \\
\leq\;& \max\{\wp'(u), max\{\wp'(v') \oplus \Omega(v) \mid (v,v') \in E\}\} && \text{(Lemma 3.9)} \\
=\;& \mathrm{Lift}_v(\wp')(u).
\end{aligned}$$

□

Next, we characterize the simultaneous pre-fixed-points of the lift-operators as progress measures.

Lemma 3.13. *Let $\mathcal{G}$ be a parity game. Then, $\wp \in \mathcal{P}_{\mathcal{G}}$ is a progress measure for $\mathcal{G}$ if, and only if, $\wp$ is a pre-fixed-point of all lift-operators, i.e., $\mathrm{Lift}_v(\wp) \sqsubseteq \wp$ for all $v \in V$.*

Proof. Let $\mathcal{G} = (\mathcal{A}, \textsc{Parity}(\Omega))$ with $\mathcal{A} = (V, V_0, V_1, E)$.

First, assume that $\wp$ is a progress measure and let $v \in V$ be arbitrary. By Definition 3.10, we have to show $\wp(u) \geq \mathrm{Lift}_v(\wp)(u)$ for all $u \in V$. We distinguish three cases.

Case 1. "$v \neq u$". We have $\mathrm{Lift}_v(\wp)(u) = \wp(u)$ and accordingly $\wp(u) \geq \wp(u)$.

Case 2. "$v = u$ and $u \in V_0$": Then, we have

$$\begin{aligned}
& \wp(u) \geq \wp(u') \oplus \Omega(u) \text{ for some successor } u' \text{ of } u \\
\Rightarrow\;& \wp(u) \geq \min\{\wp(u') \oplus \Omega(u) \mid (u,u') \in E\} \\
\Rightarrow\;& \wp(u) \geq \max\{\wp(u), \min\{\wp(u') \oplus \Omega(u) \mid (u,u') \in E\}\} \\
\Rightarrow\;& \wp(u) \geq \mathrm{Lift}_v(\wp)(u).
\end{aligned}$$

Case 3: "$v = u$ and $u \in V_1$": then, we have

$$\begin{aligned}
& \wp(u) \geq \wp(u') \oplus \Omega(u) \text{ for all successors } u' \text{ of } u \\
\Rightarrow\;& \wp(u) \geq \max\{\wp(u') \oplus \Omega(u) \mid (u,u') \in E\} \\
\Rightarrow\;& \wp(u) \geq \max\{\wp(u), \max\{\wp(u') \oplus \Omega(u) \mid (u,u') \in E\}\} \\
\Rightarrow\;& \wp(u) \geq \mathrm{Lift}_v(\wp)(u).
\end{aligned}$$

For the other implication, let $\mathrm{Lift}_v(\wp) \sqsubseteq \wp$ for all $v \in V$. By Definition 3.10, we have $\wp(u) \geq \mathrm{Lift}_v(\wp)(u)$ for all $u, v \in V$. We have to verify the two requirements of a progress measure.

First, we show that every $v \in V_0$ has a successor v' such that $\wp(v) \geq \wp(v') \oplus \Omega(v)$. We have

$$\begin{aligned}
& \wp(v) \geq \mathrm{Lift}_v(\wp)(v) \\
\Rightarrow\;& \wp(v) \geq \max\{\wp(v), \min\{\wp(v') \oplus \Omega(v) \mid (v,v') \in E\}\} \\
\Rightarrow\;& \wp(v) \geq \min\{\wp(v') \oplus \Omega(v) \mid (v,v') \in E\}
\end{aligned}$$

Thus, there exists indeed a successor v' of v with $\wp(v) \geq \wp(v') \oplus \Omega(v)$, namely the one realizing the minimum above.

Now, it remains to prove $\wp(v) \geq \wp(v') \oplus \Omega(v)$ for every $v \in V_1$ and every successor v' of v. We have

$$\begin{aligned}
& \wp(v) \geq \mathrm{Lift}_v(\wp)(v) \\
\Rightarrow\;& \wp(v) \geq \max\{\wp(v), \max\{\wp(v') \oplus \Omega(v) \mid (v,v') \in E\}\} \\
\Rightarrow\;& \wp(v) \geq \max\{\wp(v') \oplus \Omega(v) \mid (v,v') \in E\}
\end{aligned}$$

Thus, we have indeed $\wp(v) \geq \wp(v') \oplus \Omega(v)$ for every $v \in V_1$ and every successor v' of v. □

Thus, the pre-fixed-points of the lift-operators coincide with the progress measures and we prove that a least pre-fixed-point has the largest evaluation, which then has to be equal to the winning region of Player 0. The following construction yields the least pre-fixed-point.

Construction 3.3 (Small Progress Measures)**.** *Let $\mathcal{G} = (\mathcal{A}, \textsc{Parity}(\Omega))$ be a parity game with vertex set V. Furthermore, let $\wp_0$ be the unique element in $\mathcal{P}_\mathcal{G}$ such that $\wp_0(v) = (0, \dots, 0)$ for all $v \in V$.*

The small progress measure construction on $\mathcal{G}$ is then defined for all $n \in \mathbb{N}$ via $\wp^0_\mathcal{G} = \wp_0$ and

$$\wp^{n+1}_\mathcal{G} = \begin{cases} \mathrm{Lift}_v(\wp^n_\mathcal{G}) & \text{if there is a } v \in V \text{ with } \wp^n_\mathcal{G} \sqsubset \mathrm{Lift}_v(\wp^n_\mathcal{G}), \\ \wp^n_\mathcal{G} & \text{if } \mathrm{Lift}_v(\wp^n_\mathcal{G}) \sqsubseteq \wp^{n-1}_\mathcal{G} \text{for all } v \in V. \end{cases}$$

Finally, we define $\wp_\mathcal{G} = \max\{\wp^n_\mathcal{G} \mid n \in \mathbb{N}\}$, which is well-defined, as there are only finitely many functions in $\mathcal{P}_\mathcal{G}$.

Note that the construction as described above is non-deterministic, as we do not specify which lift-operator to apply in case several ones are applicable. The following results do not depend on how this non-determinism is resolved. However, the running time of the algorithm might depend on this choice.

Lemma 3.14. *Let $\mathcal{G} = (\mathcal{A}, \textsc{Parity}(\Omega))$ be a parity game with vertex set V and let $\wp_\mathcal{G}$ the result of the progress measure algorithm. Then, $W_0(\mathcal{G}) = \|\wp_\mathcal{G}\|$ and $W_1(\mathcal{G}) = V \setminus \|\wp_\mathcal{G}\|$.*

Proof. By Lemma 3.10(2) there exists a progress measure $\wp^*$ with $\|\wp^*\| = W_0(\mathcal{G})$. We have the following properties:

- $\wp_\mathcal{G}$ is the least pre-fixed-point of the Lift_v-operators, which can be shown similarly to the proof of Theorem A.1.
- $\wp^*$ is a pre-fixed-point of the Lift_v-operators (see Lemma 3.13).

Thus, we can conclude $\wp_\mathcal{G} \sqsubseteq \wp^*$, i.e., $\wp_\mathcal{G}(v) \leq \wp^*(v)$ for all $v \in V$. Thus, $\|\wp^*\| = W_0(\mathcal{G})$ implies $\wp_\mathcal{G}(v) \leq \wp^*(v) < \top$ for all $v \in W_0(\mathcal{G})$. It follows that $W_0(\mathcal{G}) \subseteq \|\wp_\mathcal{G}\|$ and together with Lemma 3.10(1) we obtain $W_0(\mathcal{G}) = \|\wp_\mathcal{G}\|$. Finally, Theorem 3.5 yields $W_1(\mathcal{G}) = V \setminus \|\wp_\mathcal{G}\|$. □

Reconsider the example given in Figure 3.7. We solve this game using the progress measure algorithm. An example execution is depicted in Figure 3.12. The resulting progress measure is the one presented above and yields both winning regions.

The following theorem wraps up the previous results and contains upper bounds on the time and space consumption of the algorithm.

Theorem 3.7. *Every parity game $\mathcal{G} = (\mathcal{A}, \textsc{Parity}(\Omega))$ with $\mathcal{A} = (V, V_0, V_1, E)$ and with k odd colors can be solved in time $\mathcal{O}(k \cdot \log|V| \cdot |E| \cdot \left(\frac{|V|}{k}\right)^k)$ time and space $\mathcal{O}(|V| \cdot \log|V| \cdot k)$.*

Proof. We use Construction 3.3 to solve parity games. By Lemma 3.14 we obtain the winning regions for both players and from the proof of Lemma 3.10(1) it follows that we also obtain a positional winning strategy for Player 0. Finally, as parity games are self-dual we obtain a winning strategy for Player 1 by solving the dual game.

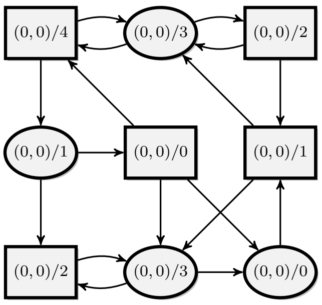
In Construction 3.3 we only need to store the current $\wp^n_\mathcal{G}$ of each iteration n. Each $\wp^n_\mathcal{G}$ consists of a score sheet for every vertex $v \in V$ and score sheets are tuples of length k. Finally, a score sheet only contains values of size at most $|V|$; hence, they can be stored in space $\log|V|$. Thus, the algorithm requires at most $\mathcal{O}(k \cdot |V| \cdot \log|V|)$ space.

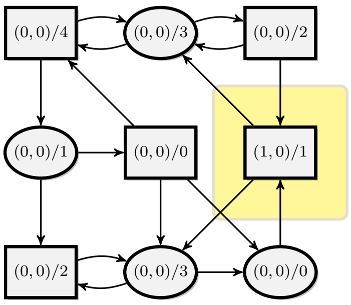
The number of iterations is bounded by $|\mathrm{Sh}(\mathcal{G})|$ and in each iteration we have to execute a lift operator, which requires us to inspect each edge at most once. Reading and writing the score sheets needs at most time $k \cdot \log|V|$ such that we obtain an upper bound of

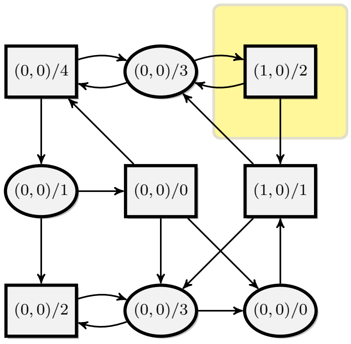
$$|\mathrm{Sh}(\mathcal{G})| \quad \leq \prod_{c \in \Omega(V) odd} (n_c + 1) \quad \leq \quad \left(\frac{|V|}{k}\right)^k$$

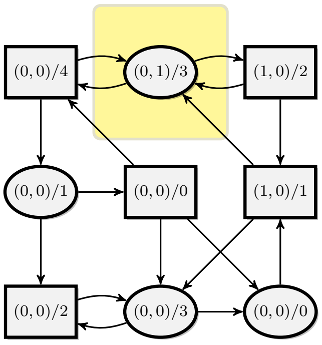
on the runtime of the algorithm. Here, the last inequality follows from the fact that the sum of all n_c exactly sums up to at most $|V|$ and choosing equally-sized n_c for each color c maximizes the product. □

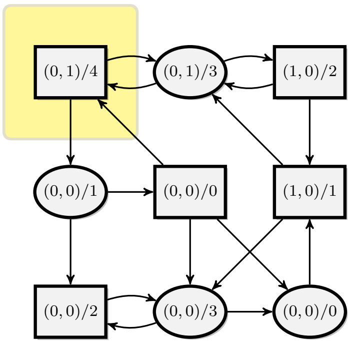
To conclude let us mention that there are examples where the algorithm requires an exponential running time. At the time of writing, it is an open problem whether parity games can be solved in polynomial time.

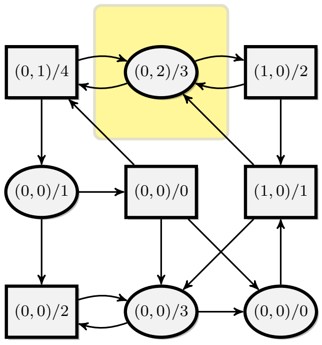

(0, 0)/4
(0, 0)/3
(0, 0)/2
(0, 0)/1
(0, 0)/0
(0, 0)/1
(0, 0)/2
(0, 0)/3
(0, 0)/0

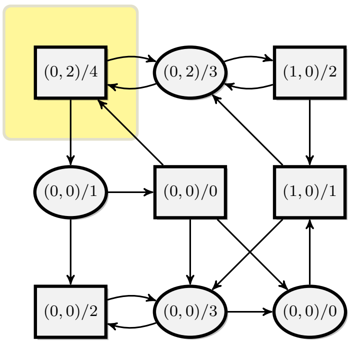

(0, 0)/4
(0, 0)/3
(0, 0)/2
(0, 0)/1
(0, 0)/0
(1, 0)/1
(0, 0)/2
(0, 0)/3
(0, 0)/0

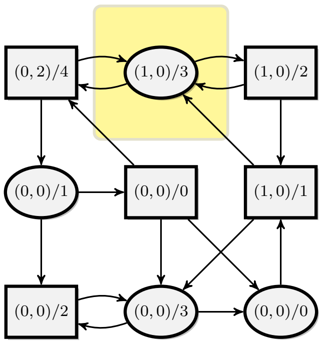

(0, 0)/4
(0, 0)/3
(1, 0)/2
(0, 0)/1
(0, 0)/0
(1, 0)/1
(0, 0)/2
(0, 0)/3
(0, 0)/0

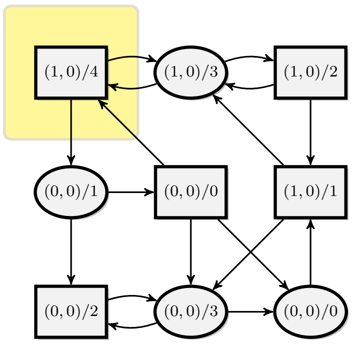

(0, 0)/4
(0, 1)/3
(1, 0)/2
(0, 0)/1
(0, 0)/0
(1, 0)/1
(0, 0)/2
(0, 0)/3
(0, 0)/0

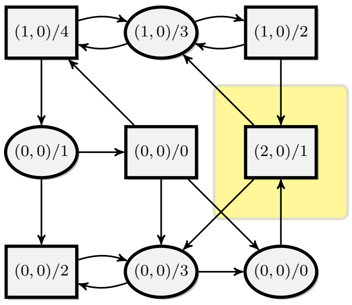

(0, 1)/4
(0, 1)/3
(1, 0)/2
(0, 0)/1
(0, 0)/0
(1, 0)/1
(0, 0)/2
(0, 0)/3
(0, 0)/0

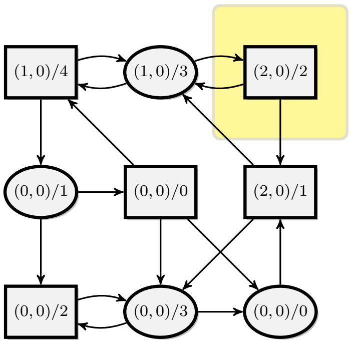

(0, 1)/4
(0, 2)/3
(1, 0)/2
(0, 0)/1
(0, 0)/0
(1, 0)/1
(0, 0)/2
(0, 0)/3
(0, 0)/0

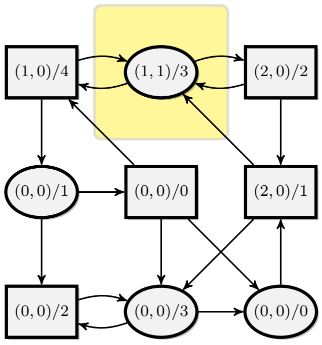

(0, 2)/4
(0, 2)/3
(1, 0)/2
(0, 0)/1
(0, 0)/0
(1, 0)/1
(0, 0)/2
(0, 0)/3
(0, 0)/0

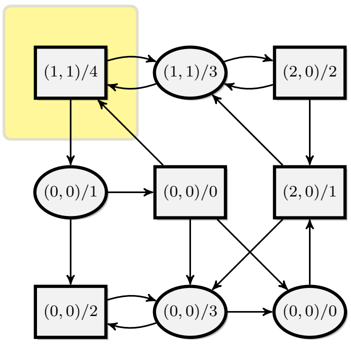

(0, 2)/4
(1, 0)/3
(1, 0)/2
(0, 0)/1
(0, 0)/0
(1, 0)/1
(0, 0)/2
(0, 0)/3
(0, 0)/0

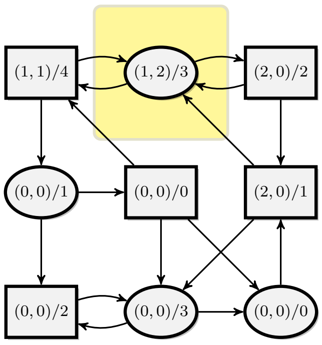

(1, 0)/4
(1, 0)/3
(1, 0)/2
(0, 0)/1
(0, 0)/0
(1, 0)/1
(0, 0)/2
(0, 0)/3
(0, 0)/0

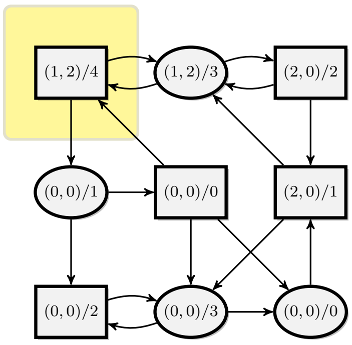

(1, 0)/4
(1, 0)/3
(1, 0)/2
(0, 0)/1
(0, 0)/0
(2, 0)/1
(0, 0)/2
(0, 0)/3
(0, 0)/0


(1, 0)/4
(1, 0)/3
(2, 0)/2
(0, 0)/1
(0, 0)/0
(2, 0)/1
(0, 0)/2
(0, 0)/3
(0, 0)/0


(1, 0)/4
(1, 1)/3
(2, 0)/2
(0, 0)/1
(0, 0)/0
(2, 0)/1
(0, 0)/2
(0, 0)/3
(0, 0)/0


(1, 1)/4
(1, 1)/3
(2, 0)/2
(0, 0)/1
(0, 0)/0
(2, 0)/1
(0, 0)/2
(0, 0)/3
(0, 0)/0


(1, 1)/4
(1, 2)/3
(2, 0)/2
(0, 0)/1
(0, 0)/0
(2, 0)/1
(0, 0)/2
(0, 0)/3
(0, 0)/0


(1, 2)/4
(1, 2)/3
(2, 0)/2
(0, 0)/1
(0, 0)/0
(2, 0)/1
(0, 0)/2
(0, 0)/3
(0, 0)/0

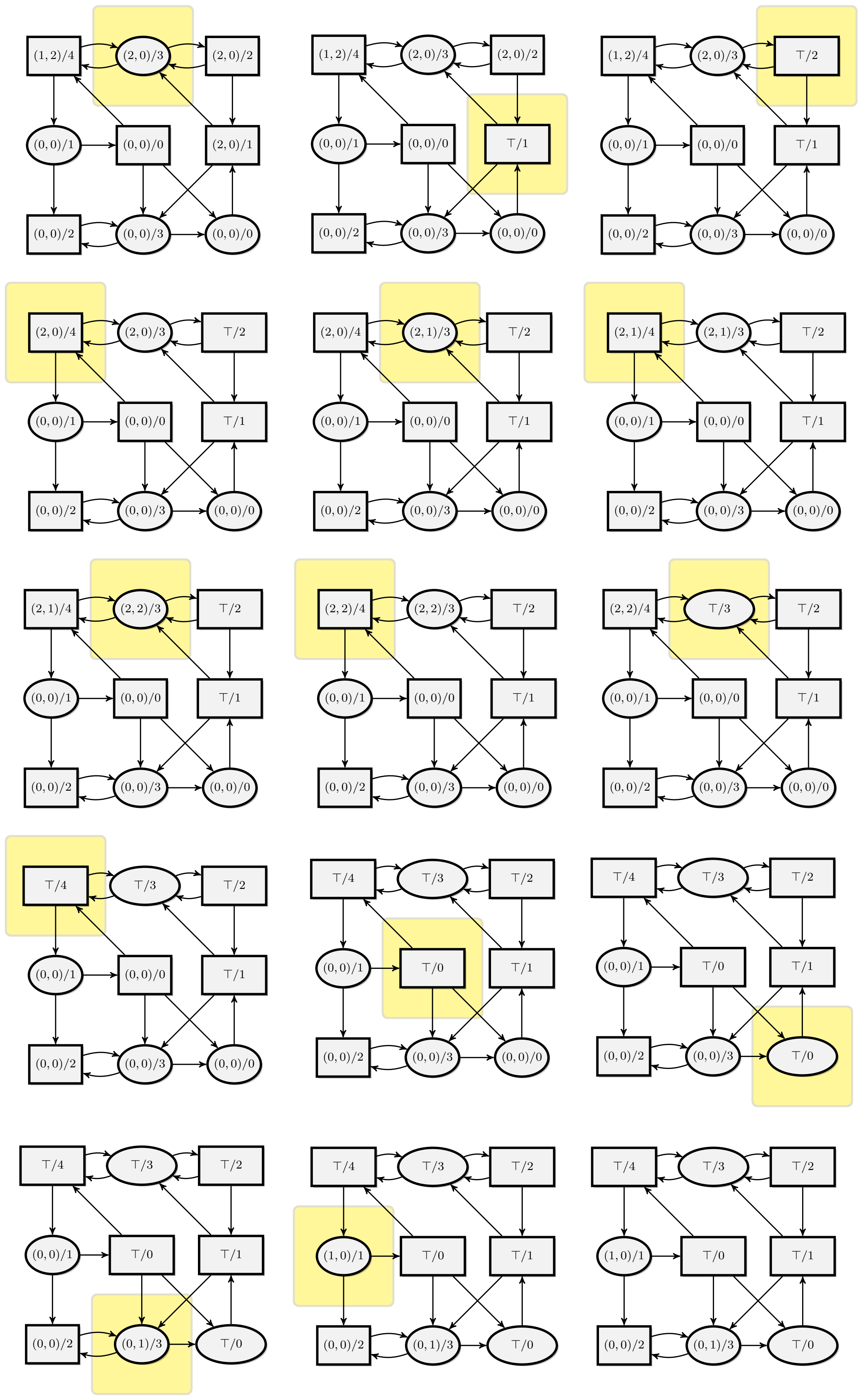
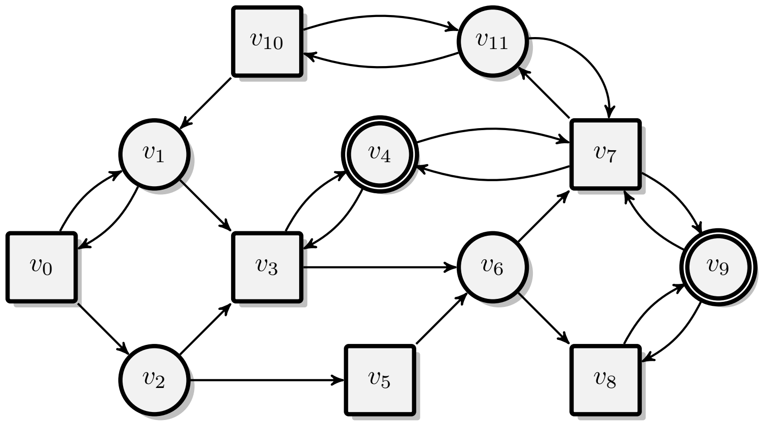


Figure 3.12: Progress measure algorithm on the game from Figure 3.7. The lifted nodes are marked.

3.6 Exercises

Exercise 3.1. Consider the reachability game $\mathcal{G} = (\mathcal{A}, \textsc{Reach}(R))$ depicted below.



1. Determine the attractor sets $\mathrm{Attr}_0^n(R)$ for all $n \in \mathbb{N}$.
2. Give the uniform winning strategies for both players resulting from the attractor construction. Argue how you constructed them.

Exercise 3.2. Let $\mathcal{A} = (V, V_0, V_1, E)$ be an arena and $R, R' \subseteq V$.

- Prove $\mathrm{CPre}_0(R) = V \setminus \mathrm{CPre}_1(V \setminus R)$
- Prove $R' \subseteq R$ implies $\mathrm{CPre}_i(R)\mathrm{CPre}_i(R')$ and $\mathrm{Attr}_i(R') \subseteq \mathrm{Attr}_i(R)$
- Prove or disprove $\mathrm{Attr}_i(R \cap R') = \mathrm{Attr}_i(R) \cap \mathrm{Attr}_i(R')$
- Prove or disprove $\mathrm{Attr}_i(R \cup R') = \mathrm{Attr}_i(R) \cup \mathrm{Attr}_i(R')$
- $\mathrm{Attr}_i(R) \setminus \mathrm{Attr}_i(R') = \mathrm{Attr}_i(R \setminus R')$
- $V \setminus \mathrm{Attr}_i(R) = \mathrm{Attr}_i(V \setminus R)$

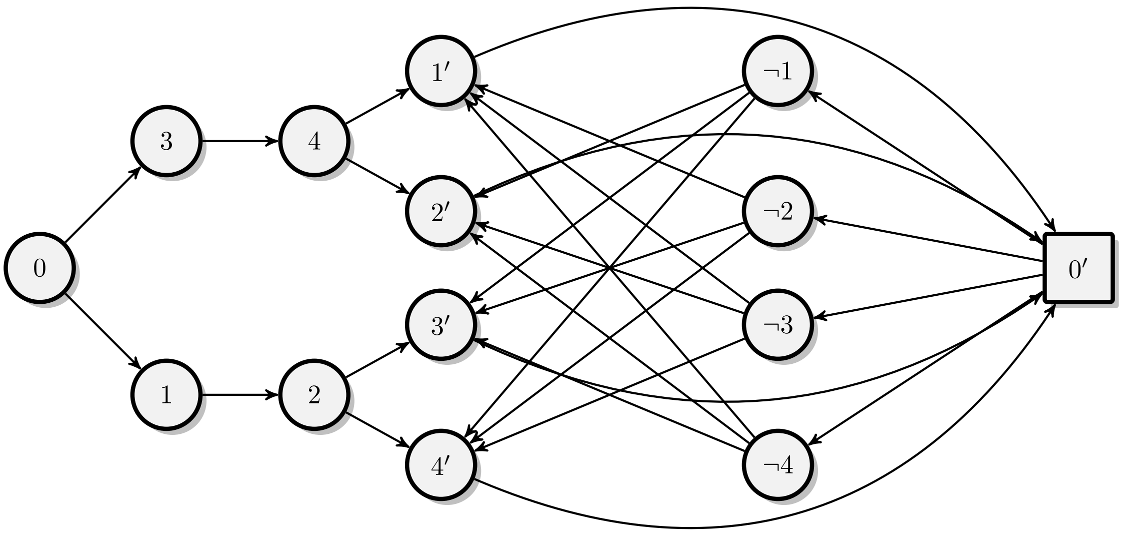
Exercise 3.3. Prove Lemma 3.2.

Exercise 3.4. Fix an arena $\mathcal{A} = (V, V_0, V_1, E)$. The generalized reachability condition for a family of subsets $\mathcal{R} \subseteq 2^V$ is defined as

$$\textsc{GenReach}(\mathcal{R}) = \{\rho \in V^\omega \mid \mathrm{Occ}(\rho) \cap R \neq \emptyset \text{ for all } R \in \mathcal{R}\}.$$

We call a game $\mathcal{G} = (\mathcal{A}, \textsc{GenReach}(\mathcal{R}))$ a *generalized reachability game.*

Let $\mathcal{A}$ be the arena depicted below:



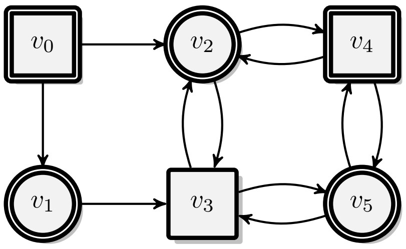
Consider the generalized reachability game $\mathcal{G} = (\mathcal{A}, \textsc{GenReach}(\mathcal{R}))$ with

$$\mathcal{R} := \{\{1, 1'\}, \{2, 2'\}, \{3, 3'\}, \{4, 4'\}\},$$

i.e., in order for a play to be winning for Player 0, it has to visit vertex j or vertex j' for each $j \in \{1, 2, 3, 4\}$.

1. Show that Player 1 has a winning strategy from vertex 0.
2. Show that Player 1 does not have a positional winning strategy from vertex 0.

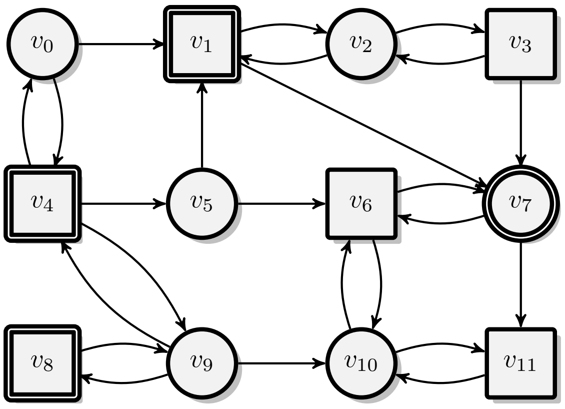
Exercise 3.5. Consider the safety game depicted below.



Solve the game by transforming it into an equivalent reachability game first, solving the reachability game and transforming back the results to the actual safety game.

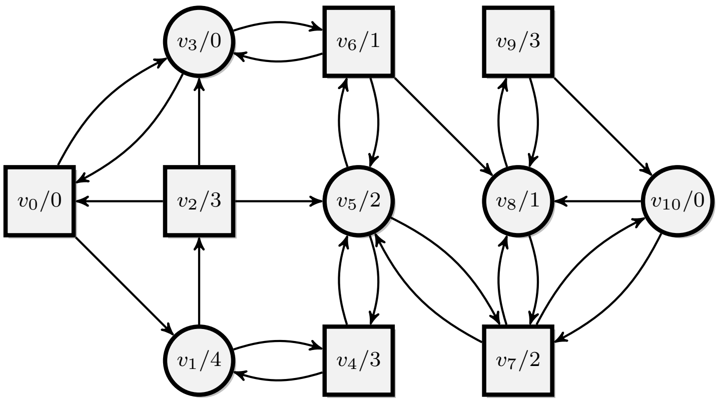
Exercise 3.6. Show for $i \in \{0, 1\}$: a winning strategy σ for Player i from a vertex $v \in V$ in a game $\mathcal{G}$ is also a winning strategy for Player $1 - i$ from v in the dual game $\overline{\mathcal{G}}$.

Exercise 3.7. Consider the Büchi game $\mathcal{G} = (\mathcal{A}, \textsc{Büchi}(F))$ depicted below.



Compute the winning region and a corresponding uniform positional winning strategy for each Player i.

Exercise 3.8. Consider the parity game $\mathcal{G}$ depicted below.



Compute the winning regions and uniform positional winning strategies for both players using the idea underlying the proof of Theorem 3.5.

Exercise 3.9. Prove Lemma 3.7.

Exercise 3.10. Prove Lemma 3.8.

Exercise 3.11. In a parity game, the goal for Player i is to ensure that the minimal color occuring infinitely often has parity i. By replacing "infinitely often" with "at least once" we obtain a definition for a weaker variant of parity games: the weak parity condition $\text{wParity}(\Omega)$ for a coloring $\Omega\colon V \to \mathbb{N}$ and an arena $\mathcal{A} = (V, V_0, V_1, E)$ is defined as

$$\text{wParity}(\Omega) := \{\rho \in V^\omega \mid \min \text{Occ}(\Omega(\rho_0)\Omega(\rho_1)\Omega(\rho_2)\cdots) \text{ is even}\}.$$

We call a game $\mathcal{G} = (\mathcal{A}, \text{wParity}(\Omega))$ a *weak parity game* with coloring Ω.

1. Give a polynomial-time algorithm that computes the winning regions and uniform positional winning strategies for both players in a weak parity game $\mathcal{G}$.
2. Consider the game $\mathcal{G} = (\mathcal{A}, \text{wParity}(\Omega))$ where $\mathcal{A}$ and Ω are defined as in Exercise 3.8.

   Determine the winning regions and uniform positional winning strategies of both players using your algorithm given in Part a).

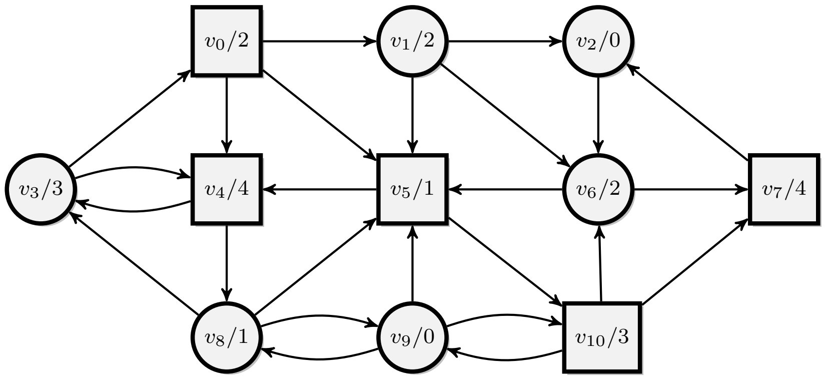
Exercise 3.12. A single-player game for Player i is a game played in an arena where every vertex of Player $1 - i$ has exactly one successor.

1. Let σ be a positional strategy for Player i in a game $(\mathcal{A}, \text{Win})$.

   Show that σ is a winning strategy for Player i from a vertex v in $\mathcal{G}$ if, and only if, v is in the winning region of Player i in the single-player game $(\mathcal{A}_\sigma, \text{Win})$.
2. Show that single-player parity games can be solved in polynomial time.

Exercise 3.13. Prove Lemma 3.10(1).

Exercise 3.14. Consider the parity game $\mathcal{G} = (\mathcal{A}, \text{Parity}(\Omega))$ with arena $\mathcal{A}$ and coloring Ω depicted below and apply the progress measure algorithm to compute the winning regions of the game.



Exercise 3.15 (Challenge)**.** A game $\mathcal{G} = (\mathcal{A}, \text{Win})$ with $\mathcal{A} = (V, V_0, V_1, E)$ is called *undirected* if $(v, v') \in E$ implies $(v', v) \in E$, i.e., the edge relation is symmetric. It is called *bipartite* if $E \subseteq V_0 \times V_1 \cup V_1 \times V_0$, i.e., the players move alternatingly.

Prove that undirected, bipartite parity games can be solved in polynomial time in the number of edges of the arena.

Exercise 3.16 (Challenge)**.** Prove that solving generalized reachability games (see Exercise 3.4) is Pspace-hard. Here, the size of a generalized reachability game $\mathcal{G} = (\mathcal{A}, \text{GenReach}(\mathcal{R}))$ is defined to be $|\mathcal{A}| + |\mathcal{R}|$.

4 Finite-state Strategies and Reductions

All games we have considered until now, e.g., reachability and safety, Büchi and co-Büchi, and parity, are determined with uniform positional winning strategies. Now, we move beyond positional strategies by considering games where such strategies do not suffice to win.

4.1 Finite-state Strategies

As an example, consider the game $\mathcal{G} = (\mathcal{A}, \mathrm{Win})$ where $\mathcal{A}$ is depicted in Figure 4.1 and where Win contains exactly those plays that visit all vertices infinitely often, i.e., $\rho \in \mathrm{Win}$ if, and only if, $\mathrm{Inf}(\rho) = \{v_0, v_1, v_2\}$. Player 0 has no positional winning strategy, as such a strategy either always moves from v_1 to v_0 or always moves from v_1 to v_2, i.e., v_2 respectively v_0 are visited at most once (namely, if the play starts in this vertex). On the other hand, she has a simple winning strategy that alternates between moving from v_1 to v_0 and to v_2, e.g., $\sigma(wv_1) = v_2$ if wv_1 contains an odd number of occurrences of v_1 and $\sigma(wv_1) = v_0$ otherwise. This strategy can be seen as switching between two positional strategies every time the vertex v_1 is visited.

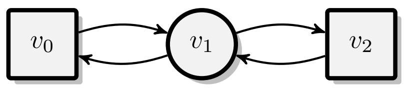


Figure 4.1: A game where Player 0 has no positional strategy that visits all vertices infinitely often.

As presented above, this strategy is an infinite object, as it has an infinite domain. Nevertheless, there is an informal finite description: remember the parity of the number of occurrences of v_1 in the the play thus far and if at v_1 move to v_2, if it is odd. As this description has only a finite number of states, namely two, one calls such a strategy *finite-state*. To formalize the intuitive description of finite-state strategies, we employ deterministic finite automata with output. The automaton reads the current play prefix and outputs the successor to move to. An automaton implementing the strategy σ is depicted in Figure 4.2. Note that unlike classical finite automata, the automaton does not have a single initial state, but an initial arrow for each vertex of the arena. For example, if the first vertex to be processed is v_0, then the runs starts in q_0 and processing v_1 then leads the automaton to state q_1. This definition mimics the formal definition, which employs an initialization function for technical reasons that become clear when we discuss reductions.

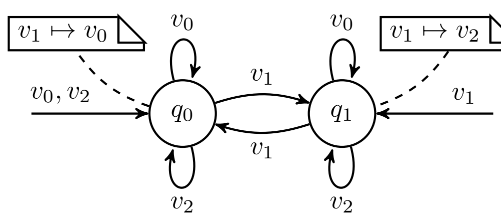


Figure 4.2: Automaton implementing the winning strategy σ for the game in Figure 4.1.

Furthermore, the outputs of the automaton are given as positional strategies, one attached to each state of the automaton. To illustrate the semantics, consider the play prefix $w = v_0v_1v_0v_1v_2v_1v_2v_1$. Processing w brings the automaton to the state q_0. As the last vertex of w is v_1, i.e., it is Player 0's turn, she uses the positional strategy attached to q_0, i.e., she moves to v_0.

Next, we introduce finite-state strategies formally. For reasons that become clear in the next subsection, we split the definition of automata with output into two parts. First, we define a variant of classical finite automata (without output) that is suitable for our purposes. Such automata are called *memory structures*, since they represent the memory needed to implement a finite-state strategy. Then, we define the output function of such an automaton as a separate object, called a *next-move function.* A finite-state strategy is then given by a memory-structure and a next-move function.

Definition 4.1 (Memory Structure)**.** *Let $\mathcal{A} = (V, V_0, V_1, E)$ be an arena. A* memory structure *$\mathcal{M} = (M, \mathrm{init}, \mathrm{upd})$ for $\mathcal{A}$ consists of*

- *a finite set M of memory states,*
- *an initialization function $\mathrm{init}\colon V \to M$ and*
- *an update function $\mathrm{upd}\colon M \times V \to M$.*

The size of $\mathcal{M}$, denoted by $|\mathcal{M}|$, is defined as $|M|$.

Unlike standard finite automata, a memory structure has no initial state, but an initialization function, i.e., the initial state depends on the first letter of the input. On the other hand, the update function is a classical transition function, i.e., it maps a state and an input letter to a successor state.

Also note that a memory structure is a deterministic device. Hence, one can define the unique state reached after processing an input as usual: we define $\mathrm{upd}^*\colon V^+ \to M$ inductively via $\mathrm{upd}^*(v) = \mathrm{init}(v)$ for $v \in V$ and $\mathrm{upd}^*(wv) = \mathrm{upd}(\mathrm{upd}^*(w), v)$.

Next, we define the output functions for memory structures. Intuitively, given a play prefix w, such a function takes the last letter v of w and the memory state $\mathrm{upd}^*(w)$ and maps them to a successor of v.

Definition 4.2 (Next-Move Function)**.** *Let $\mathcal{A} = (V, V_0, V_1, E)$ be an arena and $\mathcal{M} = (M, \mathrm{init}, \mathrm{upd})$ be a memory structure for $\mathcal{A}$. A* next-move function *for Player i is a mapping $\mathrm{nxt}\colon V_i \times M \to V$ satisfying $(v, \mathrm{nxt}(v, m)) \in E$ for all $v \in V_i$ and $m \in M$.*

As already explained, a memory structure and a next-move function induce a strategy.

Definition 4.3 (Finite-state Strategy)**.** *Let $\mathcal{A} = (V, V_0, V_1, E)$ be an arena, let $\mathcal{M} = (M, \mathrm{init}, \mathrm{upd})$ be a memory structure for $\mathcal{A}$, and let nxt be a next-move function for Player i that is compatible with $\mathcal{A}$ and $\mathcal{M}$. Then, $\mathcal{M}$ and nxt implement the strategy σ given by $\sigma(wv) = \mathrm{nxt}(v, \mathrm{upd}^*(wv))$ for all $w \in V^*$ and $v \in V_i$.*

A strategy σ is a finite-state *strategy, if it is implemented by some memory structure and some next-move function.*

If a strategy σ is implemented by some memory structure with n memory states, then we say that σ has size $|M|$. This is slightly abusive, since σ might also be implementable by smaller memory structures. However, we only use this notion, if the memory structure that implements σ is clear from context.

Finally, let us remark that we recover positional strategies as a special case of finite-state strategies.

Remark 4.1. *A strategy is positional if, and only if, it is implementable by a memory structure of size one.*

4.2 Reductions

After defining finite-state strategies in the previous subsection, we now consider the problem of computing such strategies. Obviously, one can construct such strategies by hand, e.g., as done in the example from the previous subsection. A more general approach is to use reductions between games. As usual in computer science, a reduction simplifies a problem. In the case of infinite games, this means simplifying the winning condition: given a game $\mathcal{G} = (\mathcal{A}, \mathrm{Win})$ one takes the product $\mathcal{A} \times \mathcal{M}$ of $\mathcal{A}$ with a suitable memory structure $\mathcal{M}$ to obtain a new game $\mathcal{G}' = (\mathcal{A} \times \mathcal{M}, \mathrm{Win}')$ where Player 0 is known to have a positional winning strategy. If the new winning condition Win' satisfies a certain reduction property, then a positional winning strategy for Player 0 in $\mathcal{G}'$ can effectively be turned into a finite-state winning strategy for Player 0 in the original game $\mathcal{G}$. Thus, reducing $\mathcal{G}$ to $\mathcal{G}'$ simplifies the winning condition (in the sense that positional strategies are sufficient to win $\mathcal{G}'$), but increases the size (by taking the product).

From such a reduction (and the solution of the reduced game $\mathcal{G}'$), we obtain the winning regions of the original game $\mathcal{G}$ as well as finite-state winning strategies for both players. However, note that we have to provide the memory structure for the reduction, i.e., we only obtain the next-move function from solving $\mathcal{G}'$.

We begin by introducing the product construction between an arena and a memory structure. Intuitively, the memory structure is used to keep track of the state $\mathrm{upd}^*(w)$ reached along a play prefix w.

Definition 4.4 (Product Arena)**.** *Let $\mathcal{A} = (V, V_0, V_1, E)$ be an arena and $\mathcal{M} = (M, \mathrm{init}, \mathrm{upd})$ be a memory structure for $\mathcal{A}$. The* product $\mathcal{A} \times \mathcal{M}$ *of $\mathcal{A}$ and $\mathcal{M}$ is defined as $\mathcal{A} \times \mathcal{M} = (V \times M, V_0 \times M, V_1 \times M, E')$ with $((v, m), (v', m')) \in E'$ if, and only if, $(v, v') \in E$ and $\mathrm{upd}(m, v') = m'$ for all $v, v' \in V$ and all $m, m' \in M$.*

Now, let $\rho = \rho_0\rho_1\rho_2 \cdots \in \mathrm{Plays}(\mathcal{A})$. We define the extended play $\mathrm{ext}(\rho) = (\rho_0, m_0)(\rho_1, m_1)(\rho_2, m_2) \cdots \in \mathrm{Plays}(\mathcal{A} \times \mathcal{M})$ *with $m_0 = \mathrm{init}(\rho_0)$ and $m_{n+1} = \mathrm{upd}(m_n, \rho_{n+1})$ for all $n \in \mathbb{N}$.*

Let ρ and $\mathrm{ext}(\rho)$ be as in the definition. A simple induction shows $m_n = \mathrm{upd}^*(\rho_0 \cdots \rho_n)$ for every n, i.e., the product does indeed keep track of memory states reached when processing the prefixes of ρ.

Now, we formalize the notation of (game) reductions. Most importantly, we need to relate the winning conditions in the original game in the arena $\mathcal{A}$ and in the reduced game in the product arena in order to obtain winning regions and winning strategies from solving the reduced game.

Definition 4.5 (Reduction)**.** *Let $\mathcal{G} = (\mathcal{A}, \mathrm{Win})$ and $\mathcal{G}' = (\mathcal{A}', \mathrm{Win}')$ be two games and let $\mathcal{M}$ be a memory structure for $\mathcal{A}$. We say* $\mathcal{G}$ is reducible to $\mathcal{G}'$ *via the memory structure $\mathcal{M}$, denoted by $\mathcal{G} \leq_{\mathcal{M}} \mathcal{G}'$,* *if $\mathcal{A}' = \mathcal{A} \times \mathcal{M}$ and if for every $\rho \in V^\omega$: ρ and $\mathrm{ext}(\rho)$ have the same winner, i.e., $\rho \in \mathrm{Win}$ if, and only if, $\mathrm{ext}(\rho) \in \mathrm{Win}'$.*

Next, we prove the reduction lemma, which formalizes the relation between the winning regions of the original and the reduced game. Here, we assume the case where the reduced game allows for positional winning strategies. A more general result is discussed in Exercise 4.1.

Lemma 4.1. *Let $\mathcal{G} \leq_\mathcal{M} \mathcal{G}'$ with $\mathcal{M} = (M, \mathrm{init}, \mathrm{upd})$. Also, let V' be a subset of the set of vertices of $\mathcal{G}$. If Player i has a positional winning strategy for $\mathcal{G}'$ from $\{(v, \mathrm{init}(v)) \mid v \in V'\}$, she also has a finite-state winning strategy with memory $\mathcal{M}$ for $\mathcal{G}$ from V'.*

Proof. Let $\mathcal{G} = (\mathcal{A}, \mathrm{Win})$ with $\mathcal{A} = (V, V_0, V_1, E)$. By Definition 4.5, we have $\mathcal{G}' = (\mathcal{A} \times \mathcal{M}, \mathrm{Win}')$ for some winning condition Win' and $\mathcal{A} \times \mathcal{M} = (V \times M, V_0 \times M, V_1 \times M, E')$. Furthermore, let σ' the positional winning strategy for Player i for $\mathcal{G}'$ from $\{(v, \mathrm{init}(v)) \mid v \in V'\}$. For $v \in V_i$ and $m \in M$ we define $\mathrm{nxt}(v, m) = v'$, if $\sigma'(v, m) = (v', m')$ for some $v' \in V$ and some $m' \in M$. Finally, let σ be the strategy implemented by $\mathcal{M}$ and nxt.

We have to show that σ is winning from every $v \in V'$. So let $v \in V'$ and let $\rho \in \mathrm{Plays}(\mathcal{A}, v, \sigma)$ be arbitrary. Furthermore, let $\rho' = \mathrm{ext}(\rho)$, which is a play of $\mathcal{G}'$ starting in $(v, \mathrm{init}(v))$. We first show that $\rho' \in \mathrm{Plays}(\mathcal{A} \times \mathcal{M}, (v, \mathrm{init}(v)), \sigma')$. To do so, we show by induction that every prefix $(\rho_0, m_0) \cdots (\rho_n, m_n)$ of ρ' is consistent with σ'.

As a play prefix of length one is consistent with any strategy, the induction start for $n = 0$ is trivial.

For the induction step with $n > 0$ consider a prefix $(\rho_0, m_0) \cdots (\rho_n, m_n)$. By induction hypothesis $(\rho_0, m_0)(\rho_1, m_1) \cdots (\rho_{n-1}, m_{n-1})$ is consistent with σ'. If it is not Player i's turn at (ρ_{n-1}, m_{n-1}), i.e.,$(\rho_{n-1}, m_{n-1}) \in V_{1-i} \times M$, then the full prefix is also consistent with σ'. Thus, assume we have $(\rho_{n-1}, m_{n-1}) \in V_i \times M$. Hence, $\rho_{n-1} \in V_i$ and thus

$$\rho_n = \sigma(\rho_0 \cdots \rho_{n-1}) = \mathrm{nxt}(\rho_{n-1}, \mathrm{upd}^*(\rho_0 \cdots \rho_{n-1})).$$

Thus, by definition of nxt, there is an $m \in M$ such that

$$\sigma'(\rho_{n-1}, \mathrm{upd}^*(\rho_0 \cdots \rho_{n-1})) = \sigma'(\rho_{n-1}, m_{n-1}) = (\rho_n, m).$$

It remains to show that $m = m_n$: We have $m_n = \mathrm{upd}(m_{n-1}, \rho_n)$ by definition of the extended play. Also, $\sigma'(\rho_{n-1}, m_{n-1}) = (\rho_n, m)$ implies $((\rho_{n-1}, m_{n-1}), (\rho_n, m)) \in E'$. Hence, the definition of E' yields $m = \mathrm{upd}(m_{n-1}, \rho_n)$. Thus, both m and m_n are equal to $\mathrm{upd}(m_{n-1}, \rho_n)$, i.e., $m = m_n$.

Thus, $\mathrm{ext}(\rho)$ is consistent with σ' and starts in $(v, \mathrm{init}(v))$ for every play ρ in $\mathcal{G}$ that is consistent with σ and starts in V'. As σ' is a winning strategy from $(v, \mathrm{init}(v))$, we have $\mathrm{ext}(\rho) \in \mathrm{Win}'$. Applying Definition 4.5 yields $\rho \in \mathrm{Win}$. Thus, σ is winning for Player 0 from every $v \in V'$. □

Typically, we reduce to games which are determined with uniform positional winning strategies. For this situation, we can formulate the following variant of the reduction lemma.

Corollary 4.1. *Let $\mathcal{G} \leq_\mathcal{M} \mathcal{G}'$ such that $\mathcal{G}'$ is determined with uniform positional winning strategies. Then, $\mathcal{G}$ is determined with uniform finite-state strategies implemented by $\mathcal{M}$ and the winning regions are given $W_i(\mathcal{G}) = \{v \in V \mid (v, \mathrm{init}(v)) \in W_i(\mathcal{G}')\}$ for $i \in \{0, 1\}$.*

To illustrate the usefulness of reductions, we show how to reduce weak Muller games to weak parity games, which are positionally determined (see Exercise 3.11). In a weak Muller game, the winner of a play only depends on the occurrence set of the play. More formally, there is a family of occurrence sets that are winning for Player 0, all others are winning for Player 1.

Definition 4.6 (Weak Muller Game)**.** *Let $\mathcal{A} = (V, V_0, V_1, E)$ be an arena and let $\mathcal{F} \subseteq 2^V$ be a family of subsets of $\mathcal{A}$'s vertices. Then, the weak Muller condition $\textsc{wMuller}(\mathcal{F})$ is defined as*

$$\textsc{wMuller}(\mathcal{F}) := \{\rho \in V^\omega \mid \mathrm{Occ}(\rho) \in \mathcal{F}\}.$$

We call a game $\mathcal{G} = (\mathcal{A}, \textsc{wMuller}(\mathcal{F}))$ a weak Muller game.

Recall that Player i wins a play in a weak parity game, if the parity of the minimal color occurring during the play is i. In the following, it is useful to consider the max-variant of the weak parity condition. Here, Player i wins a play, if the parity of the *maximal* color occurring during the play is i. We denote this winning condition by $\textsc{wMaxParity}(\Omega)$, i.e.,

$$\textsc{wMaxParity}(\Omega) := \{\rho \in V^\omega \mid \max \mathrm{Occ}(\Omega(\rho_0)\Omega(\rho_1)\Omega(\rho_2) \cdots) \text{ is even}\}.$$

As with parity games, it is straightforward to show that the max-variant is equivalent to the min-variant, i.e., every weak max-parity condition can be turned into a weak min-parity condition (with the same number of colors) and vice versa.

We now show how to reduce weak Muller games to weak max-parity games. The memory structure keeps track of the vertices visited along a play. The coloring of the weak max-parity game assigns even colors to vertices (v, S) where S, the set of already visited vertices, is in $\mathcal{F}$, and odd colors if F is not in $\mathcal{F}$. As the set of visited vertices increases until it gets stationary at some point, larger sets are assigned larger colors than smaller sets. This is the reason we use the weak max-parity condition: the occurrence set of the play is the largest set S that is encountered as a memory state, hence it has to have the deciding, i.e., the largest color. Using the colors $2 \cdots |S|$ respectively $2 \cdots |S| + 1$ ensures our requirements are satisfied.

Lemma 4.2. *Weak Muller games are reducible to weak max-parity games.*

Proof. Let $\mathcal{G} = (\mathcal{A}, \textsc{wMuller}(\mathcal{F}))$ be a weak Muller game with $\mathcal{A} = (V, V_0, V_1, E)$. We define the memory structure $\mathcal{M} = (M, \mathrm{init}, \mathrm{upd})$ with $M = 2^V \setminus \{\emptyset\}$, $\mathrm{init}(v) = \{v\}$ and $\mathrm{upd}(S, v) = S \cup \{v\}$ for all $v \in V$ and $S \in M$. A straightforward induction over w shows that $\mathrm{upd}^*(w) = \mathrm{Occ}(w)$ for all $w \in V^+$, i.e., $\mathcal{M}$ does indeed keep track of the occurrence set of a play prefix. In particular, upd^* is monotonic in the following sense: We have $\mathrm{upd}^*(w) \subseteq \mathrm{upd}^*(ww')$ for all $w, w' \in V^*$.

We show $\mathcal{G} = (\mathcal{A}, \textsc{wMuller}(\mathcal{F})) \leq_{\mathcal{M}} (\mathcal{A} \times \mathcal{M}, \textsc{wMaxParity}(\Omega)) = \mathcal{G}'$, where the coloring Ω is defined as

$$\Omega(v, S) = \begin{cases} 2 \cdot |S| & \text{if } S \in \mathcal{F}, \\ 2 \cdot |S| - 1 & \text{if } S \notin \mathcal{F}. \end{cases}$$

Now, it remains to show that a play in $\mathcal{G}$ and its extended play in $\mathcal{G}'$ have the same winner. Thus, let $\rho \in \mathrm{Plays}(\mathcal{A})$ be arbitrary. There is a position n of ρ such that

1. $\mathrm{Occ}(\rho_0 \cdots \rho_n) = \mathrm{Occ}(\rho)$, and
2. $\mathrm{Occ}(\rho_0 \cdots \rho_m) \subsetneq \mathrm{Occ}(\rho)$ for all $m < n$.

Now consider the extended play $\mathrm{ext}(\rho) = (\rho_0, S_0)(\rho_1, S_1)(\rho_2, S_2) \cdots \in \mathrm{Plays}(\mathcal{A} \times \mathcal{M})$. Then, the choice of n and the definition of Ω imply

1. $\Omega(\rho_m, S_m) = \Omega(\rho_n, S_n)$ for all $m > n$, and
2. $\Omega(\rho_m, S_m) < \Omega(\rho_n, S_n)$ for all $m < n$.

Intuitively, these two properties formalize that the monotonicity of upd^* is "preserved" bu the coloring Ω. Hence, the maximal color occurring in $\mathrm{ext}(\rho)$ is $\Omega(\rho_n, S_n)$, which is even if, and only if, $S_n = \mathrm{Occ}(\rho) \in \mathcal{F}$. Hence, $\rho \in \textsc{wMuller}(\mathcal{F})$ if, and only if, $\mathrm{ext}(\rho) \in \textsc{wMaxParity}(\Omega)$. □

Thus, an application of Corollary 4.1 yields the following corollary of Lemma 4.2.

Corollary 4.2. *Weak Muller games are determined with finite-state strategies of size 2^n, where n is the number of vertices, and can be solved in exponential time.*

The natural follow-up is whether the size of the strategies is optimal or whether smaller finite-state strategies exist in general. The next result rules this out. It presents a family of weak Muller games $\mathcal{G}_n$ of linear size in n where Player 0 wins from a designated initial vertex, but only with strategies of exponential size. Note that the size of a weak Muller game $(\mathcal{A}, \textsc{wMuller}(\mathcal{F}))$ is measured both in the size of $\mathcal{A}$ *and* in the size of $\mathcal{F}$, as $\mathcal{F}$ cannot necessarily be encoded in linear size in the size of $\mathcal{A}$.

If we measure the size in the game only in the size of the arena, then there is a very simple weak Muller game that has the same property. However, keeping $\mathcal{F}$ *small* as well complicates the situation and requires a clever trick due to Christof Löding.

Theorem 4.1. *There exists a family $\mathcal{G}_n = (\mathcal{A}_n, \textsc{wMuller}(\mathcal{F}_n))$ of weak Muller games, each having a designated vertex v, such that*

- *$|\mathcal{A}_n| \in \mathcal{O}(n)$ and $|\mathcal{F}_n| = 2$,*
- *Player 0 has a finite-state winning strategy from v, but*
- *Player 0 has no finite-state winning strategy from v with less than 2^n states.*

Proof. Consider the arena $\mathcal{A}_n$, depicted in Figure 4.3, with vertex set V_n. It contains n widgets, each containing the vertices s_j, h_j, u_j, and d_j for all $j \in \{1, \dots, n\}$. Additionally, the arena contains the vertices c_0, c_1, d, v_A and v_B. At every vertex s_j Player 0 has to decide to go to u_j, to d_j or to h_j. At every h_j, Player 1 has then to decide to move to u_j or d_j. Then, the play proceeds to the next gadget, or to c_0, if $j = n$.

We define $\mathcal{F}_n = \{V_n \setminus \{v_A\}, V_n \setminus \{v_B\}\}$, i.e.,Player 0 has to visit all vertices except for v_A or all vertices except for v_B. Notice that in case a play reaches the vertex c_1 twice, Player 1 can win by moving to v_A when at c_1 for the first time and to v_B the second time. Trivially, $\mathcal{A}_n$ and $\mathcal{F}_n$ satisfy the requirements formulated by the first item of the lemma.

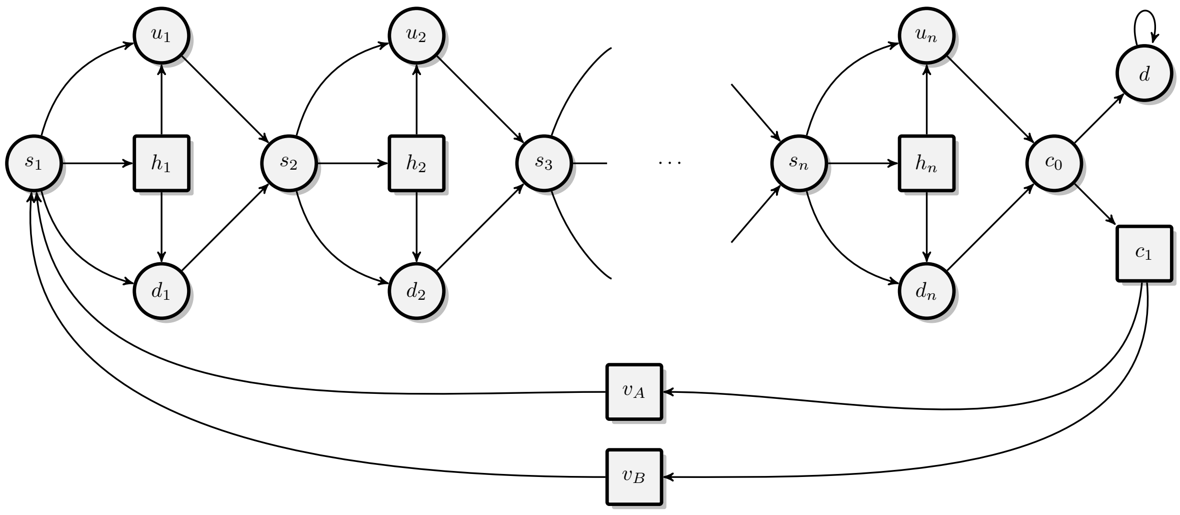


Figure 4.3: The arena $\mathcal{A}_n$ for the weak Muller game $\mathcal{G}_n$ of Theorem 4.1.

Now, consider the vertex $v = s_1$. First, we show that Player 0 has a winning strategy from this vertex. This strategy can be described as follows: When at some s_j for the first time she moves to h_j from where Player 1 can either go to u_j or to d_j in the next move. When at vertex c_0 for the first time, Player 0 moves to c_1 from where Player 1 either visits v_A or v_B and then moves to s_1 again. Now, when at some s_j for the second time, Player 0 moves to d_j if Player 1 moved to u_j from h_j and vice versa. This way every vertex in every widget is visited at least once. Finally, when at c_0 for the second time she moves to d and stays there forever. Correspondingly, the strategy ensures that every vertex except either v_A or v_B (but not both) is visited at least once.

Now, assume that Player 0 has a finite-state winning strategy σ from s_1 implementable with less than 2^n memory states. We start by showing that, when at vertex s_j for some $j \in \{1, \dots, n\}$ for the first time, the strategy prescribes a move to h_j. Assume this is not the case, i.e.,say it prescribes a move to u_j (the case of d_j is analogous). Then, after coming back in the second round, which has to be taken by Player 0 as otherwise c_1 is not visited, the strategy prescribes a move to either d_j or h_j or to u_j again. In case it moves to h_j, Player 1 answers by moving to u_j again. This way the vertex d_j is not visited yet. In case it does not move to h_j, h_j itself is not visited yet. Accordingly, Player 0 has to move to c_1 again, when at c_0. Thereby, she allows Player 1 to visit the missing vertex of the set $\{v_A, v_B\}$, not visited in the first round. At this point, Player 0 has lost the play since $\mathcal{F}_n$ contains no set containing both v_A and v_B.

As a consequence, Player 0 always has to move to h_j when at vertex s_j in the first round. From each h_j Player 1 has two choices. Hence, there are 2^n different play prefixes leading from s_1 to c_0. As there are less than 2^n memory states, at least two of them lead into the same memory state, i.e.,$\mathrm{upd}^*(w) = \mathrm{upd}^*(w')$ for two different play prefixes w and w' from s_1 to c_0. Thus, the memory structure implementing σ cannot distinguish w and w'. Hence, from c_0 onward, the strategy makes the same moves to extend the play prefixes w and w', i.e., we have $\sigma(wx) = \sigma(w'x)$ for all prolongations x. As argued before, in the second round, Player 0 has to move, when at s_j, to the vertex d_j or u_j not visited by Player 1 during the first round. Note that w and w' differ at least for one j^* in the choice made by Player 1 at h_{j^*}. Say, w.l.o.g., w visited u_{j^*} and w' visited d_{j^*}. Hence, if Player 1 moves from c_1 to v_A to extend w and w', then Player 0 continues both play prefixes the same way always moving from s_j to either d_j or u_j and then to the sink d. At s_{j^*} this choice does not complement the choice made by Player 1 during w or during w', respectively, i.e., if σ prescribes a move to u_{j^*} in the second round, then the continuation of w is losing for her. Dually, in case σ prescribes a move to d_{j^*} in the second round, then the continuation of w' is losing for her.

Hence, one of the corresponding plays will be losing contradicting the fact that σ is a winning strategy for Player 0 from s_1. □

4.3 Muller Games

In the previous subsection, we reduced weak Muller games to weak parity games to illustrate the usefulness of reductions. In this section, we consider the *strong* variant of the Muller condition. As with parity and weak parity games, it is obtained by considering the set of vertices visited infinitely often instead of the set of vertices visited at least once.

Definition 4.7 (Muller Game)**.** *Let $\mathcal{A} = (V, V_0, V_1, E)$ be an arena and let $\mathcal{F} \subseteq 2^V$ be a family of subsets of $\mathcal{A}$'s vertices. Then, the Muller condition $\textsc{Muller}(\mathcal{F})$ is defined as*

$$\textsc{Muller}(\mathcal{F}) := \{\rho \in V^\omega \mid \mathrm{Inf}(\rho) \in \mathcal{F}\}.$$

We call a game $\mathcal{G} = (\mathcal{A}, \textsc{Muller}(\mathcal{F}))$ a Muller game.

As an example, consider the family of Muller games[5] $\mathcal{DJW}_n = (\mathcal{A}_n, \textsc{Muller}(\mathcal{F}_n))$ with arena $\mathcal{A}_n = (V^n, V_0^n, V_1^n, E^n)$ where

- $V^n = V_0^n \cup V_1^n$,
- $V_0^n = \{v_1, \dots, v_n\}$,
- $V_1^n = \{v'_1, \dots, v'_n\}$,
- $E^n = V_0^n \times V_1^n \cup V_1^n \times V_0^n$, and
- $\mathcal{F}_n = \{F \subseteq V^n \mid |F \cap V_0^n| = \max\{j \mid v'_j \in F\}\}$.

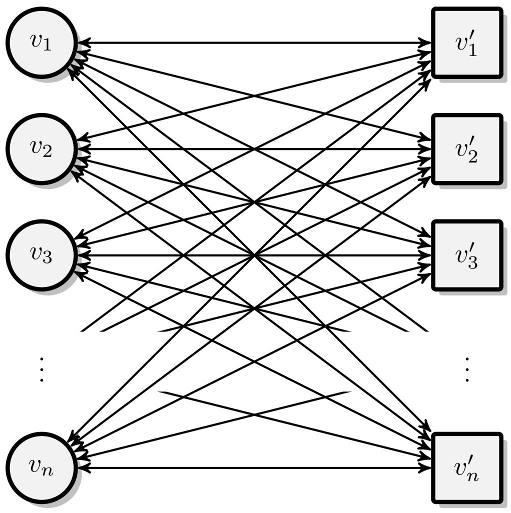


Figure 4.4: The arena for the Muller game $\mathcal{DJW}_n$.

The arena $\mathcal{A}_n$ is depicted in Figure 4.4: Player 0 can move from each of her vertices to every vertex of Player 1 and vice versa. The goal of Player 0 in a play ρ of $\mathcal{DJW}_n$ is to visit the vertex v'_j with $j = |\mathrm{Inf}(\rho) \cap V_0^n|$ infinitely often, but vertices $v'_{j'}$ with $j' > j$ only finitely often (visiting ones with smaller j' infinitely often has no influence). Note that the set $\mathrm{Inf}(\rho) \cap V_0^n$ is determined by Player 1 and depends on the whole play. During this play, Player 0 has to make sure that she visits v'_j again and again, but no larger vertex, without knowing what the cardinality of this set is going to be.

Which player wins the games $\mathcal{DJW}_n$? We claim that Player 0 has a uniform finite-state winning strategy from every vertex. The crucial point here is picking the right memory structure to implement a winning strategy: Player 0 has to *approximate* the cardinality j of the set $\mathrm{Inf}(\rho) \cap V_0^n$ from below, i.e., infinitely often moving to the vertex v'_j and from some point onwards only visiting vertices with smaller or equal index. This is possible by keeping track of the order in which the vertices in V_0^n were visited for the last time. Vertices visited only finitely often are in the limit the largest elements in this order while vertices visited infinitely often will be the smallest elements, as from some point onwards, the vertices seen infinitely often appear again and again while the other do not appear any more. Hence, the last occurrence of the vertices visited infinitely often is more recent than the last occurrence of a vertex only visited finitely often.

To illustrate this principle, consider the game $\mathcal{DJW}_4$ and the play prefix (restricted to vertices from V_0^4)

$$w = v_1\, v_3\, v_1\, v_2\, v_2\, v_1\, v_2\, v_4\, v_4\, v_4\, v_2.$$

The last vertex from V_0^4 that appears is v_2, the second-to-last one is v_4, and the third-to-last one is also v_4, which had a later occurrence, so it is ignored. The next vertices in this order are v_1 and then v_3. Hence, the memory state for this play encodes the list $v_2\, v_4\, v_1\, v_3$ (note that we reverse the order, i.e., the last vertex of the play is the first vertex in the list). Updating this list when visiting a vertex $v \in V_0^4$ is done by removing v from the list, if it appears, and putting it in front of the list again. This preserves

[5]These games where introduced by Stefan Dziembowski, Marcin Jurdziński, and Igor Walukiewicz.

the property that the list encodes the order in which the vertices were visited for the last time. There is no update when visiting vertices in V_1^4.

We need one additional piece of information to implement a winning strategy. We mark the position where v is deleted during an update by a special symbol $\sharp$, e.g., when visiting v_3 we update $v_1\sharp v_2 v_3 v_4$ to $v_3 v_1 v_2 \sharp v_4$. We call the vertices to the left of the $\sharp$ the *hit-set* of the list. The sequence of hit-sets occurring during a play approximates the infinity set from below in the following sense: During a play, the vertices from V_0^4 seen only finitely often accumulate at the end of the list while those seen infinitely often are in the front as they are the only ones that are moved to the front again and again. More formally, the following two properties are satisfied in every play: From some point onwards, the hit-set only contains vertices visited infinitely often and infinitely often the hit-set is equal to $\mathrm{Inf}(\rho) \cap V_0^4$. Hence, moving to the cardinality of the hit-set results in only finitely often seeing a vertex $v'_{j'}$ with $j' > j = |\mathrm{Inf}(\rho) \cap V_0^4|$ and infinitely often seeing the vertex v'_j. This strategy is winning for Player 0 from every vertex.

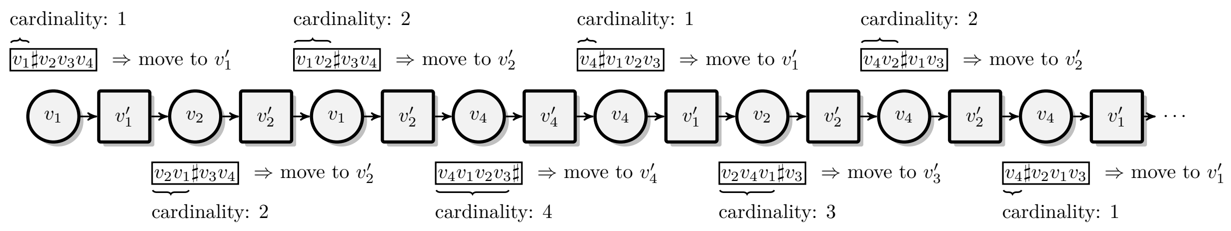

cardinality: 1
$\boxed{\overbrace{v_1}\sharp v_2 v_3 v_4}$ ⇒ move to v'_1
cardinality: 2
$\boxed{\overbrace{v_1 v_2}\sharp v_3 v_4}$ ⇒ move to v'_2
cardinality: 1
$\boxed{\overbrace{v_4}\sharp v_1 v_2 v_3}$ ⇒ move to v'_1
cardinality: 2
$\boxed{\overbrace{v_4 v_2}\sharp v_1 v_3}$ ⇒ move to v'_2
v_1
v'_1
v_2
v'_2
v_1
v'_2
v_4
v'_4
v_4
v'_1
v_2
v'_2
v_4
v'_2
v_4
v'_1
$\cdots$
$\boxed{\underbrace{v_2 v_1}\sharp v_3 v_4}$ ⇒ move to v'_2
cardinality: 2
$\boxed{\underbrace{v_4 v_1 v_2 v_3}\sharp}$ ⇒ move to v'_4
cardinality: 4
$\boxed{\underbrace{v_2 v_4 v_1}\sharp v_3}$ ⇒ move to v'_3
cardinality: 3
$\boxed{\underbrace{v_4}\sharp v_2 v_1 v_3}$ ⇒ move to v'_1
cardinality: 1

Figure 4.5: Example play consistent with the strategy for $\mathcal{DJW}_4$ described above. The current memory state is depicted in the boxes above respectively below the vertices of Player 0.

We illustrate the update of the memory and the choice of moves for Player 0 in the example play of $\mathcal{DJW}_4$ depicted in Figure 4.5. The memory states reached are depicted in the boxes above and below the vertices in V_0^4, where we start at some arbitrary initial list. Using this memory, Player 0 always moves to the vertex v'_j where j is the cardinality of the hit set of the current memory state. For example, her first move leads to v'_1, since only one vertex is in the hit set of $v_1\sharp v_2 v_3 v_4$. Her second move leads to v'_2 since there are two vertices in the hit set of $v_2 v_1 \sharp v_3 v_4$.

In the following we show that a generalization of the bookkeeping of the latest appearance of vertices suffices for general Muller games, not only for the games $\mathcal{DJW}_n$: The so-called "latest-appearance-record" (LAR) memory structure allows to reduce Muller games to parity games. In the general case, we keep a list of all vertices of the arena, instead of a list of a subset as in the case of the games $\mathcal{DJW}_n$, but the update is defined as above.

Definition 4.8 (Latest Appearance Record)**.** *Let V be a finite set and let $\sharp \notin V$ be some fresh symbol. A* latest appearance record LAR *over V is a word over the alphabet $V \cup \{\sharp\}$ where every letter from $V \cup \{\sharp\}$ appears exactly once and whose first letter is from V. The hit-set of an* LAR *$\ell = v_0 v_1 \cdots v_m \sharp v_{m+1} \cdots v_n$ is defined as $\mathrm{hit}(\ell) = \{v_0, \ldots, v_m\}$.*

We denote the set of all LAR's over V by LAR_V. Also, note that the hit-set is never empty. Now, we show how to use LAR's to reduce Muller games to parity games by generalizing the reasoning for the special case of the $\mathcal{DJW}_n$ games. For convenience, we use the max-parity condition.

Theorem 4.2. *Muller games are reducible to parity games.*

Proof. Let $\mathcal{G} = (\mathcal{A}, \textsc{Muller}(\mathcal{F}))$ be a Muller game with arena $\mathcal{A} = (V, V_0, V_1, E)$ and $\mathcal{F} \subseteq 2^V$. We define the memory structure $\mathcal{M} = (\mathrm{LAR}_V, \mathrm{init}, \mathrm{upd})$ where $\mathrm{init}\colon V \to \mathrm{LAR}_V$ is some arbitrary function and

$$\mathrm{upd}(v_0 \cdots v_m \sharp v_{m+1} \cdots v_n, v_j) = v_j v_0 v_1 \cdots v_{j-1} \sharp v_{j+1} \cdots v_n$$

for all $v_0 v_1 \cdots v_m \sharp v_{m+1} \cdots v_n \in LAR_V$, i.e., the vertex v_j we update with is moved to the front of the list and the $\sharp$ is moved to v_j's previous position. Before we define the coloring for the parity game in the arena $\mathcal{A} \times \mathcal{M}$, we first prove some general facts about the evolution of the memory state along a play.

Let $\rho = \rho_0 \rho_1 \rho_2 \cdots \in \mathrm{Plays}(\mathcal{A})$ be arbitrary and let $\ell = \ell_0 \ell_1 \ell_2 \cdots$ be the sequence of LAR's reached during ρ, i.e.,$\ell_n = \mathrm{upd}^*(\rho_0 \cdots \rho_n)$ for all $n \in \mathbb{N}$.

We begin by showing that the hit-set of the ℓ_n is from some position onwards always a subset of the infinity set of ρ, i.e., there is an n^* such that $\mathrm{hit}(\ell_n) \subseteq \mathrm{Inf}(\rho)$ for all $n > n^*$. First, let n_0 be a position such that $\mathrm{Occ}(\rho_{n_0} \rho_{n_0+1} \rho_{n_0+2} \cdots) = \mathrm{Inf}(\rho)$, i.e., the prefix up to position $n_0 - 1$ contains all occurrences of the vertices that appear only finitely often in ρ. Now, let $n^* > n_0$ be a position such that the infix $\rho_{n_0} \cdots \rho_{n^*-1}$ contains every vertex from $\mathrm{Inf}(\rho)$ at least once. We claim that this position has the desired property.

The infix $\rho_{n_0} \cdots \rho_{n^*-1}$ contains every vertex of $\mathrm{Inf}(\rho)$ at least once, but not other vertices, due to the choice of n_0. Hence, the first $|\mathrm{Inf}(\rho)|$ non-$\sharp$ entries of ℓ_{n^*-1} are exactly the vertices from $\mathrm{Inf}(\rho)$. Furthermore, the vertex ρ_{n^*} is also one of these vertices. Hence, also the first $|\mathrm{Inf}(\rho)|$ non-$\sharp$ entries of ℓ_{n^*} are exactly the vertices from $\mathrm{Inf}(\rho)$ and additionally the $\sharp$ is one of the first $|\mathrm{Inf}(\rho)| + 1$ entries. Hence, the hit-set of ℓ_{n^*} is a subset of $\mathrm{Inf}(\rho)$. The same reasoning can now be applied inductively to show the same for every ℓ_n with $n > n^*$.

Next, we show that there are infinitely many n with $\mathrm{hit}(\ell_n) = \mathrm{Inf}(\rho)$. Let $n > n^*$. It suffices to show that there is some $n' > n$ with $\mathrm{hit}(\ell_{n'}) = \mathrm{Inf}(\rho)$. As argued above, the first $|\mathrm{Inf}(\rho)|$ non-$\sharp$ entries of ℓ_n are the vertices from $\mathrm{Inf}(\rho)$. Let v be the last one of these in ℓ_n, i.e., the $|\mathrm{Inf}(\rho)|$-th non-$\sharp$ entry, and let $n' > n$ be the first position where v appears in ρ after position n. As v does not appear in $\rho_{n+1} \cdots \rho_{n'-1}$, it is also the $|\mathrm{Inf}(\rho)|$-th non-$\sharp$ entry of $\ell_{n'-1}$. Hence, every other vertex from $\mathrm{Inf}(\rho)$ is in front of it. Now, updating $\ell_{n'-1}$ to $\ell_{n'}$, which consists of moving v to the first position and moving the $\sharp$ to its position, results in the hit-set $\mathrm{hit}(\ell_{n'}) = \mathrm{Inf}(\rho)$. Thus, the position n' has the desired properties.

Relying on these two properties we define the coloring for the parity game in the arena $\mathcal{A} \times \mathcal{M}$: LAR's with a hit-set in $\mathcal{F}$ are assigned an even color while LAR's with a hit-set in the complement are assigned an odd color. Furthermore, as the sequence of hits-sets underapproximates the infinity set, larger hit-sets are more relevant than smaller ones. Thus, we assign more important colors to larger sets. Hence, it is natural to reduce to a max-parity game and relate the cardinality of the hit-set with the color we assign.

Hence, we define the coloring $\Omega \colon V \times \mathrm{LAR}_V \to \mathbb{N}$ as

$$\Omega(v, \ell) = \begin{cases} 2 \cdot |\mathrm{hit}(\ell)| & \text{if } \mathrm{hit}(\ell) \in \mathcal{F}, \\ 2 \cdot |\mathrm{hit}(\ell)| - 1 & \text{if } \mathrm{hit}(\ell) \notin \mathcal{F}. \end{cases}$$

This is well-defined, as the hit-set is always non-empty.

It remains to show $\rho \in \textsc{Muller}(\mathcal{F})$ if, and only if, $\mathrm{ext}(\rho) = (\rho_0, \ell_0)(\rho_1, \ell_1)(\rho_2, \ell_2) \cdots \in \textsc{MaxParity}(\Omega)$: We have

$$\begin{aligned} & \rho \in \textsc{Muller}(\mathcal{F}) \\ \Rightarrow\ & \mathrm{Inf}(\rho) \in \mathcal{F} \\ \Rightarrow\ & \exists F \in \mathcal{F}\, (\exists n^* \text{ s.t. } \mathrm{hit}(\ell_n) \subseteq F \text{ for all } n > n^*) \text{ and } (\mathrm{hit}(\ell_n) = F \text{ for infinitely many } n) \\ \Rightarrow\ & \exists F \in \mathcal{F}\, (\exists n^* \text{ s.t. } \Omega(\rho_n, \ell_n) \leq 2 \cdot |F| \text{ for all } n > n^*) \text{ and } (\Omega(\rho_n, \ell_n) = 2 \cdot |F| \text{ for infinitely many } n) \\ \Rightarrow\ & \exists F \in \mathcal{F} \max \mathrm{Inf}(\Omega(\rho_0, \ell_0)\Omega(\rho_1, \ell_1)\Omega(\rho_2, \ell_2) \cdots) = 2 \cdot |F| \\ \Rightarrow\ & \mathrm{ext}(\rho) \in \textsc{MaxParity}(\Omega). \end{aligned}$$

Dually, we have the exact same argument if ρ does not satisfy the Muller condition:

$$\begin{aligned} & \rho \notin \textsc{Muller}(\mathcal{F}) \\ \Rightarrow\ & \mathrm{Inf}(\rho) \notin \mathcal{F} \\ \Rightarrow\ & \exists F \in 2^V \setminus \mathcal{F}\, (\exists n^* \text{ s.t. } \mathrm{hit}(\ell_n) \subseteq F \text{ for all } n > n^*) \text{ and } (\mathrm{hit}(\ell_n) = F \text{ for infinitely many } n) \\ \Rightarrow\ & \exists F \in 2^V \setminus \mathcal{F}\, (\exists n^* \text{ s.t. } \Omega(\rho_n, \ell_n) \leq 2 \cdot |F| - 1 \text{ for all } n > n^*) \text{ and } (\Omega(\rho_n, \ell_n) = 2 \cdot |F| - 1 \text{ for inf. many } n) \\ \Rightarrow\ & \exists F \in 2^V \setminus \mathcal{F} \max \mathrm{Inf}(\Omega(\rho_0, \ell_0)\Omega(\rho_1, \ell_1)\Omega(\rho_2, \ell_2) \cdots) = 2 \cdot |F| - 1 \\ \Rightarrow\ & \mathrm{ext}(\rho) \notin \textsc{MaxParity}(\Omega). \end{aligned}$$

□

Applying Corollary 4.1 and positional determinacy of parity games yields finite-state determinacy of Muller games and a first algorithm for solving Muller games.

Theorem 4.3. *Every Muller game $\mathcal{G} = (\mathcal{A}, \textsc{Muller}(\mathcal{F}))$ with n vertices is determined with uniform finite state winning strategies of size $n \cdot n!$ and can be solved in exponential time.*

Proof. Finite-state determinacy follows directly from Corollary 4.1, while the size of the strategies stems from the fact that there are exactly $n \cdot n!$ LAR's in an arena with n vertices.

Furthermore, the corollary implies that $\mathcal{G}$ can be solved by solving a parity game of exponential size, namely $n^2 \cdot n!$, with $2n$ colors. This is possible, as the running time of the small progress measure algorithm is only exponential in the number of colors, i.e., it has an exponential running time. □

The upper bound $n \cdot n!$ on the memory requirements in a Muller game with n vertices is complemented by a matching lower bound exhibited by the family $\mathcal{DJW}_n$ presented above.

Theorem 4.4. *Player 0 wins the Muller game $\mathcal{DJW}_n$ from every vertex, but not with finite-state strategies with less than $n!$ states.*

Proof. See Exercise 4.11. □

The reduction to parity games yields an exponential-time algorithm for solving Muller games. However, this is not optimal. Indeed, depending on the representation of the family $\mathcal{F}$, the complxity of solving Muller games ranges from being in polynomial time to being complete for polynomial space.

Note that the issue of representing the winning condition does not appear for the games we considered in the previous section, e.g., reachability, Büchi, and parity, as their winning conditions are induced by a single subset of the vertices or by a coloring of the vertices. The size of these objects can be bounded by the size of the arena. However, in (weak) Muller games, the family $\mathcal{F}$ might be exponential in the size of the arena and can be represented compactly in different formalisms. We conclude this subsection by introducing several representations and the complexity of solving Muller games given in these representations.

Explicit representation: The simplest way to represent $\mathcal{F}$ is by giving it explicitly as a list of all subsets, e.g.,

$$\mathcal{F} = \{\{v_0\}, \{v_1\}, \{v_2\}, \{v_0, v_3\}, \{v_1, v_2\}, \{v_0, v_1, v_2\}, \{v_0, v_1, v_2, v_3\}\}.$$

Boolean formula: We can also encode the family $\mathcal{F}$ as a boolean formula over variables x_v for all vertices v. Then, F is in $\mathcal{F}$, if, and only if, the evaluation mapping exactly the x_v with $v \in F$ to true satisfies the formula. A formula representing the family $\mathcal{F}$ from above is given below:

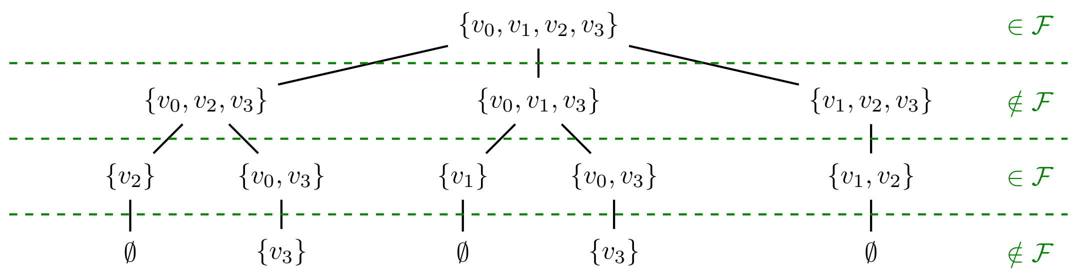
$$\begin{array}{lllllllllll}
(\overline{x_{v_0}} & \wedge & x_{v_1} & \wedge & \overline{x_{v_2}} & \wedge & \overline{x_{v_3}}) & \vee & & \rightarrow & \{v_1\} \\
(\overline{x_{v_0}} & \wedge & \overline{x_{v_1}} & \wedge & x_{v_2} & \wedge & \overline{x_{v_3}}) & \vee & & \rightarrow & \{v_2\} \\
(x_{v_0} & \wedge & \overline{x_{v_1}} & \wedge & \overline{x_{v_2}} & \wedge &) & \vee & & \rightarrow & \{v_0, v_3\}, \{v_0\} \\
(& & x_{v_1} & \wedge & x_{v_2} & \wedge & \overline{x_{v_3}}) & \vee & & \rightarrow & \{v_0, v_1, v_2\}, \{v_1, v_2\} \\
(x_{v_0} & \wedge & x_{v_1} & \wedge & x_{v_2} & \wedge &) & \vee & & \rightarrow & \{v_0, v_1, v_2, v_3\}, \{v_0, v_1, v_2\}
\end{array}$$

Boolean circuit: Every boolean formula can be represented as a circuit, which is possibly smaller, as it allows to eliminate shared subformulas.

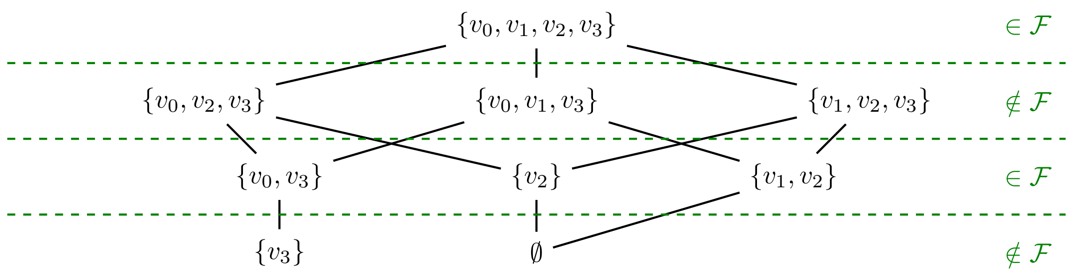
Zielonka tree: A family $\mathcal{F}$ can be represented as a tree, which is given by the following inductive definition:

- The root is labeled by the set of all vertices.
- Children of a node labeled with $F \in \mathcal{F}$ are the $\subseteq$-maximal subsets $F' \subseteq F$ with $F' \notin \mathcal{F}$.
- Children of a node labeled with $F \notin \mathcal{F}$ are the $\subseteq$-maximal subsets $F' \subseteq F$ with $F' \in \mathcal{F}$.

For example, our running example is encoded by the following Zielonka tree:



Directed Acyclic Graph: Again, the previous representation can be reduced by merging equal subtrees, i.e., we obtain a directed acyclic graph.



Coloring: We can color v by a coloring $\Omega\colon V \to \mathbb{N}$ and give an explicit representation of a family $\mathcal{F}_c \subseteq 2^{\Omega(V)}$ of sets of colors representing the family

$$\mathcal{F} = \{F \subseteq V\{\Omega\}(F) \in \mathcal{F}_c\}$$

of sets of vertices. If the number of colors is much smaller than the number of vertices, then this representation can be much smaller than an explicit list of $\mathcal{F}$ itself.

Important Subset: Here, a $W \subseteq V$ of important vertices is fixed and an explicit representation of a family $\mathcal{F}_i \subseteq 2^W$ of subsets of W represents the family

$$\mathcal{F} = \{F \subseteq V \mid F \cap W \in \mathcal{F}_i\}$$

of subsets of V. Again, if the set W is much smaller than V, then this representation can be much smaller than an explicit list of $\mathcal{F}$ itself.

Theorem 4.5.

1. *Solving Muller games is in P if $\mathcal{F}$ is given as an explicit list.*
2. *Solving Muller games is in NP $\cap$ Co-NP if $\mathcal{F}$ is given as a Zielonka tree.*
3. *Solving Muller games is PSPACE-complete if $\mathcal{F}$ is given in any of the other representations.*

4.4 Borel Determinacy and Limits on Reductions

In the previous subsection, we have presented several reductions, e.g., from Muller games to parity games. A natural question concerns the limits of such reductions, e.g., can Büchi games be reduced to reachability games? It turns out that this is impossible: A Büchi condition is harder than a reachability condition in a very formal way. On an intuitive level, this hardness can be phrased as follows: In a Büchi game, vertices from F have to be visited infinitely often while in a reachability game it suffices to visit R once, which is a much weaker condition. In particular, assume there is a reduction from Büchi games to reachability games and consider a play ρ satisfying the Büchi condition. Thus, the extended play has to satisfy the reachability condition, i.e., there is a prefix of length n that satisfies the reachability condition. Now, consider a play ρ' shares its prefix of length n with ρ, but does not satisfy the Büchi condition. The extended play $\mathrm{ext}(\rho')$ shares its prefix of length n with $\mathrm{ext}(\rho)$, hence it satisfies the reachability condition. This is a contradiction, as $\mathrm{ext}(\rho')$ has to violate the reachability condition, as ρ' violates the Büchi condition. The underlying problem here is that reductions are continuous, i.e., if ρ and ρ' share a prefix of length n, then so do $\mathrm{ext}(\rho)$ and $\mathrm{ext}(\rho')$.

In this subsection, we introduce tools to make such informal statements precise. In particular, we introduce the Borel hierarchy, which captures the complexity of languages (and therefore also winning conditions for infinite games) in terms of their topological complexity. Starting from very simple languages it consists of infinitely many levels of languages of increasing complexity.

Then, we show that reductions cannot go "down" the hierarchy: A complicated language cannot be reduced to a simpler one. To complement this impossibility result, we conclude by mentioning Martin's determinacy result, the most far-reaching determinacy result: every game with a winning condition in the Borel hierarchy is determined. This result subsumes all determinacy results proved thus far in these notes. However, it yields only arbitrary strategies, not necessarily positional or finite-state strategies where they exist.

We begin by introducing the Borel hierarchy, whose basic sets are so-called open sets, which can be understood as very general reachability conditions[6].

[6]But should not be confused with generalized reachability conditions as introduced in Exercise 3.4.

Definition 4.9 (Open Set). *Let V be a finite set. A set $L \subseteq V^\omega$ is* open *if it is of the form KV^ω for some $K \subseteq V^*$.*

Thus, as soon as $\rho \in V^\omega$ has a prefix in K, ρ itself is in L. However note, that K might be arbitrarily complicated, e.g., the set $P = \{a^p \mid p \in \mathbb{N} \text{ is prime}\}\{a, b\}^\omega$ is open[7]. As expected, reachability winning conditions are open, as we have $\textsc{Reach}(R) = (V^*R)V^\omega$.

Starting with the open sets, the Borel hierarchy is obtained by closing these sets under complementation and countable unions. The levels of the hierarchy correspond to the number of applications of countable unions.

Definition 4.10 (Borel Hierarchy). *Let V be a finite set. The* Borel hierarchy *(over V^ω) consists of levels $\mathbf{\Sigma}_n \subseteq 2^{V^\omega}$ and $\mathbf{\Pi}_n \subseteq 2^{V^\omega}$ for every $n > 0$, which are defined inductively as follows:*

- $\mathbf{\Sigma}_1 = \{L \subseteq V^\omega \mid L \textit{ is open}\}$,
- $\mathbf{\Pi}_n = \{L \subseteq V^\omega \mid V^\omega \setminus L \in \mathbf{\Sigma}_n\}$ *for every $n > 0$, and*
- $\mathbf{\Sigma}_{n+1} = \{L \subseteq V^\omega \mid L = \bigcup_{j\in\mathbb{N}} L_j \textit{ with } L_j \in \mathbf{\Pi}_n \textit{ for every } j\}$ *for every $n > 0$.*

Note that we require the L_j to be from $\mathbf{\Pi}_n$ when defining $\mathbf{\Sigma}_{n+1}$. This is not a restriction, as we prove that every *lower* level is a subset of $\mathbf{\Pi}_n$ (see Lemma 4.3). Also, the Borel hierarchy has levels $\mathbf{\Sigma}_\alpha$ and $\mathbf{\Pi}_\alpha$ for countably infinite ordinals α. We omit the formal definition, since all winning conditions we consider here are in the first three levels of the hierarchy, i.e., in $\mathbf{\Sigma}_n$ or $\mathbf{\Pi}_n$ for $n \leq 3$.

First, we show that the classes $\mathbf{\Sigma}_n$ and $\mathbf{\Pi}_n$ indeed form a hierarchy, as already mentioned above.

Lemma 4.3. *For every $n \in \mathbb{N}^+$, we have $\mathbf{\Sigma}_n \cup \mathbf{\Pi}_n \subseteq \mathbf{\Sigma}_{n+1} \cap \mathbf{\Pi}_{n+1}$.*

Proof. Note that we trivially have $\mathbf{\Pi}_n \subseteq \mathbf{\Sigma}_{n+1}$ for every n, as each $L \in \mathbf{\Pi}_n$ can be expressed as $L = \bigcup_{j\in\mathbb{N}} L_j$ with $L_j = L$ for every j. Hence, $L \in \mathbf{\Sigma}_{n+1}$. Applying this inclusion and duality, we also obtain

$$L \in \mathbf{\Sigma}_n \Rightarrow V^\omega \setminus L \in \mathbf{\Pi}_n \Rightarrow V^\omega \setminus L \in \mathbf{\Sigma}_{n+1} \Rightarrow \; L \in \mathbf{\Pi}_{n+1}.$$

To conclude the proof, we prove the implications $L \in \mathbf{\Sigma}_n \Rightarrow L \in \mathbf{\Sigma}_{n+1}$ and $L \in \mathbf{\Pi}_n \Rightarrow L \in \mathbf{\Pi}_{n+1}$ by induction over n.

For the induction start, let $L \in \mathbf{\Sigma}_1$ be arbitrary. We first show $L \in \mathbf{\Sigma}_2$, i.e., $L = \bigcup_{j\in\mathbb{N}} L_j$ with $L_j \in \mathbf{\Pi}_1$ for every j. Given some finite word $w = w_0 \cdots w_k \in V^*$ let

$$K_w = \{w_0 \cdots w_{k'} v \mid k' < k \text{ and } v \neq w_{k'+1}\}$$

be the set of shortest words that are not equal to a prefix of w. As L is in $\mathbf{\Sigma}_1$ and therefore open, $L = KV^\omega$ holds for some $K \subseteq V^*$. We show

$$L = \bigcup_{w\in K} V^\omega \setminus K_w V^\omega.$$

This implies $L \in \mathbf{\Sigma}_2$ as every $V^\omega \setminus K_w V^\omega$ is in $\mathbf{\Pi}_1$ (every $K_w V^\omega$ is open) and as K is countable.

We show the statement by proving $v_0v_1v_2 \cdots \in L$ if, and only if, $v_0v_1v_2 \cdots \in \bigcup_{w\in K} V^\omega \setminus K_w V^\omega$. For the direction from left to right, let $v_0v_1v_2 \cdots \in L$ be arbitrary. Since $L = KV^\omega$, there exists a prefix $w = v_0 \cdots v_n \in K$. If $w = \varepsilon$, then $K_w = \emptyset$ and $V^\omega \setminus K_w V^\omega = V^\omega$. As a consequence, $v_0v_1v_2 \cdots \in \bigcup_{w\in K} V^\omega \setminus K_w V^\omega$. Now, consider the case $w \neq \varepsilon$. Then, we have $v_0v_1v_2 \cdots \notin K_w V^\omega$ and therefore $v_0v_1v_2 \cdots \in V^\omega \setminus K_w V^\omega$. Thus, also $v_0v_1v_2 \cdots \in \bigcup_{w\in K} V^\omega \setminus K_w V^\omega$.

We show the other direction of the implication by contraposition. To this end, let $v_0v_1v_2 \cdots \notin L$ and $w \in K$ be arbitrary. Note that $w \neq \varepsilon$ since $\varepsilon \in K$ implies $L = KV^\omega = V^\omega$ which contradicts $v_0v_1v_2 \cdots \notin L$. Furthermore, $v_0v_1v_2 \cdots \in K_w V^\omega$, since w is not a prefix of $v_0v_1v_2 \cdots$. Thus, $v_0v_1v_2 \cdots \notin V^\omega \setminus K_w V^\omega$. As this reasoning holds for every $w \in K$ we conclude $v_0v_1v_2 \cdots \notin \bigcup_{w\in K} V^\omega \setminus K_w V^\omega$. Thus, as argued above, $L \in \mathbf{\Sigma}_1$ implies $L \in \mathbf{\Sigma}_2$ for all $L \subseteq V^\omega$.

It remains to consider $\mathbf{\Pi}_1$. The fact that $L \in \mathbf{\Pi}_1$ implies $L \in \mathbf{\Pi}_2$ directly follows from the result $L \in \mathbf{\Sigma}_1 \Rightarrow L \in \mathbf{\Sigma}_2$ just proven and from duality, i.e.,

$$L \in \mathbf{\Pi}_1 \Rightarrow V^\omega \setminus L \in \mathbf{\Sigma}_1 \Rightarrow V^\omega \setminus L \in \mathbf{\Sigma}_2 \Rightarrow L \in \mathbf{\Pi}_2.$$

[7]However, P is equal to $aa\{a, b\}^\omega$. A more interesting set is $\{a^p b^p \mid p \text{ is prime}\}\{a, b\}^\omega$.

Now, consider the induction step, i.e., $n > 1$. We have

$$L \in \mathbf{\Sigma}_n \Rightarrow L = \bigcup_{j \in \mathbb{N}} L_j \text{ with } L_j \in \mathbf{\Pi}_{n-1} \overset{IH}{\Rightarrow} L = \bigcup_{j \in \mathbb{N}} L_j \text{ with } L_j \in \mathbf{\Pi}_n \ \Rightarrow L \in \mathbf{\Sigma}_{n+1},$$

where we apply the induction hypothesis to every L_j with $j \in \mathbb{N}$. The proof for $\mathbf{\Pi}_n$ is similar to the proof for the induction start. We have

$$L \in \mathbf{\Pi}_n \Rightarrow V^\omega \setminus L \in \mathbf{\Sigma}_n \Rightarrow V^\omega \setminus L \in \mathbf{\Sigma}_{n+1} \Rightarrow L \in \mathbf{\Pi}_{n+1}$$

where we again apply the result just proven for $\mathbf{\Sigma}_n$. □

All winning conditions we have considered thus far can be placed in the first three levels of the Borel hierarchy.

	$\mathbf{\Sigma}_1$	$\mathbf{\Pi}_1$	$\mathbf{\Sigma}_2 \cap \mathbf{\Pi}_2$	$\mathbf{\Sigma}_2$	$\mathbf{\Pi}_2$	$\mathbf{\Sigma}_3 \cap \mathbf{\Pi}_3$
REACH(R)	×		×	×	×	×
SAFETY(S)		×	×	×	×	×
wPARITY(Ω)			×	×	×	×
wMULLER($\mathcal{F}$)			×	×	×	×
BÜCHI(F)					×	×
coBÜCHI(C)				×		×
PARITY(Ω)						×
MULLER($\mathcal{F}$)						×

Furthermore, for each type of winning condition appearing in the table above, there is a condition which is in the corresponding class, but not in a smaller class or in the complement-class[8]. For example, there is a parity condition PARITY(Ω) (which is also a Muller condition) with PARITY(Ω) $\notin \mathbf{\Sigma}_2 \cup \mathbf{\Pi}_2$. Similarly, there is a Büchi condition BÜCHI(F) with BÜCHI(F) $\notin \mathbf{\Sigma}_1 \cup \mathbf{\Pi}_1$ and BÜCHI(F) $\notin \mathbf{\Sigma}_2$ (see Exercise 4.14 for tools to prove such results).

Next, we study a generalized notion of reductions via continuous mappings between languages. We later prove that game reductions are a special case of them. Here, we define continuity of functions via preservation of open sets. An equivalent approach is to define a suitable metric on infinite words and then using the standard ε-δ-definition for functions over the reals known from analysis.

Definition 4.11 (Continuous Function). *A function $f\colon U^\omega \to V^\omega$ is called continuous if $f^{-1}(L)$ is open for every open set $L \subseteq V^\omega$.*

Now, a Wadge reduction is defined to be a continuous function that preserves membership in the languages under consideration.

Definition 4.12 (Wadge Reduction). *A set $L \subseteq U^\omega$ is Wadge-reducible to a set $L' \subseteq V^\omega$, denoted by $L \leq L'$, if, and only if, there exists a continuous function $f\colon U^\omega \to V^\omega$ such that $f^{-1}(L') = L$.*

This definition, while simple, is rather impractical to apply. Later, we characterize the existence of a reduction by a certain two-player game (cf. Theorem 4.6). Constructing a winning strategy in such a game, which implies the existence of a continuous function with the desired properties, is typically simpler than defining a function and then verifying that it is continuous and has the desired properties. Even more so, a winning strategy for the opponent witnesses that no reduction exists. Constructing such a strategy is again simpler than arguing that every possible function violates the requirements.

First, we mention a simple consequence of the definition of Wadge reductions.

Remark 4.2. *Let V and U be finite sets and $L \subseteq U^\omega$, $L' \subseteq V^\omega$. Then $L \leq L'$ implies $U^\omega \setminus L \leq V^\omega \setminus L'$.*

The following lemma proves restrictions on the existence of reductions. Intuitively, there are no reductions "down" the Borel hierarchy.

Lemma 4.4. *Let $L \subseteq U^\omega$ and $L' \subseteq V^\omega$ for two finite sets U and V such that $L \leq L'$. Then,*

1. *$L' \in \mathbf{\Sigma}_n \Rightarrow L \in \mathbf{\Sigma}_n$, and*
2. *$L' \in \mathbf{\Pi}_n \Rightarrow L \in \mathbf{\Pi}_n$.*

[8]The latter claim only applies to the winning conditions which are in $\mathbf{\Sigma}_n$ or $\mathbf{\Pi}_n$, but not in their intersection.

Proof. We prove both statements simultaneously by induction over $n > 0$.

For the induction start, let $L' \in \mathbf{\Sigma}_1$, i.e., L' is open. By $L \leq L'$ there exists a continuous function f with $f^{-1}(L') = L$. Thus, by continuity of f, L is also open and correspondingly $L \in \mathbf{\Sigma}_1$.

Now, let $L' \in \mathbf{\Pi}_1$. We have $V^\omega \setminus L' \in \mathbf{\Sigma}_1$ and therefore also $U^\omega \setminus L \in \mathbf{\Sigma}_1$ by the proof for $\mathbf{\Sigma}_1$ and Remark 4.2. Hence, $L \in \mathbf{\Pi}_1$.

Now, consider an $n > 1$ and let $L' \in \mathbf{\Sigma}_n$, i.e., $L' = \bigcup_{j \in \mathbb{N}} L'_j$ for some $L'_j \in \mathbf{\Pi}_{n-1}$. We define $L_j = f^{-1}(L'_j)$ for all $j \in \mathbb{N}$, where f is again the continuous function witnessing the reduction. Correspondingly, $L_j \leq L'_j$ for every $j \in \mathbb{N}$. It follows $L_j \in \mathbf{\Pi}_{n-1}$ for all $j \in \mathbb{N}$ by induction hypothesis. Also,

$$x \in L \Leftrightarrow f(x) \in L' \Leftrightarrow \; f(x) \in L'_j \text{ for some } j \;\Leftrightarrow\; x \in L_j \text{ for some } j \;.$$

Hence, $L = \bigcup_{j \in \mathbb{N}} L_j$ with $L_j \in \mathbf{\Pi}_{n-1}$ for all $j \in \mathbb{N}$, i.e., $L \in \mathbf{\Sigma}_n$.

Finally, consider $L' \in \mathbf{\Pi}_n$. Then, we have $V^\omega \setminus L' \in \mathbf{\Sigma}_n$ and $U^\omega \setminus L \leq V^\omega \setminus L'$ due to Remark 4.2. Thus, as already shown, $U^\omega \setminus L \in \mathbf{\Sigma}_n$ and therefore $L \in \mathbf{\Pi}_n$. ☐

There is an alternative characterization of Wadge reductions via games. A Wadge game $W(L, L')$ consists of two languages $L \subseteq U^\omega$ and $L' \subseteq V^\omega$ and is played in rounds by two players, called I and II. In each round n, I picks a letter $x_n \in U$ and then II picks a word $y_n \in V^*$. After ω many rounds, the pair (x, y) with $x = x_0x_1x_2\cdots$ and $y = y_0y_1y_2\cdots$ determines the winner. Player II wins if, and only if, y is infinite and the equivalence $x \in L \Leftrightarrow y \in L'$ is satisfied.

Here, a strategy for I is a mapping $\sigma\colon V^* \to U$ while a strategy for II is a mapping $\tau\colon U^+ \to V^*$. A play $\rho = x_0y_0x_1y_1\cdots$ is consistent with σ if $x_n = \sigma(y_0\cdots y_{n-1})$ for all $n \in \mathbb{N}$ and it is consistent with τ if $y_n = \tau(x_0\cdots x_n)$ for every $n \in \mathbb{N}$. A strategy is winning, if every play that is consistent with the strategy is winning. For a play $\rho = x_0y_0x_1y_1\cdots$ we call the pair (x, y) the outcome of the play.

As an example consider the game $W(0^*1(0{+}1)^\omega, (0^*1)^\omega)$ and note that the first condition is essentially a reachability one while the second one is essentially a Büchi condition: The first language contains the words having at least one occurrence of 1 while the second languages contains the word having infinitely many occurrences of 1. The following strategy is winning for II: If I has already played a 1, then pick a 1 too, and a 0 otherwise. Consider an outcome (x, y) that is consistent with this strategy. If x contains a 1, then y ends with the suffix 1^ω. Otherwise, if x contains no 1, then y is equal to 0^ω. Hence, we have $x \in 0^*1(0 + 1)^\omega$ if, and only if, $y \in (0^*1)^\omega$. Hence, this strategy is indeed winning for II.

In contrast, we claim that I wins the game $W((0^*1)^\omega, 0^*1(0 + 1)^\omega)$ where we have swapped the two languages. His strategy to do so is to pick 1, if II has not yet picked a 1, and 0 otherwise. Consider an outcome (x, y) that is consistent with this strategy. We only have to consider the case where y is infinite, as I wins in the other case by default. If y contains a 1, then x ends with 0^ω, i.e., $x \notin (0^*1)^\omega$, but $y \in 0^*1(0 + 1)^\omega$. On the other hand, if y does not contain a 1, i.e., $y = 0^\omega$, then x is equal to 1^ω. Thus, $x \in (0^*1)^\omega$, but $y \notin 0^*1(0 + 1)^\omega$. Thus, the strategy is indeed winning for I. Note that this winning strategy for I is a formalization of the intuitive reasoning presented above arguing that Büchi games cannot be reduced to reachability games in general.

We now show that Wadge games characterize Wadge reductions.

Theorem 4.6. *Let $L \subseteq U^\omega$ and $L' \subseteq V^\omega$ for two finite sets U and V. Then $L \leq L'$ if, and only if,* II *has a winning strategy for the game $W(L, L')$.*

Proof. First, let $L \leq L'$, i.e., there is a continuous function f such that $f^{-1}(L') = L$. We have to construct a winning strategy for II in $W(L, L')$. Assume we are in round n. Then I has picked letters $x_0, \ldots, x_n$ and II has picked possibly empty words $y_0, \ldots, y_{n-1}$ and has to pick y_n. By construction of our strategy, the set $\{x_0\cdots x_n\}U^\omega$ is partitioned into $\{f^{-1}(\{y_0\cdots y_{n-1}v\}V^\omega) \mid v \in V\}$. If there is a v such that $\{x_0\cdots x_n\}U^\omega \subseteq f^{-1}(\{y_0\cdots y_{n-1}v\}V^\omega)$, then II picks $y_n = v$, otherwise II picks $y_n = \varepsilon$. Note, that this choice preserves the partition property required above.

It remains to show that $y_0y_1y_2\cdots$ is an infinite word. If this is the case, then $y_0y_1y_2\cdots = f(x_0x_1x_2\cdots)$ and therefore $x_0x_1x_2\cdots \in L$ if, and only if, $y_0y_1y_2\cdots = f(x_0x_1x_2\cdots) \in L'$ due to $L = f^{-1}(L')$. Hence, the strategy is indeed winning.

Towards a contradiction, assume II picks $y_{n'} = \varepsilon$ for every $n' \geq n$ and some fixed $n \in \mathbb{N}$ as answer to $x = x_0x_1x_2\cdots$. Then, $\{x_0\cdots x_n\cdots x_{n+k}\}U^\omega$ is not a subset of $f^{-1}(\{y_0\cdots y_{n-1}v\}V^\omega)$ for every $v \in V$ and every $k \in \mathbb{N}$. As every $f(\{x_0\cdots x_n\cdots x_{n+k}\}U^\omega)$ is open and correspondingly also every $f^{-1}(\{y_0\cdots y_{n-1}v\}V^\omega)$ we have that

$$f^{-1}(\{y_0\cdots y_{n-1}v\}V^\omega) = K_vU^\omega$$

for some $K_v \subseteq U^*$. Now recall the invariant of our strategy. The set $\{x_0 \cdots x_n\}U^\omega$ is partitioned into the sets $K_v U^\omega$. Hence, there exists some prefix $x_0 \cdots x_m$ of x that is in K_v for some v. But then we also have $\{x_0 \cdots x_m\}U^\omega \subseteq K_v U^\omega$. This yields the desired contradiction.

Now, let τ be a winning strategy for II in $W(L, L')$. We define a function $f\colon U^\omega \to V^\omega$ as follows. Let $u_0 u_1 u_2 \cdots \in U^\omega$ and let the sequence $y = y_0 y_1 y_2 \cdots \in V^\omega$ be defined by $y_n = \tau(u_0 \cdots u_n)$ for all $n \in \mathbb{N}$. Then, as τ is a winning strategy, we have that y is infinite. Hence, we can define $f(u_0 u_1 u_2 \cdots) = y$. We obtain $f^{-1}(L') = L$ by the fact that τ is a winning strategy for II in $W(L, L')$. It remains to show that f is continuous. To do so, consider an open set KV^ω for some $K \subseteq V^\omega$. We have to show that $f^{-1}(KV^\omega)$ is open as well. To this end define

$$K' = \{u_0 \cdots u_n \in U^* \mid \tau(u_0)\tau(u_0 u_1) \cdots \tau(u_0 \cdots u_n) \in K\}.$$

We show $f^{-1}(KV^\omega) = K'U^\omega$. Let $x \in f^{-1}(KV^\omega)$, i.e., $f(x) \in KV^\omega$. Then there exists a prefix $u_0 \cdots u_n$ of x satisfying $\tau(u_0)\tau(u_0 u_1) \cdots \tau(u_0 \cdots u_n) \in K$. As a consequence, $u_0 \cdots u_n \in K'$ and therefore $x \in K'U^\omega$. Now, let $x \in K'U^\omega$. Then, there exists a prefix $u_0 \cdots u_n$ of x with $\tau(u_0)\tau(u_0 u_1) \cdots \tau(u_0 \cdots u_n) \in K$. Hence, $f(x) \in KV^\omega$. □

Finally, we can apply the general results we obtained so far to study game reductions as introduced previously. We show that every game reduction is a Wadge reduction. In particular, this implies that there are no game reductions “down” the Borel hierarchy, e.g., parity games cannot be reduced to safety games.[9]

Lemma 4.5. *Let $\mathcal{G} = (\mathcal{A}, \mathrm{Win})$ be reducible to $\mathcal{G}' = (\mathcal{A}', \mathrm{Win}')$. Then, $\mathrm{Win} \leq \mathrm{Win}'$.*

Proof. Let $\mathcal{M} = (M, \mathrm{init}, \mathrm{upd})$ be the memory structure witnessing the reduction. We show that II has a winning strategy for the game $W(\mathrm{Win}, \mathrm{Win}')$ which we define by $\tau(v_0 \cdots v_n) = (v_n, \mathrm{upd}^*(v_0 \cdots v_n))$. Accordingly, II constructs $\mathrm{ext}(\rho)$ as response to I picking a play $\rho \in V^\omega$. By definition of $\mathcal{G} \leq_{\mathcal{M}} \mathcal{G}'$, we have $\rho \in \mathrm{Win}$ if, and only if, $\mathrm{ext}(\rho) \in \mathrm{Win}'$. As a consequence, τ is indeed a winning strategy for II, which implies $\mathrm{Win} \leq \mathrm{Win}'$. □

We conclude this subsection by mentioning Martin’s determinacy theorem, which subsumes all determinacy results we proved so far. Note, however, that it does not impose any restriction on the type of winning strategies for both players. For example it does not imply that parity games are positionally determined, but only that they are determined. We say that a set L is Borel if it is contained in some level of the Borel hierarchy, even if the level has an ordinal index.

Theorem 4.7. *Every infinite game with Borel winning condition is determined.*

4.5 An Undetermined Game

Every game with Borel winning condition is determined, which, in particular, covers every determinacy result we have proven thus far. However, there are indeed games that are undetermined. In this subsection, we present one such game. Typically such games have a winning condition that allows the players to *steal* strategies in the following sense: Assuming σ is a winning strategy for Player i, Player $1-i$ has a strategy τ obtained by mimicking σ so that the play obtained from using τ against σ is winning for Player $1-i$. Thus, σ is not winning after all. Throughout this section, we fix the alphabet $\mathbb{B} = \{0, 1\}$.

The game we present here is based on an infinite XOR function mapping words in $\mathbb{B}^\omega$ to bits in $\mathbb{B}$. Such a function satisfies the following property: changing one bit in the input changes the output bit.

We first define the Hamming distance between two words $x, y \in \mathbb{B}^\omega$ as the number of positions at which these two words differ.

Definition 4.13 (Hamming Distance)**.** *Let $x = x_0 x_1 x_2 \cdots$ and $y = y_0 y_1 y_2 \cdots$ be in $\mathbb{B}^\omega$. The* Hamming distance *between x and y is defined as*

$$\mathrm{hd}(x, y) = |\{n \in \mathbb{N} \mid x_n \neq y_n\}| \in \mathbb{N} \cup \{\infty\}.$$

For example, we have

- $\mathrm{hd}(0101101000\cdots, 1010100000\cdots) = 5$,
- $\mathrm{hd}(1010101010\cdots, 0101010101\cdots) = \infty$, and

[9] However, see Exercise 4.15 for a relaxed notion of reduction that allows to do exactly that.

- $\mathrm{hd}(1010101010\cdots, 1111111111\cdots) = \infty$.

We define a relation $\sim$ over $\mathbb{B}^\omega$ via $x \sim y$ if $\mathrm{hd}(x,y) \neq \infty$, i.e., we relate those words having finite Hamming distance. An alternative characterization relies on the fact that having finite Hamming distance implies that x and y share a suffix that starts at the same position in both words.

Remark 4.3. *Let $x = x_0x_1x_2\cdots$ and $y = y_0y_1y_2\cdots$ in $\mathbb{B}^\omega$. Then, $x \sim y$ if, and only if, there is an n such that $x_{n'} = y_{n'}$ for all $n' > n$.*

Using this characterization, it is straightforward to show that $\sim$ is reflexive, symmetric, and transitive.

Remark 4.4. *$\sim$ is an equivalence relation.*

Definition 4.14 (Infinite XOR Function)**.** *A function $f\colon \mathbb{B}^\omega \to \mathbb{B}$ is an* infinite XOR function*, if $hd(x,y) = 1$ implies $f(x) \neq f(y)$ for all $x, y \in \mathbb{B}^\omega$.*

Theorem 4.8. *There exists an infinite XOR function.*

Proof. Let $S \subseteq \mathbb{B}^\omega$ be a set that contains exactly one element from each $\sim$-equivalence class, i.e., for every $x \in \mathbb{B}^\omega$, there is a $y \in S$ with $x \sim y$, but there are no two $y \neq y'$ in S with $y \sim y'$. Hence, for every $x \in \mathbb{B}^\omega$, let $r(x)$ denote the unique element in S with $x \sim r(x)$. Now, we define

$$f(x) = \mathrm{hd}(x, r(x)) \bmod 2$$

and show that f is an infinite XOR function. First note that $f(x)$ is indeed well-defined, since $\mathrm{hd}(x, r(x)) \in \mathbb{N}$ for all $x \in \mathbb{B}^\omega$.

Thus, let $x = x_0x_1x_2\cdots$ and $y = y_0y_1y_2\cdots$ be such that $\mathrm{hd}(x,y) = 1$, i.e., x and y differ at exactly one position. We assume w.l.o.g. that x has a 0 in this position while y has a 1: $x = w0w'$ and $y = w1w'$ for some $w \in \mathbb{B}^*$ and $w' \in \mathbb{B}^\omega$.

Furthermore, we have $x \sim y$ and therefore $r(x) = r(y)$. Call this element $r = r_0r_1r_2\cdots$ and define

$$D_x = \{n \mid x_n \neq r_n\} \qquad \text{and} \qquad D_y = \{n \mid y_n \neq r_n\}.$$

Due to x and y differing only at position $|w|$, we have

$$D_x \setminus \{|w|\} = D_y \setminus \{|w|\}$$

and exactly one of the sets contains $|w|$, say, w.l.o.g., $|w| \in D_y$. Thus, $|D_y| = |D_x| + 1$.

Then,

$$f(x) = \mathrm{hd}(x, r) \bmod 2 = |D_x| \bmod 2$$

and

$$f(y) = \mathrm{hd}(y, r) \bmod 2 = |D_y| \bmod 2 = (|D_x| + 1) \bmod 2.$$

Thus, one of the values $f(x)$ and $f(y)$ is equal to 0 and the other one is equal to 1, i.e., $f(x) \neq f(y)$. $\square$

The construction of the set S in the previous proof requires an application of the axiom of choice, which states that every family $(A_i)_{i \in I}$ of non-empty sets has a choice function, i.e., a mapping $c\colon I \to \bigcup_{i \in I} A_i$ with $c(i) \in A_i$ for every i. Here, I is an arbitrary index set. The intuitively true axiom of choice is equivalent to many less intuitive statements (e.g., Zorn's Lemma and the well-ordering principle) and has some paradoxical consequences (e.g., the Banach-Tarski paradox). Here, the study of games has very deep connections to set theory and the foundations of math.

We fix some infinite XOR function f for the remainder of this subsection and define a game $\mathcal{G}_f$ played in rounds $n = 0, 1, 2, \ldots$ between Player 0 and Player 1 as follows: in round n, first Player 0 picks $w_{2n} \in \mathbb{B}^+$, then Player 1 picks $w_{2n+1} \in \mathbb{B}^+$. The resulting play $w_0, w_1, w_2, \ldots$ is winning for Player $f(w_0w_1w_2\cdots)$.

A strategy σ for Player i in $\mathcal{G}_f$ is a mapping

$$\sigma\colon \bigcup_{n \in \mathbb{N}} (\mathbb{B}^+)^{2n+i} \to \mathbb{B}^+$$

with $(\mathbb{B}^+)^0 = \{\emptyset\}$ by convention. A play $w_0, w_1, w_2, \ldots$ is consistent with σ, if $w_{n+1} = \sigma(w_0, \ldots, w_n)$ for all n with $n \bmod 2 = i$. As usual, σ is a winning strategy for Player i, if Player i wins every play that is consistent with σ.

Note that we defined $\mathcal{G}_f$ as an abstract game without underlying arena. It is straightforward to construct an explicit variant $\mathcal{G}'_f = (\mathcal{A}, \mathrm{Win})$ of $\mathcal{G}_f$ in a finite arena $\mathcal{A}$ with some designated initial vertex v such that Player i has a winning strategy for $\mathcal{G}_f$ if, and only if, $v \in W_i(\mathcal{G}'_f)$ (see Exercise 4.16).

Theorem 4.9. *Let f be an infinite XOR function. No player has a winning strategy for $\mathcal{G}_f$.*

Proof. To simplify our notation, let $\rho(\sigma, \tau)$ be the unique play that is consistent with the strategy σ for Player 0 and the strategy τ for Player 1. Our proof is a strategy stealing argument: First, fix an arbitrary strategy τ for Player 1. From τ, we construct two strategies σ and σ' for Player 0 that mimic the behavior of τ. The difference between σ and σ' is that one of them starts by playing a 0, the other one by playing a 1. We show that one of the plays $\rho(\sigma, \tau)$ and $\rho(\sigma', \tau)$ is winning for Player 0, as they will only differ in their first position and hence, $\mathrm{hd}(\rho(\sigma, \tau), \rho(\sigma', \tau)) = 1$. This property implies that the infinite XOR function f maps one of these plays to 0, i.e., it is winning for Player 0. Thus, τ is not a winning strategy and as τ is picked arbitrarily, we conclude that Player 1 has no winning strategy. Then, we prove the dual statement for Player 0: for every strategy σ for Player 0, Player 1 has a counter strategy that witnesses that σ is not a winning strategy. Hence, both players do not have a winning strategy.

Thus, first let τ be a strategy for Player 1 in $\mathcal{G}_f$. We show that τ is not winning by defining two strategies σ and σ' for Player 0 as described above. To this end, recall that the input for the initial move of Player 0 is $\emptyset$ and define $\sigma(\emptyset) = 0$. Now, let $\tau(0) = w_1$. Then, we define $\sigma'(\emptyset) = 1w_1$, i.e., we mimic the move of Player 1 in response to Player 0 starting with the choice $w_0 = 0$, but replace the initial 0 of this play prefix by a 1. For all subsequent rounds, we define

$$\sigma(0, w_1, \ldots, w_{2n+1}) = \tau(1w_1, \ldots, w_{2n+1})$$

(note that $1w_1$ is a single word in the argument to τ) and we define

$$\sigma'(1w_1, w_2, \ldots, w_{2n}) = \tau(0, w_1, \ldots, w_{2n})$$

i.e., we continue mimicking τ. For all other input sequences, we define σ and σ' arbitrarily, as these are not relevant to our argument. The definition of the strategies is illustrated in Figure 4.6.

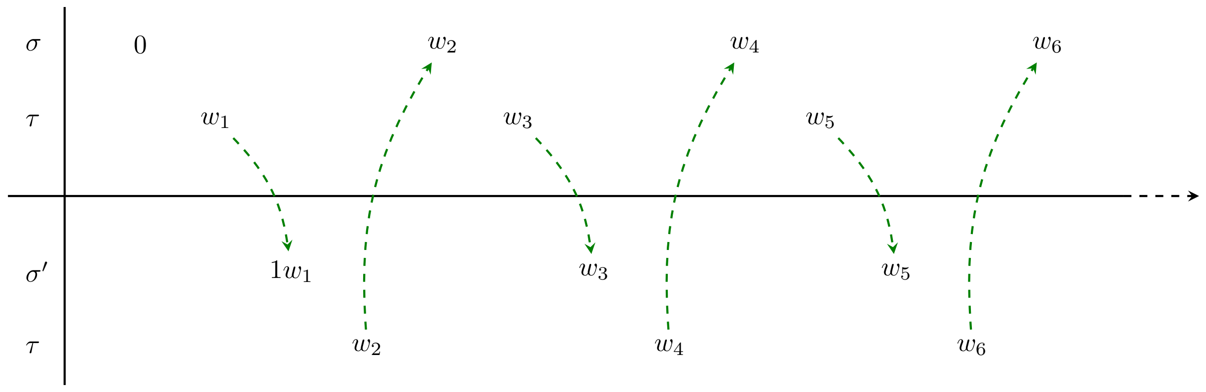


Figure 4.6: The definition of the strategies σ and σ' from τ. The resulting plays are $\rho(\sigma, \tau) = 0w_1w_2w_3\cdots$ and $\rho(\sigma', \tau) = 1w_1w_2w_3\cdots$.

Consider the plays $\rho(\sigma, \tau)$ and $\rho(\sigma', \tau)$. By construction, they differ only at their first position, where one has a 0 and the other one has a 1. Hence, f maps one of these plays to 0, i.e., Player 0 wins this play. Hence, τ is not a winning strategy, as a play that is consistent with τ is winning for Player 0.

The construction for Player 0 is analogous. Fix a strategy σ for her. Here, we define two strategies τ and τ' for Player 1. Let $w_0 = \sigma(\emptyset)$ and define $\tau(w_0) = 0$. Now, let $\sigma(w_0, 0) = w_1$ and define $\tau'(w_0) = 1w_1$.

For all subsequent rounds, we define

$$\tau(w_0, 0, w_1, \ldots, w_{2n+1}) = \sigma(w_0, 1w_1, \ldots, w_{2n+1})$$

and

$$\tau'(w_0, 1w_1, \ldots, w_{2n}) = \sigma(w_0, 0, w_1, \ldots, w_{2n}).$$

Again, for all other input sequences, we define τ and τ' arbitrarily, as these are not relevant to our argument. The definition of τ is illustrated in Figure 4.7.

Thus, the resulting plays $\rho(\sigma, \tau)$ and $\rho(\sigma, \tau')$ differ only in one position. Hence, one of them is mapped by f to 1, i.e., it witnesses that σ is not a winning strategy for Player 0. $\square$

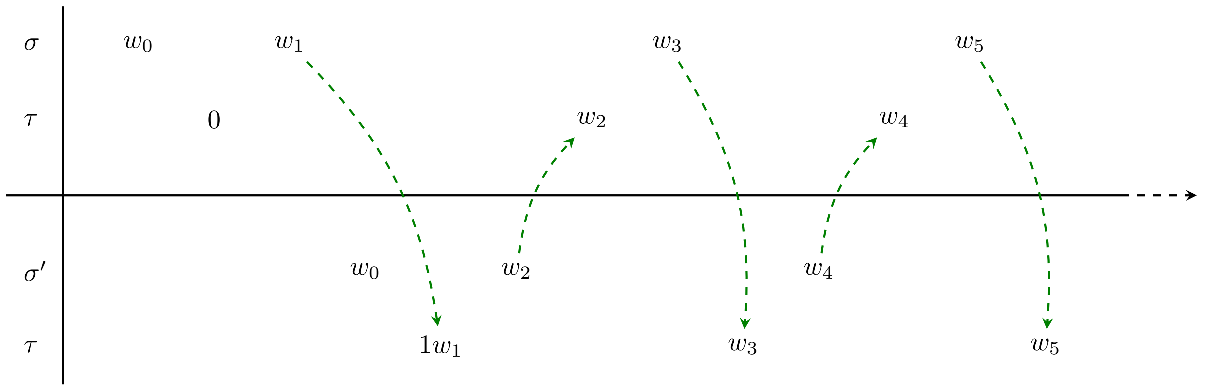


Figure 4.7: The definition of the strategies τ and τ' from σ. The resulting plays are $\rho(\sigma, \tau) = w_0 0 w_1 w_2 \cdots$ and $\rho(\sigma, \tau') = w_0 1 w_1 w_2 \cdots$.

4.6 Exercises

Exercise 4.1. Let $\mathcal{G} \leq_{\mathcal{M}} \mathcal{G}'$ with $\mathcal{M} = (M, \mathrm{init}, \mathrm{upd})$. Also, let V' be a subset of $\mathcal{G}$'s vertices. If Player i has a finite-state winning strategy for $\mathcal{G}'$ from $\{(v, \mathrm{init}(v)) \mid v \in V'\}$, she also has a finite-state winning strategy for $\mathcal{G}$ from V'.

Hint: Start by defining a suitable product memory structure $\mathcal{M} \times \mathcal{M}_f$.

Exercise 4.2.

1. Recall the definition of generalized reachability games presented in Exercise 3.4.

   Show that generalized games are reducible to reachability games.

2. Let $\mathcal{A} = (V, V_0, V_1, E)$ be an arena. Given a finite family $(Q_j, P_j)_{j=1,\ldots,k}$ of subsets $Q_j, P_j \subseteq V$, we define the request-response condition by

   $$\textsc{ReqRes}((Q_j, P_j)_{j=1,\ldots,k}) = \{\rho \in V^\omega \mid \text{for all } j = 1, \ldots, k \text{ and all } n \in \mathbb{N}\text{:}$$
   $$\rho_n \in Q_j \text{ implies } \rho_{n'} \in P_j \text{ for some } n' \geq n\}.$$

   Intuitively, a visit to Q_j is a request that has to be answered by a later response, i.e., a visit to P_j. A game $\mathcal{G} = (\mathcal{A}, \textsc{ReqRes}((Q_j, P_j)_{j=1,\ldots,k}))$ is a request-response game.

   Show that request-response games are reducible to Büchi games.

Exercise 4.3. Show that Player 0 needs exponential memory to win request-response games. To this end, construct a family $\mathcal{G}_n$ of request-response games of polynomial size in n, each with a designated vertex v, such that Player 0 wins $\mathcal{G}_n$ from v, but only with finite-state strategies of size 2^n.

Note: The size of a request-response game is measured in the number of vertices of the arena *and* in the number of request-response pairs (Q, P).

Exercise 4.4. Prove or disprove: If Player i has a finite-state winning strategy from each vertex $v \in W_i(\mathcal{G})$, in an arbitrary game $\mathcal{G}$, then Player i has a uniform finite-state winning strategy for $\mathcal{G}$.

Exercise 4.5. A (deterministic word) parity automaton $\mathscr{A} = (Q, \Sigma, q_I, \delta, \Omega)$ is a tuple consisting of

- a finite set Q of states,
- an alphabet Σ,
- an initial state $q_I \in Q$,
- a transition function $\delta \colon Q \times \Sigma \to Q$ and
- a coloring $\Omega \colon Q \to [k]$.

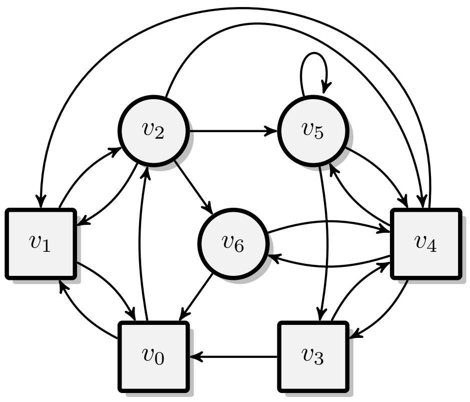
The run $r = r_0r_1r_2... \in Q^\omega$ of $\mathscr{A}$ on an infinite input word $\alpha \in \Sigma^\omega$ is defined by $r_0 = q_I$ and $r_{n+1} = \delta(r_n, \alpha_n)$ for all $n \in \mathbb{N}^+$. This run is accepting if $\min(\Omega(\inf(r)))$ is even. The language $\mathcal{L}(\mathscr{A})$ of $\mathscr{A}$ is the set of all input words whose run is accepting. A game $\mathcal{G} = (\mathcal{A}, \text{Win})$ is ω-regular if there exists a parity automaton $\mathscr{A}$ with $\mathcal{L}(\mathscr{A}) = \text{Win}$.

Prove the following statements:

1. Parity games are ω-regular.
2. Muller games are ω-regular.
3. The Büchi-Landweber Theorem: Every ω-regular game is determined with uniform finite-state winning strategies.

Exercise 4.6. Consider the Muller game $\mathcal{G}_1 = (\mathcal{A}_1, \textsc{Muller}(\mathcal{F}_1))$ with $\mathcal{A}_1$ as depicted below and

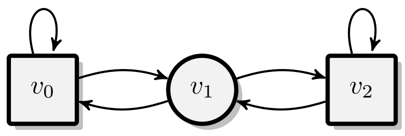
$$\mathcal{F}_1 = \{\{v_0, v_1\}, \{v_3, v_4\}, \{v_4, v_5\}, \{v_4, v_5, v_6\}, \{v_0, v_1, v_2, v_6\}\}.$$



Determine the winning regions of $\mathcal{G}_1$ and uniform finite-state winning strategies for both players. Specify the strategies by giving a memory structure (not necessarily the same for both players) and a next-move function.

Exercise 4.7. Consider the Muller game $\mathcal{G}_2 = (\mathcal{A}_2, \textsc{Muller}(\mathcal{F}_2))$ with $\mathcal{A}_2$ as depicted below and

$$\mathcal{F}_2 = \{\{v_0\}, \{v_2\}, \{v_0, v_1, v_2\}\}.$$



Apply the LAR reduction to determine the winning regions of $\mathcal{G}_2$, where constructing the vertices reachable from $\{(v, \text{init}(v)) \mid v \in \{v_0, v_1, v_2\}\}$ suffices.

Exercise 4.8. A family $\mathcal{F} \subseteq 2^V$ of sets is union-closed, if $F \cup F' \in \mathcal{F}$ for all $F, F' \in \mathcal{F}$. The family $\mathcal{F}$ is doubly union-closed, if $\mathcal{F}$ and $2^V \setminus \mathcal{F}$ are union-closed.

Show that doubly union-closed Muller conditions are equivalent to parity conditions, i.e.

1. for every doubly union-closed $\mathcal{F} \subseteq 2^V$ there exists a coloring $\Omega \colon V \to \mathbb{N}$ with $\textsc{Muller}(\mathcal{F}) = \textsc{Parity}(\Omega)$, and
2. for every coloring $\Omega \colon V \to \mathbb{N}$ there exists a doubly union-closed $\mathcal{F} \subseteq 2^V$ such that $\textsc{Parity}(\Omega) = \textsc{Muller}(\mathcal{F})$.

Exercise 4.9. Fix a Muller game $\mathcal{G} = (\mathcal{A}, \textsc{Muller}(\mathcal{F}))$ with vertex set V. Furthermore, let $W \subseteq V$ be such that for all $F, F' \subseteq V$: $F \cap W = F' \cap W$ implies $F \in \mathcal{F}$ if, and only if, $F' \in \mathcal{F}$, i.e., membership in $\mathcal{F}$ only depends on the vertices in W.

Show: Both players have uniform finite-state winning strategies for $\mathcal{G}$ using the LAR's over W as memory states.

Exercise 4.10 (Challenge)**.** Consider a request-response game $\mathcal{G} = (\mathcal{A}, \textsc{ReqRes}((Q_j, P_j)_{j=1,\ldots,k}))$ with $\mathcal{A} = (V, V_0, V_1, E)$ as defined in Exercise 4.2. We define the waiting time for condition j, denoted by $\mathrm{wt}_j \colon V^* \to \mathbb{N}$, inductively via $\mathrm{wt}_j(\varepsilon) = 0$ and

$$\mathrm{wt}_j(wv) = \begin{cases} 1 & \text{if } \mathrm{wt}_j(w) = 0 \text{ and } v \in Q_j \setminus P_j, \\ 0 & \text{if } \mathrm{wt}_j(w') = 0 \text{ and } v \notin Q_j \setminus P_j, \\ \mathrm{wt}_j(w) + 1 & \text{if } \mathrm{wt}_j(w) > 0 \text{ and } v \notin P_j, \\ 0 & \text{if } \mathrm{wt}_j(w) > 0 \text{ and } v \in P_j. \end{cases}$$

Intuitively, wt_j measures the waiting time between a request and its (earliest) response (ignoring additional requests of condition j while wt_j is already non-zero). The waiting times measure the quality of plays and strategies.

Give a family of request-response games $\mathcal{G}_k = (\mathcal{A}_k, \textsc{ReqRes}((Q_j, P_j)_{j=1,\ldots,k}))$ with $|\mathcal{A}_k| \in \mathcal{O}(k)$ and k pairs (Q_j, P_j) such that every $\mathcal{A}_k$ has a distinguished vertex v satisfying:

- Player 0 has a winning strategy from v, but
- every winning strategy for Player 0 from v has a play $\rho \in \mathrm{Plays}(\mathcal{A}_k, v, \sigma)$ satisfying $\mathrm{wt}_j(w) \geq 2^k$ for some prefix w of ρ.

Can you derive an upper bound on wt_j from your reduction in Exercise 4.2? Does this bound (asymptotically) match the lower bound 2^k?

Exercise 4.11 (Challenge)**.** Prove Theorem 4.4.

Hint: By induction over n using the fact that $\mathcal{DJW}_{n+1}$ contains $n+1$ different copies of $\mathcal{DJW}_n$.

Exercise 4.12. Show that $\mathbf{\Sigma}_n$ and $\mathbf{\Pi}_n$ are closed under union and intersection for every $n > 0$.

Exercise 4.13. Let V be a finite set. Prove each membership in the Borel hierarchy stated below.

1. $\textsc{wMuller}(\mathcal{F}) = \{\rho \in V^\omega \mid \mathrm{Occ}(\rho) \in \mathcal{F}\} \in \mathbf{\Sigma}_2 \cap \mathbf{\Pi}_2$ for every $\mathcal{F} \subseteq 2^V$.
2. $\textsc{coBüchi}(C) = \{\rho \in V^\omega \mid \mathrm{Inf}(\rho) \subseteq C\} \in \mathbf{\Sigma}_2$ for every $C \subseteq V$.
3. $\textsc{Parity}(\Omega) = \{\rho \in V^\omega \mid \min(\mathrm{Inf}(\Omega(\rho_0)\Omega(\rho_1)\Omega(\rho_2)\cdots)) \text{ is even}\} \in \mathbf{\Sigma}_3 \cap \mathbf{\Pi}_3$ for every $\Omega \colon V \to \mathbb{N}$.

Hint: Use the closure properties proven in Exercise 4.12.

Exercise 4.14. Fix $\mathbb{B} = \{0, 1\}$. A language $L \subseteq \mathbb{B}^\omega$ is *complete* for a level $\mathbf{\Sigma}_n$ of the Borel hierarchy over $\mathbb{B}$ if $L \in \mathbf{\Sigma}_n$ and $L' \leq L$ for every $L' \subseteq \mathbb{B}^\omega$ with $L' \in \mathbf{\Sigma}_n$. Completeness for $\mathbf{\Pi}_n$ is defined similarly.

1. Show that $0^*1(0+1)^\omega$ is complete for $\mathbf{\Sigma}_1$.
2. Show that $(0^*1)^\omega$ is complete for $\mathbf{\Pi}_2$.
3. Show that $(0^*1)^\omega$ is not in $\mathbf{\Sigma}_1 \cup \mathbf{\Pi}_1$.

Exercise 4.15 (Challenge)**.** Let $\mathcal{G} = (\mathcal{A}, \textsc{Parity}(\Omega))$ be a parity game. Construct a safety game $\mathcal{G}_s = (\mathcal{A} \times \mathcal{M}, \textsc{Safety}(S))$ for some memory structure $\mathcal{M} = (M, \mathrm{init}, \mathrm{upd})$ such that for all $v \in V$:

$$v \in W_0(\mathcal{G}) \Leftrightarrow (v, \mathrm{init}(v)) \in W_0(\mathcal{G}_s).$$

Hint: Revisit the small progress measure algorithm for parity games.

Exercise 4.16. Let $f \colon \mathbb{B}^\omega \to \mathbb{B}$. Construct an explicit variant $\mathcal{G}'_f = (\mathcal{A}, \mathrm{Win})$ of $\mathcal{G}_f$ in a *finite* arena $\mathcal{A}$ with some designated initial vertex v such that Player i has a winning strategy for $\mathcal{G}_f$ if, and only if $v \in W_i(\mathcal{G}'_f)$.

5 Infinite Games in Infinite Arenas

Until now, we only have considered games where the underlying arena is a finite object. In this chapter, we overcome this restriction and allow arenas to be countably infinite. Plays, strategies, winning regions, etc. are still defined as usual. However, there are some important differences between games in finite arenas and games in infinite arenas:

- The input to the solution algorithms is now an infinite object.
- Even positional strategies are infinite objects.
- The infinity set $\mathrm{Inf}(\rho)$ of a play ρ might be empty.
- Paths of bounded, finite length might not be sufficient for the attractor construction. For an example, see Figure 5.1. Here, the vertex v will never by added to the attractor after a fixed number of n iterations. Accordingly, we also have that $v \notin \bigcup_{n\in\mathbb{N}} \mathrm{Attr}_0^n(R) = \mathrm{Attr}_0(R)$. But Player 0 can force every play from v to v' as Player 1 has to choose one outgoing edge and then the number of steps after v' is reached is bounded.

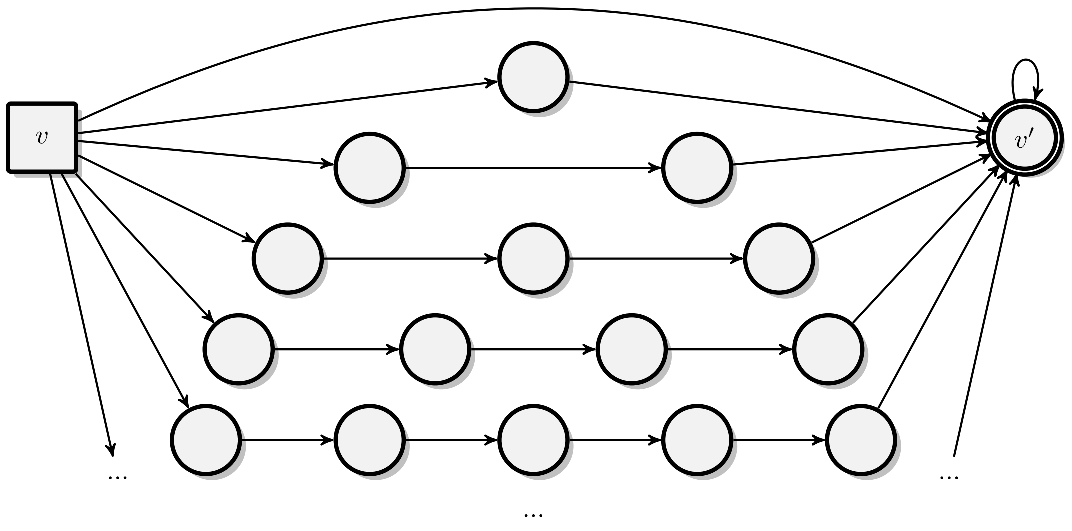


Figure 5.1: There is no $n \in \mathbb{N}$ such that v is in $\mathrm{Attr}_0^n(\{v'\})$.

Accordingly, to solve this game by an attractor construction we need more than infinitely many steps. Every vertex but v is eventually in some level of the attractor. Hence, doing "infinitely many and then one more" step adds v to the attractor as well.

More formally, using ordinals one can define higher levels of the attractor construction as follows:

$$\begin{aligned}
\mathrm{Attr}_0^{\omega}(R) &= \bigcup_{n\in\mathbb{N}} \mathrm{Attr}_0^n(R) \\
\mathrm{Attr}_0^{\omega+1}(R) &= \mathrm{Attr}_0^{\omega}(R) \cup \mathrm{CPre}_0(\mathrm{Attr}_0^{\omega}(R)) \\
\mathrm{Attr}_0^{\omega+2}(R) &= \mathrm{Attr}_0^{\omega+1}(R) \cup \mathrm{CPre}_0(\mathrm{Attr}_0^{\omega+1}(R)) \\
&\dots
\end{aligned}$$

We then have that $v \in \mathrm{Attr}_0^{\omega+1}(R)$. However, if v has further predecessors, we even need more iterations and possibly even infinitely many limit steps[10]. However, there is always a level at which the attractor gets stationary.

- Consider a parity game $\mathcal{G} = (\mathcal{A}, \textsc{Parity}(\Omega)$ with $\Omega \colon V \to \{0, 1, \dots, k\})$ for some $k \in \mathbb{N}$. As the image of Ω is finite, there is a color appearing infinitely often on every play, even if every vertex only appears once. For such games, one can still prove positional determinacy, using an induction

[10] $\mathrm{Attr}_0^{\omega}(R)$ is a limit step, as it is the union of all smaller attractors

on the number of colors and the adapted attractor construction. Note, that it is indeed crucial here that the image of Ω is finite, as otherwise the minimal color seen infinitely often may be undefined.

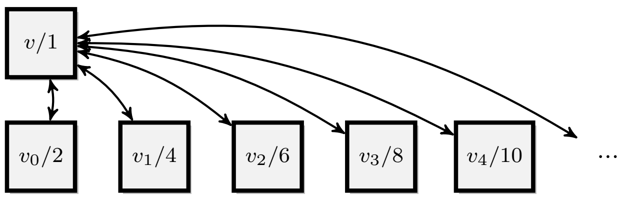


Figure 5.2: Max-parity game which cannot be translated into an equivalent min-parity game and where Player 1 needs a winning strategy of infinite size.

There is also a difference between min-parity and max-parity when we allow Ω to range over $\mathbb{N}$ and even infinite memory might be necessary. As an example consider the max-parity game depicted in Figure 5.2. Here, Player 1 has a winning strategy by visiting each of the lower vertices only once, which implies that only v is visited infinitely often. But with a finite-state strategy, he could only visit finitely many of the lower states such that the maximal color occurring infinitely often is even.

Also, we cannot represent this game as an equivalent min-parity game, as both players have positional winning strategies in min-parity games, even with infinitely many colors.[11]

5.1 Pushdown Parity Games

In the following, we want to consider a class of infinite games in infinite arenas that have a better behavior than the examples discussed so far: parity games that are played on the configuration graphs of pushdown machines. Such machines are similar to usual pushdown automata, but with discarded language acceptance features. These games are finitely represented by the underlying pushdown machines and therefore suitable for algorithmic analysis.

Definition 5.1 (Pushdown System). *A pushdown system $\mathcal{P} = (Q, \Gamma, \Delta, q_I)$ is a tuple consisting of*

- *a non-empty, finite set of states Q,*
- *a finite stack alphabet Γ,*
- *a transition relation $\Delta \subseteq Q \times \Gamma_\perp \times Q \times \Gamma_\perp^{\leq 2}$ and*
- *an initial state $q_I \in Q$,*

where $\Gamma_\perp$ denotes the extended stack alphabet $\Gamma_\perp = \Gamma \cup \{\perp\}$. We call $\perp$ the initial stack symbol, which we require to be fresh, i.e., $\perp \notin \Gamma$. Furthermore, a pushdown system $\mathcal{P}$ has to fulfil the following requirements:

- *The system is deadlock free: For all $q \in Q$ and all $A \in \Gamma_\perp$ there exists a $q' \in Q$ and an $\alpha \in \Gamma_\perp^{\leq 2}$ such that $(q, A, q', \alpha) \in \Delta$.*
- *The initial stack symbol is never deleted nor written on the stack: $(q, \perp, q', \varepsilon) \notin \Delta$ for all $q, q' \in Q$ and $(q, A, q', \perp X) \in \Delta$ implies $A = \perp$ and $X = \varepsilon$.*

We call a transition $(q, A, q', \alpha) \in \Delta$

- a pop transition if $|\alpha| = 0$,
- a skip transition if $|\alpha| = 1$, and
- a push transition if $|\alpha| = 2$.

A *stack content* of $\mathcal{P}$ is a word $\gamma \in \Gamma^*\{\perp\}$. A *configuration* of $\mathcal{P}$ is a tuple $(q, \gamma) \in Q \times \Gamma^*\{\perp\}$ consisting of a state and a stack content. The *stack height* of such a configuration is defined as by $|(q, \gamma)| = |\gamma| - 1$.

A transition from one configuration (q, γ) to another one (q, γ), is defined as $(q, \gamma) \rightarrow (q', \gamma')$ if, and only if, $(q, \gamma_0, q', \alpha) \in \Delta$ and $\gamma' = \alpha\gamma_1\gamma_2 \ldots \gamma_{|(q,\gamma)|}$. Now, we define the configuration graph of a pushdown system.

[11]For further details on this topic, see the paper “Positional determinacy of games with infinitely many priorities” by Grädel and Walukiewicz (LMCS, 2006).

Definition 5.2 (Configuration Graph)**.** *Let $\mathcal{P} = (Q, \Gamma, \Delta, q_I)$ be a pushdown system. The* configuration graph *of $\mathcal{P}$, denoted by $\mathscr{G}(\mathcal{P})$, is then defined as $\mathscr{G}(\mathcal{P}) = (V, E)$ where*

- *$V = Q \times \Gamma^*\{\bot\}$ is the set of configurations of $\mathcal{P}$ and*
- *$E = \{(v, v') \in V \times V \mid v \twoheadrightarrow v'\}$ is the set of transitions.*

The configuration graph has no terminal vertices, as $\mathcal{P}$ is deadlock free. To construct an arena, it remains to define the positions of the players and to color the vertices appropriately.

Definition 5.3 (Pushdown Parity Game)**.** *Let $\mathcal{P} = (Q, \Gamma, \Delta, q_I)$ be a pushdown system, let $\{Q_0 Q_1\}$ be a partition of Q, and let $\Omega' \colon Q \to \mathbb{N}$ be a coloring of Q. Then, we define the parity game $\mathcal{G}$ by $\mathcal{G} = (\mathcal{A}, \textsc{Parity}(\Omega))$ with $\mathcal{A} = (V, V_0, V_1, E)$ such that*

- *$(V, E) = \mathscr{G}(\mathcal{P})$,*
- *$V_i = \{(q, \gamma) \in V \mid q \in Q_i\}$ for $i \in \{0, 1\}$, and*
- *$\Omega((q, \gamma)) = \Omega'(q)$ for all $(q, \gamma) \in V$.*

The call such a game a pushdown parity game*.*

An example for pushdown parity game is depicted in Figure 5.3. The corresponding pushdown system is given by $\mathcal{P} = (\{q_I, q_1, q_2\}, \{A\}, \Delta, q_I)$ with Δ defined as the set

$$\{(q_I, X, q_I, AX), (q_I, X, q_1, AX), (q_1, A, q_1, \varepsilon), (q_1, \bot, q_2, \bot), (q_2, A, q_2, \varepsilon), (q_2, \bot, q_2, \bot) \mid X \in \{A, \bot\}\},$$

the partition $Q_1 = \{q_{in}\}$ and $Q_0 = \{q_1, q_2\}$, and the coloring Ω with $\Omega(q) = 0$ if $q \neq q_1$, and $\Omega(q_1) = 1$.

In this game, Player 1 can either increase the stack forever or move to some vertex v'_j where it is Player 0's turn. If he increases the stack forever, the minimal color seen infinitely often is 0. This play is winning for Player 0. However, if Player 0 is allowed to move, she simply can empty the stack again and move to q_2. As removing all elements from the stack only needs finitely many steps and q_2 has an even color, odd colors occur only finitely often. Accordingly, Player 0 wins this game from every vertex.

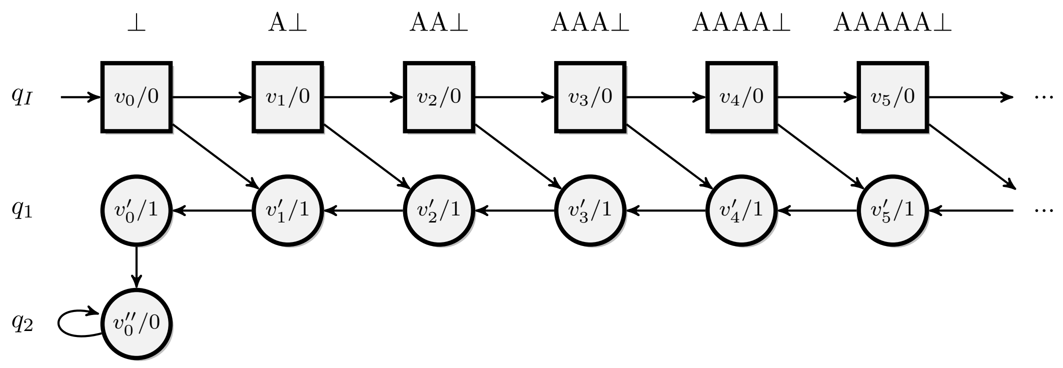


Figure 5.3: An example for a pushdown game.

We now are interested in solving such games with infinite arenas. In general we want to tackle the following problem: Given a pushdown game $\mathcal{G}$, represented by a pushdown system $\mathcal{P}$, a partition of its states and a coloring Ω of its states, and some $i \in \{0, 1\}$, determine whether Player i wins $\mathcal{G}$ from $(q_I, \bot)$ and, if yes, compute a corresponding winning strategy for her.

5.2 Pushdown Transducer

Recall that we claimed that parity games with finitely many colors are positionally determined, even if the arena is countably infinite. This applies to pushdown parity games, as the set of configurations is countable. However, even a positional winning strategy in a pushdown game is an infinite object, as there are infinitely many vertices.

In the following we introduce a finitely-representable machine model that implements winning strategies for pushdown games. Finite-state strategies does not suffice[12], but, not surprisingly, using pushdown machines suffices.

In the following, we define pushdown automata with output, so-called pushdown transducers.

Definition 5.4 (Pushdown Transducers)**.** *A pushdown transducer $\mathcal{T} = (Q, \Gamma, \Delta, q_I, \Sigma_I, \Sigma_O, \lambda)$ is a tuple consisting of*

- *a non-empty, finite set of states Q,*
- *a finite stack alphabet Γ,*
- *a transition relation $\Delta \subseteq Q \times \Gamma_\perp \times (\Sigma_I \cup \{\varepsilon\}) \times Q \times \Gamma_\perp^{\leq 2}$,*
- *an initial state q_I,*
- *a finite input alphabet Σ_I,*
- *a finite output alphabet Σ_O and*
- *a partial output function $\lambda\colon Q \rightharpoonup \Sigma_O$.*

where $\Gamma_\perp$ is defined similar as for pushdown systems and $\varepsilon \notin \Sigma_I$.

We say that $\mathcal{T}$ is *deterministic* if, and only if, for all $q \in Q$, all $A \in \Gamma_\perp$, and all $a \in \Sigma_I$ we have

$$|\{(q', \alpha) \mid (q, A, a, q', \alpha) \in \Delta \text{ or } (q, \varepsilon, a, q', \alpha) \in \Delta\}| \leq 1.$$

Informally, in a deterministic, there is always at most one transition applicable.

A *run r of $\mathcal{T}$* on a finite input word $w \in \Sigma_I^*$ is defined as

$$(q_0, \gamma_0)(q_1, \gamma_1)...(q_m, \gamma_m) \in (Q \times \Gamma^*\{\perp\})^*$$

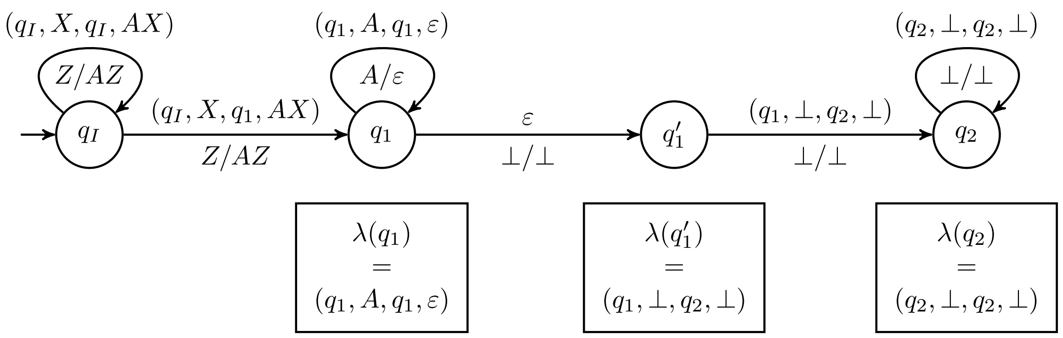
with $(q_0, \gamma_0) = (q_{in}, \perp)$ and with $(q_j, \gamma_j) \overset{a_j}{\rightarrow} (q_{j+1}, \gamma_{j+1})$ for all $j \in [m]$ and $a_j \in \Sigma_I \cup \{\varepsilon\}$ such that $a_0 a_1 ... a_{m-1} = w$ and $\{(q, \alpha) \mid (q_m, \gamma_0, \varepsilon, q, \alpha) \in \Delta\} = \emptyset$. We use $(q, \gamma) \overset{a}{\rightarrow} (q', \gamma')$ to denote that $(q, \gamma_0, a, q', x) \in \Delta$ with $\gamma' = \alpha\gamma(1)...\gamma(|\gamma| - 1)$. The last condition enforces that after reading w no more ε-transitions are possible. If $\mathcal{T}$ is deterministic, the corresponding run of w is unique and we denote it by $\mathrm{run}(w)$.

Definition 5.5 (Function Computed by a Transducer)**.** *Let $\mathcal{T} = (Q, \Gamma, \Delta, q_I, \Sigma_I, \Sigma_O, \lambda)$ be a deterministic pushdown transducer. Then $\mathcal{T}$* computes *the partial function $f_\mathcal{T}\colon (\Sigma_I)^* \rightharpoonup \Sigma_O$ with $f_\mathcal{T}(w) = \lambda(\mathrm{pr}_Q(\mathrm{Lst}(\mathrm{run}(w))))$ for all $w \in \Sigma_I^*$, if such a run exists. Here, pr_Q denotes the projection of a configuration to its state.*

To implement pushdown strategies in a pushdown game we will use pushdown transducers. To have a finite input alphabet, we represent play prefixes here by sequences of transitions and not by sequences of configurations. Notice, that both representations can easily be converted into each other. Furthermore, the output will be the next transition to be chosen by Player i instead of the next configuration. Hence, we use the set of transitions of the pushdown system defining the pushdown game for both, the input and the output alphabet of the pushdown transducer. Accordingly, we have $\Sigma_I = \Sigma_O = \Delta$, where Δ is the transition relation of the pushdown system underlying the game arena. This way, the transducer consumes a play prefix in the pushdown graph represented by a sequence of transitions and outputs the transition which Player i should choose next (in case the last configuration of the play prefix is a Player i configuration). Thus, we have to require the output transition to be executable from the last configuration of the play prefix induced by the input sequence.

As an example, consider the pushdown transducer depicted in Figure 5.4, which implements the winning strategy for Player 0 in the game $\mathcal{G}$ depicted in Figure 5.3.

[12]Even more problematic is that the update and the next-move function have infinite domain, as there are infinitely many vertices.



(q_I, X, q_I, AX)
Z/AZ
(q_I, X, q_1, AX)
Z/AZ
$(q_1, A, q_1, \varepsilon)$
A/ε
ε
$\perp/\perp$
$(q_1, \perp, q_2, \perp)$
$\perp/\perp$
$(q_2, \perp, q_2, \perp)$
$\perp/\perp$
q_I
q_1
q_1'
q_2
$\lambda(q_1) = (q_1, A, q_1, \varepsilon)$
$\lambda(q_1') = (q_1, \perp, q_2, \perp)$
$\lambda(q_2) = (q_2, \perp, q_2, \perp)$

Figure 5.4: Pushdown transducer implementing a winning strategy for Player 0 from $(q_I, \perp)$ in the game depicted in Figure 5.3.

5.3 Walukiewicz's Reduction

In this section, we show how to simulate a pushdown parity game by a parity game in a finite arena. This entails that we cannot store the whole information about the stack content of a configuration. Instead, we only store the topmost stack symbol, which allows us to deal with push and skip transitions, but not with pop transitions. Such transitions remove the topmost stack symbol, revealing the symbol below it, which we discarded. Hence, we change the evolution of a play: the players are assigned different tasks, one of them makes predictions and the other one verifies them.

A prediction $P = (P_0, ..., P_{k-1})$ contains for every $c \in \{0, 1, \dots, k-1\}$ a subset $P_c \subseteq Q$ of states, where $k-1$ is the largest color appearing in the game. Whenever a push-transition is to be simulated the predicting player has to make a prediction P about the future round t when the same stack height as before performing the push-transition is reached again for the first time (if it is reached at all). With this prediction, the predicting player claims that if the current push-transition is performed, then in round t some state $q \in P_c$ will be reached if $c \in \{0, 1, \dots, k-1\}$ is the minimal color seen in between. Once a prediction P is proposed, the verifying player has two ways of reacting, either believing that P is correct or not. In the first case, he is not interested in verifying P, so the push-transition is not performed and the verifying player chooses a color $c \in \{0, 1, \dots, k-1\}$ and a state $q \in P_c$, for some $P_c \neq \emptyset$, and skips a part of the simulated play by jumping to an appropriate position in the play. In the other case, he wants to verify the correctness of P, so the push-transition is performed and when the top of the stack is eventually popped it will turn out whether P is correct or not. The predicting player wins if P turns out to be correct and otherwise the verifying player wins. So after a pop-transition the winner is certain. For the other case, where no pop-transition is performed at all, the parity condition determines the winner.

In the following, let Player 0 take the role of the predicting player and Player 1 the role of the verifying one. Let $\mathcal{G} = (\mathcal{A}, \textsc{Parity}(\Omega))$ be a pushdown game with arena $\mathcal{A} = (V, V_0, V_1, E)$ induced by $\mathcal{P} = (Q, \Gamma, \Delta, q_I)$ with partition $Q_0 \cup Q_1$ of Q and coloring $\Omega \colon Q \to \{0, 1, \dots, k-1\}$. To simulate $\mathcal{G}$ by a game on a finite arena the information stored on the stack is encoded by some finite memory structure. The essential component of this structure is the set of predictions $\mathsf{Pred} = (2^Q)^k$.

We define the game $\mathcal{G}' = (\mathcal{A}', \textsc{Parity}(\Omega'))$ with $\mathcal{A}' = (V', V_0', V_1', E')$ as follows: for all states $q \in Q$, stack symbols $A, B \in \Gamma_\perp$, colors $c, d \in \{0, 1, \dots, k-1\}$ and predictions $P, R \in \mathsf{Pred}$, the set V' contains the vertices

- $\mathsf{Check}[q, A, P, c, d]$ (which correspond to the configurations of $\mathcal{G}$),
- $\mathsf{Push}[P, c, q, AB]$ (to signalize the intention of performing a push-transition),
- $\mathsf{Claim}[P, c, q, AB, R]$ (to make a new prediction),
- $\mathsf{Jump}[q, A, P, c, d]$ (to skip a part of a simulated play), and
- $\mathsf{Win}_0[q]$ and $\mathsf{Win}_1[q]$ (sink vertices which are reached when a prediction is verified).

The set E' consists of the following edges (for the sake of readability, we denote an edge $(v_1, v_2) \in E'$ here by $v_1 \to v_2$). For every skip-transition $\delta = (q, A, p, B) \in \Delta$ there are edges

$$\mathsf{Check}[q, A, P, c, d] \to \mathsf{Check}[p, B, P, \min\{c, \Omega(p)\}, \Omega(p)]\,,$$

for $P \in \mathsf{Pred}$ and $c, d \in \{0, 1, \dots, k-1\}$. Thus, the first two components of the Check-vertices are updated according to δ, the prediction P remains untouched, the last but one component is used to keep track of the minimal color for being able to check the prediction for correctness and the last component determines the color of the current Check-vertex. For every push-transition $\delta = (q, A, p, BC) \in \Delta$ there are edges

$$\mathsf{Check}[q, A, P, c, d] \to \mathsf{Push}[P, c, p, BC]\,,$$

for all $P \in \mathsf{Pred}$ and $c, d \in \{0, 1, \dots, k-1\}$. Here, a player states that a push-transition is to be performed such that the current state q has to be changed to p and the top of the stack A has to be replaced by BC. The information containing the current prediction P and the minimal color c is carried over, as this is needed in the case where the verifying player decides to skip. Moreover, to make a new prediction R, all edges

$$\mathsf{Push}[P, c, p, BC] \to \mathsf{Claim}[P, c, p, BC, R]$$

for every $R \in \mathsf{Pred}$ are needed. In case a new prediction is to be verified, a push-transition is finally performed using edges of the form

$$\mathsf{Claim}[P, c, p, BC, R] \to \mathsf{Check}[p, B, R, \Omega(p), \Omega(p)]$$

where the prediction P, the color c and the lower stack symbol C are discarded, since they are no longer needed. For the other case, where the verifying player intends to skip a part of a play, all edges

$$\mathsf{Claim}[P, c, p, BC, R] \to \mathsf{Jump}[q', C, P, c, e]$$

with $q' \in R_e$ are contained in E'. Here, the verifying player chooses a color $e \in \{0, 1, \dots, k-1\}$ for the minimal color of the skipped part and a state q from the corresponding component R_e of the prediction R. Now, the lower stack symbol C, the prediction P and the color c additionally have to be carried over, whereas B and R are discarded. Then, all edges

$$\mathsf{Jump}[q, C, P, c, e] \to \mathsf{Check}[q, C, P, \min\{c, e, \Omega(q)\}, \min\{e, \Omega(q)\}]$$

are contained in E' where the last component of the Check-vertex is set to be the minimum of the color of the current state q and the minimal color of the part just skipped. For the last but one component, we also have to account for the color c, which is necessary for eventually checking P for correctness. Finally, we have for every pop-transition $(q, A, p, \varepsilon) \in \Delta$, the edges

$$\begin{aligned} \mathsf{Check}[q, A, P, c, d] &\to \mathsf{Win}_0[p] \quad \text{if } p \in P_c\,, \text{ and} \\ \mathsf{Check}[q, A, P, c, d] &\to \mathsf{Win}_1[p] \quad \text{if } p \notin P_c\,, \end{aligned}$$

for $P \in \mathsf{Pred}$ and $c, d \in \{0, 1, \dots, k-1\}$, which lead to the sink vertex $\mathsf{Win}_0[p]$ of the predicting Player 0 if the prediction P turns out to be correct or to the sink vertex $\mathsf{Win}_1[p]$ of the verifying Player 1 otherwise. Moreover, we have $(\mathsf{Win}_i[q], \mathsf{Win}_i[q]) \in E'$, for $i \in \{0, 1\}$ and $q \in Q$.

The set of vertices V_0 of Player 0 (who in addition has to make the predictions) is defined to consist of all Push-vertices, as there new predictions are made, and of those $\mathsf{Check}[p, A, P, c, d]$ vertices where $p \in Q_i$. Accordingly, all other vertices belong to Player 1., especially all Claim-vertices belong to Player 1, as he accepts a prediction or challenges it at these vertices. Similarly, all Jump-vertices belong to Player 1, since he can pick a configuration from the current prediction to continue the play at this configuration.

The coloring $\Omega' \colon V' \to \{0, 1, \dots, k-1\}$ is defined by $\Omega'(\mathsf{Check}[p, A, P, c, d]) = d$ and $\Omega'(\mathsf{Win}_i[q]) = i$, for $i \in \{0, 1\}$. All other vertices are colored by the maximal color k (which does not appear in $\mathcal{G}$), since they are auxiliary vertices and should have no influence on the minimal color seen infinitely often. Note that there is no play in $\mathcal{G}'$ with minimal color k, as every play visits infinitely many Check-vertices or ends up in one of the sinks, which all have a smaller color.

The following lemma claims that solving $\mathcal{G}'$ suffices to determine the winner of $\mathcal{G}$. Furthermore, in the proof we will construct a pushdown transducer implementing a winning strategy for Player 0, if she is the winner. The proof shows that $\mathcal{G}'$ simulates $\mathcal{G}$. To this end, we need to specify the vertex of $\mathcal{G}'$ where the simulation starts. To this end, we define the initial prediction $P^{\text{in}} = \emptyset^k$, coinciding with the fact that the stack bottom symbol cannot be deleted from the stack.

Lemma 5.1. *Let $\mathcal{G}$ be a pushdown parity game and $\mathcal{G}'$ the parity game constructed as above. We have $(q_I, \bot) \in W_i(\mathcal{G})$ if and only if $\mathsf{Check}[q_I, \bot, P^{\text{in}}, \Omega(q_I), \Omega(q_I)] \in W_i(\mathcal{G})$ for $i \in \{0, 1\}$.*

Proof. We show that $(q_I, \bot) \in W_0(\mathcal{G})$ implies $\mathsf{Check}[q_I, \bot, P^{\text{in}}, \Omega(q_I), \Omega(q_I)] \in W_0(\mathcal{G})$ and vice versa. This proves the claim for $i = 1$, too, as both games are determined.

In the following, we only consider plays and play prefixes beginning in $(q_I, \bot)$ in $\mathcal{A}$ respectively in $\mathsf{Check}[q_I, \bot, P^{\text{in}}, \Omega(q_I), \Omega(q_I)]$ in $\mathcal{A}'$. For the sake of readability, when we refer to plays, we will always mean plays starting in one of these vertices.

First, let σ be a positional winning strategy for Player 0 from $(q_I, \bot)$ in $\mathcal{G}$. We need to construct a winning strategy σ' for Player 0 from $\mathsf{Check}[q_I, \bot, P^{\text{in}}, \Omega(q_I), \Omega(q_I)]$ in $\mathcal{G}'$. We define σ' and a function f mapping finite play prefixes of $\mathcal{G}'$ ending in a Check vertex to play prefixes in $\mathcal{A}$ simultaneously by induction over the length of the play prefixes. This is necessary, as f depends on choices made by σ' while σ is defined using f, which simulates play prefixes in $\mathcal{G}'$ by play prefixes in $\mathcal{G}$. Then, σ' mimics σ when applied to the simulated play prefix. Both functions are subjects to some invariants. First, f maps a play prefix w in $\mathcal{G}'$ ending in $\mathsf{Check}[q, A, P, c, d]$ to $f(w)$ ending in $(q, A\gamma)$ for some γ. Secondly, we define $f(w)$ and σ' only for those w that are consistent with σ' as defined thus far. This is sufficient as all other values of σ can be defined arbitrarily, since we only want to define a winning strategy from $\mathsf{Check}[q_I, \bot, P^{\text{in}}, \Omega(q_I), \Omega(q_I)]$.

We begin by defining $\sigma'(w)$ for those w ending in a Check vertex, say w ends in $\mathsf{Check}[q, A, P, c, d]$, as the definition is the same for the induction start and the induction step. Here, we apply the invariant: $f(w)$ ends in a configuration of the form $(q, A\gamma)$. If it is Player 0's turn at the last vertex of w, then also at the last vertex of $f(w)$. Hence, let $\sigma(f(w)) = (p, B\gamma')$ and let δ be the transition inducing the edge from $(q, A\gamma)$ to $(p, B\gamma')$. If δ is a skip-transition, then $\delta = (q, A, p, B)$. Then, we define σ' to mimic σ by defining

$$\sigma'(w) = \mathsf{Check}[p, B, P, \min\{c, \Omega(p)\}, \Omega(p)] \ .$$

If δ is a push-transition, then we again want to mimic σ. However, Player 0 cannot execute the push-transition, she can only signal her intent and make a prediction, and then let Player 1 decide whether he actually simulates the push-transition or whether he wants to skip a portion of the play. So, we define

$$\sigma'(w) = \mathsf{Push}[P, c, p, BC] \ .$$

Finally, if δ is a pop-transition, we move to the unique successor of the form $\mathsf{Win}_i[p]$ for some $i \in \{0, 1\}$. Note that this might be a losing sink for Player 0 (i.e., the vertex $\mathsf{Win}_1[p]$), if $p \notin P_c$. We will see later that this is never the case.

Now, consider a play prefix w' of the form $w\mathsf{Push}[P, c, p, BC]$ so that $f(w)$ is already defined. If $\mathsf{Check}[q, A, P, c, d]$ is the last vertex of w, then the last vertex of $f(w)$ is of the form $(q, A\gamma)$ (here, we again apply the invariant). As there is an edge from $\mathsf{Check}[q, A, P, c, d]$ to $w\mathsf{Push}[P, c, p, BC]$, $\delta = (q, A, p, BC)$ is a transition of $\mathcal{P}$, which is executable at the last vertex of $f(w)$ leading to the configuration $(p, BC\gamma)$. Consider the set R_c all those $(q^*, C) \in Q \times \Gamma_\bot$ such that there is a finite prolongation $f(w)(p, BC\gamma)(q_1, \gamma_1)...(q_n, \gamma_n)$ of $f(w)(p, BC\gamma)$ that are consistent with σ and satisfy the following requirements: $q^* = q_n$ and C is the topmost symbol of γ_n), $|q_n, \gamma_n| = |q, A\gamma|$, $|q_{n'}, \gamma_{n'}| > |q, A\gamma|$, and $c = \min(\{\Omega(p) \cup \{\Omega(q_{n'}) \mid 1 \leq n' < n\}\})$. Hence, R_c contains all those states of configurations that are reachable from $(q, A\gamma)$ via a play prefix that is consistent with σ after first taking the push-transition δ (replacing the topmost A by BC) and then reaching the same stack height as $(q, A\gamma)$ for the first time, while the minimal color seen after $(q, A\gamma)$ is c. Note that this implies that the topmost stack symbol of such a configuration is C. We collect these set in the prediction $R = (R_0, ..., R_{k-1})$ and define $\sigma'(w') = \mathsf{Claim}[P, c, p, BC, R]$. This completes the definition of σ'.

It remains to define f. For the induction start, consider a play prefix of length one. As all our plays start in the initial Check vertex $\mathsf{Check}[q_I, P^{\text{in}}, \Omega(q_I), \Omega(q_I)]$, we only have to consider the play prefix $\mathsf{Check}[q_I, P^{\text{in}}, \Omega(q_I), \Omega(q_I)]$ and define

$$f(\mathsf{Check}[q_I, P^{\text{in}}, \Omega(q_I), \Omega(q_I)]) = (q_I, \bot) \ .$$

Now consider a play prefix w in $\mathcal{G}'$ ending in a Check vertex $\mathsf{Check}[q, A, P, c, d]$ that is consistent with σ as defined so far. By the invariant, we have that the last vertex of $f(w)$ is of the form $(q, A\gamma)$ for some γ. We consider the different cases how w can be prolonged so that the next Check-vertex is reached.

First, consider $w' = w\mathsf{Check}[p, B, P, \min\{c, \Omega(p)\}, \Omega(p)]$, i.e., the skip-transition (q, A, p, B) is simulated. This transition is executable at $(q, A\gamma)$, the last vertex of $f(w)$, leading to the configuration $(p, B\gamma)$. Hence, we can define $f(w') = f(w)(p, B\gamma)$.

Secondly, consider the simulation of a push-transition (q, A, p, BC). To simulate such a transition in $\mathcal{G}'$, one of the players moves to a Push-vertex. From there, Player 0 moves to a Claim-vertex, while making a prediction R during this move. Note that we can assume that R is the prediction as in the definition of σ' above, as we only have to consider play prefixes that are consistent with σ'. After Player 0 has made the prediction, there are two possible types of moves for Player 1: either, he accepts prediction and executes the push-transition and moves from the Claim-vertex to the corresponding Check-vertex, or he skips a portion of the play using the Jump-vertex. we consider both cases independently. First, let

$$w' = w\mathsf{Push}[P, c, p, BC]\mathsf{Claim}[P, c, p, BC, R]\mathsf{Check}[p, B, R, \Omega(p), \Omega(p)] \ ,$$

i.e., Player 0 makes the prediction R and Player 1 chooses to execute the push-transition. The push-transition is also executable at $(q, A\gamma)$, the last vertex of $f(w)$, leading to the configuration $(p, BC\gamma)$. Hence, we can define $f(w') = f(w)(p, BC\gamma)$. On the other hand, assume Player 1 skips a portion of the play using a Jump-vertex, i.e., we have

$$w' = w\mathsf{Push}[P, c, p, BC]\mathsf{Claim}[P, c, p, BC, R]\mathsf{Jump}[qC, P, c, e]\mathsf{Check}[q, C, P, \min\{c, e, \Omega(q)\}, \min\{e, \Omega(q)\}] \ ,$$

where $q \in R_e$ for some color e. By the choice of the prediction R_e by σ', there exists a prolongation $f(w)(p, BC\gamma)(q_1, \gamma_1)...(q_n, \gamma_n)$ of $f(w)(p, BC\gamma)$ that is consistent with σ and satisfies $q^* = q_n$ and that C is the topmost symbol of γ_n). Thus, we can define $f(w') = f(w)(p, BC\gamma)(q_1, \gamma_1)...(q_n, \gamma_n)$. Notice that in all cases the invariant is satisfied. This completes the definition of f.

It remains to show that σ' is a winning strategy for Player 0 from $\mathsf{Check}[q_I, \bot, P^{\text{in}}, \Omega(q_I), \Omega(q_I)]$. To this end, pick some arbitrary play ρ starting in this vertex that is consistent with σ'. First, we show that ρ does not contain a sink-vertex $\mathsf{Win}_1[p]$, as they are losing for Player 0. To the contrary, assume ρ visits $\mathsf{Win}_1[q]$. Then, the last vertex before the first visit to the sink is of the form $\mathsf{Check}[q, A, P, c, d]$ (and is the last Check-vertex in ρ) such that there is a pop-transition $\delta = (q, A, p, \varepsilon)$ with $p \notin P_c$. Note that $A \neq \bot$, since there is no pop-transitions deleting $\bot$. This implies that there was a last simulated push-transition during ρ, at which point Player 0 picked the prediction P according to σ'. Afterwards, only skip-transitions are simulated or push-transitions where Player 0 decides to skip a portion of the play are executed, since the prediction P would be replaced in any other case.

Let w denote the prefix of ρ up to and including the last Check-vertex of ρ (immediately before the sink is reached), and let w' the prefix of w ending in the Check-vertex reached before Player 0 picked the prediction P. Then, $f(w')$ is a prefix of $f(w)$ and there is some non-empty x such that $f(w')x = f(w)$. By the choice of x as a suffix of $f(w)$ we know that is of the form $(q, A\gamma)$. Thus, the pop-transition δ is applicable, leading to the configuration (p, γ). Furthermore, $x(p, \gamma)$ is a path that witnesses $p \in P_e$, where e is the minimal color seen during x. This color is equal to c as encoded in the vertex $\mathsf{Check}[q, A, P, c, d]$, as c is equal to the minimal color visited, if only skip-transitions or push-transitions via Jump-vertices are simulated in ρ. Hence, we have $p \in P_e = P_c$. This contradicts $p \notin P_c$. Thus, no losing sink for Player 0 is ever reached by ρ.

If a sink of the form $\mathsf{Win}_0[q]$ is reached, then Player 0 wins ρ. Hence, it remains to consider the case where ρ visits no sink at all. As $f(\rho_0...\rho_{n'})$ is a prefix of $f(\rho_0...\rho_n)$ for every $n' < n$, there is a unique play $f(\rho)$ such that $f(\rho_0...\rho_n)$ is a prefix of $f(\rho)$ for every n. By construction, the play $f(\rho)$ is consistent with σ. Let $x_0 = f(\rho_0)$ and x_n be such that $f(\rho_0...\rho_n) = f(\rho_0...\rho_{n-1})x_n$ for every $n > 0$, which is non-empty as $f(\rho_0...\rho_n)$ is a proper prolongation of $f(\rho_0...\rho_{n-1})$. Note that this implies $f(\rho) = x_0x_1x_2....$

Let $n_0 < n_1 < n_2 < ...$ be the enumeration of the positions of the Check-vertices in ρ. There are infinitely many, since ρ never visits a sink vertex. Inspecting all three cases of the definition of f, we obtain that the color of the j-th Check-vertex is equal to the minimal color occurring in x_{n_j}. As $f(\rho) = x_0x_1x_2...$ we conclude that the minimal color occurring infinitely often in $f(\rho)$ (which is even, as $f(\rho)$ is winning for Player 0) is equal to the minimal color labeling infinitely many Check-vertices in ρ. Finally, all other vertices in ρ have a larger color, hence the minimal color seen infinitely often ρ is even, i.e., ρ is winning for Player 0. This concludes the first part of the proof.

Now, let us describe how a winning strategy σ for Player i in $\mathcal{G}$ can be constructed from a positional winning strategy σ' for Player 0 in $\mathcal{G}'$. The idea is to simulate σ' in $\mathcal{G}$. This works out fine as long as only skip- and push-transitions are involved. As soon as the first pop-transition is used, σ' leads to a sink Win_0-vertex at which the future moves of σ' are no longer useful for playing in the original game $\mathcal{G}$. Note that no Win_1-vertex is reached, since they are losing for Player 0, and therefore not reachable via a winning strategy. To overcome this, the strategy σ uses a stack to store Claim-vertices visited during the simulated play. This allows us to reset the simulated play and to continue from the appropriate successor Jump-vertex of the Claim-vertex stored on the stack.

Formally, let $\mathcal{A}'|_{\sigma'} = (V'|_{\sigma'}, V_0'|_{\sigma'}, V_1'|_{\sigma'}, E'|_{\sigma'})$ be $\mathcal{A}'$ restricted to the vertices and edges visited by σ' when starting in to play in $\mathsf{Check}[q_I, \bot, P^{\text{in}}, \Omega(q_I), \Omega(q_I)]$. This implies that every vertex from $V_0'|_{\sigma'}$ has a unique successor in $\mathcal{A}'|_{\sigma'}$ and that Win_1-vertices are not contained in $V|_{\sigma'}$. The pushdown transducer $\mathcal{T}_\sigma$ implementing σ is obtained from σ' by employing $\mathcal{A}'|_{\sigma'}$ for its finite control and the Claim-vertices as its stack symbols. The PDT implementing σ is defined by $\mathcal{T}_\sigma = (Q^\sigma, \Gamma^\sigma, \Delta^\sigma, q_I^\sigma, \Sigma_I^\sigma, \Sigma_O^\sigma, \lambda^\sigma)$, where

- $Q^\sigma = V'|_{\sigma'}$,
- $\Gamma^\sigma = \{v \in V'|_{\sigma'} \mid v \text{ is a } \mathsf{Claim}\text{-vertex in } \mathcal{A}'|_\sigma\}$,
- $q_I^\sigma = \mathsf{Check}[q_I, \bot, P^{\text{in}}, \Omega(q_I), \Omega(q_I)]$, and
- $\Sigma_I^\sigma = \Sigma_O^\sigma = \Delta$.

Recall that Δ is the transition relation of the pushdown system $\mathcal{P}$ inducing the arena. To define Δ^σ, we first define the labeling $\ell\colon E'|_{\sigma'} \to \Delta \cup \{\varepsilon\}$ which assigns to every edge in $E'|_{\sigma'}$ its corresponding transition $\delta \in \Delta$ by

$$\ell(v, v') = \begin{cases} (q, A, p, B) & \text{if } (v, v') = (\mathsf{Check}[q, A, P, c, d], \mathsf{Check}[p, B, P, c', d'])\,, \\ (q, A, p, BC) & \text{if } (v, v') = (\mathsf{Check}[q, A, P, c, d], \mathsf{Push}[P, c, p, BC])\,, \\ (q, A, p, \varepsilon) & \text{if } (v, v') = (\mathsf{Check}[q, A, P, c, d], \mathsf{Win}_0[p])\,, \\ \varepsilon & \text{otherwise.} \end{cases}$$

Now, the transition relation Δ^σ is defined by considering several cases for every edge $(v, v') \in E'|_{\sigma'}$.

- If v is not a Claim-vertex *and* v' is not a Win_0-vertex, then $(v, Z, \ell(v, v'), v', Z) \in \Delta^\sigma$, for every $Z \in \Gamma_\bot^\sigma$.
- If v is a Claim-vertex and v' is a Check-vertex, then $(v, Z, \ell(v, v'), v', vZ) \in \Delta^\sigma$ for $Z \in \Gamma_\bot^\sigma$, i.e., the Claim-vertex v is pushed onto the stack.
- If $(v, v') = (\mathsf{Check}[q, A, P, c, d], \mathsf{Win}_0[p])$, then $(v, Z, \ell(v, v'), \mathsf{Jump}[p, C, R, e, c], \varepsilon) \in \Delta^\sigma$ for every $Z \in \Gamma^\sigma$ of the form $Z = \mathsf{Claim}[R, e, q', BC, R']$, i.e., the topmost symbol $\mathsf{Claim}[R, e, q', BC, R']$ is popped from the stack and the pushdown transducer proceeds to the state $\mathsf{Jump}[p, C, R, e, c]$ which would be reached in $\mathcal{A}'|_{\sigma'}$ if Player 1 would have chosen color c and state $p \in R_c$ to determine the successor of $\mathsf{Claim}[R, e, q', BC, R']$.

To complete the definition of $\mathcal{T}_\sigma$, we define the output function λ^σ by $\lambda^\sigma(v) = \ell(v, v')$ if $v \in V_0'|_{\sigma'}$ is a Check-vertex and $(v, v') \in E'|_{\sigma'}$, i.e., the labeling of the edge chosen by σ' determines the output of $\mathcal{T}_\sigma$.

It remains to show that $\mathcal{T}_\sigma$ implements a winning strategy from $(q_I, \bot)$. To show this, let $\rho = (q_0, \gamma_0)(q_1, \gamma_1)(q_2, \gamma_2)\ldots$ be a play that is consistent with the strategy σ implemented by $\mathcal{T}$, starting in $(q_I, \bot)$. To prove that ρ is winning, we need some additional notation. The position n is a stair of ρ, if $|q_n, \gamma_n| \geq |q_{n'}, \gamma_{n'}|$ for every $n' \geq n$, i.e., no smaller stack height is reached after n. Every play has infinitely many stairs and if n is a stair and n' the next one, then either $|q_n, \gamma_n| = |q_{n'}, \gamma_{n'}|$ or $|q_n, \gamma_n| = |q_{n'}, \gamma_{n'}| - 1$.

Let $n_0 < n_1 < n_2 < \ldots$ be the ascending enumeration of ρ's stairs. We chop ρ into infinitely many pieces, leading from one stair to the next by defining $x_0 = (q_0, \gamma_0)\ldots(q_{n_0}, \gamma_{n_0})$ and

$$x_j = (q_{n_{j-1}+1}, \gamma_{n_{j-1}+1})\ldots(q_{n_j}, \gamma_{n_j}) \enspace .$$

Then, we have $\rho = x_0 x_1 x_2 \ldots$ and furthermore, the minimal color seen infinitely often during ρ is the same as the minimal color seen infinitely often in the sequence $c_0 c_1 c_2 \ldots$, where c_j is the minimal color in x_j.

Towards a contradiction, assume ρ is not winning for Player 0, i.e., the minimal color seen infinitely often in $c_0 c_1 c_2 \ldots$ is odd. We construct a play ρ in $\mathcal{G}'$ that is consistent with σ', but losing for Player 0, which will yield the desired contradiction. Intuitively, Player 1 uses Jump-vertices to skip the portions between stairs, thus the simulated play uses only push- and skip-transitions and never leads to a sink-vertex.

More formally, we define ρ' by induction over the stairs, satisfying the following invariants: after simulating a stair, the simulated play is in a Check-vertex whose first two components encode the state and the topmost stack symbol of the stair configuration. Secondly, the simulation in $\mathcal{G}'$ will be consistent with σ.

We have that $n_0 = 0$ as the stack bottom symbol is never deleted. Accordingly, we can define ρ_0' as $\rho_0' = \mathsf{Check}[q_I, \bot, P^{\text{in}}, \Omega(q_I), \Omega(q_I)]$. Now, assume we have simulated ρ up to stair j, defining the play prefix $\rho_0' \ldots \rho_m'$, which is consistent with σ'. We need to prolong $\rho_0' \ldots \rho_m'$ to simulate the $(j+1)$-st stair, while staying consistent with σ'. To this end, we have to consider several cases.

If $n_{j+1} = n_j + 1$ and $|q_{n_{j+1}}, \gamma_{n_{j+1}}| = |q_{n_j}, \gamma_{n_j}|$, then the n_{j+1}-st configuration of ρ is reached via a skip-transition. This skip-transition can be simulated in $\mathcal{G}'$ to extend $\rho_0' \ldots \rho_m'$ by the Check-vertex which is reached by this simulation step. This satisfies the first invariant. Furthermore, if it is Player 1' turn at the n_{j+1}-st configuration of ρ, then also at ρ'_m, hence the prolongation is consistent with σ'. If it is Player 0's turn, then the skip-transition is the one specified by $\mathcal{T}$, which is exactly the one specified by σ' in $\mathcal{G}'$, too. Thus, the prolongation is again consistent with σ'.

If $n_{j+1} = n_j + 1$ and $|q_{n_{j+1}}, \gamma_{n_{j+1}}| = |q_{n_j}, \gamma_{n_j}| + 1$, then the n_{j+1}-st configuration of ρ is reached via a push-transition. This transition is simulated similarly as the skip-transition before, the only difference being that Player 1 decides to simulate the transition by moving from the Claim-vertex to the Check-successor. Thus, we can prolong $\rho'_0 \ldots \rho'_m$ by the corresponding Push-vertex, the Claim-vertex picked by σ' and the unique Check-successor of it. This choice again satisfies our invariant.

Finally, we consider the most involved case, where $n_{j+1} > n_j + 1$, which implies $|q_{n_{j+1}}, \gamma_{n_{j+1}}| = |q_{n_j}, \gamma_{n_j}|$ and $|q_{n_j+1}, \gamma_{n_j+1}| = |q_{n_j}, \gamma_{n_j}| + 1$, i.e., a push-transition is executed first. In this case, the state q of the $(j+1)$-th stair configuration is contained in P_c, where P is the prediction made by Player 0 when simulating the push-transition and c is the minimal color seen between the stairs j and $j+1$. Hence, Player 1 can use the Jump-vertex to reach a Check-vertex that encodes the state q and the second (lower) stack symbol pushed onto the stack during the push-transition. Thus, we can prolong $\rho'_0 \ldots \rho'_m$ by the corresponding Push-vertex, the Claim-vertex picked by σ', the Jump-successor and the Check-vertex encoding the jump to q. This is again consistent with σ'.

To conclude we note that the sequence of colors seen at the Check-vertices during ρ' is exactly the sequence $c_0 c_1 c_2 \ldots$. This is true for the first two cases of the simulation, since the Check-vertex that is reached has exactly the same color as the $(j+1)$-st stair configuration. In the third case, the Check-vertex reached has color c, which is exactly the smallest one seen between the stairs, i.e., the minimal color of x_{j+1}, which is c_{j+1} by definition.

All other colors appearing in ρ' are larger than the colors of the (infinitely many) check-vertices, hence they are irrelevant. We have constructed an infinite play that is consistent with σ', starts in $\mathsf{Check}[q_I, \bot, P^{\text{in}}, \Omega(q_I), \Omega(q_I)]$, and is winning for Player 1. This contradicts the fact that σ' is winning from $\mathsf{Check}[q_I, \bot, P^{\text{in}}, \Omega(q_I), \Omega(q_I)]$. $\square$

Since the parity game $\mathcal{G}'$ is of exponential size and has just one more color than $\mathcal{G}$, it can be solved in exponential time. Hence, we obtain the following complexity result.

Theorem 5.1. *Solving pushdown parity games is in* EXPTIME.

Furthermore, one can show that solving pushdown parity games is also EXPTIME-hard, and therefore EXPTIME-complete.[13]

5.4 Exercises

Exercise 5.1. Let $\mathcal{A} = (V, V_0, V_1, E)$ be a, possibly countably infinite, arena. Given a coloring $\Omega \colon V \to \mathbb{N}$ and $i_\emptyset, i_\infty \in \{0, 1\}$ we define

$$\textsc{InfParity}(\Omega, i_\emptyset, i_\infty) = \{\rho \in V^\omega \mid \mathrm{Inf}(\Omega(\rho)) \text{ is finite and } (\max(\mathrm{Inf}(\Omega(\rho)))) \text{ is even}\} \cup P_\emptyset \cup P_\infty,$$

where $P_\emptyset$ and P_∞ are defined as

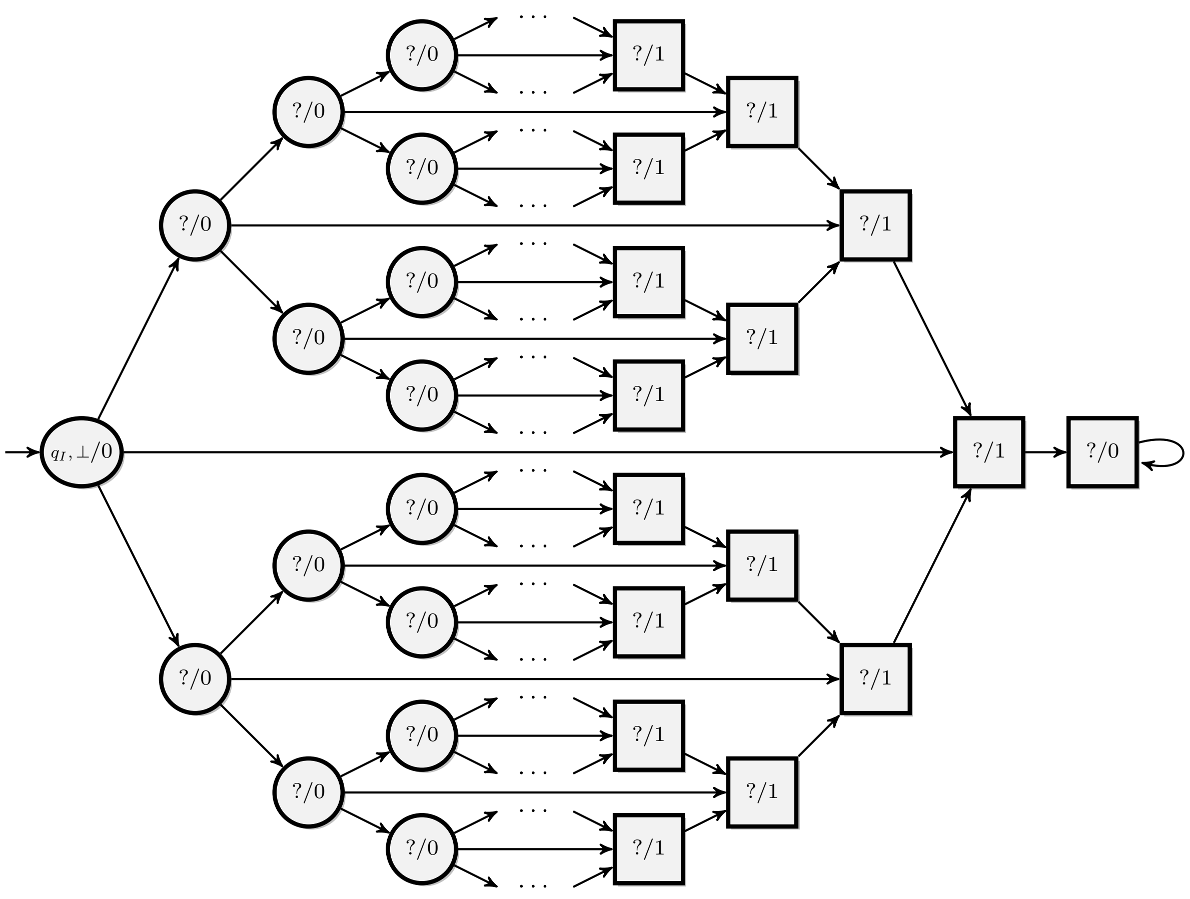
$$P_\emptyset = \begin{cases} \emptyset & \text{if } i_\emptyset = 1, \\ \{\rho \mid \mathrm{Inf}(\Omega(\rho)) = \emptyset\} & \text{if } i_\emptyset = 0, \end{cases} \quad \text{and} \quad P_\infty = \begin{cases} \emptyset & \text{if } i_\infty = 1, \\ \{\rho \mid \mathrm{Inf}(\Omega(\rho)) \text{ is infinite}\} & \text{if } i_\infty = 0. \end{cases}$$

Here, $\Omega(\rho)$ denotes the sequence of colors seen during ρ, i.e. $\Omega(\rho) = \Omega(\rho_0)\Omega(\rho_1)\Omega(\rho_2) \ldots \in \Omega(V)^\omega$. Accordingly, if only finitely many colors are seen infinitely often, then the parity of the maximal one determines the winner. If no color is seen infinitely often, then Player $i_\emptyset$ wins, while Player i_∞ wins if infinitely many colors are seen infinitely often.

Give an arena $\mathcal{A} = (V, V_0, V_1, E)$ and a coloring $\Omega \colon V \to \mathbb{N}$ such that the winning regions of the games $\mathcal{G}_{i_\emptyset, i_\infty} = (\mathcal{A}, \textsc{InfParity}(\Omega, i_\emptyset, i_\infty))$ are pairwise different for every value of $(i_\emptyset, i_\infty)$, that is if $(i_\emptyset, i_\infty) \neq (i'_\emptyset, i'_\infty)$ then $W_0(\mathcal{G}_{i_\emptyset, i_\infty}) \neq W_0(\mathcal{G}_{i'_\emptyset, i'_\infty})$.

Exercise 5.2. Consider the following arena.

[13] See Walukiewicz's paper "Pushdown processes: games and model checking" (Information and Computation 164, 2001) for details.



1. Give a pushdown system, a partition of its states, and a coloring inducing the pushdown parity game sketched above, where only the part reachable from $(q_{in}, \bot)$ is sketched.
2. Give a pushdown transducer implementing a winning strategy for one of the players from $(q_{in}, \bot)$.

Exercise 5.3. Consider the family of pushdown systems $\mathcal{P}_n = (Q_n, \{A\}, \Delta_n, q_I)$ for $n \in \mathbb{N}^+$ with Q_n and Δ_n defined as follows.

- $Q_n = \{q_I, q_A\} \cup \bigcup_{j=1}^{n} \{q_j^t \mid 0 \leq t < j\}$, where p_j denotes the j-th prime number, i.e. $p_1 = 2$, $p_2 = 3$, $p_3 = 5$ and so on.
- $\Delta_n = \{(q_I, X, q_I, AX), (q_I, X, q_A.AX) \mid X \in \{A, \bot\}\} \cup$
  $\{(q_A, A, q_j^0, A) \mid 1 \leq j \leq n\} \cup$
  $\{(q_j^t, A, q_j^{t'}, \varepsilon) \mid 1 \leq j \leq n, 0 \leq t < p_j, t' = (t+1) \bmod p_j\} \cup$
  $\{(q, \bot, q, \bot) \mid q \in Q_n \setminus \{q_I\}\}$

Furthermore, let $Q_0 = Q_n \setminus \{q_A\}$ and $Q_1 = \{q_A\}$ be a partition of Q_n with $n \in \mathbb{N}^+$ and let $\Omega\colon Q_n \to \mathbb{N}$ be the coloring defined by

$$\Omega(q) = \begin{cases} 0 & \text{if there exists a } j \in \{1, 2, \ldots, n\} \text{ with } q = q_j^0 \\ 1 & \text{otherwise} \end{cases}$$

Let $\mathcal{G}_n$ be the pushdown parity game induced by $\mathcal{P}_n$ and the partition and coloring defined above.

a) Draw $\mathcal{G}_2$ up to stack height 8 and give a positional winning strategy from $(q_I, \bot)$ for one of the players.

b) Give a winning strategy from $(q_I, \bot)$ for one of the players for every $\mathcal{P}_n$ with $n \in \mathbb{N}^+$.

Exercise 5.4. Show how to solve pushdown parity games that are induced by a pushdown system without pop transitions, some partition, and some coloring function.

6 Rabin's Theorem

In this chapter, we present an application of infinite games to automata theory by proving Rabin's theorem, which states that satisfiability of monadic second order logic over labeled infinite binary trees (S2S for short) is decidable. This logic subsumes many important logics used in verification, e.g., Linear Temporal Logic and Computation Tree Logic. Thus, satisfiability is decidable for these logics as well. For this reason, Rabin's theorem is referred to as the "mother of all decidability results".

Rabin's original proof is purely combinatorial and has been characterized as cumbersome and complicated. Later, a simpler proof was published, which relies on determinacy of parity games to overcome the key obstacle of the proof. The overall proof technique is to show that S2S is equivalent to parity tree automata, a type of non-deterministic tree automaton running on infinite trees. Thus, a formula φ can be translated into an automaton $\mathscr{A}_\varphi$ such that φ is satisfiable if, and only if, the language of $\mathscr{A}_\varphi$ is non-empty. As the latter problem is decidable[14], satisfiability is decidable, too. The translation is by induction over the structure of φ, which amounts to showing that these automata are closed under intersection (conjunction in S2S), union (disjunction in S2S), complementation (negation in S2S), and projection (quantification in S2S).

The most involved step in this translation is showing that parity tree automata are closed under complementation, i.e., to construct an automaton $\mathscr{A}'$ that accepts a tree t, if it is not accepted by a given automaton $\mathscr{A}$. Hence, there exists an accepting run of $\mathscr{A}'$ on t if, and only if, *all* runs of $\mathscr{A}$ on t are rejecting. This universal statement is not well-suited to be checked by a non-deterministic automaton $\mathscr{A}'$, since such automata are better suited to testing existential statements using their non-determinism. Thus, one needs to turn the universal statement "every run is rejecting" into an existential one, i.e., one needs a quantifier switch.

Here, determinacy of games comes into play, which can be seen as a quantifier switch. In a determined game, Player 0 does not have a winning strategy, i.e., every strategy is losing (a universal statement), if, and only if, Player 1 has a winning strategy (an existential statement). Thus, in a determined infinite game we can turn a universal statement into an existential one.

This property is applied as follows: we define a parity game $\mathcal{G}(\mathscr{A}, t)$ in which Player 0 has a winning strategy from some fixed initial vertex if, and only if, $\mathscr{A}$ accepts t. Thus, if t is not accepted by $\mathscr{A}$, then Player 1 has a winning strategy for $\mathcal{G}(\mathscr{A}, t)$ from the initial vertex. Hence, we construct an automaton $\mathscr{A}'$ which checks whether Player 1 has a winning strategy for the game $\mathcal{G}(\mathscr{A}, t)$ while running on t, essentially guessing the strategy and verifying that it is winning. This automaton recognizes the complement of the language of $\mathscr{A}$.

In the following, we introduce the logic S2S, parity tree automata, prove their equivalence, and then show how to check the automata for emptiness, which solves the satisfiability problem for S2S.

6.1 Infinite Trees

In this subsection, we introduce our notation for (infinite binary) trees. To this end, fix $\mathbb{B}^* = \{0, 1\}^*$, the (infinite) set of finite words over $\mathbb{B}$. We interpret an occurrence of a 0 or a 1 as a direction. By convention, when drawing trees, 0 encodes going to the left and 1 encodes going to the right. Then, we can interpret the set $\mathbb{B}^*$ as the nodes of the infinite binary tree, as drawn in Figure 6.1. Note that edges are defined implicitly, i.e., the edge relation contains all pairs (w, wb) with $w \in \mathbb{B}^*$ and $b \in \mathbb{B}$.

$$\begin{tikzcd}
& & & \varepsilon \arrow[dll] \arrow[drr] & & & \\
& 0 \arrow[dl] \arrow[dr] & & & & 1 \arrow[dl] \arrow[dr] & \\
00 \arrow[d, dash, dashed] & & 01 \arrow[d, dash, dashed] & & 10 \arrow[d, dash, dashed] & & 11 \arrow[d, dash, dashed] \\
 & & & & & &
\end{tikzcd}$$

Figure 6.1: The complete binary tree

This tree is now used as underlying structure to define inputs for automata and as structures to evaluate formulas in. To this end, we label the position of the tree by letters from an alphabet Σ.

Definition 6.1 (Infinite Labeled Trees)**.** *An* (infinite binary labeled) tree *over an alphabet Σ is a mapping $t \colon \mathbb{B}^* \to \Sigma$.*

[14]Most easily seen by defining an emptiness game with parity winning condition, another application of infinite games to automata theory.

Consider the example tree t^e with

$$t^e(w) = \begin{cases} a & \text{if } w = 1^*0, \\ b & \text{otherwise.} \end{cases}$$

We show a graphical representation of t^e in Figure 6.2.

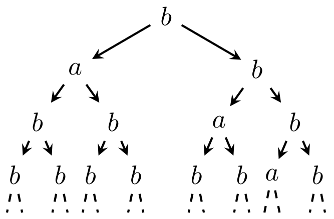


Figure 6.2: Graphical representation of the example tree t^e.

We close this section with some useful definitions to reason about trees.

Definition 6.2 (Subtree). *Let $t\colon \mathbb{B}^* \to \Sigma$ be a tree. The* subtree *of t rooted in $w \in \mathbb{B}^*$ is denoted by t_w and defined by $t_w(w') = t(ww')$ for all $w' \in \mathbb{B}^*$.*

For an example, the subtree t^e_0 of t^e rooted in 0 is depicted in Figure 6.3. Other subtrees are $t^e_1 = t^e_{11} = t^e_{111} = \cdots = t$, for example.

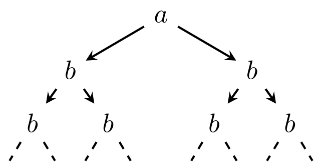


Figure 6.3: The subtree t^e_0 of t^e rooted at node 0.

Definition 6.3 (Branch). *A* branch *π is an infinite word $\pi \in \mathbb{B}^\omega$. The label of $\pi = b_0b_1b_2\cdots$ in a tree $t\colon \mathbb{B}^* \to \Sigma$ is the word*

$$t_{|\pi} = t(\varepsilon)t(b_0)t(b_0b_1)t(b_0b_1b_2)\cdots \in \Sigma^\omega.$$

Some labels of branches of t^e are the following:

$$t^e_{|0^\omega} = bab^\omega \qquad t^e_{|(10)^\omega} = bbab^\omega$$
$$t^e_{|1^\omega} = b^\omega \qquad t^e_{|(01)^\omega} = bab^\omega$$

6.2 The Monadic Second-Order Logic of Two Successors

Next, we introduce *Monadic Second-Order Logic of two Successors*, S2S for short. Here, "two successors" refers to the fact that we consider binary trees, where every node has exactly two successors. Furthermore, "second-order" means that there are two types of quantification, i.e., quantification over nodes (first-order quantification) and quantifications over relations of nodes (second-order quantification). Finally, the qualifier "monadic" refers to restricting second-order quantification to unary relations, i.e., to sets.

Correspondingly, there are two different types of variables in S2S. First-order variables are denoted by lowercase letters from the end of the latin alphabet, e.g., $x, y, z, \ldots$. Second-order variables are denoted by uppercase letters from the end of the latin alphabet, e.g., $X, Y, Z, \ldots$. First-order variables refer to nodes of the binary tree while second-order variables refer to sets of nodes. Formally, we use two disjoint infinite sets $\mathcal{V}_1$ and $\mathcal{V}_2$ of first-order and second-order variables, respectively. Furthermore, we require $\varepsilon \notin \mathcal{V}_1 \cup \mathcal{V}_2$.

We begin our introduction to S2S with the definition of its syntax, which is defined in three steps: terms are combined into atomic formulas which are combined into formulas.

Definition 6.4 (Syntax of S2S). *A* term *of S2S is either the symbol ε or a first-order variable $x \in \mathcal{V}_1$. The* atomic formulas *of S2S are defined inductively as follows:*

- *$s = s'$ is an atomic formula for all terms s, s'.*
- *$s \sqsubseteq s'$ is an atomic formula for all terms s, s'.*
- *$P_a(s)$ is an atomic formula for every $a \in \Sigma$ and every term s.*
- *$S_i(s, s')$ is an atomic formula for every $i \in \{0, 1\}$ and all terms s, s'.*
- *$s \in X$ is an atomic formula for every second-order variable X and all terms s.*

The formulas *of S2S are defined inductively as follows:*

- *Every atomic formula is a formula.*
- *$\neg\varphi$ is a formula for every formula φ.*
- *$\varphi \wedge \varphi'$ is a formula for all formulas φ, φ'.*
- *$\varphi \vee \varphi'$ is a formula for all formulas φ, φ'.*
- *$\exists x\varphi$ is a formula for every formula φ.*
- *$\forall x\varphi$ is a formula for every formula φ.*
- *$\exists X\varphi$ is a formula for every formula φ.*
- *$\exists X\varphi$ is a formula for every formula φ.*

As usual, we use the boolean connectives like $\rightarrow$ and $\leftrightarrow$ as syntactic sugar, i.e., $\varphi \rightarrow \varphi' = \neg\varphi \vee \varphi'$ and $\varphi \leftrightarrow \varphi' = (\varphi \rightarrow \varphi') \wedge (\varphi' \rightarrow \varphi)$.

In order to define the semantics of S2S we first interpret the first-order and second-order variables using a valuation function μ.

Definition 6.5 (Variable Valuation)**.** *A* variable valuation *is a function $\mu\colon \mathcal{V}_1 \rightarrow \mathbb{B}^* \cup \mathcal{V}_2 \rightarrow 2^{(\mathbb{B}^*)} \cup \{\varepsilon\} \rightarrow \{\varepsilon\}$ mapping first-order variables x to a node $\mu(x)$, every second-order variable X to a set of nodes $\mu(X) \subseteq \mathbb{B}^*$, and ε to ε.*

Given a variable valuation μ, a first-order variable $x \in \mathcal{V}_1$, and $w \in \mathbb{B}^*$, we define the update of μ in x by w*, denoted by $\mu[x \mapsto w]$, as*

$$\mu[x \mapsto w](y) = \begin{cases} w & \text{if } y = x, \\ \mu(y) & \text{if } y \neq x. \end{cases}$$

Given a variable valuation μ, a second-order variable $X \in \mathcal{V}_2$, and $B \subseteq \mathbb{B}^*$, we define the update of μ in X by B*, denoted by $\mu[X \mapsto B]$, as*

$$\mu[X \mapsto B](Y) = \begin{cases} B & \text{if } Y = X, \\ \mu(Y) & \text{if } Y \neq X. \end{cases}$$

We now define what it means for a given tree t and a variable evaluation μ to evaluate a formula. Formally, the semantics of S2S are given by a satisfaction relation $\models$ between a tree t, a variable valuation μ interpreting the variables, and a formula.

Definition 6.6 (Semantics of S2S)**.** *The* semantics of S2S *are defined recursively for a tree t, a variable valuation μ, and a formula φ as follows:*

- *$t, \mu \models s = s'$ if, and only if, $\mu(s) = \mu(s')$.*
- *$t, \mu \models s \sqsubseteq s'$ if, and only if, $\mu(s)$ is a prefix of $\mu(s')$.*
- *$t, \mu \models P_a(s)$ if, and only if, $t(\mu(s)) = a$.*
- *$t, \mu \models S_i(s, s')$ if, and only if, $\mu(s') = \mu(s) \cdot i$, for $i \in \{0, 1\}$.*
- *$t, \mu \models s \in X$ if, and only if, $\mu(s) \in \mu(X)$.*
- *$t, \mu \models \neg\varphi$ if, and only if, it is not the case that $t, \mu \models \varphi$.*

- *$t, \mu \models \varphi \wedge \varphi'$ if, and only if, $t, \mu \models \varphi$ and $t, \mu \models \varphi'$.*
- *$t, \mu \models \varphi \vee \varphi'$ if, and only if, $t, \mu \models \varphi$ or $t, \mu \models \varphi'$.*
- *$t, \mu \models \exists x \varphi$ if, and only if, $t, \mu[x \mapsto w] \models \varphi$ for some $w \in \mathbb{B}^*$.*
- *$t, \mu \models \forall x \varphi$ if, and only if, $t, \mu[x \mapsto w] \models \varphi$ for all $w \in \mathbb{B}^*$.*
- *$t, \mu \models \exists X \varphi$ if, and only if, $t, \mu[X \mapsto B] \models \varphi$ for some $B \subseteq \mathbb{B}^*$.*
- *$t, \mu \models \forall X \varphi$ if, and only if, $t, \mu[X \mapsto B] \models \varphi$ for all $B \subseteq \mathbb{B}^*$.*

If an occurrence of a variable x or X in a formula φ is under the scope of a quantifier, then the satisfaction of φ does not depend on the valuation $\mu(x)$ or $\mu(X)$, respectively. Such an occurrence is called *bound*. In particular, if a formula only has bound occurrences, then its satisfaction only depends on the tree t but not on the variable valuation under consideration.

Definition 6.7 (Sentences)**.** *Let φ be an S2S formula. An occurrence of a variable in φ is called* free, *if it is not under the scope of a quantifier, otherwise it is called* bound. *A formula without free occurrences of variables is called a* sentence.

In Figure 6.4, this definition is illustrated on an example formula, which is not a sentence. Here, x and y appears both free and bound in φ and X is quantified but never appears in a term.

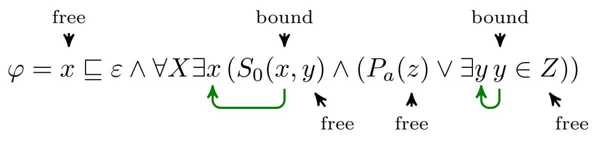

free
bound
bound
$\varphi = x \sqsubseteq \varepsilon \wedge \forall X \exists x\, (S_0(x,y) \wedge (P_a(z) \vee \exists y\, y \in Z))$
free
free
free

Figure 6.4: Example formula φ with free and bound variables.

As argued above, satisfaction of sentences is independent of the variable valuation under consideration, i.e., for every sentence φ it holds true that $t, \mu \models \varphi$ if, and only if, $t, \mu' \models \varphi$. Accordingly, it suffices to write $t \models \varphi$ instead of $t, \mu \models \varphi$ and we say that t satisfies φ. We are interested in the satisfiability problem for S2S:

> Given a sentence φ, is there some tree t such that $t \models \varphi$?

If we define the language of φ to be the set $\mathcal{L}(\varphi) = \{t \colon \mathbb{B}^* \to \Sigma \mid t \models \varphi\}$ of trees satisfying φ, then we can reformulate the problem as:

> Given a sentence φ, is $\mathcal{L}(\varphi)$ non-empty?

In the upcoming sections, we solve the satisfaction problem for S2S.

We close this section by some examples of S2S formulas and their satisfaction in our example tree t^e from Figure 6.2.

- $t^e \models P_b(\varepsilon)$, as the root is labeled with b.
- $t^e \not\models P_a(\varepsilon)$, as the root is not labeled with a.
- $t^e \models \exists x S_1(\varepsilon, x) \wedge P_b(x)$, as the right child of the root is labeled with b.
- $t^e \models \exists X\, \textsc{Brnch}(X) \wedge \forall x(x \in X \to P_b(x))$ where $\textsc{Brnch}(X)$ is defined as

$$\begin{aligned}\textsc{Brnch}(X) = \varepsilon \in X \wedge \forall x\big(x \in X \wedge \neg\exists y\, (S_0(y,x) \vee S_1(y,x))\big) \to x = \varepsilon \wedge \\ \forall x\big(x \in X \to \exists y(y \in X \wedge (S_0(x,y) \vee S_1(x,y))) \wedge \\ \forall y'(y' \in X \wedge (S_0(x,y') \vee S_1(x,y'))) \to y = y'\big),\end{aligned}$$

  expressing that the root is in X and that it is the only node in X without predecessor, and that each node in X has a successor in X, but not two.

  This formula holds in t^e, as the branch 1^ω is labeled by b^ω.

- $t^e \models \exists X \exists Y \exists Z\, \textsc{Brnch}(X) \land \textsc{Partitions}(X,Y,Z) \land \textsc{Alternate}(Y,Z) \land \psi(X,Y)$ where $\textsc{Brnch}(X)$ is defined as above, where

$$\begin{aligned}\textsc{Partitions}(X,Y,Z) = {} & Y \subseteq X \land Z \subseteq X \land \\ & \forall x\, (x \in X \to ((x \in Y \lor x \in Z) \land \neg(x \in Y \land x \in Z)))\end{aligned}$$

  expresses that Y and Z partition X (here, we use $\subseteq$ as syntactic sugar; See Exercise 6.1), where

$$\begin{aligned}\textsc{Alternate}(Y,Z) = {} & \varepsilon \in Y \land \forall x \forall x'\, ((S_0(x,x') \lor S_1(x,x')) \to \\ & (\neg(x \in Y \land x' \in Y) \land \neg(x \in Z \land x' \in Z)))\end{aligned}$$

  expresses that membership in Y and Z alternates (under the assumption that $\textsc{Partitions}(X,Y,Z)$ is satisfied), and where

$$\psi(X,Y) = \exists y(y \in Y \land P_a(y) \land \forall x((x \in X \land \neg(x = y)) \to P_b(x)))$$

  expresses that there is a position on the branch X and in Y where a holds and every other position has a b. As y contains exactly the even positions along the branch induced by X, the conjunction of the formulas requires the existence of a branch that is labeled by a sequence from $(bb)^*ab^\omega$.

  This formula holds in t^e, as the branch 10^ω is labeled by such a sequence, i.e., by $bbab^\omega$.

6.3 Parity Tree Automata

Recall that we want to decide the satisfiability problem for S2S. However, as working in a logic is typically cumbersome, the usual approach is to devise an equi-expressive automaton model for the logic. Then, one first translates a sentence φ into an automaton $\mathscr{A}_\varphi$ recognizing exactly the trees satisfying φ. In this situation, one only has to solve the emptiness problem for this automaton model, which is typically simpler. The automaton model we consider here is that of parity tree automata. These automata process an input tree top-down, meaning that a run starts in the initial state at the root and every transition of the automaton takes the current state and input letter at the current node of the tree and yields two successor states, one for the left child and one for the right child. Finally, acceptance is defined by a parity condition which has to hold on every branch of the run. Such an automaton is necessarily non-deterministic, as a deterministic automaton cannot even check, e.g., whether there is an a-labeled node in the tree (see Exercise 6.7).

Definition 6.8 (Parity Tree Automaton)**.** *A* parity tree automaton $\mathscr{A} = (Q, \Sigma, q_I, \Delta, \Omega)$ *consists of*

- *a finite set Q of states,*
- *an alphabet Σ,*
- *an initial state $q_I \in Q$,*
- *a transition relation $\Delta \subseteq Q \times \Sigma \times Q \times Q$, and*
- *a coloring $\Omega\colon Q \to \mathbb{N}$.*

We often graphically depict a transition $\tau = (q, a, q_0, q_1)$ by the tree shown below:

$$\begin{tikzcd}
 & q,a \arrow[dl, dash] \arrow[dr, dash] & \\
q_0 & & q_1
\end{tikzcd}$$

Such a transition can be seen as a binary tree of height one, whose vertices are labeled by states of the automaton and whose root is additionally labeled by a letter from the input alphabet. Then, a run of the automaton can be understood as a tiling of a given input tree by such transitions that is locally consistent and has the initial state in the root.

Definition 6.9 (Run)**.** *A* run *of a parity tree automaton $\mathscr{A} = (Q, \Sigma, q_I, \Delta, \Omega)$ on a tree $t\colon \mathbb{B}^* \to \Sigma$ is a mapping $r\colon \mathbb{B}^* \to Q$ such that $r(\varepsilon) = q_I$ and $(r(w), t(w), r(w0), r(w1)) \in \Delta$ for all $w \in \mathbb{B}^*$.*

Note, that a run is a Q-labeled binary tree, i.e., all our notations for trees are applicable.

Definition 6.10 (Accepting Run)**.** *A run $r\colon \mathbb{B}^* \to Q$ of a parity tree automaton $\mathscr{A} = (Q, \Sigma, q_I, \Delta, \Omega)$ is* accepting *if for every branch $\pi \in \mathbb{B}^\omega$ the label $r_{|\pi} = q_0q_1q_2\cdots \in Q^\omega$ of the branch satisfies the parity condition, i.e.,*

$$\min(\mathrm{Inf}(\Omega(q_0)\Omega(q_1)\Omega(q_2)\cdots))$$

is even.

The language of a parity tree automaton $\mathscr{A}$, denoted by $\mathcal{L}(\mathscr{A})$, is defined as usual, i.e.,

$$\mathcal{L}(\mathscr{A}) = \{t\colon \mathbb{B}^* \to \Sigma \mid \mathscr{A} \text{ has an accepting run on } t\}.$$

A parity tree automaton $\mathscr{A} = (Q, \Sigma, q_I, \Delta, \Omega)$ is complete if, for every state $q \in Q$ and every letter $a \in \Sigma$ there are states $q_0, q_1 \in Q$ such that $(q, a, q_0, q_1) \in \Delta$. A complete automaton has a run on every input tree. By adding a new sink state which is colored by an odd color one can complete each parity tree automaton without changing the language it accepts. Thus, from now on we can assume every automaton to be complete, if necessary. In examples, we typically keep them incomplete to improve readability.

Next, we present example parity tree automata recognizing the following tree languages.

1. $L_1 = \{t\colon \mathbb{B}^* \to \{a, b\} \mid t_{|\pi} = b^\omega \text{ for some branch } \pi\}$, the language of trees containing a branch labeled with b^ω.
2. $L_2 = \{t\colon \mathbb{B}^* \to \{a, b\} \mid b \in \mathrm{Inf}(t_{|\pi}) \text{ for some branch } \pi\}$, the language of trees containing a branch labeled with infinitely many b's.
3. $L_3 = \{t^e\}$, the language containing exactly our example tree t^e from above.

1. We construct a parity tree automaton $\mathscr{A}_1$ that accepts every tree containing a branch completely labeled with b's, i.e., such that $\mathcal{L}(\mathscr{A}_1) = L_1$. As parity tree automata are non-deterministic, $\mathscr{A}_1$ can guess such a branch and verify that it only contains b's. For all other nodes in the tree we use a dummy state allowing an arbitrary label. Accordingly, we define $\mathscr{A}_1 = (Q_1, \{a, b\}, q_I, \Delta_1, \Omega_1)$ with

   - $Q_1 = \{q_I, q_*\}$,
   - $\Omega_1(q_I) = \Omega_1(q_*) = 0$, and
   - $$\Delta_1 = \left\{ \begin{tikzcd} & q_I, b \arrow[dl, dash] \arrow[dr, dash] & \\ q_I & & q_* \end{tikzcd}, \begin{tikzcd} & q_I, b \arrow[dl, dash] \arrow[dr, dash] & \\ q_* & & q_I \end{tikzcd}, \begin{tikzcd} & q_*, a \arrow[dl, dash] \arrow[dr, dash] & \\ q_* & & q_* \end{tikzcd}, \begin{tikzcd} & q_*, b \arrow[dl, dash] \arrow[dr, dash] & \\ q_* & & q_* \end{tikzcd} \right\}.$$

   The automaton starts in the initial state and propagates it to exactly one of its successors, thereby guessing the designated branch. On this branch, only b's are allowed: the automaton gets stuck if it encounters an a in state q_I. On all nodes of the guessed branch, the dummy state q_* is used. Thus, the acceptance condition is irrelevant, we just have to ensure that every run is accepting. Hence, every state has color 0.

2. Now we want to construct a parity tree automaton $\mathscr{A}_2$ that accepts every tree containing a branch with infinitely many b's. Similar to $\mathscr{A}_1$ we can guess the corresponding branch. We only need to adjust the coloring to control the infinite behavior. Furthermore, the automaton may no longer get stuck if it encounters an a on the guessed branch. Our corresponding automaton $\mathscr{A}_2 = (Q_2, \{a, b\}, q_a, \Delta_2, \Omega_2)$ is then given by

   - $Q_2 = \{q_*, q_a, q_b\}$,
   - $\Omega_2(q_b) = \Omega_2(q_*) = 0$ and $\Omega_2(q_a) = 1$, and
   - $$\Delta_2 = \left\{ \begin{tikzcd} & q_a, a \arrow[dl, dash] \arrow[dr, dash] & \\ q_a & & q_* \end{tikzcd}, \begin{tikzcd} & q_a, a \arrow[dl, dash] \arrow[dr, dash] & \\ q_* & & q_a \end{tikzcd}, \begin{tikzcd} & q_a, b \arrow[dl, dash] \arrow[dr, dash] & \\ q_b & & q_* \end{tikzcd}, \begin{tikzcd} & q_a, b \arrow[dl, dash] \arrow[dr, dash] & \\ q_* & & q_b \end{tikzcd}, \begin{tikzcd} & q_*, a \arrow[dl, dash] \arrow[dr, dash] & \\ q_* & & q_* \end{tikzcd}, \right.$$
     $$\left. \begin{tikzcd} & q_b, a \arrow[dl, dash] \arrow[dr, dash] & \\ q_a & & q_* \end{tikzcd}, \begin{tikzcd} & q_b, a \arrow[dl, dash] \arrow[dr, dash] & \\ q_* & & q_a \end{tikzcd}, \begin{tikzcd} & q_b, b \arrow[dl, dash] \arrow[dr, dash] & \\ q_b & & q_* \end{tikzcd}, \begin{tikzcd} & q_b, b \arrow[dl, dash] \arrow[dr, dash] & \\ q_* & & q_b \end{tikzcd}, \begin{tikzcd} & q_*, b \arrow[dl, dash] \arrow[dr, dash] & \\ q_* & & q_* \end{tikzcd} \right\}.$$

We use the states q_a and q_b along the guessed branch to denote the last label seen. It follows that $\mathcal{L}(\mathscr{A}_2) = L_2$. Note that using q_b as initial state does not change the language accepted by the automaton.

3. Finally we want to construct a parity tree automaton $\mathscr{A}_3$ that only accepts our example tree t^e form the previous section. We construct $\mathscr{A}_3 = (Q_3, \{a, b\}, q_r, \Delta_3, \Omega_3)$ to get $\mathcal{L}(\mathscr{A}_3) = L_3$ as follows.

   - $Q_3 = \{q_*, q_r, q_l\}$,
   - $\Omega_3(q_r) = \Omega_3(q_l) = \Omega_3(q_r) = 0$, and
   - $\Delta_3 = \left\{ \begin{array}{ccc} & q_r, b & \\ / & & \backslash \\ q_l & & q_r \end{array} , \begin{array}{ccc} & q_l, a & \\ / & & \backslash \\ q_* & & q_* \end{array} , \begin{array}{ccc} & q_*, b & \\ / & & \backslash \\ q_* & & q_* \end{array} \right\}.$

6.4 S2S and Parity Tree Automata are Equivalent

In this section, we prove that S2S and parity tree automata are equally expressive, i.e., for every parity tree automaton $\mathscr{A}$ there exists an S2S sentence $\varphi_{\mathscr{A}}$ such that $t \in \mathcal{L}(\mathscr{A})$ if, and only if, $t \models \varphi_{\mathscr{A}}$ and that for every S2S sentence φ there is an automaton $\mathscr{A}_\varphi$ such that $t \models \varphi$ if, and only if, $t \in \mathcal{L}(\mathscr{A}_\varphi)$.

We begin by showing the simpler direction, i.e., we show that every language recognized by a parity tree automaton is also definable by some S2S sentence.

Theorem 6.1. *For every parity tree automaton $\mathscr{A}$ with alphabet Σ there exists an S2S sentence $\varphi_{\mathscr{A}}$ such that*

$$t \in \mathcal{L}(\mathscr{A}) \Leftrightarrow t \models \varphi_{\mathscr{A}}$$

for all trees $t \colon \mathbb{B}^* \to \Sigma$.

Proof. Let $\mathscr{A} = (Q, \Sigma, q_I, \Delta, \Omega)$. We construct a sentence $\varphi_{\mathscr{A}}$ that expresses that there exists an accepting run of $\mathscr{A}$ on t. Without loss of generality, let $Q = \{q_0, \cdots, q_{n-1}\}$ and $q_I = q_0$. Intuitively, we express the existence of n disjoint sets $X_0, \ldots, X_{n-1}$ such that X_j contains exactly those nodes labeled by state q_j. Additionally, we formulate the local requirements on a run as well as the acceptance condition.

Formally, we define $\varphi_{\mathscr{A}}$ as

$$\begin{aligned}
\varphi_{\mathscr{A}} = \exists X_0 \exists X_1 \cdots \exists X_{n-1} \big(& \qquad (1)\\
& \forall x \bigvee_{j=0}^{n-1} \big(x \in X_j \wedge \bigwedge_{j' \neq j} \neg(x \in X_{j'})\big) \wedge \qquad (2)\\
& \varepsilon \in X_0 \wedge \qquad (3)\\
& \forall x\, \exists x_l \exists x_r \big[S_0(x, x_l) \wedge S_1(x, x_r) \wedge \qquad (4)\\
& \qquad \bigvee_{(q_t, a, q_i, q_j) \in \Delta} \big(x \in X_t \wedge P_a(x) \wedge x_l \in X_i \wedge x_r \in X_j\big)\big] \wedge \qquad (5)\\
& \forall X\, \textsc{Brnch}(X) \rightarrow \bigvee_{\substack{c \in \Omega(Q),\\ c \text{ even}}} \qquad (6)\\
& \qquad \forall x\big[x \in X \rightarrow \exists y(y \in X \wedge x \sqsubseteq y \wedge \bigvee_{\substack{j \in [n],\\ \Omega(q_j) = c}} y \in X_j)\big] \wedge \qquad (7)\\
& \qquad \exists x\big[x \in X \wedge \forall y((y \in X \wedge x \sqsubseteq y) \rightarrow \bigvee_{\substack{j \in [n],\\ \Omega(q_j) \geq c}} y \in X_j)\big]\big) \qquad (8)
\end{aligned}$$

The formula expresses the existence of n sets $X_0, \ldots, X_{n-1}$ (line 1), one for each state q_j, that partition the set of nodes (line 2). Furthermore, the root is in the set corresponding to the initial state (line 3) and the transition relation is respected locally (lines 4 and 5). Thus, the sets have to encode a run. Finally, for every branch there is an even color (line 6) that appears infinitely often on the branch (line 7), but from some point onwards, no smaller color appears on the branch (line 8). Thus, the last three lines express that the encoded run is accepting. ☐

The backward direction turns out to be more complicated. First note that due to logical equivalences it suffices to consider S2S sentences consisting of atomic formulas, as well as disjunction, existential quantification and negation. We translate S2S sentences into parity tree automata inductively over the structure of the formulas, i.e., we have to show that we can build automata for the atomic formulas and show that they are closed under union, projection, and complement, which yields the inductive step for disjunction, existential quantification, and negation. The only non-trivial step is closure of parity tree automata under complement. As already alluded to in the introduction we rely on infinite games to show this result. As a first step, we characterize the acceptance of a tree t by a parity tree automaton $\mathscr{A}$ using a parity game $\mathcal{G}(\mathscr{A}, t)$ such that $t \in \mathcal{L}(\mathscr{A})$ if, and only if, Player 0 wins $\mathcal{G}(\mathscr{A}, t)$ from a fixed initial vertex. By determinacy, this means that Player 1 wins $\mathcal{G}(\mathscr{A}, t)$ from this vertex if, and only if, t is not accepted by $\mathscr{A}$, i.e., t is in the complement language. As the tree t is infinite, the arena of $\mathcal{G}(\mathscr{A}, t)$ is infinite as well.

Definition 6.11 (Acceptance Game)**.** *Let $\mathscr{A} = (Q, \Sigma, q_I, \Delta, \Omega)$ be a complete parity tree automaton and let $t\colon \mathbb{B}^* \to \Sigma$ be some tree. The* acceptance game *$\mathcal{G}(\mathscr{A}, t) = (\mathcal{A}, \textsc{Parity}(\Omega'))$ with $\mathcal{A} = (V, V_0, V_1, E)$ is a parity game given by*

- $V = V_0 \cup V_1$,
- $V_0 = \mathbb{B}^* \times Q$,
- $V_1 = \mathbb{B}^* \times \Delta$,
- $E = \{((w, q), (w, \tau)) \in V_0 \times V_1 \mid \tau = (q, t(w), q_0, q_1)$ *for some states* $q_0, q_1 \in Q\} \cup$
  $\{((w, (q, t(w), q_0, q_1)), (wb, q_b)) \in V_1 \times V_0 \mid b \in \{0, 1\}\}$,
- $\Omega'(w, q) = \Omega(q)$ *for all* $(w, q) \in V_0$, *and*
- $\Omega'(w, (q, a, q_0, q_1)) = \Omega(q)$ *for all* $(w, (q, a, q_0, q_1)) \in V_1$.

The idea behind this construction is that Player 1 picks directions and thereby builds a branch π. During this, Player 0 has to pick transitions along this branch in a way that is compatible with t and such that the parity condition of $\mathscr{A}$ is satisfied by the sequence of states labeling the roots of the transitions. Since Player 1 might pick any branch π, Player 0 has to be prepared to construct a labeling of the whole binary tree $\mathbb{B}^*$ using the states in a way such that every branch is accepting. As a consequence, Player 0 has a winning strategy if, and only if, there is an accepting run of $\mathscr{A}$ on t. Due to their roles described above, the players in the acceptance game are often also called "Automaton" and "Pathfinder". Since we require the automaton $\mathscr{A}$ to be complete, the arena is well-defined, i.e., every Player 0 vertex has at least one successor since at least one transition is applicable.

Lemma 6.1. *Let $\mathscr{A}$ be a parity tree automaton over Σ and t be a tree over Σ. Then it holds that*

$$t \in \mathcal{L}(\mathscr{A}) \Leftrightarrow (\varepsilon, q_I) \in W_0(\mathcal{G}(\mathscr{A}, t)).$$

Proof. Let $\mathscr{A} = (Q, \Sigma, q_I, \Delta, \Omega)$ and $\mathcal{G}(\mathscr{A}, t) = (\mathcal{A}, \textsc{Parity}(\Omega'))$ with $\mathcal{A} = (V, V_0, V_1, E)$. For the direction from left to right, see Exercise 6.8.

For the other direction, let σ be a positional winning strategy for Player 0 from (ε, q_I) in $\mathcal{G}(\mathscr{A}, t)$. We have to construct an accepting run r of $\mathscr{A}$ on t.

By construction, for every $w \in \mathbb{B}^*$ there is a unique play prefix p_w that starts in (ε, q_I), is consistent with σ, and ends in a vertex of the form (w, q) for some $q \in Q$. Accordingly, we define the run r by $r(w) = q$, where q is given by $\mathrm{Lst}(p_w) = (w, q)$.

We show that r is an accepting run of $\mathscr{A}$. First, we have that $r(\varepsilon) = q_I$ since $p_\varepsilon = (\varepsilon, q_I)$. Now let $w \in \mathbb{B}^+$ be arbitrary and let the corresponding play prefix p_w end in (w, q). Consider the vertex

$$\sigma(w, q) = (w, (q, t(w), q_0, q_1)) \in \mathbb{B}^* \times \Delta$$

picked by the strategy σ. Then, we have

$$p_{wb} = p_w\,(w, (q, t(w), q_0, q_1))\,(wb, q_b)$$

for both $b \in \{0, 1\}$. Hence, $q_b = r(wb)$ and we have $(r(w), t(w), r(w0), r(w1)) = (q, t(w), q_0, q_1)$ which is a transition from Δ. Hence, r satisfies both requirements on the run.

Now let π be a branch of r and let $\rho \in \text{Plays}(\mathcal{A}, (\varepsilon, q_I), \sigma)$ be the play where Player 1 picks the directions according to π and where Player 0 plays according to σ. This play has the same sequence of colors as $r_{|\pi}$ with every occurrence of a color doubled. Accordingly, $r_{|\pi}$ fulfills the parity condition given by the fact that ρ is winning. As this holds for every branch π it follows that r is an accepting run. □

Before we continue proving closure of parity tree automata under complement, we take a short detour and show that the acceptance game can be modified to solve the emptiness problem for such automata. The arena of the acceptance game $\mathcal{G}(\mathscr{A}, t)$ is infinite, since it encodes the the infinite tree t. In this game, Player 0 has to construct an accepting run of $\mathscr{A}$ on the fixed tree t. By projecting away the tree t and letting Player 0 construct it implicitly, we obtain a finite game that is won by Player 0 if, and only if, there is a tree that is accepted by $\mathscr{A}$.

Definition 6.12 (Emptiness Game)**.** *Let $\mathscr{A} = (Q, \Sigma, q_I, \Delta, \Omega)$ be a complete parity tree automaton. The* emptiness game *$\mathcal{G}(\mathscr{A}) = (\mathcal{A}, \text{PARITY}(\Omega'))$ with $\mathcal{A} = (V, V_0, V_1, E)$ is a parity game defined as*

- *$V = V_0 \cup V_1$ with $V_0 = Q$ and $V_1 = \Delta$,*
- *$E = \{(q, \tau) \in V_0 \times V_1 \mid \tau = (q, a, q_0, q_1)$ for some $a \in \Sigma$ and some $q_0, q_1 \in Q\} \cup$*
  $\{((q, a, q_0, q_1), q_b) \in V_1 \times V_0 \mid b \in \{0, 1\}\}$, and
- *$\Omega'(q) = \Omega(q)$ for all $q \in V_0$ and $\Omega'(q, a, q_0, q_1) = \Omega(q)$ for all $(q, a, q_0, q_1) \in V_1$.*

Thus, by making a move in the emptiness game, Player 0 picks a letter and an applicable transition, thereby constructing a tree and a run on it simultaneously (one branch per play) while Player 1 is still in charge of picking the branch π. Player 0 wins if, and only if, the branch π of the run satisfies the parity condition.

We show that $\mathcal{G}(\mathscr{A})$ characterizes emptiness of $\mathscr{A}$.

Theorem 6.2. *Let $\mathscr{A}$ be a parity tree automaton. Then, $\mathcal{L}(\mathscr{A}) \neq \emptyset$ if, and only if, $q_I \in W_0(\mathcal{G}(\mathscr{A}))$.*

Proof. Let $\mathscr{A} = (Q, \Sigma, q_I, \Delta, \Omega)$ and $\mathcal{G}(\mathscr{A}) = (\mathcal{A}, \text{PARITY}(\Omega'))$ with $\mathcal{A} = (V, V_0, V_1, E)$, i.e., $V_0 = Q$ and $V_1 = \Delta$. For the direction from right to left, see Exercise 6.9.

For the other direction, fix some $t \in \mathcal{L}(\mathscr{A})$ and an accepting run r of $\mathscr{A}$ on t. We define a strategy σ for Player 0 from q_I in $\mathcal{G}(\mathscr{A})$ that implicitly constructs the branches of t and r corresponding to the branch picked by Player 1 during the play.

For the unique play prefix of length one starting in q_I, we define

$$\sigma(q_I) = (r(\varepsilon), t(\varepsilon), r(0), r(1)).$$

Now, consider a play prefix

$$w = q_0\,(q_0, a_0, q_0^0, q_0^1)\,q_1\,(q_1, a_1, q_1^0, q_1^1)\,q_2\,(q_2, a_2, q_2^0, q_2^1)\,\cdots\,q_{n-1}\,(q_{n-1}, a_{n-1}, q_{n-1}^0, q_{n-1}^1)\,q_n$$

starting in q_I such that $\sigma(w')$ is already defined for every strict prefix w' of w ending in a vertex of Player 0. By construction, for every j in the range $0 \leq j \leq n-1$ there is a bit $b_j \in \mathbb{B}$ such that $q_{j+1} = q_j^{b_j}$, as Player 1 has to move to one of the vertices q_j^0 and q_j^1 when it is his turn at the vertex (q_j, a_j, q_j^0, q_j^1). Let $x = b_0 b_1 \cdots b_{n-1}$ be the sequence of these bits. Then, we define

$$\sigma(w) = (r(x), t(x), r(x0), r(x1)).$$

Clearly, $\sigma(w) \in \Delta$, since r is a run. Hence, σ is indeed a strategy. We show that σ is winning for Player 0 from q_I. To this end, let $\rho \in \text{Plays}(\mathcal{A}, q_I, \sigma)$ be arbitrary and $b_0 b_1 b_2 \cdots$ be the infinite sequence of associated directions defined as above. Furthermore, let $c_0 c_1 c_2 \cdots$ be the sequence of colors seen during ρ, which satisfies $c_{2n} = c_{2n+1}$ for all $n \in \mathbb{N}$ by construction of $\mathcal{G}(\mathscr{A})$. Also by construction, the sequence $c_0 c_2 c_4 \cdots$ is equal to $\Omega(r(\varepsilon))\Omega(r(b_0))\Omega(r(b_0 b_1)) \cdots$, i.e., to the colors along the branch $r_{|b_0 b_1 b_2 \cdots}$ of r. Since this sequence satisfies the parity condition, r is an accepting run. Hence, it follows that also ρ satisfies the parity condition, i.e., σ is winning from q_I. □

Thus, we have shown that the emptiness problem for parity tree automata is decidable by constructing and then solving the emptiness game. The size of the resulting game is polynomial in the size of the automaton. One can also show that the problem of solving parity games can be reduced to the emptiness problem of parity tree automata, which have polynomial size in the number of vertices of the game (see

Exercise 6.12). Hence, these two problems are equivalent under polynomial-time reductions and therefore share their complexity theoretic status: they are both in NP $\cap$ Co-NP, but no polynomial-time algorithm is known for either problem.

After this detour, we continue with showing closure of parity tree automata under complement. Recall that we defined the acceptance game $\mathcal{G}(\mathscr{A}, t)$ which characterizes acceptance of t by $\mathscr{A}$, i.e., we have $t \in \mathcal{L}(\mathscr{A})$ if, and only if, $(\varepsilon, q_I) \in W_0(\mathcal{G}(\mathscr{A}, t))$. Thus, t is in the complement language of $\mathscr{A}$ if, and only if, $(\varepsilon, q_I) \notin W_0(\mathcal{G}(\mathscr{A}, t))$. Applying (positional) determinacy of parity games (which also holds true for countably infinite arenas, i.e., in particular for $\mathcal{G}(\mathscr{A}, t)$) yields that $(\varepsilon, q_I) \notin W_0(\mathcal{G}(\mathscr{A}, t))$ is equivalent to $(\varepsilon, q_I) \in W_1(\mathcal{G}(\mathscr{A}, t))$. Thus, a winning strategy for Player 1 witnesses that t is in the complement language. We will construct a parity tree automaton that recognizes a tree t and an encoding s of a strategy for Player 1 if, and only if, the strategy encoded by s is winning for him in $\mathcal{G}(\mathscr{A}, t)$. Then, we show that projecting away the second component encoding the strategy yields the desired automaton recognizing the complement language.

Note that the arena of the acceptance game is an infinite tree and that every Player 1 vertex has exactly two successors identified by the directions 0 and 1. First, we encode a strategy for Player 1 as a tree labeled by a finite alphabet, which can then be processed by an automaton. Due to positional determinacy of parity games in countable arenas, we only have to consider positional strategies. For Player 1, such a strategy τ has the form

$$\tau \colon \mathbb{B}^* \times \Delta \to \mathbb{B},$$

since Player 1's positions in $\mathcal{G}(\mathscr{A}, t)$ are of them form $(w, (q, a, q_0, q_1))$ and have two successors, $(w0, q_0)$ and $(w1, q_1)$, which we identify with their directions 0 and 1. Equivalently, applying currying, one can denote such a strategy τ as

$$\tau \colon \mathbb{B}^* \to (\Delta \to \mathbb{B}).$$

Since the set $\mathbb{B}^\Delta$ of functions from Δ to $\mathbb{B}$ is finite, τ is a $\mathbb{B}^\Delta$-labeled tree. We call an element from $\mathbb{B}^\Delta$ a local strategy, since it encodes Player 1's reaction to Player 0 picking a certain transition from Δ. This is local in the sense that it does not take the position w, which is also encoded in the vertices, into account.

A tree $\tau \colon \mathbb{B}^* \to \mathbb{B}^\Delta$ is referred to as a strategy tree. Furthermore, we call a strategy tree winning for t, if it encodes a winning strategy for Player 1 in the game $\mathcal{G}(\mathscr{A}, t)$. By using the characterization of acceptance via $\mathcal{G}(\mathscr{A}, t)$ and positional determinacy, we can express non-acceptance of t by the existence of a winning strategy tree.

Remark 6.1. *$t \notin \mathcal{L}(\mathscr{A})$ if, and only if, there is a winning strategy tree for t.*

Next, we construct a parity tree automaton recognizing pairs of trees t and s such that s is winning for t. To this end, we first construct a word automaton $\mathscr{M}$ that checks for every branch π and every strategy for Player 0, whether the parity condition of $\mathscr{A}$ is satisfied if Player 1 picks the branch π. If this is the case, s is not winning. Thus, we are ultimately interested in the complement language, but the language described above is simpler to encode by an automaton.

Recall that $\mathscr{A} = (Q, \Sigma, q_I, \Delta, \Omega)$ is the parity tree automaton we want to complement. We define the non-deterministic parity word automaton[15] $\mathscr{M} = (Q, \Sigma', q_I, \Delta', \Omega)$ where the set of states, the initial state, and the coloring are as in $\mathscr{A}$, with $\Sigma' = \mathbb{B}^\Delta \times \Sigma \times \mathbb{B}$ and with $(q, (f, a, b), q') \in \Delta'$ if, and only if, there is a transition $\tau = (q, a, q_0, q_1) \in \Delta$ of the tree automaton such that $f(\tau) = b$ and $q' = q_b$. Note that the state q and the letter a appear in both transitions and the direction b appearing in $\mathscr{M}$'s transition is equal to $f(\tau)$. Intuitively, the non-determinism of $\mathscr{M}$ simulates Player 0 picking a transition τ that is applicable at state q and letter a (as she does in $\mathcal{G}(\mathscr{A}, t)$) and Player 1's reaction to τ, as encoded by the local strategy f, is the direction $b = f(\tau)$, which leads the automaton $\mathscr{M}$ from state q to state q_b.

Fix some tree $t \colon \mathbb{B}^* \to \Sigma$ and a strategy tree $s \colon \mathbb{B}^* \to \mathbb{B}^\Delta$ and let $\pi = b_0 b_1 b_2 \cdots \in \mathbb{B}^\omega$ be a branch. We define the word $w(s, t, \pi)$ by

$$w(s,t,\pi) = (s(\varepsilon), t(\varepsilon), b_0)\,(s(b_0), t(b_0), b_1)\,(s(b_0b_1), t(b_0b_1), b_2)\,(s(b_0b_1b_2), t(b_0b_1b_2), b_3) \cdots \in (\Sigma')^\omega$$

and the define the word language $L(s, t) \subseteq (\Sigma')^\omega$ by

$$L(s,t) = \{w(s,t,\pi) \mid \pi \in \mathbb{B}^\omega\} \ .$$

Thus, $L(s, t)$ contains the labels of s and t along the branch π, together with the branch itself.

Using these definitions, we can characterize s being a winning strategy tree for t in terms of the automaton $\mathscr{M}$.

[15]See Exercise 4.5

Lemma 6.2. *The strategy tree s is winning for t if, and only if, $L(s,t) \cap \mathcal{L}(\mathscr{M}) = \emptyset$.*

Proof. First, let s be winning for t and assume towards a contradiction that there is a branch $\pi = b_0 b_1 b_2 \cdots$ such that $w(s,t,\pi)$ is accepted by $\mathscr{M}$, say with run $r = q_0 q_1 q_2 \cdots$. Then, for every $j \geq 0$,

$$(q_j, (s(b_0 \cdots b_{j-1}), t(b_0 \cdots b_{j-1}), b_j), q_{j+1})$$

is a transition of $\mathscr{M}$. By definition of $\mathscr{M}$, $s(b_0 \cdots b_{j-1}) = f$ satisfies $f(\tau_j) = b_j$ for some transition

$$\tau_j = (q_j, t(b_0 \cdots b_{j-1}), q'_0, q'_1)$$

such that $q_{j+1} = q'_{b_j}$.

The transitions τ_j and the strategy for Player 1 encoded by s induce a play ρ in $\mathcal{G}(\mathscr{A}, t)$. The sequence of colors visited by ρ is the same one as the sequence of colors of the run r of $\mathscr{M}$ on $w(s,t,\pi)$, except that each occurrence of a color is doubled. As r is accepting, ρ is winning for Player 0, which yields the desired contradiction to s being winning.

For the other direction, let $L(s,t) \cap \mathcal{L}(\mathscr{M}) = \emptyset$. We show that an arbitrary play ρ that is consistent with the strategy encoded by s is winning for Player 1, which proves that s is winning for t.

For every $j \in \mathbb{N}$, let $(w_j, (q_j, t(w_j), q'_0, q'_1))$ be the unique Player 1 vertex visited by ρ where the first component is of length j. Since w_{j+1} is obtained from w_j by appending a single bit, the w_j induce an infinite branch $\pi = b_0 b_1 b_2 \cdots$ such that $w_j = b_0 \cdots b_{j-1}$.

Consider the sequence $r = q_0 q_1 q_2 \cdots$ of states appearing in the first components of the transitions. A straightforward induction proves that r is a run of $\mathscr{M}$ on $w(s,t,\pi)$. This run is rejecting, since $w(s,t,\pi) \in L(s,t)$ and therefore $w(s,t,\pi) \notin \mathcal{L}(\mathscr{M})$.

Furthermore, the play ρ has the same sequence of colors as r, except that every occurrence of a color is doubled. Hence, the play does not satisfy the parity condition and is therefore winning for Player 1. $\square$

Note that sequences in $\mathcal{L}(\mathscr{M})$ are good for Player 0, since they satisfy the acceptance condition of $\mathscr{A}$. Hence, we are actually interested in the complement language $(\Sigma')^\omega \setminus \mathcal{L}(\mathscr{M})$. We use the following result about parity word automata without proof[16].

Theorem 6.3. *Parity word automata can be determinized and are closed under complement.*

Thus, there is a deterministic parity word automaton $\mathscr{S} = (Q', \Sigma', q'_I, \delta', \Omega')$ with deterministic transition function $\delta' \colon Q' \times \Sigma' \to Q'$ such that $\mathcal{L}(\mathscr{S}) = (\Sigma')^\omega \setminus \mathcal{L}(\mathscr{M})$. By simulating this automaton along all the branches of s and t we can check whether $L(s,t) \cap \mathcal{L}(\mathscr{M}) = \emptyset$. To this end, we define the combined tree $t^\frown s \colon \mathbb{B}^* \to \Sigma \times \mathbb{B}^\Delta$ by $t^\frown s(w) = (t(w), s(w))$.

Now, we define the parity tree automaton $\mathscr{B} = (Q', \Sigma \times \mathbb{B}^\Delta, q'_I, \Delta', \Omega')$ where the set of states, the initial state, and the coloring are as in $\mathscr{S}$ and where $(q, (a,f), q_0, q_1) \in \Delta'$ if, and only if, $\delta'(q, (f, a, b)) = q_b$ for both $b \in \mathbb{B}$. Thus, $\mathscr{B}$ indeed simulates $\mathscr{S}$ along every branch of $t^\frown s$. Next, we show that $\mathscr{B}$ recognizes winning strategies for Player 1 in $\mathcal{G}(\mathscr{A}, t)$. Note that this simulation is only possible because $\mathscr{S}$ is deterministic.

Lemma 6.3. *The strategy tree s is winning for t if, and only if, $t^\frown s \in \mathcal{L}(\mathscr{B})$.*

Proof. Let s be winning for t. Thus, by Lemma 6.2, we have $L(s,t) \cap \mathcal{L}(\mathscr{M}) = \emptyset$ and therefore $L(s,t) \subseteq \mathcal{L}(\mathscr{S})$. Hence, for every branch π, we have $w(s,t,\pi) \in \mathcal{L}(\mathscr{S})$.

For a sequence $w = b_0 \cdots b_j \in \mathbb{B}^*$, let $r(w)$ be the unique state that $\mathscr{S}$ reaches while processing

$$(s(\varepsilon), t(\varepsilon), b_0)\,(s(b_0), t(b_0), b_1)\,(s(b_0 b_1), t(b_0 b_1), b_2) \cdots (s(b_0 b_1 \cdots b_{j-1}), t(b_0 b_1 \cdots b_{j-1}), b_j)\ .$$

As $\mathscr{S}$ is deterministic, r is a run of $\mathscr{B}$ on $t^\frown s$. Furthermore, $r_{|\pi}$ is equal to the unique run r' of $\mathscr{S}$ on $w(s,t,\pi)$. As $w(s,t,\pi)$ is accepted by $\mathscr{S}$, r' and therefore also $r_{|\pi}$ satisfy the parity condition. As π is an arbitrary branch, this implies that r is accepting, i.e., $t^\frown s \in \mathcal{L}(\mathscr{B})$.

For the other direction, let r be an accepting run of $\mathscr{B}$ on $t^\frown s$. Let π be an arbitrary branch. As before, we have that $r_{|\pi}$ is equal to the unique run r' of $\mathscr{S}$ on $w(s,t,\pi)$. As $r_{|\pi}$ satisfies the parity condition, so does r'. Hence, $w(s,t,\pi)$ is accepted by $\mathscr{S}$ and therefore not in $\mathcal{L}(\mathscr{M})$. As we picked π arbitrarily, we obtain $L(s,t) \cap \mathcal{L}(\mathscr{M}) = \emptyset$, which implies that s is winning for t by Lemma 6.2. $\square$

[16]For a proof, see, e.g., Erich Grädel, Wolfgang Thomas, Thomas Wilke: Automata, Logics, and Infinite Games: A Guide to Current Research [outcome of a Dagstuhl seminar, February 2001]. Lecture Notes in Computer Science 2500, Springer 2002

Now, we can finish our argument by applying closure of parity tree automata under projection (see Exercise 6.6), i.e., our final automaton $\mathscr{B}'$ runs on t, guesses a strategy tree s, and uses $\mathscr{B}$ to verify that s is winning for t.

Theorem 6.4. *Parity tree automata are closed under complement.*

Proof. Let $\mathscr{B}'$ recognize the projection of $\mathcal{L}(\mathscr{B})$ to the first component. Then, $t \in L(\mathscr{B}')$ if, and only if, there exists a strategy tree s such that $t^\frown s \in \mathcal{L}(\mathscr{B})$. The latter statement is equivalent to s being winning for t. Thus, $t \in \mathcal{L}(\mathscr{A})$ if, and only if, there is a winning strategy tree for t. Thus, due to Lemma 6.1, $\mathscr{B}'$ recognizes the complement of $\mathcal{L}(\mathscr{A})$. ☐

After having proved closure of parity tree automata under complement, we are now in a position to translate S2S into automata. To this end, we first simplify S2S by eliminating syntactic sugar and first-order quantification, obtaining the logic $\mathrm{S2S}_0$. In particular, first-order quantification is simulated by second-order quantification of singletons. In order to do so, we need new atomic formulas that replace the atomic formulas of S2S to deal with these singleton sets.

Definition 6.13 (Syntax of $\mathrm{S2S}_0$). *Let $X, Y \in \mathcal{V}_2$ be second-order variables. The* atomic formulas *of $\mathrm{S2S}_0$ are*

- $\mathrm{Sing}(X)$,
- $X \subseteq P_a$ *for* $a \in \Sigma$,
- $X \subseteq Y$, *and*
- $S_i(X, Y)$ *for* $i \in \mathbb{B}$.

Furthermore, every atomic formula is a formula *of $S2S_0$. If φ and ψ are formulas and $X \in \mathcal{V}_2$ is a second-order variable, then $\neg\varphi$, $\varphi \vee \psi$, and $\exists X \varphi$ are formulas as well.*

Intuitively, $\mathrm{Sing}(X)$ expresses that X is a singleton, $X \subseteq P_a$ expresses that letter a is at every position contained in X, while $S_i(X, Y)$ holds if $X = \{x\}$ and $Y = \{y\}$ are singletons and y is the i-successor of x. Formally, we define the semantics as follows.

Definition 6.14 (Semantics of $\mathrm{S2S}_0$). *The* semantics of $\mathrm{S2S}_0$ *are defined recursively by the satisfaction relation $\models$. Given a tree t and a variable valuation μ, we have*

- $t, \mu \models \mathrm{Sing}(X)$ *if, and only if,* $|\mu(X)| = 1$,
- $t, \mu \models X \subseteq P_a$ *if, and only if,* $t(w) = a$ *for every* $w \in \mu(X)$,
- $t, \mu \models X \subseteq Y$ *if, and only if,* $\mu(X) \subseteq \mu(Y)$
- $t, \mu \models S_i(X, Y)$ *if, and only if,* $\mu(X) = \{w\}$ *and* $\mu(Y) = \{wi\}$ *for some* $w \in \mathbb{B}^*$,
- $t, \mu \models \neg\varphi$ *if, and only if, it is not the case that* $t, \mu \models \varphi$,
- $t, \mu \models \varphi \vee \psi$ *if, and only if,* $t, \mu \models \varphi$ *or* $t, \mu \models \psi$, *and*
- $t, \mu \models \exists X \varphi$ *if, and only if,* $t, \mu[X \mapsto B] \models \varphi$ *for some* $B \subseteq \mathbb{B}^*$.

Now, we can show that we can turn every S2S-sentence into an equivalent $\mathrm{S2S}_0$-sentence.

Lemma 6.4. *For every S2S-sentence φ there is an $S2S_0$ sentence φ' such that for every tree t: $t \models \varphi$ if, and only if, $t \models \varphi'$.*

Proof. First, we showed in Exercise 6.1 that ε and $\sqsubseteq$ are syntactic sugar and can be eliminated. Hence, we assume that these symbols do not appear in φ, which implies that every term in φ is a first-order variable. Now, we replace every occurrence of a universally quantified subformula $\forall x \psi$ by the equivalent formula $\neg\exists x \neg\psi$ respectively $\forall X \psi$ by $\neg\exists X \neg\psi$. Also, we replace conjunctions by disjunctions using De Morgans's law.

After these rewriting steps, we define φ' by induction over the construction of S2S-formulas using the remaining atomic formulas, boolean connectives, and existential quantification.

- $(x = y)' = \mathrm{Sing}(X) \wedge \mathrm{Sing}(Y) \wedge X \subseteq Y \wedge Y \subseteq X$,

- $(P_a(x))' = X \subseteq P_a$,
- $(S_i(x, y))' = S_i(X, Y)$,
- $(x \in Y)' = \mathrm{Sing}(X) \wedge X \subseteq Y$,
- $(\neg\varphi)' = \neg(\varphi')$,
- $(\varphi \vee \psi)' = \varphi' \vee \psi'$,
- $(\exists x \varphi)' = \exists X\, (\mathrm{Sing}(X) \wedge \varphi')$, and
- $(\forall X \varphi)' = \exists X \varphi'$.

Here, one has to ensure that the newly introduced second-order variables simulating the first-order variables are fresh, i.e., not already appearing in φ to avoid "variable capturing".

Note that this rewriting introduces new conjunctions, which can again be replaced by disjunctions using De Morgan's law. The resulting formula is then in $\mathrm{S2S}_0$ and a straightforward induction shows that it is indeed equivalent. □

Thus, it suffices to show how to translate $\mathrm{S2S}_0$ into parity tree automata. To this end, we have to deal with free variables, which were assigned meaning by a variable valuation. We encode this valuation by a tree. Formally, fix an $\mathrm{S2S}_0$ formula φ with free variables $X_0, \ldots, X_{n-1}$ and a variable valuation μ. We define the tree $t_\mu \colon \mathbb{B}^* \to \mathbb{B}^n$ via $t_\mu(w) = (b_0, \ldots, b_{n-1})$ where

$$b_j = \begin{cases} 0 & \text{if } w \notin \mu(X_j), \\ 1 & \text{if } w \in \mu(X_j). \end{cases}$$

Theorem 6.5. *For every $\mathrm{S2S}_0$-formula φ there is a parity tree automaton $\mathscr{A}_\varphi$ such that $t, \mu \models \varphi$ if, and only if, $t^\frown t_\mu \in \mathcal{L}(\mathscr{A}_\varphi)$ for every tree t and every variable valuation μ.*

In particular, if φ is a sentence, then we have $t \models \varphi$ if, and only if, $t \in \mathcal{L}(\mathscr{A}_\varphi)$.

Proof. We construct the automata $\mathscr{A}_\varphi$ by induction over the structure of φ. The automata for the atomic formulas are straightforward:

- The automaton $\mathscr{A}_{\mathrm{Sing}(X_j)}$ has to verify that there is a single position in the input tree whose label has a 1 in the component encoding X_j. This can be done by guessing a finite branch to such a position and requiring that every other position has a 0 in this component. The parity condition is used to ensure that the guessed branch eventually finds a 1.

- The automaton $\mathscr{A}_{X_j \subseteq P_a}$ works similarly as $\mathscr{A}_{\mathrm{Sing}(X_j)}$, but additionally checks that the position with a 1 is labeled by a in t.

- The automaton $\mathscr{A}_{X_j \subseteq X_{j'}}$ has a single state that allows every label except for those where the component encoding X_j has a 1, but the component encoding $X_{j'}$ has a 0.

- Finally, the automaton $\mathscr{A}_{S_i(X_j, X_{j'})}$ also guesses a branch to a position where the component encoding X_j has a 1, checks that the i-successor has a 1 in the component encoding $X_{j'}$ as well, and checks that there is exactly one 1 in each of these two components in the input tree.

It remains to consider the boolean connectives and existential quantification.

- $\mathscr{A}_{\neg\varphi}$ is the automaton recognizing the complement of $\mathcal{L}(\mathscr{A}_\varphi)$, see Theorem 6.4.

- $\mathscr{A}_{\varphi \vee \psi}$ is the automaton recognizing $\mathcal{L}(\mathscr{A}_\varphi) \cup \mathcal{L}(\mathscr{A}_\psi)$, see Exercise 6.6.

- $\mathscr{A}_{\exists X_j \varphi}$ is the automaton recognizing the projection of $\mathcal{L}(\mathscr{A}_\varphi)$ to all components but the one encoding X_j, see Exercise 6.6. □

In conclusion, we can now prove Rabin's Theorem: satisfiability of S2S sentences is decidable.

Theorem 6.6. *The following problem is decidable: Given an S2S sentence φ, is φ satisfiable?*

Proof. Given φ, first construct an equivalent S2S$_0$ sentence φ', then the equivalent parity tree automaton $\mathscr{A}_{\varphi'}$, and then the emptiness game $\mathcal{G}(\mathscr{A}_{\varphi'})$. Due to the equivalences, this game is won by Player 0 from the vertex representing the initial state of $\mathscr{A}_{\varphi'}$, if and only if, φ is satisfiable. As the emptiness game can be effectively constructed from φ and can be solved, the satisfiability problem is decidable by a reduction to solving parity games. □

The proof of Rabin's Theorem presented here can straightforwardly be extended to infinite trees of higher arities: instead of considering binary trees one deals with trees where every node has n successors, for some fixed arity $n > 0$. Hence, the corresponding logic, called SnS, has n successor relations $S_0, \ldots, S_{n-1}$ and one proves it to be effectively equivalent to parity tree automata working on n-ary trees. Such automata have transitions of the form $(q, a, q_0, \ldots, q_{n-1})$ sending state q_j to the j-th successor. All results proven in this section for the case $n = 2$ can be generalized to the case for arbitrary n, in particular the translation of logic into automata and the emptiness game for these automata.

Corollary 6.1. *SnS satisfiability is decidable for every $n > 0$.*

The case of trees with countably infinite branching, i.e., where one considers trees of the form $\mathbb{N}^* \to \Sigma$, is slightly more involved. Now, the resulting logic, called SωS has a successor relation S_j for every $j \in \mathbb{N}$, although each formula can only use a finite number of them. The automata-based approach cannot be adapted as before, as this entails using sending a state to each successor, i.e., each transition is an infinite object. Such automata are in general not suitable for algorithms.

Instead, one "interprets" the underlying structure $\mathbb{N}^*$ with the successor relations $(S_j)_{j\in\mathbb{N}}$ in a binary tree with domain $\{0,1\}^*$ and relations S_0 and S_1. The roots of both trees are identified and the j-th successor j of the root ε is identified by the node 0^j1. In general, the node $j_0j_1\ldots j_k$ of $\mathbb{N}^*$ is mapped to $0^{j_0}1\,0^{j_1}1\cdots0^{j_k}1$ of $\{0,1\}^*$.Thus, the j-th successor relation corresponds to taking S_0 j times and then S_1 once. Note that the image of the function mapping $\mathbb{N}^*$ to $\{0,1\}^*$ is the set $D = \{\varepsilon\} \cup \{w1 \mid w \in \{0,1\}^*\}$, which is characterized by the formula

$$\varphi_D = x = \varepsilon \vee \exists y S_1(y, x)$$

with a single free variable x. Similarly, the j-th successor relation is characterized by the formula

$$\varphi_{S_j} = \exists y_1 \cdots \exists y_j S_0(x, y_1) \wedge \bigwedge_{k=1}^{j-1} S_0(x_k, x_{k+1}) \wedge S_1(x_j, x')$$

for $j > 0$ and $\varphi_{S_0} = S_1(x, x')$, each with free variables x and x', which express that x' is the j-th successor of x.

Now, given an SωS sentence φ, one rewrites it by "guarding" each quantification with the domain formula φ_D, i.e., one replaces $\exists x\psi$ by $\exists x(\varphi_D(x) \wedge \psi)$ and $\forall x\psi$ by $\forall x(\varphi_D(x) \to \psi)$. Furthermore, one replaces each occurrence of S_j by the formula phi_{S_j}, i.e., each atomic formula $S_j(x, x')$ by $\varphi_{S_j}(x, x')$. Call the resulting S2S sentence φ'. Then, one can show that φ is satisfiable if, and only if, φ' is satisfiable, by translating models for φ into models of φ' and vice versa.

Corollary 6.2. *SωS satisfiability is decidable.*

Similarly, one can show decidability of monadic second-order logic on total orders that are interpretable in the infinite binary tree.

6.5 Exercises

Exercise 6.1. Show that ε and $\sqsubseteq$ are syntactic sugar.

1. Give a formula $\varphi_\varepsilon(x)$ not containing ε and $\sqsubseteq$ with one free first-order variable x and no free second-order variable such that $t, \mu \models \varphi_\varepsilon$ if, and only if, $\mu(x) = \varepsilon$.

2. Give a formula $\varphi_\preceq(x, y)$ not containing $\preceq$ and ε with two free first-order variables x and y and no free second-order variable such that $t, \mu \models \varphi_\preceq$ if, and only if, $\mu(x)$ is a prefix of $\mu(y)$.

Exercise 6.2. Give S2S formulas defining the following tree languages over the alphabet $\Sigma = \{a, b, c\}$:

1. The language of trees containing an a-labeled vertex whose left subtree contains a b-labeled vertex and whose right subtree contains a c-labeled vertex.

2. The language of trees t satisfying $t_{|0^\omega} \in (aa)^* b^\omega$.

3. The language of trees containing at least one a-labeled vertex and at least one b-labeled vertex.

Exercise 6.3. Let L be the language of all trees over $\Sigma = \{a, b\}$ containing only finitely many a-labeled vertices.

1. Give an S2S formula defining L.
2. Give a parity tree automaton recognizing L.

In both cases, argue informally that your solution is correct.

Exercise 6.4. Show that the subset-relation and “being connected“ is expressible in S2S.

1. Give a formula $\varphi_\subseteq(X, Y)$ with two free second-order variables X and Y and no free first-order variable such that $t, \mu \vDash \varphi_\subseteq$ if and only if $\mu(X) \subseteq \mu(Y)$.
2. Give a formula $\varphi_c(X)$ with one free second-order variable X and no free first-order variable such that $t, \mu \vDash \varphi_c$ if, and only if, $\mu(X)$ is connected, i.e., if w and w' are in $\mu(X)$ and w' is a descendant of w, then all vertices on the path between w and w' are in $\mu(X)$, as well.

Exercise 6.5. Give parity tree automata recognizing the languages from Exercise 6.2.

Exercise 6.6. Show that languages recognized by parity tree automata are closed under union and projection.

1. Given two parity tree automata $\mathscr{A}_1$ and $\mathscr{A}_2$ over the same alphabet construct a parity tree automaton $\mathscr{A}$ such that $\mathcal{L}(\mathscr{A}) = \mathcal{L}(\mathscr{A}_1) \cup \mathcal{L}(\mathscr{A}_2)$. Show formally that your automaton recognizes $\mathcal{L}(\mathscr{A}_1) \cup \mathcal{L}(\mathscr{A}_2)$.
2. Given a tree $t\colon \mathbb{B}^* \to \Sigma \times \Gamma$ we define its projection $p_\Sigma(t)\colon \mathbb{B}^* \to \Sigma$ to its first component by $p_\Sigma(t)(w) = a$ for every $w \in \mathbb{B}^*$ with $t(w) = (a, b)$.

   Given a parity tree automaton $\mathscr{A}_e$ over the alphabet $\Sigma \times \Gamma$ construct a parity tree automaton $\mathscr{A}$ such that $\mathcal{L}(\mathscr{A}) = \{p_\Sigma(t) \mid t \in \mathcal{L}(\mathscr{A}_e)\}$. Show formally that your automaton recognizes this language.

Exercise 6.7. A parity tree automaton $(Q, \Sigma, q_I, \Delta, \Omega)$ is deterministic, if for every state q and every letter a there is at most one pair (q_0, q_1) of states such that $(q, a, q_0, a_1) \in \Delta$. Thus, a deterministic automaton has a unique run on every tree.

Let L be the language of trees over that alphabet $\{a, b\}$ that contain at least one a.

1. Show that there is a parity tree automaton that recognizes L.
2. Show that there is no deterministic parity tree automaton that recognizes L.

Exercise 6.8. Prove the left-to-right implication of Lemma 6.1: if $t \in \mathcal{L}(\mathscr{A})$, then $(\varepsilon, q_I) \in W_0(\mathcal{G}(\mathscr{A}, t))$.

Exercise 6.9. Prove the right-to-left implication of Lemma 6.2: if $q_I \in W_0(\mathcal{G}(\mathscr{A}))$, then $\mathcal{L}(\mathscr{A}) \neq \emptyset$.

Exercise 6.10. Consider the parity tree automaton $\mathscr{A} = (\{q_0, q_1, q_2, q_3, q_4\}, \{a, b\}, q_0, \Delta, \Omega)$ where Δ and Ω are defined by $\Omega(q_3) = 1$, $\Omega(q_0) = \Omega(q_4) = 2$, $\Omega(q_1) = \Omega(q_2) = 3$, and

$$\Delta = \left\{ \begin{array}{ccc} & q_0,a & \\ \diagup & & \diagdown \\ q_0 & & q_1 \end{array},\ \begin{array}{ccc} & q_0,b & \\ \diagup & & \diagdown \\ q_0 & & q_3 \end{array},\ \begin{array}{ccc} & q_1,a & \\ \diagup & & \diagdown \\ q_3 & & q_0 \end{array},\ \begin{array}{ccc} & q_1,b & \\ \diagup & & \diagdown \\ q_0 & & q_2 \end{array},\ \begin{array}{ccc} & q_2,a & \\ \diagup & & \diagdown \\ q_0 & & q_3 \end{array},\ \begin{array}{ccc} & q_2,b & \\ \diagup & & \diagdown \\ q_0 & & q_1 \end{array},\right.$$

$$\left. \begin{array}{ccc} & q_2,b & \\ \diagup & & \diagdown \\ q_0 & & q_4 \end{array},\ \begin{array}{ccc} & q_3,a & \\ \diagup & & \diagdown \\ q_0 & & q_3 \end{array},\ \begin{array}{ccc} & q_3,b & \\ \diagup & & \diagdown \\ q_0 & & q_3 \end{array},\ \begin{array}{ccc} & q_4,a & \\ \diagup & & \diagdown \\ q_0 & & q_1 \end{array},\ \begin{array}{ccc} & q_4,b & \\ \diagup & & \diagdown \\ q_3 & & q_0 \end{array} \right\}.$$

1. Construct the emptiness game $\mathcal{G}(\mathscr{A})$ and determine the winner from (ε, q_0).
2. Give a tree $t \in \mathcal{L}(\mathscr{A})$ as function $t\colon \mathbb{B}^* \to \{a, b\}$.
3. Give a precise description of $\mathcal{L}(\mathscr{A})$ using natural language.

Exercise 6.11. A parity tree automaton $(Q, \Sigma, q_I, \Delta, \Omega)$ is deterministic, if for every state q and every letter a there is exactly one pair (q_0, q_1) of states such that $(q, a, q_0, a_1) \in \Delta$. Thus, a deterministic automaton has a unique run on every tree.

Let L be the language of trees over that alphabet $\{a, b\}$ that contain at least one a.

1. Show that there is a parity tree automaton that recognizes L.
2. Show that there is no deterministic parity tree automaton that recognizes L.

Exercise 6.12. In Theorem 6.2, we proved that the emptiness problem for parity tree automata $\mathscr{A}$ is reducible to solving parity games of polynomial size in $|\mathscr{A}|$.

Prove the converse, i.e., show that for every parity game $\mathcal{G}$ and vertex v of $\mathcal{G}$, there is a parity tree automaton $\mathscr{A}_{\mathcal{G},v}$ such that $v \in W_0(\mathcal{G})$ if, and only if, $\mathcal{L}(\mathscr{A}_{\mathcal{G},v}) \neq \emptyset$. Furthermore, $|\mathscr{A}_{\mathcal{G},v}|$ should be polynomial in $|\mathcal{G}|$.

A Appendix: Basic Definitions and Notations

Here, we introduce our notation and some basic definitions.

A.1 Basic Notations

We use the symbol $\mathbb{N}$ to denote the set of non-negative integers and $\mathbb{N}^+$ to denote the set of positive integers. Arbitrary elements of $\mathbb{N}$ are abbreviated by lowercase Latin letters, preferably j, m, and n. For every $n \in \mathbb{N}$ we define $[n] = \{0, 1, \dots, n-1\}$. In particular, $[0] = \emptyset$. The parity of an integer $n \in \mathbb{N}$ is denoted by $\mathrm{Par}(n)$, i.e.,

$$\mathrm{Par}(n) = \begin{cases} 0 & \text{if } n \text{ is even,} \\ 1 & \text{if } n \text{ is odd.} \end{cases}$$

To denote sets, we use uppercase Latin letters like P, Q, and S. The cardinality of a set S is denoted by $|S|$ and its power set by 2^S.

A.2 Alphabets and Words

An alphabet is a non-empty, finite set of symbols, usually denoted by Σ. The elements of an alphabet are called letters.

The concatenation $w = w_0 w_1 \cdots w_{n-1}$ of finitely many letters of Σ is a finite word over Σ. The length n of w is denoted by $|w|$. The only word of length 0 is the empty word which is denoted by ε. For a non-empty finite word w, $\mathrm{Lst}(w)$ denotes the last letter of w.

The concatenation of infinitely many letters is an infinite word , i.e., a word of infinite length. We usually use lowercase Latin letters like w to denote finite words and lowercase Greek letters like α, β, γ to denote infinite words. If we just talk about words, we mean either finite or infinite words and fall back to denoting them by lowercase Latin letters. The set of all finite words over Σ is denoted by Σ^* and the set of all infinite words by Σ^ω. For $\Sigma^* \setminus \{\varepsilon\}$ we use the shortcut Σ^+.

Like letters, we can concatenate a finite word w with a finite or infinite words w' to form a new word $w'' = ww'$. For such a word $w = w'w''$ we call w' a prefix of w and w'' a suffix. If a word w is a prefix of a word w' we denote this by $w \sqsubseteq w'$ and use $\mathrm{Prefs}(w)$ to denote the set of all prefixes of w. In particular, $w \sqsubseteq w$ for all $w \in \Sigma^*$. The reversal of a finite word w is denoted by w^R, i.e., $(w_0 \cdots w_n)^R = w_n \cdots w_0$.

For a finite or infinite word $w = w_0 w_1 w_2 \cdots$ we define

$$\mathrm{Occ}(w) := \{a \in \Sigma \mid w_n = a \text{ for some } n\}$$

to be the set of letters occurring in a word w, also called the occurrence set of w. Also, we define

$$\mathrm{Inf}(w) := \{a \in \Sigma \mid w_n = a \text{ for infinitely many } n\}$$

to be the set of letters occurring infinitely often in the word w, called the infinity set of w. The infinity set of a finite word is always empty.

A.3 Regular Languages

If Σ is an alphabet then each subset of Σ^* is a language over finite words and each subset of Σ^ω is a language over infinite words. The concatenation of two languages $K \subseteq \Sigma^*$ and $M \subseteq \Sigma^*$ or $M \subseteq \Sigma^\omega$ is denoted by KM and defined as $KM := \{ww' \mid w \in K \text{ and } w' \in M\}$.

To specify languages over finite words we use regular expressions, which are given by the following grammar, where a ranges over Σ:

$$r \ := \ \emptyset \mid \varepsilon \mid a \mid r + r \mid rr \mid r^*$$

Additionally, we use r^+ as a shortcut for rr^*. The language $\mathcal{L}(r)$ of a regular expression r is defined using the following semantics. Languages definable this way are called regular.

- $\mathcal{L}(\emptyset) := \emptyset$
- $\mathcal{L}(\varepsilon) := \{\varepsilon\}$
- $\mathcal{L}(a) := \{a\}$
- $\mathcal{L}(r_1 r_2) := \mathcal{L}(r_1)\mathcal{L}(r_2)$
- $\mathcal{L}(r_1 + r_2) := \mathcal{L}(r_1) \cup \mathcal{L}(r_2)$
- $\mathcal{L}(r^*) := \{w_0 \cdots w_{n-1} \mid n \in \mathbb{N} \text{ and } w_j \in \mathcal{L}(r) \text{ for every } j\}$. [17]

[17] Here, the empty concatenation is defined to be ε.

To define languages over infinite words we use ω-regular expressions, given by the grammar:

$$e \;:=\; e + e \mid rr^{\omega}$$

Here, r ranges over the regular expressions as defined above. The language $\mathcal{L}(e)$ of an ω-regular expression e is defined using the following semantics. Languages definable this way are called ω-regular.

- $\mathcal{L}(e_1 + e_2) = \mathcal{L}(e_1) \cup \mathcal{L}(e_2)$
- $\mathcal{L}(r_1(r_2)^{\omega}) := \mathcal{L}(r_1)\{w_0w_1w_2\cdots \mid w_j \in \mathcal{L}(r_2) \text{ for every } j \in \mathbb{N}\}$

As usual, we often drop the operator $\mathcal{L}$ assigning a language to an expression and instead identify a regular expression with its language, i.e., $(b^*a)^{\omega}$ is both an ω-regular expression and the language of infinite words over $\{a, b\}$ with infinitely many a's.

A.4 Fixed-points and the Knaster-Tarski Theorem

Fix a set L. A binary relation $\leq$ over L is a *partial order*, if it is

- reflexive: $a \leq a$ for all $a \in L$,
- antisymmetric: $a \leq b$ and $b \leq a$ imply $a = b$ for all $a, b \in L$, and
- transitive: $a \leq b$ and $b \leq c$ imply $a \leq c$ for all $a, b, c \in L$.

In this case, we call $(L, \leq)$ a partially ordered set (*poset* for short). We write $a \geq b$ if, and only if, $b \leq a$.

Let $L' \subseteq L$ be a subset of L. An element $a \in L$ is a *lower bound* of L' if $a \leq a'$ for all $a' \in L'$. A lower bound $a \in L$ is an *infimum* of L', if $b \leq a$ for every lower bound b of L', i.e., a is a greatest lower bound of L'. Upper bounds and suprema are defined dually: an element $a \in L$ is an *upper bound* of L' if $a \geq a'$ for all $a' \in L'$. An upper bound $a \in L$ is a *supremum* of L', if $b \geq a$ for every upper bound b of L', i.e., a is a least upper bound of L'. Note that neither infima nor suprema may not always exist, but are unique if they do.

A *complete lattice* is a poset in which every subset $L' \subseteq L$ has an infimum and a supremum. A complete lattice is never empty, e.g., it contains an infimum of the empty set. Furthermore, the only infimum of the empty set is the least element of L, an element a such that $a \leq b$ for every $b \in L$.

A function $f\colon L \to L$ is monotonic, if $a \leq b$ implies $f(a) \leq f(b)$. A fixed-point of f is an element a with $f(a) = a$ and a pre-fixed-point of f is an element a with $f(a) \leq a$.

Theorem A.1 (Knaster-Tarski)**.** *Let L be a complete lattice and $f\colon L \to L$ monotonic. Then, the set of fixed-points of f is a complete lattice.*

In particular, every monotonic function f on a complete lattice has a least fixed-point, i.e., a fixed-point a satisfying $a \leq b$ for every other fixed-point b of f.

If L is finite, then the least fixed-point can be computed as follows: let $\bot$ be the minimal element of L and consider the sequence

$$\bot = f^0(\bot) \leq f^1(\bot) \leq f^2(\bot) \leq \cdots .$$

As L is finite, there has to be an $n \leq |L|$ with $f^n(\bot) = f^{n+1}(\bot)$.

Lemma A.1. *$e =: f^n(\bot)$ is the least fixed-point of f.*

Proof. First, we show that e is indeed a fixed-point:

$$e = f^n(\bot) = f^{n+1}(\bot) = f(f^n(\bot)) = f(e).$$

Thus, it remains to show that e is the least fixed-point, i.e., $e \leq a$ for every fixed-point a of f. Fix one such fixed-point a. We prove $e = f^n(\bot) \leq a$ by showing $f^j(\bot) \leq a$ by induction over j. The induction start $f^0(\bot) = \bot \leq a$ follows from $\bot$ being the least element of L. Now, assume $f^j(\bot) \leq a$. Then,

$$f^{j+1}(\bot) = f(f^j(\bot)) \leq f(a) \leq a$$

where the first inequality follows from monotonicity of f and the second one from a being a fixed-point. □

Furthermore, the least fixed-point is also the least pre-fixed-point, which can be proven by the same argument presented in the second part of the proof of Lemma A.1: there, we only used the fact $f(a) \leq a$, i.e., that a is a pre-fixed-point.