Infinite Games

Lecture Notes

Winter Term 13/14 – Saarland University

by **Martin Zimmermann** and **Felix Klein**

(In case of questions, errors or suggestions for improvements please send a mail to: *fklein@cs.uni-saarland.de*)

Contents

1 Basic Definitions and Notations

In this section we introduce our notation and some basic definitions. Its main purpose is to provide a reference for everything that is unfamiliar to you. Nevertheless, we recommend you to read this section at least once to know what is in there. Consider that there are some definitions that are first used in later lectures. So don't be confuesed if you cannot put them into any context at the moment.

1.1 Integers and Sets

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 small roman letters, preferably n, m and j. For all $n, m \in \mathbb{N}$ with $n \leq m$ we use $[n, m]$ to denote the set $\{n, n+1, ..., m\}$ and shorten the special case of $[0, n-1]$ by $[n]$. Consider that $[0] = \emptyset$ and $[\infty] = \mathbb{N}$. To talk about the parity of an integer $n \in \mathbb{N}$ we use the function Par with

$$\mathrm{Par}(n) = \begin{cases} 0 & \text{if } n \text{ is even} \\ 1 & \text{if } n \text{ is odd} \end{cases}$$

To denote sets, we use big roman letters like S, P and Q. The cardinality of a set S is denoted by $|S|$ where for infinite sets S we define $|S|$ to be ∞. The power set of S is denoted by 2^S.

1.2 Functions, Tuples and Relations

To denote functions we use small roman letters, mostly f, g and h. For a function $f : A \to B$ and $C \subseteq A$ we use $f(C)$ as a shorthand for the set $\{\, b \in B \mid c \in C \wedge f(c) = b \,\}$.

If X is a set and $n \in \mathbb{N}$ we use X^n to denote the set of n-ary tuples over X and number its components from 0 to $n-1$. By $X^{>n}$ we denote the set $\bigcup_{j=n+1}^{\infty} X^j$ and the function pr_j for $j \in \mathbb{N}$ can be used to project to the j-th component of a tuple of arbitrary length. If the tuple has no such component pr_j is undefined. To denote concrete tuple instances, we use round brackets, e.g. $(1, 2, 3) \in \mathbb{N}^3$.

The relations $<, \leq, =, >, \geq$ over $\mathbb{N}$ are defined as usual. For $n \in \mathbb{N}$ and $a, b \in \mathbb{N}^n$ we use the lexicographical ordering to compare a and b, e.g. $(0, 3, 2) < (1, 2, 0)$ and $(1, 1, 2) > (1, 1, 1)$. Finally, with $<_j, \leq_j, =_j, \geq_j, >_j$ for $j \in \mathbb{N}$, we can compare two tuples $a, b \in \mathbb{N}^{>j}$ by lexicographically comparing them from component 0 to j, e.g. $(1, 2, 3, 4) <_2 (1, 2, 4)$ and $(7, 3, 5) \geq_1 (7, 3, 9)$.

1.3 Alphabets and Words

An alphabet is a non-empty, finite set of symbols, usually denoted by Σ or Υ. The elements of an alphabet are called letters. Let an alphabet Σ be given, then the concatenation $w = w_0 w_1 ... w_{n-1}$ of finitely many letters of Σ is called a finite word over Σ, where n defines the length of w also denoted by $|w|$. The only word of length 0 is the empty word denoted by ε.

The concatenation of infinitely many letters defines an infinite word which has infinite length. We usually use small roman letters like w to denote finite words and small greek letters like α, β, γ denote infinite words. If we just talk about words, we mean either finite or infinite words and fall back to denote them by small roman 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^+. Given some word w we have that for all $n \in [|w|]$ the n-th letter of w is denoted by w_n and the first letter is at w_0. In case w is finite and non-empty we use $\mathrm{Lst}(w)$ to extract the last letter of w.

Like letters, we can concatenate finite words w with words w' to new words $w'' = ww'$. For 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 $\mathrm{Pref}(w)$ to denote the set of all prefix of w.

For $n, m, j \in \mathbb{N}$ and $n \leq m$ we also use $w[j]$ to denote the unique prefix $w' \in \mathrm{Pref}(w)$ with $|w| = j+1$ and $w^{>j}$ to denote the unique suffix of w with $w = w[j]w^{>j}$ and use $w[n, m]$ as shortcut for $w[m]^{>n-1}$. Finally, for a word w over Σ and $a \in \Sigma$ we use $\mathrm{Idx}(w, a) := \{\, n \in \mathbb{N} \mid w_n = a \,\}$ to denote the set of all indices n of a word w with an a at index n. This set is also called the index set of w and a. Further, we use $\mathrm{Occ}(w) := \{\, a \in \Sigma \mid \exists n \in \mathbb{N}.\ w_n = a \,\}$ to denote the set of all letters occurring in a word w, also called the occurrence set of w, and $\mathrm{Inf}(w) := \{\, a \in \Sigma \mid \forall n \in \mathbb{N}.\ \exists m \in \mathbb{N}.\ m \geq n \wedge w_m = a \,\}$ to denote the set of all letters occurring infinitely often in the word w, called the infinity set of w.

1.4 Languages and Regular Expressions

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 $M \subseteq \Sigma^* \cup \Sigma^\omega$ is denoted by KM and defined as $KM := \{ ww' \mid w \in K \wedge w' \in M \}$. To define languages over finite words we can use regular expressions. To construct a regular expression we can use the following grammar for arbitrary $a \in \Sigma$:

$$R \;:=\; \emptyset \mid \varepsilon \mid a \mid RR \mid R + R \mid R^*$$

Additionally, we use R^+ as a shortcut for RR^*. From a regular expression R to a language $\mathcal{L}(R)$ we get by using the semantics given in the following. Languages definable this way will be called regular languages.

- $\mathcal{L}(\emptyset) := \emptyset$
- $\mathcal{L}(\varepsilon) := \{\varepsilon\}$
- $\forall a \in \Sigma.\ \mathcal{L}(a) := \{a\}$
- $\mathcal{L}(R_1 R_2) := \mathcal{L}(R_1)\mathcal{L}(R_2)$
- $\mathcal{L}(R_1 + R_2) := \mathcal{L}(R_1) \cup \mathcal{L}(R_2)$
- $\mathcal{L}(R^*) := \{ w_0 w_1 ... w_n \in \Sigma^* \mid \exists p_0 p_1 ... p_m \in \mathbb{N}^*.\ \forall j \in [m].$
  $p_0 = 0 \wedge p_m = n \wedge p_j \leq p_{j+1} \wedge w[p_j, p_{j+1}] \in \mathcal{L}(R) \}$

To define languages over infinite words we use ω-regular expressions, given by the grammar:

$$E \;:=\; RR^\omega \mid E + E$$

Here, R refers to the already defined grammar of regular expressions. Finally, the semantics for the new grammatical elements E is given by:

- $\mathcal{L}(E_1 + E_2) = \mathcal{L}(E_1) \cup \mathcal{L}(E_2)$
- $\mathcal{L}(R_1 (R_2)^\omega) := \mathcal{L}(R_1) \{ w_0 w_1 w_2 ... \in \Sigma^\omega \mid \exists p_0 p_1 p_2 ... \in \mathbb{N}^\omega.\ \forall j \in \mathbb{N}.$
  $p_0 = 0 \wedge p_j \leq p_{j+1} \wedge w[p_j, p_{j+1}] \in \mathcal{L}(R) \}$

As for regular expressions we call languages definable this way ω-regular languages. Consider that we assume for these definitions that Σ is always given and fixed. However, to make things more readable, we often just infer Σ from the context, meaning that each single symbol in an (ω-)regular expression not defined somewhere else and unequal to ε and $\emptyset$ is assumed to be in Σ. This way giving the (ω-)regular expression is enough to properly define the language. To provide a more readable notation we will use big roman letters for variables and small letters for all other symbols. Sometimes we will also drop the $\mathcal{L}(\ )$ part if it is clear that we talk about a language.

2 Introduction

Many of todays problems in computer science are no longer concerned with programs that transform data and then terminate, but with non-terminating systems which have to interact with a possibly antagonistic environment. An example is the controller regulating the anti-lock braking system in a car. It receives a constant input stream of sensor readings like wheel speed of each wheel and selectively applies the brakes on one wheel to maintain a uniform wheel speed. Thus, we are not interested in whether the terminator terminates and generates correct 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.

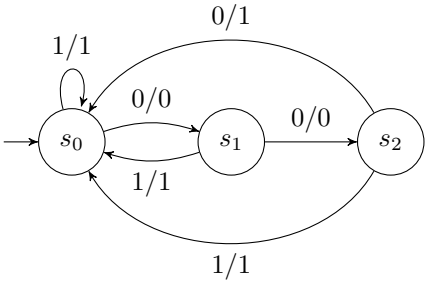
The emergence of so-called reactive systems requires new approaches to verification and synthesis. Over the course of the last fifty years it turned out to be very fruitful to model and analyze reactive systems in a game-theoretic framework, which captures the antagonistic and strategic nature of the 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 automaton with output in the following where the label 1/1 stands for "process a 1 and output a 1".



An example run of the automaton 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 will see how to model such a synthesis problem as a two-player game and how to compute a finite-state representation of a strategy satisfying the requirements.

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.

In this framework, 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). Hence, the size of the automaton implementing the winning strategy influences the size of the synthesized controller for the reactive system.

Controllers for reactive systems can be synthesized by solving infinite games, which amounts to determining for every vertex the player who has a winning strategy and to compute such a strategy. The computational complexity of solving a game and the size of finite-state winning strategies for a game are influenced by the expressiveness and succinctness of the formalism employed to specify the winning condition.

2.1 Course Overview

This course is divided into an introduction and three main parts, each dealing with one aspect of infinite games.

In the introduction, we give all necessary definitions to reason about infinite games and introduce basic winning conditions based on acceptance conditions for automata on infinite objects: reachability and safety, Büchi and co-Büchi, and parity.

In the first main part, 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 will discuss the tight relation between finite-state strategies and game reductions, the classical way to compute finite-state strategies.

In the second main part, 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 last part, 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 will introduce monadic second-order logic and parity tree automata, which are equally expressive. Infinite games are used to prove that such automata are closed under negation (which is on step we need to do 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.

2.2 Arenas, Strategies and Games

First we define the main structure of a game: its arena. The idea of an arena is to describe the rules the two players have to follow and additionally, the order in which the players make their moves. To get an intuition of what an arena looks like, consider the graphical example in Figure 1.

In general an arena is a directed graph consisting of vertices and edges. To assign the players, the vertices are partitioned into two classes, the vertices of Player 0 and the vertices of Player 1. Graphically, we distinguish this assignment by using round vertices for Player 0 (round as the symbol for zero) and angled vertices for Player 1 (angled as the symbol for one). We also say the players own these vertices.

To play on an arena, we assume the existence of a token on one of the vertices. If the vertex is owned by Player 0, she has to move the token, otherwise Player 1 can 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, has the possibility to choose one of them.

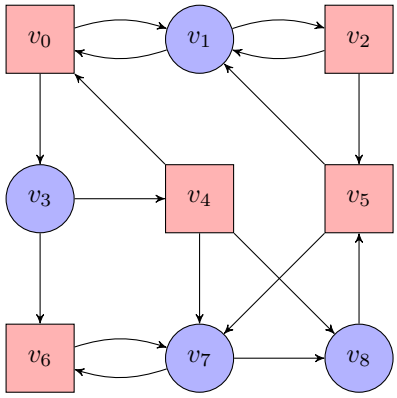


Figure 1: Graphical example for an arena. The round vertices describe the vertices owned by Player 0, the angled ones the vertices owned by Player 1. The edges in between describe the moves the players can do.

Finally, since we want to play games with infinite behaviour, we only consider arenas that have at least one outgoing edge for each vertex. This way, we ensure that the token cannot end in a deadlock after some time. We summarize these mentioned ideas in the following formal definition.

Definition 2.1. An *arena* $\mathcal{A} = (V, V_0, V_1, E)$ is a tuple where

- V is a finite set of vertices,
- $V_0 \subseteq V$ is the set of vertices owned by Player 0,
- $V_1 = V \setminus V_0$ is the set of vertices owned by Player 1,
- $E \subseteq V \times V$ is a set of directed edges

and each vertex $v \in V$ has at least one outgoing edge, formally: $\forall v \in V.\ \exists v' \in V.\ (v, v') \in E$. The size of $\mathcal{A}$, denoted by $|\mathcal{A}|$, is defined to be $|V|$.

Sometimes, we only want to consider some part of an arena like in the example the part only consisting of the vertices v_4, v_5, v_7 and v_8 and the edges between these vertices (Figure 2).

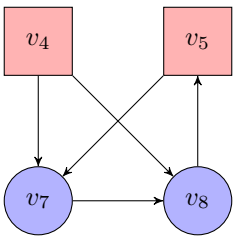


Figure 2: Sub-arena of the arena given in Figure 1.

We then also call this part a sub-arena of the original arena. As the vertices, the arena is restricted to, are enough to clearly describe the corresponding sub-arena we can introduce the following definition for a sub-arena. Consider that we have to ensure that the resulting arena is valid, meaning there is an outgoing edge for each remaining vertex.

Definition 2.2. Let $\mathcal{A} = (V, V_0, V_1, E)$ be an arena and $V' \subseteq V$ such that every vertex in V' has a successor vertex in V'. The *sub-arena of* $\mathcal{A}$, 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 continue by adding some further useful definitions. We already have seen that the players move a token through the arena. If we do that forever, these moves define an infinite sequence of vertices the token visits. Such an infinite sequence we also call a play on the arena. An example for a play is given in Figure 3.

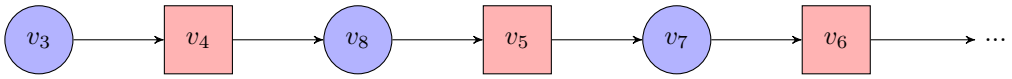


Figure 3: Some play on the arena of Figure 1 starting in v_3.

Consider that a play is always an infinite object and depends on the initial vertex the token is placed on. Formally, we get:

Definition 2.3. A *play* on an arena $\mathcal{A} = (V, V_0, V_1, E)$ is an infinite sequence $\rho = \rho_0\rho_1\rho_2... \in V^\omega$ such that for all $n \in \mathbb{N}$ holds $(\rho_n, \rho_{n+1}) \in E$. We say a play starts in vertex v iff $v = \rho_0$. We denote the set of all possible plays on $\mathcal{A}$ with $\text{Plays}(\mathcal{A})$ and the set of all possible plays starting in vertex v with $\text{Plays}(\mathcal{A}, v)$.

Coming along with the definition of a play is the definition of a strategy which fixes all possible decisions of a player regarding the moves already made. On the one hand, these decisions depend on the structure of the arena, on the other hand they also depend on the decisions of the other player. Accordingly, a player should have the possibility to react differently, if the other player chooses a different outgoing edge in the previous move. We say that a strategy may depend on the history, where the history fixes the finite prefix of a play that has already been played. To describe this formally, we define a strategy of a player as a function that gets the actual history of moves and then chooses the next outgoing edge. This way we fix all possible behaviors of that player without considering the strategy of the other player. If a play then results from using this strategy, we call it consistent with the strategy. Consider that there can by multiple plays that are consistent with the strategy of one of the players but just one play that is consistent with both strategies. We formally put these ideas into the following definitions.

Definition 2.4. A *strategy* for Player i in an arena $\mathcal{A} = (V, V_0, V_1, E)$ is a function $\sigma : V^*V_i \to V$ such that whenever $\sigma(wv) = v'$ then also $(v, v') \in E$.

By convention we use i to denote an arbitrary player. The strategies for an arbitrary Player i is denoted by σ, the strategies of the corresponding other Player $i - 1$ is denoted by τ. If we talk about concrete players we use σ for Player 0 and τ for Player 1.

Definition 2.5. A play ρ on an arena $\mathcal{A} = (V, V_0, V_1, E)$ is *consistent* with a strategy σ in $\mathcal{A}$ iff for all $n \in \mathbb{N}$ with $\rho_n \in V_i$ we have that $\sigma(\rho[n]) = \rho_{n+1}$. We denote the set of all plays consistent with σ and starting in some vertex $v \in V$ with $\text{Plays}(\mathcal{A}, \sigma, v)$.

Note that for every arena $\mathcal{A}$, every strategy σ and τ and every initial vertex v there exists only one play that is consistent with τ and σ or formally $|\text{Plays}(\mathcal{A}, \sigma, v) \cap \text{Plays}(\mathcal{A}, \tau, v)| = 1$. An example for a strategy of Player 0 that is consistent with the play given in Figure 3 is indicated below. Consider that we indeed only fix the decisions of Player 0.

w	v_3	$v_3v_4v_8$	$v_3v_4v_8v_5v_7$	$v_3v_4v_8v_5v_7v_6v_7$	...
$\sigma(w)$	v_4	v_5	v_6	v_6	...

One might observe that our definition of a strategy is too expressive for practical usage, since it may have to fix infinitely many different decisions. Accordingly, we may be interested in some weaker representations. One of them is the class of a positional strategies. Here, the strategy is not allowed to depend on the actual history, meaning at each vertex the strategy chooses always the same successor. Consequently, when using a positional strategy, a player has always to chose the same outgoing edge if the token is in a vertex he owns. The formal definition is given in the following. Later in the lecture, we will also see some more complex representations that arise if we additionally allow the usage of some finite memory.

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

As a positional strategy for a player i is practically equivalent to a function σ just mapping vertices owned by player i to some successor vertex (formally: $f : V_i \to V$) we usually will express them as such functions. Nevertheless, technically we still mean a strategy in the original sense, since otherwise notions like consistency of plays are not properly defined anymore.

We now are able to define the notion of a game. So far, the players are only able to play infinitely long on an arena, but there is no notion of when a player is winning that play. Therefore, we start with a very general characterization. Regarding the set of all possible plays, we just specify a subset of this set and say Player 0 is winning the play if it is in the set. We get the following definition of a game.

Definition 2.7. A *game* $\mathcal{G} = (\mathcal{A}, \mathrm{Win})$ is a tuple containing an arena $\mathcal{A}$ and a set of winning plays $\mathrm{Win} \subseteq \mathrm{Plays}(\mathcal{A})$. We call a play ρ winning for Player 0 iff $\rho \in \mathrm{Win}$ and winning for Player 1 otherwise.

As an example consider the game $\mathcal{G}_e = (\mathcal{A}, \mathrm{Win})$ where $\mathcal{A} = (V, V_0, V_1, E)$ is defined by the arena depicted in Figure 1 and the winning condition is defined as $\mathrm{Win} = \{ \rho \in \mathrm{Plays}(\mathcal{A}) \mid \mathrm{Occ}(\rho) \neq V \}$. A play is winning for Player 0 in this game iff at least one of the vertices of the arena is not visited. Next, we characterize when a strategy is winning or not:

Definition 2.8. Let $\mathcal{G} = (\mathcal{A}, \mathrm{Win})$ be a game with $\mathcal{A} = (V, V_0, V_1, E)$ and σ be a strategy for Player i on $\mathcal{A}$. The strategy σ is a *winning strategy* from vertex $v \in V$ for Player i iff every play $\rho \in \mathrm{Plays}(\mathcal{A}, v)$ consistent with σ is winning for Player i.

Consider that we defined the notion of winning strategies per initial vertex. This way, we can introduce the notion of a winning region.

Definition 2.9. The *winning region* $W_i(\mathcal{G})$ of a game $\mathcal{G} = (\mathcal{A}, \mathrm{Win})$ with $\mathcal{A} = (V, V_0, V_1, E)$ is defined for Player i as the set of vertices $v \in V$ for which there exists a winning strategy starting from vertex v for Player i.

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 for all vertices. She just has always to take the edges $(v_1, v_0), (v_3, v_4)$ and (v_7, v_8) if in the vertices v_1, v_3 and v_7 correspondingly. This way she avoids visiting the vertices v_2 and v_6 and since at least two vertices can be avoided, starting a play in one of them cannot help. Consequently, we have for the winning regions: $W_0 = V$ and $W_1 = \emptyset$. Further note that this is a positional strategy.

Remark 2.1. *For all games $\mathcal{G}$ it holds that $W_0(\mathcal{G}) \cap W_1(\mathcal{G}) = \emptyset$.*

Proof. Let $\mathcal{G} = (\mathcal{A}, \mathrm{Win})$ with $\mathcal{A} = (V, V_0, V_1, E)$. Suppose that there exists $v \in W_0(\mathcal{G}) \cap W_1(\mathcal{G}) \neq \emptyset$. By Definition 2.9 there exists a strategy σ winning for Player 0 from v and a strategy τ winning for Player 1 from v. Let $\rho = \rho_0\rho_1\rho_2...$ be the single, unique element of $\mathrm{Plays}(\mathcal{A}, \sigma, v) \cap \mathrm{Plays}(\mathcal{A}, \tau, v)$. It follows by Definition 2.8 that ρ is winning for Player 0 and ρ is winning for Player 1. By Definition 2.7 it follows that $\rho \in \mathrm{Win}$ and $\rho \notin \mathrm{Win}$. This is a contradiction against our supposition concluding the proof. □

We already have seen that the winning regions of the two players are always disjoint. Accordingly, there is no vertex in the arena that can be in both winning regions at the same time. Another interesting question is whether it is possible that a vertex is in none of the two regions, or in other words whether the winning regions build a partition of the vertex set. This property is also known as determinacy of a game and defined as follows:

Definition 2.10. A game $\mathcal{G} = (\mathcal{A}, \text{Win})$ with $\mathcal{A} = (V, V_0, V_1, E)$ is *determined* iff the winning regions of both players describe a partition of V.

Consider that, in contrast to Remark 2.1, we introduce determinacy as a definition here. As a consequence, one can reason that not all games are determined in general. However, the construction of a game that is not determined is a complicated task and only from theoretical interest. Correspondingly, we just want to suggest the interested reader the book *Classical descriptive set theory* by *Alexander Kechris* here.

We conclude this section with two further definitions. First we extend the idea of positional strategies to games, where we require that in every vertex one of the two players can win the game with a positional strategy. Consider that this definition implies determinacy of the game which is why we call it positional determinacy.

Definition 2.11. A game $\mathcal{G} = (\mathcal{A}, \text{Win})$ with $\mathcal{A} = (V, V_0, V_1, E)$ is *positionally determined* iff for every vertex $v \in V$ there exists a positional winning strategy σ for some Player i.

Further, we introduce the idea of a uniform winning strategy. In contrast to usual winning strategies a uniform winning strategy can be used no matter in which vertex the corresponding play starts in.

Definition 2.12. Let the game $\mathcal{G} = (\mathcal{A}, \text{Win})$ with $\mathcal{A} = (V, V_0, V_1, E)$ be given and σ be a strategy for Player i on $\mathcal{A}$. The strategy σ is a *uniform winning strategy* iff it is winning from every vertex in $W_i(\mathcal{G})$.

Consider that not for every positionally determined game there exists a uniform positional winning strategy (Exercise 1.4).

2.3 Reachability Games

We continue by analyzing some common types of games, where we start with reachability games. Here, we are interested whether Player 0 is able to move the token into a specific area of the arena, where an area is just a set of vertices. We call this condition the reachability condition, since we ask whether the token can reach that area. The set of vertices, from which at least one vertex should be reached, is called the reachability set. Accordingly, a reachability game then consists of an arena and a reachability set, usually denoted by R. Formally we have:

Definition 2.13. Let the reachability condition $\textsc{reach}(R)$ on a set $R \subseteq V$ for an arena $\mathcal{A} = (V, V_0, V_1, E)$ be defined as:

$$\textsc{reach}(R) := \{\, \rho \in \text{Plays}(\mathcal{A}) \mid \text{Occ}(\rho) \cap R \neq \emptyset \,\}$$

Then we call the game $\mathcal{G} = (\mathcal{A}, \textsc{reach}(R))$ a *reachability game* with reachability set R.

An example for a reachability game is given in Figure 4, where we graphically denote the reachability set by using doubly framed vertices. In this game $\mathcal{G}_r$ Player 0 has to reach either v_4 or v_5. This is possible from each $v \in W_0(\mathcal{G}_r) = \{v_3, v_4, v_5, v_6, v_7, v_8\}$ as the following uniform winning strategy σ shows.

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

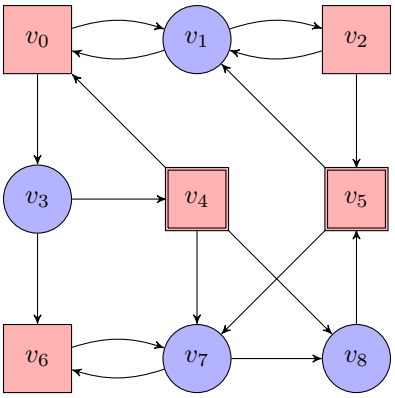


Figure 4: Example for a reachability game. We use doubly framed vertices to denote the reachability set.

We can already make some general observations from this strategy. For example if we start in a vertex of the reachability set, like in vertex v_5, we already have won so it does not matter which successor we or our opponent choose. We also win if we start in a vertex next to the reachability set that we control, like vertex v_8. We then just can move into the reachability set and have won in the next step.
But how can we win if we do not control a vertex next to the reachability set? In general there are two possibilities. Either Player 1 has no other choice than moving into the reachability set or he can avoid moving there. In the first case, we can derive that we also win from that vertex, in the second case we cannot derive anything so far.

It remains to argue about the remaining vertices that cannot reach the reachability set in one step, like vertex v_7. The idea is to recursively use the arguments made already before. If we own the vertex and we can move into a vertex where we already have argued that we will win from that vertex, we will also win from the predecessor vertex. If our opponent owns that vertex, we only can win if he is forced to move to vertices where we know that we will win.

This general construction of arguing about the winning regions is called the attractor construction and can be formally described as follows.

Construction 2.1. Let an arena $\mathcal{A} = (V, V_0, V_1, E)$ be given. The *attractor construction* on $\mathcal{A}$ is defined for each Player i, for all $n \in \mathbb{N}$ and $R \subseteq V$ as:

$$\begin{aligned}
\mathrm{CPre}_i(R) &= \{\, v \in V_i \mid \exists v' \in V.\ (v, v') \in E \wedge v' \in R \,\} \\
&\quad \cup \{\, v \in V_{1-i} \mid \forall v' \in V.\ (v, v') \in E \Rightarrow v' \in R \,\} \\
\mathrm{Attr}_i^0(R) &= R \\
\mathrm{Attr}_i^{n+1}(R) &= \mathrm{Attr}_i^n(R) \cup \mathrm{CPre}_i(\mathrm{Attr}_i^n(R)) \\
\mathrm{Attr}_i(R) &= \bigcup_{n \in \mathbb{N}} \mathrm{Attr}_i^n(R)
\end{aligned}$$

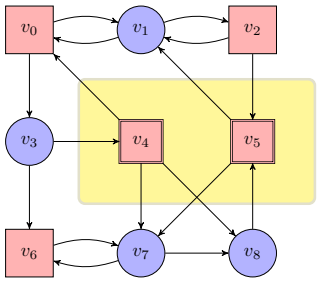
An execution of the attractor construction for the game in Figure 4 is depicted in Figure 5. Consider that $\mathrm{Attr}_0^3(\{v_4, v_5\}) = \mathrm{Attr}_0^4(\{v_4, v_5\})$ such that it follows that $\mathrm{Attr}_0^n(\{v_4, v_5\}) = \mathrm{Attr}_0^m(\{v_4, v_5\}) = \mathrm{Attr}_0(\{v_4, v_5\})$ for all $n, m \in \mathbb{N}$ greater than two. Consider that in general we always reach such a situation after at least $|\mathcal{A}|$ many steps, since we have $\mathrm{Attr}_i^0(R) \subseteq \mathrm{Attr}_i^1(R) \subseteq \mathrm{Attr}_i^2(R) \subseteq \ldots$.

Remark 2.2. *For every reachability game $\mathcal{G} = (\mathcal{A}, \textsc{reach}(R))$ and every Player i it holds:*

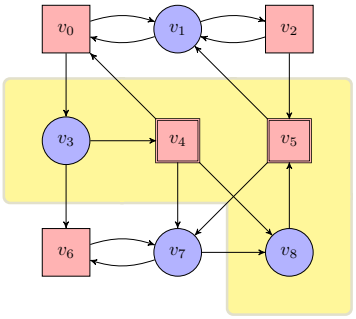
$$\forall n, m \geq |\mathcal{A}|.\ \mathrm{Attr}_i^n(R) = \mathrm{Attr}_i^m(R) = \mathrm{Attr}_i(R)$$

Remark 2.3. *The attractor $\mathrm{Attr}_i(R)$ on an arena $\mathcal{A} = (V, V_0, V_1, E)$ for some Player i and vertex set $R \subseteq V$ can be computed in linear time in $|E|$.*

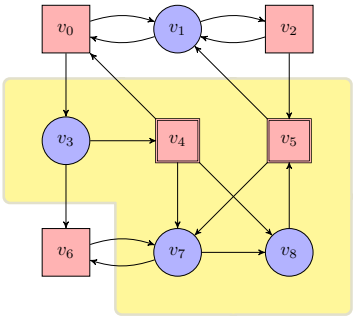
Proof. See Exercise 2.2. □



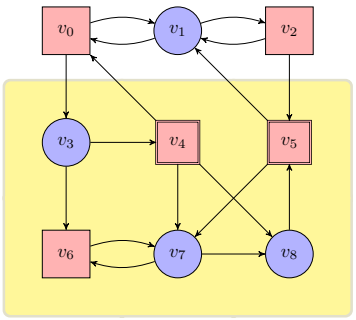
$\text{Attr}_0^0(\{v_4, v_5\}) = \{v_4, v_5\}$



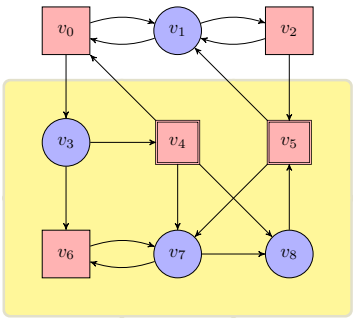
$\text{Attr}_0^1(\{v_4, v_5\}) = \{v_4, v_5\} \cup \{v_3, v_8\}$



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



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



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

Figure 5: Visual representation of the attractor construction for the game of Figure 4. The corresponding attractor sets are denoted by the areas marked yellow.

For our example game $\mathcal{G}_r$ holds that $W_0(\mathcal{G}_R)$ equals $\text{Attr}_0(R)$. We can generalize this result as follows.

Lemma 2.1. *For every reachability game* $\mathcal{G} = (\mathcal{A}, \text{REACH}(R))$ *with* $\mathcal{A} = (V, V_0, V_1, E)$ *it holds that* $W_0(\mathcal{G}) = \text{Attr}_0(R)$ *and* $W_1(\mathcal{G}) = V \setminus \text{Attr}_0(R)$.

Proof. To show the lemma we first show $\text{Attr}_0(R) \subseteq W_0(\mathcal{G})$. Therefore, we introduce the notion of a so called distance function. The distance function $\delta : \text{Attr}_0(R) \to \mathbb{N}$ for a reachability game $\mathcal{G} = (\mathcal{A}, \text{REACH}(R))$ with $\mathcal{A} = (V, V_0, V_1, E)$ is defined as

$$\delta(v) = min\{\, n \in \mathbb{N} \mid v \in \text{Attr}_0^n(R) \,\} \tag{1}$$

where $min\, \emptyset = \infty$. From Construction 2.1 we can derive the following properties for vertices $v \in \text{Attr}_0(R)$.

- $\delta(v) > 0 \wedge v \in V_0 \Rightarrow (\exists v' \in V.\ (v, v') \in E \wedge \delta(v') < \delta(v))$ (2)

  Let $\delta(v) = n$, then by (1) it follows that $v \in \text{Attr}_0^n(R)$ and $v \notin \text{Attr}_0^{n-1}(R)$. By construction it follows that $v \in \{\, v \in V_0 \mid \exists v' \in V.\ (v, v') \in E \wedge v' \in \text{Attr}_0^{n-1}(R) \,\} \subseteq \text{CPre}_0(\text{Attr}_0^{n-1}(R))$ so the corresponding v' exists and by (1) we finally get $\delta(v') \leq n - 1 < n$.

- $\delta(v) > 0 \wedge v \in V_1 \Rightarrow (\forall v' \in V.\ (v, v') \in E \Rightarrow \delta(v') < \delta(v))$ (3)

  Let $\delta(v) = n$, then by (1) it follows that $v \in \text{Attr}_0^n(R)$ and $v \notin \text{Attr}_0^{n-1}(R)$. By construction it follows that $v \in \{\, v \in V_1 \mid \forall v' \in V.\ (v, v') \in E \wedge v' \in \text{Attr}_0^{n-1}(R) \,\} \subseteq \text{CPre}_0(\text{Attr}_0^{n-1}(R))$ and by (1) we finally get for the corresponding v' that $\delta(v') \leq n - 1 < n$.

We are now able to construct a winning strategy σ for each play starting in $\text{Attr}_0(R)$.

$$\sigma(v) = \begin{cases} v' \text{ for some } (v,v') \in E \text{ with } \delta(v') < \delta(v) & \text{if } \infty > \delta(v) > 0 \\ v' \text{ for some } (v,v') \in E & \text{otherwise} \end{cases} \tag{4}$$

Consider that by (2) we can always construct such a strategy and that our strategy is positional. It remains to prove that σ is indeed a winning strategy. Let $\rho = \rho_0\rho_1\rho_2... \in \text{Plays}(\mathcal{A}, \sigma, \text{Attr}_0(R))$ be an arbitrary play consistent with σ and starting in $\text{Attr}_0(R)$. We show by induction on $\delta(\rho_0) = n$ that there exists an index m with $\rho_m \in R$.

Induction Base: $n = 0$

From $\delta(\rho_0) = 0$ it follows by (1) that $\rho_0 \in \text{Attr}_0^0(R)$ and by construction that $\rho_0 \in R$.

Induction Hypothesis:

$\forall n \in \mathbb{N}.\ \forall \rho \in \text{Plays}(\mathcal{A}, \sigma, \text{Attr}_0(R)).\ \delta(\rho_0) = n \Rightarrow \exists m \in \mathbb{N}.\ \rho_m \in R$

Induction Step: $n > 0$

If $\rho_0 \in V_0$ it follows by consistency that $\delta(\rho_1) < n$. The same follows by (3) if $\rho_0 \in V_1$. Then since $\rho_1 \in \text{Attr}_0(R)$ and since σ is positional we have the play $\rho' = \rho_1\rho_2\rho_3 \cdots \in \text{Plays}(\mathcal{A}, \sigma, \text{Attr}_0(R))$. By induction hypothesis there exists an $m' \in \mathbb{N}$ with $\rho'_{m'} \in R$. Correspondingly, for $m = m' + 1$ we have $\rho_m \in R$.

It follows that Player 0 has a winning strategy from each vertex $v \in \text{Attr}_0(R)$ and consequently $\text{Attr}_0(R) \subseteq W_0(\mathcal{G})$. Next, we show $V \setminus \text{Attr}_0(R) \subseteq W_1(\mathcal{G})$. Let $X = V \setminus \text{Attr}_0(R)$. From Construction 2.1 we get the following for arbitrary $v \in V$.

- $v \in (X \cap V_0) \Rightarrow \forall (v,v') \in E.\ v' \in X$ (5)

  Assume there exists a $v' \in V$ with $(v,v') \in E$ and $v' \notin X$. Then by definition it follows $v' \in \text{Attr}_0(R)$ and correspondingly, there exists an $n \in \mathbb{N}$ such that $v' \in \text{Attr}_0^n(R)$. But then by Construction 2.1 it follows that $v \in \text{Attr}_0^{n+1}(R)$ and also $v \in \text{Attr}_0(R)$. Accordingly, such a v' cannot exist.

- $v \in (X \cap V_1) \Rightarrow \exists (v,v') \in E.\ v' \in X$ (6)

  Assume that for all $v' \in V$ with $(v,v') \in E$ we have $v' \notin X$. Then by definition it follows $v' \in \text{Attr}_0(R)$ and there exists an $n \in \mathbb{N}$ such that $v' \in \text{Attr}_0^n(R)$. But then by construction it follows that $v \in \text{Attr}_0^{n+1}(R)$ and $v \in \text{Attr}_0(R)$ concluding that the corresponding v' exists.

We can now give a strategy τ for Player 1, defined arbitrarily for $v \notin X$, that wins from all vertices in X.

$$\tau(v) = v' \quad \text{for some } (v,v') \in E \text{ with } v' \in X \tag{7}$$

Consider that by (6) such a strategy can always be constructed and that τ is a positional strategy. We show that τ is indeed a winning strategy, so let $\rho = \rho_0\rho_1\rho_2... \in \text{Plays}(\mathcal{A}, \tau, X)$ be an arbitrary play. We show by induction on $n \in \mathbb{N}$ that all $\rho_n \in X$. By definitions of X it follows then also $\rho_n \notin R$ and that ρ is a winning play for Player 0.

Induction Base: $n = 0$

By definition $\rho_0 \in X$.

Induction Hypothesis:

$\forall n \in \mathbb{N}.\ \forall \rho \in \text{Plays}(\mathcal{A}, \tau, X).\ \rho_n \in X$

Induction Step: $n > 0$

By induction hypothesis we know that $\rho_{n-1} \in X$. We distinguish two cases. If $\rho_{n-1} \in V_0$ then by (5) it follows that each successor is an element of X, so also $\rho_n \in X$. If $\rho_{n-1} \in V_1$ then $\rho_n = \tau(\rho_{n-1}) \in X$ by (7) and consistency.

We have shown that $\text{Attr}_0(R) \subseteq W_0(\mathcal{G})$ and that $V \setminus \text{Attr}_0(R) \subseteq W_1(\mathcal{G})$. Together with the result of Remark 2.1 we finally get $W_0(\mathcal{G}) = \text{Attr}_0(R)$ and $W_1(\mathcal{G}) = V \setminus \text{Attr}_0(R)$. □

We can conclude the following theorem from the proof of Lemma 2.1, where solving a game is defined as computing the winning regions and the winning strategies from these regions for both players.

Theorem 2.1. *Reachability games are determined with uniform positional winning strategies. They can be solved in linear time in the number of edges of the underlying arena.*

Proof. We already have shown in the proof of Lemma 2.1 that a reachability game $\mathcal{G} = (\mathcal{A}, \textsc{reach}(R))$ with $\mathcal{A} = (V, V_0, V_1, E)$ is determined since $W_0(\mathcal{G}) = \text{Attr}_0(R)$ and $W_1(\mathcal{G}) = V \setminus \text{Attr}_0(R)$. Further, we also have seen that players only need positional strategies to win the game form the corresponding winning regions. Finally consider that the strategies, defined in the proof, are winning independent from which state in the winning regions the plays start. Altogether, it follows that the game is positionally determined with uniform strategies. For the given complexity results see Exercise 2.2. □

2.4 Safety Games

The next type of winning condition we want to analyze is the so called safety condition. Where for the reachability condition Player 0 is asked to reach a specific region of the arena, in the safety condition Player 0 is not allowed to leave a specific region. We also call this region the safe region of the arena. Formally we get the following game:

Definition 2.14. Let the safety condition $\textsc{safe}(S)$ on the set $S \subseteq V$ for an arena $\mathcal{A} = (V, V_0, V_1, E)$ be defined as:

$$\textsc{safe}(S) := \{\, \rho \in \text{Plays}(\mathcal{A}) \mid \text{Occ}(\rho) \subseteq S \,\}$$

Then we call the game $\mathcal{G} = (\mathcal{A}, \textsc{safe}(S))$ a *safety game* with safety set S.

An example for such a game is given in Figure 6. As for reachability games we mark the safe region of the game with double framed vertices. Here Player 0 can win exactly from the set $W_0 = \{v_1, v_2, v_5, v_7, v_8\}$ since in v_4 Player 1 can move to the unsafe vertex v_0 and from v_3 Player 0 can only move to v_6, which is unsafe, or v_4 from which we already have seen that Player 1 can move to an unsafe region. Correspondingly, $W_1 = \{v_0, v_3, v_4, v_5\}$.

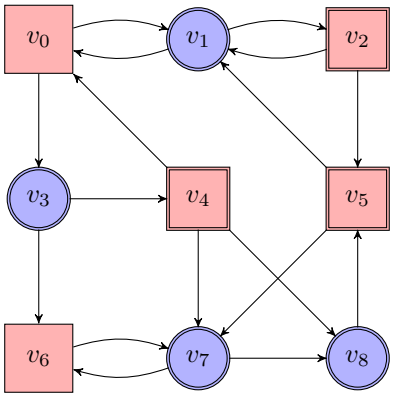


Figure 6: Example for a safety game. The doubly framed vertices mark the safe region.

As we can observe in the game above, the goal of both players exactly have swapped in comparison to reachability games. Player 1 now tries to reach the unsafe region of the arena and Player 0 has to avoid this. To generalize this observation we first introduce the notion of a complemented arena and finally get to the lemma given below.

Definition 2.15. For an arena $\mathcal{A} = (V, V_0, V_1, E)$ we define the *complemented arena* $\bar{\mathcal{A}}$ as:

$$\bar{\mathcal{A}} = (V, V_1, V_0, E)$$

Lemma 2.2. *Let $\mathcal{A} = (V, V_0, V_1, E)$ be an arena, $\mathcal{G} = (\mathcal{A}, \text{SAFE}(S))$ be a safety game with $S \subseteq V$ and $\mathcal{G}' = (\bar{\mathcal{A}}, \text{REACH}(V \setminus S))$ be a reachability game. Then every strategy σ winning for Player i from a vertex $v \in V$ in the game $\mathcal{G}'$ is also a winning strategy for Player $1 - i$ from v in the game $\mathcal{G}$.*

Proof. See Exercise 2.3. □

Corollary 2.1. *Let $\mathcal{A} = (V, V_0, V_1, E)$ be an arena, $\mathcal{G} = (\mathcal{A}, \text{SAFE}(S))$ be a safety game with $S \subseteq V$ and $\mathcal{G}' = (\bar{\mathcal{A}}, \text{REACH}(V \setminus S))$ be a reachability game. Then $W_i(\mathcal{G}) = W_{1-i}(\mathcal{G}')$ for each Player i.*

Proof. See Exercise 2.3. □

Finally we can conclude the following theorem.

Theorem 2.2. *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.*

Proof. The result directly follows from Corollary 2.1 and Theorem 2.1. □

2.5 Büchi Games

We already have seen games that are won by Player 0 or her opponent by reaching some set of vertices in the arena. After this goal is reached, it is clear who has won the game such that theoretically the players could stop playing the game.

We will now consider games where we overcome this. Therefore, we extend the condition of reaching a set once to reaching it infinitely often. Correspondingly, Player 0 not only has to find a strategy to reach the set but her strategy also has to be able to come back after every visit. This winning condition is called the Büchi condition and yields to the following game definition.

Definition 2.16. Let the Büchi condition $\text{BÜCHI}(F)$ on a set $F \subseteq V$ for an arena $\mathcal{A} = (V, V_0, V_1, E)$ be defined as:

$$\text{BÜCHI}(F) := \{\, \rho \in \text{Plays}(\mathcal{A}) \mid \text{Inf}(\rho) \cap F \neq \emptyset \,\}$$

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

We also call the vertices of the recurrence set accepting vertices. An example for a Büchi game is given by Figure 7. In this game, the goal of Player 0 is to visit v_4 or v_6 infinitely often. This is only achievable from the vertices v_3, v_6 and v_7 since Player 0 can cycle between v_6 and v_7 forever and move into this area from v_3. From the other vertices Player 1 can force the play into the set $\{v_0, v_1, v_2\}$ and can stay there by always moving back to v_1 from v_0 and v_2. Accordingly, we have $W_0 = \{v_3, v_6, v_7\}$ and $W_1 = \{v_0, v_1, v_2, v_4, v_5, v_8\}$.

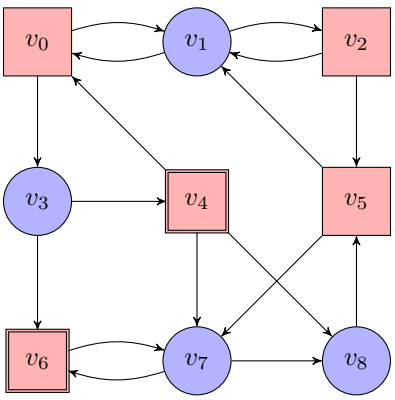
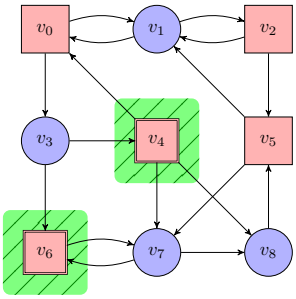
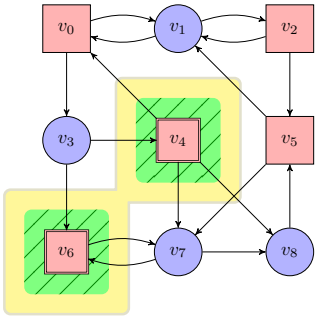


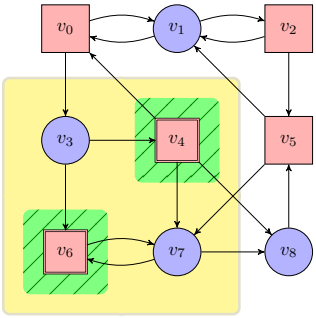
Figure 7: A Büchi game where the recurrence set is denoted by doubly framed vertices.



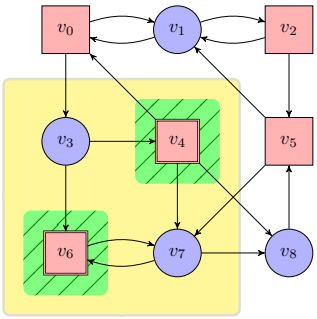
$\mathrm{Recur}^0(\{v_4, v_6\}) = \{v_4, v_6\}$



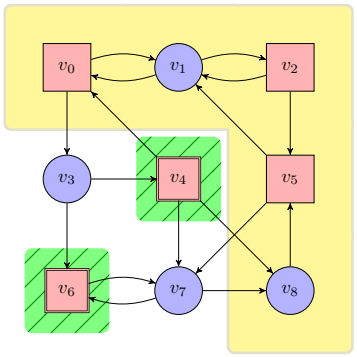
$\mathrm{Attr}_0^0(\{v_4, v_6\}) = \{v_4, v_6\}$



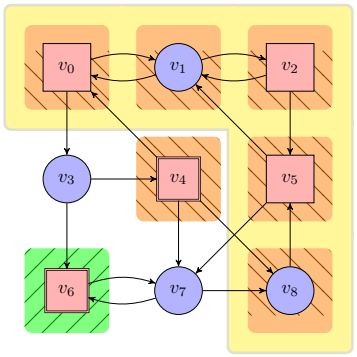
$\mathrm{Attr}_0^1(\{v_4, v_6\}) = \{v_4, v_6\} \cup \{v_3, v_7\}$



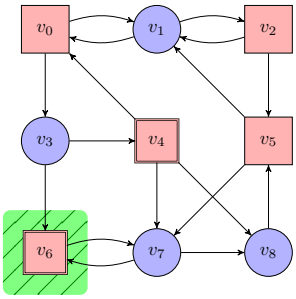
$\mathrm{Attr}_0(\{v_4, v_6\}) = \{v_3, v_4, v_6, v_7\}$



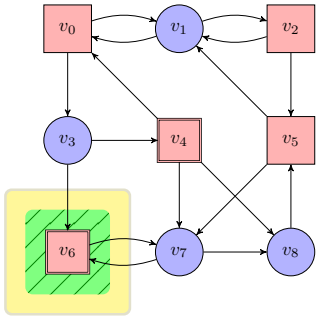
$W_1^0(\{v_4, v_6\}) = \{v_0, v_1, v_2, v_5, v_8\}$



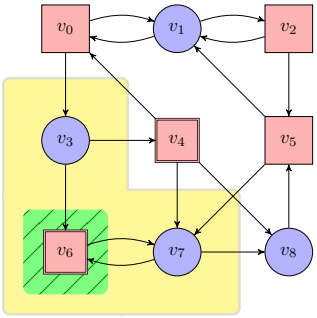
$\mathrm{CPre}_1(\{v_0, v_1, v_2, v_5, v_8\}) = \{v_0, v_1, v_2, v_4, v_5, v_8\}$



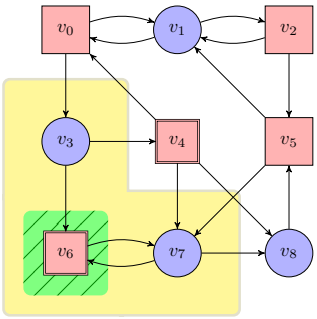
$\mathrm{Recur}^1(\{v_4, v_6\}) = \{v_6\}$



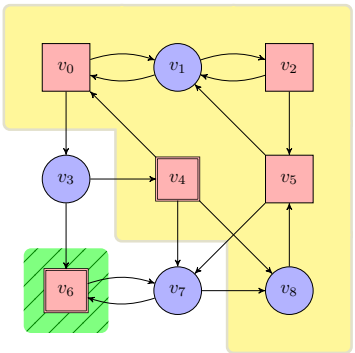
$\mathrm{Attr}_0^0(\{v_6\}) = \{v_6\}$



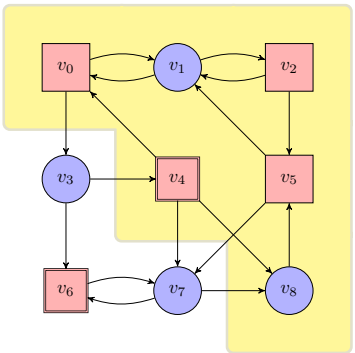
$\mathrm{Attr}_0^1(\{v_6\}) = \{v_6\} \cup \{v_3, v_7\}$



$\mathrm{Attr}_0(\{v_6\}) = \{v_3, v_6, v_7\}$



$W_1^1(\{v_4, v_6\}) = \{v_0, v_1, v_2, v_4, v_5, v_8\}$



$W_1(\mathcal{G}) = \{v_0, v_1, v_2, v_4, v_5, v_8\}$

Figure 8: Execution of the recurrence construction on the example of Figure 7. The essential parts of the calculations of $\mathrm{Recur}^n(F)$, $\mathrm{Attr}_i^n(R)$ and $\mathrm{CPre}_i(F)$ are highlighted by ▨, ▢ and ▨, respectively.

We are interested in solving Büchi games. From our example it seems that we have somehow to distinguish the different goals of Player 0 and Player 1. Player 0 eventually needs to cycle between some vertices in F. Player 1 eventually needs to cycle between some vertices not in F. In the example these vertices are v_6 and v_7 for Player 0 and v_0, v_1 and v_2 for Player 1.

But how can we generalize this observation? The problem is that we cannot just focus on one of the two players and handle the other player later as for reachability games. The reason is that we need information of the infinite behaviour for both. Accordingly, we have to first analyze where in the arena the players can fulfill their infinite goal. Having that, we can finally fix the remaining parts. This idea is realized in the following construction.

Construction 2.2. Let an arena $\mathcal{A} = (V, V_0, V_1, E)$ be given. The *recurrence construction* on $\mathcal{A}$ is defined for each Player i, for all $n \in \mathbb{N}$ and $F \subseteq V$ as:

$$\begin{aligned} \mathrm{Recur}^0(F) &= F \\ W_1^n(F) &= V \setminus \mathrm{Attr}_0(\mathrm{Recur}^n(F)) \\ \mathrm{Recur}^{n+1}(F) &= F \setminus \mathrm{CPre}_1(W_1^n(F)) \end{aligned}$$

We start with the set of vertices F that we want to see infinitely often. These vertices can only be reached by Player 0 in their attractor. Consequently, if Player 0 is not even able to reach them we must be in the winning region of Player 1. Finally, in contrast to reachability games, it does not help to reach the vertices at least once. So even if Player 0 is in an accepting vertex, if Player 1 can force her to move into his winning region, she will loose the play. Accordingly, we have to remove these accepting vertices from our consideration. But then Player 0 also cannot win if she can only reach such a bad accepting vertex. Accordingly we have to refine our attractor until we have excluded every one of them, what can be done in finitely many steps.

Remark 2.4. *For every Büchi game $\mathcal{G} = (\mathcal{A}, \textsc{Büchi}(F))$ it holds:*

$$\forall n, m \geq |\mathcal{A}|.\ \mathrm{Recur}^n(F) = \mathrm{Recur}^m(F)$$

In the end, Player 0 wins for the remaining attractor vertices because she can reach the remaining accepting vertices by the attractor construction and their successors are elements of the attractor. Accordingly, she can revisit them again by the same reason. An example execution of the construction is depicted in Figure 8. We can summarize the complete result formally as follows.

Lemma 2.3. *For every Büchi game $\mathcal{G} = (\mathcal{A}, \textsc{Büchi}(F))$ it holds that $W_1(\mathcal{G}) = \bigcup_{n\in\mathbb{N}} W_1^n(F)$ and $W_0(\mathcal{G}) = V \setminus W_1(\mathcal{G})$.*

Proof. Let $\mathcal{G} = (\mathcal{A}, \textsc{Büchi}(F))$ be a Büchi game with $\mathcal{A} = (V, V_0, V_1, E)$ and $X = V \setminus \bigcup_{n\in\mathbb{N}} W_1^n(F)$. We first show $X \subseteq W_0(\mathcal{G})$. By Remark 2.4 we know that $\forall m \geq |\mathcal{A}|$. $\mathrm{Recur}^{m+1}(F) = \mathrm{Recur}^m(F)$ and correspondingly $X = \mathrm{Attr}_0(\mathrm{Recur}^m(F))$. It holds that:

- $\mathrm{Recur}^m(F) \subseteq \mathrm{CPre}_0(X)$ (1)

  To show this let $v \in \mathrm{Recur}^m(F)$ be arbitrary. Then by construction $v \in F \setminus \mathrm{CPre}_1(W_1^m(F))$ and correspondingly $v \notin \mathrm{CPre}_1(W_1^m(F))$. We distinguish two cases:

  Case 1: $v \in V_0$

  $$\begin{aligned} &\overset{\text{Def. CPre}}{\Rightarrow} && \exists v' \in V.\ (v, v') \in E \wedge v' \notin W_1^m(F) \\ &\Rightarrow && \exists v' \in V.\ (v, v') \in E \wedge v' \in V \setminus W_1^m(F) \\ &\overset{\text{Def. CPre}}{\Rightarrow} && v \in \mathrm{CPre}_0(V \setminus W_1^m(F)) \end{aligned}$$

  Case 1: $v \in V_1$

  $$\begin{aligned} &\overset{\text{Def. CPre}}{\Rightarrow} && \forall v' \in V.\ (v, v') \in E \Rightarrow v' \notin W_1^m(F) \\ &\Rightarrow && \forall v' \in V.\ (v, v') \in E \Rightarrow v' \in V \setminus W_1^m(F) \\ &\overset{\text{Def. CPre}}{\Rightarrow} && v \in \mathrm{CPre}_0(V \setminus W_1^m(F)) \end{aligned}$$

  $$\Rightarrow v \in \mathrm{CPre}_0(\mathrm{Attr}_0(\mathrm{Recur}^m(F))) \Rightarrow v \in \mathrm{CPre}_0(X)$$

As shown in the proof of Lemma 2.1 we can use the attractor to derive a strategy σ_A to reach $\text{Recur}^m(F)$. Using this strategy, we define the strategy σ for Player 0 as follows.

$$\sigma(v) = \begin{cases} \sigma_A(v) & \text{if } v \in \text{Attr}_0(\text{Recur}^m(F)) \setminus \text{Recur}^m(F) \\ v' \text{ for some } v' \in X \text{ with } (v, v') \in E & \text{if } v \in \text{Recur}^m(F) \end{cases} \tag{2}$$

Consider that by (1) the choice for the case $v \in \text{Recur}^m(F)$ is always possible. We show that σ is a winning strategy for Player 0 from X, so let $\rho = \rho_0\rho_1\rho_2... \in \text{Plays}(\mathcal{A}, \sigma, X)$ be arbitrary. We first show:

- $\forall n \in \mathbb{N}.\ \rho_n \in X$ *(Proof by induction on n)* (3)

  Induction Base: $n = 0$

  $\rho_0 \in X$ directly follows from $\rho \in \text{Plays}(\mathcal{A}, \sigma, X)$.

  Induction Hypothesis:

  $\forall n \in \mathbb{N}.\ \rho_n \in X$

  Induction Step: $n > 0$

  By induction hypothesis we know that $\rho_{n-1} \in X$. By (2) we only need to show the case $\rho_{n-1} \in \text{Recur}^m(F)$. All other cases can be shown analogously as in the proof of Lemma 2.1. If $\rho_{n-1} \in V_0$ then $\rho_n = \sigma(\rho_{n-1})$ and the result follows from (2) and (1). If $\rho_{n-1} \in V_1$ then $\rho_n \in X$ follows directly from (1).

We need to show that the set F is visited infinitely often by ρ, or formally: $\forall n \in \mathbb{N}.\ \exists j \geq n.\ \rho_j \in F$. Accordingly, let $n \in \mathbb{N}$ be arbitrary. By (3) we know that $\rho_n \in X$. If $\rho_n \in \text{Recur}^m(F)$ choose $j = n$ and we are done since $\text{Recur}^m(F) \subseteq F$ by construction. For $\rho_n \notin \text{Recur}^m(F)$ the result follows by Lemma 2.1 and the choice of our strategy.

It remains to show $\bigcup_{n\in\mathbb{N}} W_1^n(F) \subseteq W_1(\mathcal{G})$ what is by Remark 2.1 sufficient to complete our proof. By Remark 2.4 we know for $m = |\mathcal{A}|$ that $\bigcup_{n\in\mathbb{N}} W_1^n(F) = W_1^m(F)$. We define a distance function $\delta : W_1^m(F) \to \mathbb{N}$ as follows:

$$\delta(v) = min\{\, n \in \mathbb{N} \mid v \in W_1^n(F) \,\} \tag{4}$$

We can show the following:

- $\forall v \in W_1^m \cap F.\ \delta(v) > 0$ (5)

  Consider that $W_1^0(F) = V \setminus \text{Attr}_0(F)$ and that by Construction 2.1 we have that $F \subseteq \text{Attr}_0(F)$. Accordingly we know $W_1^0 \cap F = \emptyset$ and the result follows for $v \in W_1^m \cap F$ by (4).

- $\forall v \in W_1^m(F) \cap V_0.\ \forall v' \in V.\ (v, v') \in E \Rightarrow \delta(v') \leq \delta(v)$ (6)

  Let $v \in W_1^m(F) \cap V_0$ and $v' \in V$ with (v, v') be arbitrary. Further, let $n = \delta(v)$ be the minimal n such that that $v \in W_1^n(F)$. We show the sufficient condition that $v' \in W_1^n(F) = V \setminus \text{Attr}_0(\text{Recur}^n(F))$. For the sake of contradiction assume that $v' \notin W_1^n(F)$ and accordingly $v' \in \text{Attr}_0(\text{Recur}^n(F))$. Then, as $v \in V_0$, it follows that $v \in \text{CPre}_0(\text{Attr}_0(\text{Recur}^n(F)))$ and further $v \in \text{Attr}_0(\text{Recur}^n(F))$ by Construction 2.2. But this is a contradiction against $v \in W_1^n(F) = V \setminus \text{Attr}_0(\text{Recur}^n(F))$. As a consequence, v' must be part of $W_1^n(F)$.

- $\forall v \in W_1^m(F) \cap V_1.\ \exists v' \in V.\ (v, v') \in E \wedge \delta(v') \leq \delta(v)$ (7)

  Let $v \in W_1^m \cap V_1$ be arbitrary and $n = \delta(v)$ be the minimal n such that that $v \in W_1^n(F)$. We have to show that there exists a $v' \in V$ with $(v, v') \in E$ such that $\delta(v') \leq n$. For the sake of contradiction assume that no such v' exists. Then for all v' with $(v, v') \in E$ we have that $\delta(v') > \delta(v)$ and accordingly $v' \notin W_1^n(F)$. It follows that all v' with $(v, v') \in E$ are part of $\text{Attr}_0(\text{Recur}^n(F))$ such that we get that $v \in \text{CPre}_0(\text{Attr}_0(\text{Recur}^n(F)))$. It follows that also $v' \in \text{Attr}_0(\text{Recur}^n(F))$ by Construction 2.2 which is a contradiction against $v \in W_1^n(F)$. Accordingly, there must exist at least one v'

- $\forall v \in W_1^m(F) \cap F \cap V_1.\ \exists v' \in V.\ (v, v') \in E \wedge \delta(v') < \delta(v)$ (8)

  Let v be arbitrary with $\delta(v) = n$. By (5) we know $n > 0$ such that by (4) follows $v \in W_1^n(F)$ and $v \notin W_1^{n-1}(F)$. It follows $v \in V \setminus \text{Attr}_0(\text{Recur}^n(F))$ and accordingly $v \notin \text{Attr}_0(\text{Recur}^n(F))$. By Construction 2.1 it follows also $v \notin \text{Recur}^n(F)$ and $v \notin F \setminus \text{CPre}_1(W_1^{n-1}(F))$. It follows by $v \in F$ that also $v \in \text{CPre}_1(W_1^{n-1}(F))$. Accordingly by definition of CPre it follows that there exists a $v' \in V$ with $(v, v') \in E$ and $v' \in W_1^{n-1}(F)$. Accordingly $\delta(v')$ is at most $n - 1$.

- $\forall v \in W_1^m(F) \cap F \cap V_0.\ \forall v' \in V.\ (v, v') \in E \Rightarrow \delta(v') < \delta(v)$ (9)

  Let v and v' arbitrary with $\delta(v) = n$. As already shown in (8) it follows $v \in \text{CPre}_1(W_1^{n-1}(F))$ and by definition of CPre that $(v, v') \in E$ and $v' \in W_1^{n-1}(F)$. As before it follows $\delta(v') \leq n - 1$.

We now can construct a strategy τ for Player 1 defined on $W_1^m(F)$.

$$\tau(v) = \begin{cases} v' \text{ for some } (v, v') \in E \text{ with } \delta(v') < \delta(v) & \text{if } v \in F \cap W_1^m(F) \\ v' \text{ for some } (v, v') \in E \text{ with } \delta(v') \leq \delta(v) & \text{if } v \in W_1^m(F) \setminus F \end{cases} \tag{10}$$

Consider that case $v \in F \cap W_1^m(F)$ is always realizable by (8) and the other case by (7). It remains to show that τ is winning for Player 1 from $W_1^m(F)$. Let $\rho = \rho_0\rho_1\rho_2... \in \text{Plays}(\mathcal{A}, \tau, W_1^m(F))$ be arbitrary. By (6) and (10) we have that $\delta(\rho_n) \geq \delta(\rho_{n+1})$ for all $n \in \mathbb{N}$. Now assume ρ is not winning for Player 1, so there are infinitely many different positions $n_0, n_1, n_2, ... \in \mathbb{N}$ with $\rho_{n_j} \in F$ for all $j \in \mathbb{N}$. By (9) and (10) we know $\delta(\rho_0) > \delta(\rho_{n_0+1}) > \delta(\rho_{n_1} + 1) > ...$ 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 (5). Accordingly τ is winning for all plays starting in $W_1^m(F)$ and by that $W_1^m(F) \subseteq W_1(\mathcal{G})$ concluding the proof. □

Theorem 2.3. *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 2.3. The given complexity results are achievable directly through Construction 2.2. □

2.6 Co-Büchi Games

Analogous to reachability and safety conditions being complementary, we call the complement of a Büchi condition a co-Büchi condition and consider games with such a winning condition.

Definition 2.17. Let the co-Büchi condition $\text{coBÜCHI}(C)$ on a set $C \subseteq V$ for an arena $\mathcal{A} = (V, V_0, V_1, E)$ be defined as:

$$\text{coBÜCHI}(C) := \{\, \rho \in \text{Plays}(\mathcal{A}) \mid \text{Inf}(\rho) \subseteq C \,\}$$

Then we call the game $\mathcal{G} = (\mathcal{A}, \text{coBÜCHI}(C))$ a *co-Büchi game* with persistence set C.

For an example consider the game given in Figure 9. Here, Player 0 wins from every initial vertex. A winning strategy is given by going from v_1 to v_2, from v_3 to v_6, from v_7 to v_8 and from v_8 to v_5. Using this strategy, Player 0 avoids visiting v_4 more than once since the only incoming edge comes from v_3 and is never taken. But then also v_0 is only visited finitely often, because she does not use the edge from v_1. As a consequence, v_3 can only be visited finitely often as well and the same holds for v_6 since Player 0 never takes the edge back from v_7. Finally, we get the similar results, as for reachability and safety games, summarized in Lemma 2.4, Corollary 2.2 and Theorem 2.3.

Lemma 2.4. *Let $\mathcal{A} = (V, V_0, V_1, E)$ be an arena, $\mathcal{G} = (\mathcal{A}, \text{coBÜCHI}(C))$ be a co-Büchi game with $C \subseteq V$ and $\mathcal{G}' = (\bar{\mathcal{A}}, \text{BÜCHI}(V \setminus C))$ be a Büchi game. Then every strategy σ winning for Player i from a vertex $v \in V$ in the game $\mathcal{G}'$ is also a winning strategy for Player $1 - i$ from v in the game $\mathcal{G}$.*

Proof. Analogous to the proof of Lemma 2.2. □

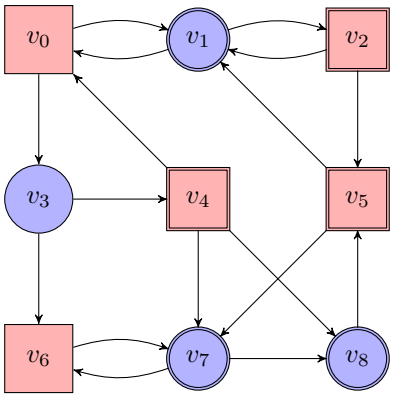


Figure 9: Example for a co-Büchi game. The doubly framed vertices mark the persistence region.

Corollary 2.2. *Let $\mathcal{A} = (V, V_0, V_1, E)$ be an arena, $\mathcal{G} = (\mathcal{A}, \text{COBÜCHI}(C))$ be a co-Büchi game with $C \subseteq V$ and $\mathcal{G}' = (\bar{\mathcal{A}}, \text{BÜCHI}(V \setminus C))$ be a Büchi game. Then $W_i(\mathcal{G}) = W_{1-i}(\mathcal{G}')$ for each Player i.*

Proof. Analogous to the proof of Corollary 2.1. □

Theorem 2.4. *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.*

Proof. The result is a direct consequence of Corollary 2.2 and Theorem 2.3. □

2.7 Parity Games

The last type of games we want to consider in this chapter are so called parity games. Here, each vertex is marked with a color, represented by natural numbers. Player 0 wins a play, if the minimal color seen infinitely often is even, Player 1 wins a play if it is odd. Accordingly, the winner of a game depends on the parity of the smallest color seen infinitely often, giving the game its name. To have a suitable representation of the coloring we use a color function Ω that returns us the color for each vertex.

Definition 2.18. Let the parity condition $\text{PARITY}(\Omega)$ on a color function $\Omega : V \to [k]$ for some arena $\mathcal{A} = (V, V_0, V_1, E)$ and some $k \in \mathbb{N}$ be defined as:

$$\text{PARITY}(\Omega) := \{\, \rho \in \text{Plays}(\mathcal{A}) \mid \text{Par}(min(\Omega(\text{Inf}(\rho)))) = 0 \,\}$$

Then we call the game $\mathcal{G} = (\mathcal{A}, \text{PARITY}(\Omega))$ a *parity game* with color function Ω.

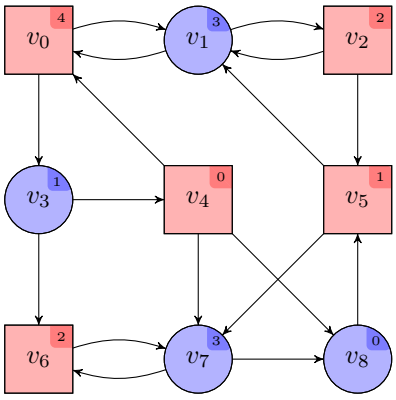


Figure 10: Example for a parity game. The color of each vertex is given in its upper right corner.

An example for a parity game is given in Figure 10. Player 0 wins this game from the vertices v_3, v_6 and v_7. Her strategy is to cycle between v_6 and v_7 such that the minimal color visited infinitely often is 2. From v_3 Player 0 can reach this region in one step. Player 1 wins from the remaining vertices.

His strategy is to either cycle between v_0 and v_1 or to cycle between v_1, v_2 and v_5, depending on which cycle Player 0 chooses infinitely often in v_1. Whatever strategy Player 0 chooses for v_1, the minimal color visited infinitely often is either 3 or 1 this way. Player 1 then also wins from v_8 by moving to v_5 and from v_4 by moving to v_0 or to v_8. Consider that both player have uniform winning strategies.

One might observe that a parity condition is a nested combination of Büchi and co-Büchi conditions, i.e. Player 0 needs to avoid seeing odd colors infinitely often and has to see even colors infinitely often. However seeing large odd colors infinitely often does not cause problems as long as a smaller even color is seen infinitely often. The dual property is given for Player 1. On the other hand, Büchi and co-Büchi games are just a special case of Parity games as shown in the following.

Remark 2.5.

1. *Every Büchi game* $\mathcal{G} = (\mathcal{A}, \textsc{büchi}(F))$ *can be represented as a parity game* $\mathcal{G}' = (\mathcal{A}, \textsc{parity}(\Omega))$.
2. *Every co-Büchi game* $\mathcal{G} = (\mathcal{A}, \textsc{cobüchi}(C))$ *can be represented as a parity game* $\mathcal{G}' = (\mathcal{A}, \textsc{parity}(\Omega))$.

Proof. We can represent a Büchi game $\mathcal{G} = (\mathcal{A}, \textsc{büchi}(F))$ with arena $\mathcal{A} = (V, V_0, V_1, E)$ by the parity game $\mathcal{G}' = (\mathcal{A}, \textsc{parity}(\Omega : V \to [3]))$ with

$$\Omega(v) = \begin{cases} 0 & \text{if } v \in F \\ 1 & \text{if } v \notin F \end{cases}$$

and we can represent a co-Büchi game $\mathcal{G} = (\mathcal{A}, \textsc{cobüchi}(C))$ with arena $\mathcal{A} = (V, V_0, V_1, E)$ by the parity game $\mathcal{G}' = (\mathcal{A}, \textsc{parity}(\Omega : V \to [2]))$ with

$$\Omega(v) = \begin{cases} 2 & \text{if } v \in C \\ 1 & \text{if } v \notin C \end{cases}$$

Then in both cases the parity condition exactly defines the Büchi and co-Büchi condition, respectively, and as a consequence we have for both cases that $\mathcal{G} = \mathcal{G}'$. □

As for all other games presented so far, we can ask ourself what is the weakest notion of strategies the players need to win parity games. The answer is given by the following theorem.

Theorem 2.5. *Parity games are determined with uniform positional winning strategies.*

Proof. Let $\mathcal{G} = (\mathcal{A}, \textsc{parity}(\Omega : V \to [k]))$ be a parity game with $\mathcal{A} = (V, V_0, V_1, E)$. We prove the theorem by induction over $|V| = n$.

Induction Base: $n = 1$

Let $V = \{v\}$, by Definition 2.1 it follows that $E = \{(v, v)\}$. Accordingly, there is only one unique play $\rho = v^\omega$, independently from the player owning the vertex and its strategy. As a consequence, Player 0 wins the game if $\text{Par}(\Omega(v)) = 0$ and Player 1 wins the game otherwise. Consider that there is only one possible strategy for the player owning v, which is positional.

Induction Hypothesis:

$\forall n \in \mathbb{N}.\ \forall \mathcal{G} = (\mathcal{A}, \textsc{parity}(\Omega)).\ |\mathcal{A}| = n \Rightarrow \mathcal{G}$ is pos. determined with uniform winning strategies

Induction Step: $n > 1$

Let $d = min(\Omega(V))$ be the minimal color occurring in the game with $\text{Par}(d) = i$. Further let $D = \{\, v \in V \mid \Omega(v) = d \,\}$ be the set of states colored with d and $A = \text{Attr}_i(D) \neq \emptyset$ be its attractor. Finally let σ_A be the strategy defined trough the attractor forcing plays from A into D.

Case 1: $A = V$

We define a winning strategy for Player i as follows

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

It holds that σ is a winning strategy for Player i from each $v \in V$. To show this let $v \in V$, $\rho = \rho_0\rho_1\rho_2... \in \text{Plays}(\mathcal{A}, \sigma, v)$ and $n \in \mathbb{N}$ be arbitrary. By $A = V$ it follows that $\rho_n \in A$ and by construction of σ we know that there must be an $m \geq n$ such that $\rho_m \in D$. Accordingly, we have that $\forall n \in \mathbb{N}.\ \exists m \geq n.\ \rho_m \in D$ what is equivalent to $\text{Inf}(\rho) \cap D \neq \emptyset$. Finally, by $\Omega(D) = d$ and minimality of d it follows with $\text{Par}(d) = i$ that $\rho \in \text{PARITY}(\Omega)$ for $i = 0$ and $\rho \notin \text{PARITY}(\Omega)$ for $i = 1$. Thus, σ is a uniform positional winning strategy from $V = W_i(\mathcal{G})$ for Player i.

Case 2: $A \subset V$

Consider the sub-arena $\mathcal{A}' = \mathcal{A}\restriction(V \setminus A)$ defining the game $\mathcal{G}' = (\mathcal{A}', \text{PARITY}(\Omega' : V \setminus A \to [k]))$ with $\Omega'(v) = \Omega(v)$ for all $v \in V \setminus A$. This sub-arena is always defined since for each $v \in V \setminus A$ there exists an edge back to $V \setminus A$. If this would not be the case then all outgoing edges of v would lead into A. But this is a contradiction against $A = \text{Attr}_i(D)$ and $v \notin A$ since then by definition of the attractor it would hold $v \in A$. It follows from $A \neq \emptyset$ that $|\mathcal{A}'| < |\mathcal{A}|$ such that the induction hypothesis is applicable. Correspondingly, the $\mathcal{G}'$ is positionally determined with uniform winning strategies σ' and τ'. We distinguish two further subcases:

Subcase 1: $W_{1-i}(\mathcal{G}') = \emptyset$

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

$$\sigma(v) = \begin{cases} \sigma'(v) & \text{if } v \in W_i(\mathcal{G}') \\ \sigma_A(v) & \text{if } v \in A \setminus D \\ v' \text{ for some } (v, v') \in E & \text{if } v \in D \end{cases} \tag{1}$$

To show that σ is indeed a winning strategy let $v \in V$ and $\rho = \rho_0\rho_1\rho_2... \in \text{Plays}(\mathcal{A}, \sigma, v)$ be arbitrary. First assume that for some position $m \in \mathbb{N}$ we have that $\rho_j \in V \setminus A$ for every $j \geq m$. Then by (1) it follows that $\rho' = \rho_m\rho_{m+1}\rho_{m+2}... \in \text{Plays}(\mathcal{A}', \sigma', V \setminus A)$ and together with the inductive properties of σ' that ρ' is winning for Player i in $\mathcal{G}'$. Accordingly, for $q = min\{\, \Omega(\rho_j) \mid j \geq m \,\}$ we have that $\text{Par}(q) = i$ and it holds that $\forall j \geq m.\ \exists p \geq j.\ \Omega(\rho_p) = q$. As a consequence the same holds for all $j \in \mathbb{N}$. Thus ρ is also winning for Player i in $\mathcal{G}$.

Next assume that for all $m \in \mathbb{N}$ there exists a $j \geq m$ such that $\rho_j \in A$. Then by definition of (1) and the attractor construction we know that there exists $p \geq j$ such that $\rho_p \in D$. As a consequence we have that $\forall m \in \mathbb{N}.\ \exists p \geq m.\ \rho_p \in D$ and ρ must be winning for Player i in $\mathcal{G}$.

Subcase 2: $W_{1-i}(\mathcal{G}') \neq \emptyset$

Let $B = \text{Attr}_{1-i}(W_{1-i}(\mathcal{G}')) \neq \emptyset$ be the attractor of $W_{1-i}(\mathcal{G}')$ in the game $\mathcal{G}$. Then the sub-arena $\mathcal{A}'' = \mathcal{A}\restriction(V \setminus B)$ defines the game $\mathcal{G}'' = (\mathcal{A}'', \text{PARITY}(\Omega'' : V \setminus B \to [k]))$ with $\Omega''(v) = \Omega(v)$ for all $v \in V \setminus B$. Again this sub-arena is defined since its vertices are in the complement of an attractor for some set in the original game. Further we know that $|\mathcal{A}''| < |\mathcal{A}|$ such that the induction hypothesis is applicable and the game $\mathcal{G}''$ is positionally determined with uniform winning strategies σ'' and τ''. We show that

$$W_i(\mathcal{G}) = W_i(\mathcal{G}'') \qquad \text{and} \qquad W_{1-i}(\mathcal{G}) = W_{1-i}(\mathcal{G}'') \cup B$$

by showing $W_i(\mathcal{G}'') \subseteq W_i(\mathcal{G})$ and $W_{1-1}(\mathcal{G}'') \cup B \subseteq W_{1-1}(\mathcal{G})$. We first show that σ'' is a positional winning strategy for Player i from $W_i(\mathcal{G}'')$ in the game $\mathcal{G}$. Accordingly, let $\rho = \rho_0\rho_1\rho_2... \in \text{Plays}(\mathcal{A}, \sigma'', W_i(\mathcal{G}''))$ be arbitrary. It holds that for all $m \in \mathbb{N}$ we have that $\rho_m \in W_i(\mathcal{G}'')$ since parity games are prefix independent for similar reasons as for Büchi and co-Büchi games and the winning regions define a trap for each Player i (see Exercise 2.3). It follows that $W_i(\mathcal{G}'')$ is also a trap on the whole arena $\mathcal{A}$ since Player i can only move to vertices in $\mathcal{A}''$, as σ'' is defined on $\mathcal{A}''$, and Player $1 - i$ cannot move to vertices $v' \in B$, not part of $\mathcal{A}''$, from some vertex v since B is defined as the attractor of Player $1 - i$ and then it would hold that $v \in B$ as well. It follows that $\rho \in \text{PARITY}(\Omega'')$ for $i = 0$ and $\rho \notin \text{PARITY}(\Omega'')$ for $i = 1$. Correspondingly also $\rho \in \text{PARITY}(\Omega)$ for $i = 0$ and $\rho \notin \text{PARITY}(\Omega)$ for $i = 1$.

It remains to show that there is a uniform positional strategy τ for Player $1 - i$ winning from $W_{1-i}(\mathcal{G})$. We define τ as follows.

$$\tau(v) = \begin{cases} \tau''(v) & \text{if } v \in W_{1-i}(\mathcal{G}'') \\ \tau_A(v) & \text{if } v \in B \setminus W_{1-i}(\mathcal{G}') \\ \tau'(v) & \text{if } v \in W_{1-i}(\mathcal{G}') \end{cases} \tag{2}$$

where τ_A is the strategy defined through the attractor of B enforcing plays from B into $W_{1-i}(\mathcal{G}')$. We show that τ is indeed a winning strategy for Player $1-i$. Let the play $\rho = \rho_0\rho_1\rho_2... \in \text{Plays}(\mathcal{A}, \tau, W_{1-i}(\mathcal{G}'') \cup B)$ be arbitrary. If there exists a position $m \in \mathbb{N}$ such that $\rho_m \in W_{1-i}(\mathcal{G}')$ we know that $\rho_j \in W_{1-i}(\mathcal{G}')$ for all $j \geq m$ since $W_{1-i}(\mathcal{G}')$ is a trap on $\mathcal{A}$. This holds since $W_{1-i}(\mathcal{G}')$ is a trap on $\mathcal{A}'$, Player $1-i$ will not leave $\mathcal{A}'$ since τ' only is defined on $\mathcal{A}'$ and Player i cannot move from some vertex $v \in W_{1-i}(\mathcal{G})$ to some vertex $v' \in A$ not in $\mathcal{A}'$ as A is defined as an attractor for Player i and then it would hold that also $v \in A$. Accordingly, the play $\rho' = \rho_m\rho_{m+1}\rho_{m+2}... \in \text{Plays}(\mathcal{A}', \tau', W_{1-i}(\mathcal{G}'))$ is winning for Player $1-i$, thus $\rho' \notin \text{PARITY}(\Omega')$ if $i = 0$ and $\rho' \in \text{PARITY}(\Omega')$ if $i = 1$. Correspondingly, also $\rho \notin \text{PARITY}(\Omega)$ if $i = 0$ and $\rho \in \text{PARITY}(\Omega)$ if $i = 1$ such that ρ is winning for Player i. It remains to consider the case, where ρ starts in $W_{i-1}(\mathcal{G}'')$ and for all $m \in \mathbb{N}$ we have $\rho_m \notin B$. For $i = 0$ it follows that $\rho \in \text{PARITY}(\Omega'')$ and correspondingly also $\rho \in \text{PARITY}(\Omega)$. For $i = 1$ it follows that $\rho \notin \text{PARITY}(\Omega'')$ and correspondingly also $\rho \notin \text{PARITY}(\Omega)$. Anyway, the play ρ is winning for Player $i-1$. □

Lemma 2.5. *For every parity game $\mathcal{G} = (\mathcal{A}, \text{PARITY}(\Omega\colon V \to [k]))$ exists a dual parity game $\mathcal{G}' = (\bar{\mathcal{A}}, \text{PARITY}(\Omega'))$ such that $\text{PARITY}(\Omega') = V^\omega \setminus \text{PARITY}(\Omega)$. We also say parity games are self-dual.*

Proof. For a parity game $\mathcal{G} = (\mathcal{A}, \text{PARITY}(\Omega : V \to [k]))$ with arena $\mathcal{A} = (V, V_0, V_1, E)$ we can create the dual game $\mathcal{G}' = (\bar{\mathcal{A}}, \text{PARITY}(\Omega' : V \to [k+1]))$ by shifting the colors by one, accordingly $\Omega'(v) = \Omega(v)+1$. Then every strategy σ', winning for Player i in $\mathcal{G}'$, is a winning strategy τ for player Player $1-i$ in $\mathcal{G}$ (even → odd, odd → even). □

Consequence 2.1. *The problem to determine whether it holds that $v \in W_0(\mathcal{G})$ for a given parity game $\mathcal{G} = (\mathcal{A}, \text{PARITY}(\Omega))$ with $\mathcal{A} = (V, V_0, V_1, E)$ and $v \in V$ is in* NP ∩ Co-NP.

Proof. We prove the statement by giving an NP-algorithm and a Co-NP-algorithm.

NP-algorithm:

To show that the problem is in NP it is sufficient to show that there are polynomially-sized certificates, proving that $v \in W_0(\mathcal{G})$, which can be verified polynomial time. We choose as certificates positional strategies σ, winning for Player 0 from v. Accordingly, let a winning strategy σ be given. Then we construct the new game $\mathcal{G}_\sigma = (\mathcal{A}_\sigma, \text{PARITY}(\Omega)))$ with the new arena $\mathcal{A}_\sigma = (V, V_0, V_1, E_\sigma)$ and $E_\sigma = E \cap (V_1 \times V \cup \sigma)$. The resulting game $\mathcal{G}_\sigma$ is a solitary parity game for Player 1 and can be solved in polynomial time (see Exercise 4.3). So we can compute the winning regions $W_0(\mathcal{G}_\sigma)$ and $W_1(\mathcal{G}_\sigma)$ and corresponding winning strategies σ and τ. But as $\text{Plays}(\mathcal{A}, \sigma, v) = \text{Plays}(\mathcal{A}_\sigma, v)$ by construction, it follows that σ is a winning strategy iff $v \in W_0(\mathcal{G}_\sigma)$.

Co-NP-algorithm:

To show that the problem is in Co-NP it is sufficient to show that there are polynomially-sized certificates, proving that $v \notin W_0(\mathcal{G})$, which can be verified in polynomial time. By determinacy, we can choose as certificates positional strategies τ, winning for Player 1 from v. Accordingly, let a winning strategy τ be given. As for the NP-algorithm we can create the corresponding solitary game $\mathcal{G}_\tau$ and solve it. As before, $\text{Plays}(\mathcal{A}, \tau, v) = \text{Plays}(\mathcal{A}_\tau, v)$ by construction such that τ is winning for Player 1 iff $v \in W_1(\mathcal{G}_\tau)$. □

We are interested in solving parity games. Consider that we already have given some kind of algorithm by the proof of Theorem 2.5. We can simply translate the inductive construction of the proof into an algorithm with recursive calls. The corresponding algorithm is also known as McNaughton's algorithm and has exponential worst case running time. However, we want to consider a different algorithm here, called the progress measure algorithm. We start with the following definition.

Definition 2.19. Let $\mathcal{G} = (\mathcal{A}, \text{PARITY}(\Omega\colon V \to [k]))$ be a parity game with $\mathcal{A} = (V, V_0, V_1, E)$, $d = min\{\, j \in \mathbb{N} \mid j \geq k-1 \wedge \text{Par}(j) = 1 \,\}$. A *score sheet* s of $\mathcal{G}$ is either $\top$ or a vector $\vec{s}$ such that

$$\vec{s} = (s_1, s_3, ..., s_{d-2}, s_d) \in \bigtimes_{\substack{c \in [d+1], \\ \text{Par}(c)=1}} [n_c + 1]$$

where $n_c = |\{\, v \in V \mid \Omega(v) = c \,\}|$. The set of all score sheets of $\mathcal{G}$ is denoted by $\text{Sh}(\mathcal{G})$.

Consider the example game $\mathcal{G}_e$ depicted in Figure 10. Here, we have $d = 5$ such that the set $\text{Sh}(\mathcal{G}_e)$ is given by the set $\text{Sh}(\mathcal{G}_e) = \{0,1,2\} \times \{0,1,2\} \times \{0\} \cup \{\top\}$.

Definition 2.20. Let $s \in \text{Sh}(\mathcal{G})$ be a score sheet of some parity game $\mathcal{G} = (\mathcal{A}, \text{PARITY}(\Omega\colon V \to [k]))$ with $\mathcal{A} = (V, V_0, V_1, E)$ and $c \in [k]$. We define the *addition of c to s*, denoted by $s \oplus c$, as follows:

$$s \oplus c = \begin{cases} (s_1, ..., s_{c-1}, 0, ..., 0) & \text{if } s = (s_1, s_3, ..., s_d) \wedge \text{Par}(c) = 0 \\ (s_1, ..., s_{c'-2}, s_{c'} + 1, 0, ..., 0) & \text{if } s = (s_1, s_3, ..., s_d) \wedge \text{Par}(c) = 1 \\ & \quad \wedge\, c' = max\{\, j \leq c \mid \text{Par}(j) = 1 \wedge s_j < n_j \,\} \\ \top & \text{otherwise} \end{cases}$$

Let e.g. $s = (1,1,0) \in \text{Sh}(\mathcal{G}_e)$, then $s \oplus 3 = (1,2,0)$, $(s \oplus 3) \oplus 3 = (2,0,0)$ and $(s \oplus 3) \oplus 2 = (1,0,0)$. It holds the following.

Lemma 2.6. *Let $\mathcal{G} = (\mathcal{A}, \text{PARITY}(\Omega\colon V \to [k]))$ be a parity game with $\mathcal{A} = (V, V_0, V_1, E)$, $d = min\{\, j \in \mathbb{N} \mid j \geq k-1 \wedge \text{Par}(j) = 1 \,\}$ and $P_1 = \{\, j \in [d+1] \mid \text{Par}(j) = 1 \,\}$. Then it holds for all $c \in P_1$ that $\oplus\, c$ is $\leq$-monotone, i.e.*

$$\forall s, s' \in \text{Sh}(\mathcal{G}).\ s \leq s' \Rightarrow s \oplus c \leq s' \oplus c$$

Proof. Let $c \in P_1$ and $s, s' \in \text{Sh}(\mathcal{G})$ be arbitrary. We have to show that $s \leq s' \Rightarrow s \oplus c \leq s' \oplus c$.

Case 1: $s' = \top$ $\Rightarrow\ s' \oplus c = \top \geq s \oplus c$

Case 2: $s = \top$ $\Rightarrow\ s' = \top\ \Rightarrow$ Case 1

Case 3: $s = s'$ $\Rightarrow\ s \oplus c = s' \oplus c$

Case 4: $s = (s_1, s_3, ..., s_d) \wedge s' = (s'_1, s'_3, ..., s'_d) \wedge s \neq s'$

Then $\exists n \in P_1.\ s_n < s'_n \wedge \forall m \in P_1.\ m < n \Rightarrow s_m = s'_m$. Let $c' = max\{\, j \in \mathbb{N} \mid j \leq c \wedge s_j \leq n_j \,\}$ and $c'' = max\{\, j \in \mathbb{N} \mid j \leq c \wedge s'_j \leq n_j \,\}$.

Subcase 1: $\text{Par}(c) = 0 \wedge n < c$

$$\begin{array}{ll} & s \oplus c \\ = & (s_1, s_3, ..., s_{c-1}, 0, ..., 0) \\ \leq & (s_1, s_3, ..., s'_n, ..., s'_{c-1}, 0, ..., 0) \\ = & (s'_1, s'_3, ..., s'_n, ..., s'_{c-1}, 0, ..., 0) \\ = & s' \oplus c \end{array}$$

Subcase 2: $\text{Par}(c) = 0 \wedge n \geq c$

$$\begin{array}{ll} & s \oplus c \\ = & (s_1, s_3, ..., s_{c-1}, 0, ..., 0) \\ = & (s'_1, s'_3, ..., s'_{c-1}, 0, ..., 0) \\ = & s' \oplus c \end{array}$$

Subcase 3: $\text{Par}(c) = 1 \wedge n < c''$

It follows that $n \leq c'$ such that

$$\begin{array}{ll} & s \oplus c \\ = & (s_1, s_3, ..., s_n, ..., s_{c'} + 1, 0, ..., 0) \\ \leq & (s_1, s_3, ..., s'_n, ..., s'_{c''} + 1, 0, ..., 0) \\ = & (s'_1, s'_3, ..., s'_n, ..., s'_{c''} + 1, 0, ..., 0) \\ = & s' \oplus c \end{array}$$

Subcase 4: $\text{Par}(c) = 1 \wedge n \geq c''$

It follows that $j \geq c' \geq c''$ such that

$$\begin{array}{ll} & s \oplus c \\ = & (s_1, s_3, ..., s_{c'} + 1, 0, ..., 0) \\ \leq & (s_1, s_3, ..., s'_{c'} + 1, 0, ..., 0) \\ = & (s'_1, s'_3, ..., s'_{c'} + 1, 0, ..., 0) \\ = & s' \oplus c \end{array}$$

Consider that in the transformation above $s'_{c'} + 1 = s_{c'} + 1$ for $j > c'$ and $s'_{c'} + 1 = s'_{c''} + 1$ for $j = c' = c''$. □

Definition 2.21. Let $\mathcal{G} = (\mathcal{A}, \text{PARITY}(\Omega\colon V \to [k]))$ with $\mathcal{A} = (V, V_0, V_1, E)$ be a parity game. A function $\wp\colon V \to \text{Sh}(\mathcal{G})$ is a *progress measure for* $\mathcal{G}$ iff

- $\forall v \in V_0.\ \exists v' \in V.\ (v, v') \in E \wedge \wp(v) \geq \wp(v') \oplus \Omega(v)$ and
- $\forall v \in V_1.\ \forall v' \in V.\ (v, v') \in E \Rightarrow \wp(v) \geq \wp(v') \oplus \Omega(v)$.

We denote the set of all progress measures for $\mathcal{G}$ by $\text{PM}(\mathcal{G})$ and define the function $\|\cdot\|\colon \text{PM}(\mathcal{G}) \to 2^V$ as $\|\wp\| = \{\, v \in V \mid \wp(v) \neq \top \,\}$.

An example for a progress measure $\wp_e$ for our game $\mathcal{G}_e$ would be given as follows:

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

In general, we can make the following observation.

Lemma 2.7. *Let $\mathcal{G} = (\mathcal{A}, \text{PARITY}(\Omega\colon V \to [k]))$ be a parity game with $\mathcal{A} = (V, V_0, V_1, E)$.*

i) For every progress measure $\wp \in \text{PM}(\mathcal{G})$ it holds that $\|\wp\| \subseteq W_0(\mathcal{G})$

ii) There exists a progress measure $\wp \in \text{PM}(\mathcal{G})$ with $\|\wp\| = W_0(\mathcal{G})$.

Proof. For the proof of i) see Exercise 5.2. It remains to show ii). By Theorem 2.5 there exists a positional winning strategy σ for Player 0 from $W_0(\mathcal{G})$. We define the function $sh\colon V^* \to \text{Sh}(\mathcal{G})$ recursively as $sh(\varepsilon) = \vec{0}$ and $sh(wv) = sh(w) \oplus \Omega(v)$. Further define $\overleftarrow{sh}\colon V^* \to \text{Sh}(\mathcal{G})$ as $\overleftarrow{sh}(w_0 w_1 ... w_n) = sh(w_n w_{n-1} ... w_0)$. We then can define the function $\wp\colon V \to \text{Sh}(\mathcal{G})$:

$$\wp(v) = \begin{cases} max\{\, \overleftarrow{sh}(w) \mid w \in \text{Pref}(\rho) \wedge \rho \in \text{Plays}(\mathcal{A}, \sigma, v) \,\} & \text{if } v \in W_0(\mathcal{G}) \\ \top & \text{if } v \in W_1(\mathcal{G}) \end{cases}$$

It remains to show the following two statements concluding the proof:

- $\forall v \in W_0(\mathcal{G}).\ \wp(v) \neq \top$

  As $\top$ is the maximal element in $\text{Sh}(\mathcal{G})$, we have to prove that for all $\rho \in \text{Plays}(\mathcal{A}, \sigma, v)$ and all $w \in \text{Pref}(\rho)$ it holds that $\overleftarrow{sh}(w) \neq \top$. Let $P_1 = \{\, j \in [d] \mid \text{Par}(j) = 1 \,\}$ and $\varsigma_c\colon \text{Sh}(\mathcal{G}) \to \mathbb{N} \cup \{\infty\}$ for $c \in P_1$ be defined as:

  $$\varsigma_c(s) = \begin{cases} \infty & \text{if } s = \top \\ s_c & \text{if } s = (s_1, s_3, ..., s_d) \end{cases}$$

  Additionally, we define the function $sc_c\colon V^* \to \mathbb{N}$ as follows:

  $$sc_c(w) = \begin{cases} sc_c(w') & \text{if } w = w'v' \wedge \Omega(v') > c \\ sc_c(w') + 1 & \text{if } w = w'v' \wedge \Omega(v') = c \\ 0 & \text{otherwise} \end{cases}$$

  Now, let $\rho \in \text{Plays}(\mathcal{A}, \sigma, v)$ be arbitrary and let $w \in \text{Pref}(\rho)$ with $m = |w|$. Consider that always $sc_c(w^R) \leq n_c$ for $w^R = w_{m-1} ... w_0$. Otherwise, there would be $j > n_c$ many occurrences of an odd color without a smaller even color in between. It follows that there would exist indices p and q with $w = w' w_p w_{p+1} ... w_q w''$, $w_p = w_q$, $\text{Par}(\Omega(w_p)) = 1$ and $\Omega(w_t) \geq \Omega(w_p)$ for all $t \in [p, q]$. But then the play $w'(w_p ... w_{q-1})^\omega$ is consistent with σ which is a contradiction against the fact that σ is a winning strategy.

  We have to show that $\varsigma_c(\overleftarrow{sh}(w)) \neq \infty$ for all $c \in P_1$. We show this by proving that

  $$\forall n \in [m].\ \forall c \in P_1.\ \varsigma_c(\overleftarrow{sh}(w_n w_{n+1} ... w_{m-1})) = sc_c(w_{m-1} w_{m-2} ... w_n)$$

  This then especially shows that $\varsigma_c(\overleftarrow{sh}(w)) \neq \infty$ for all $c \in P_1$. We show the statement by downward induction over n.

Induction Base: $n = m - 1$

$\Rightarrow\ \varsigma_c(\overleftarrow{sh}(w_{m-1})) = \varsigma_c(sh(w_{m-1})) = \varsigma_c(\vec{0} \oplus \Omega(w_{m-1})) =$

$\Omega(w_{m-1}) = c: \quad 1 = sc_c(\varepsilon) + 1 = sc_c(w_{m-1})$

$\Omega(w_{m-1}) \neq c: \quad 0 = sc_c(\varepsilon) = sc_c(w_{m-1})$

Induction Hypothesis:

$\forall n \in [m].\ \forall c \in P_1.\ \varsigma_c(\overleftarrow{sh}(w_n w_{n+1} ... w_{m-1})) = sc_c(w_{m-1} w_{m-2} ... w_n)$

Induction Step: $n < m - 1$

$\Rightarrow\ \varsigma_c(\overleftarrow{sh}(w_n w_{n+1} ... w_{m-1})) = \varsigma_c(sh(w_{m-1} ... w_n)) = \varsigma_c(sh(w_{m-1} ... w_{n+1}) \oplus \Omega(w_n))$

By induction hypothesis we have that $\varsigma_c(\overleftarrow{sh}(w_{n+1} ... w_{m-1})) = sc_c(w_{m-1} ... w_{n+1})$ for all $c \in P_1$. We distinguish three cases.

Case 1: $\Omega(w_n) = c$

Consider that we have $\varsigma_c(\overleftarrow{sh}(w_{n+1} ... w_{m-1})) = sc_c(w_{m-1} ... w_{n+1})$. It must hold that $sc_c(w_{m-1} ... w_{n+1}) < n_c$, since otherwise $sc_c(w_{m-1} ... w_{n+1} w_n) > n_c$ and we already have argued that this is impossible. It follows that

$$\begin{aligned} \varsigma_c(\overleftarrow{sh}(w_n ... w_{m-1})) = \varsigma_c(\overleftarrow{sh}(w_{n+1} ... w_{m-1})) + 1 &= sc_c(w_{m-1} ... w_{n+1}) + 1 \\ &= sc_c(w_{m-1} ... w_n) \end{aligned}$$

Case 2: $\Omega(w_n) < c$

By Case 1 we have that $\varsigma_{\Omega(w_n)}(\overleftarrow{sh}(w_n ... w_{m-1})) = sc_{\Omega(w_n)}(w_{m-1} ... w_n) \leq n_{\Omega(w_n)}$ and accordingly $\overleftarrow{sh}(w_n ... w_{m-1}) \neq \top$. It follows that

$$\varsigma_c(\overleftarrow{sh}(w_n ... w_{m-1})) = 0 = sc_c(w_{m-1} ... w_n)$$

Case 3: $\Omega(w_n) > c$

By Case 1 we have that $\varsigma_{\Omega(w_n)}(\overleftarrow{sh}(w_n ... w_{m-1})) = sc_{\Omega(w_n)}(w_{m-1} ... w_n) \leq n_{\Omega(w_n)}$ and accordingly $\overleftarrow{sh}(w_n ... w_{m-1}) \neq \top$. It follows that

$$\varsigma_c(\overleftarrow{sh}(w_n ... w_{m-1})) = \varsigma(\overleftarrow{sh}(w_{n+1} ... w_{m-1})) = sc_c(w_{m-1} ... w_{n+1}) = sc_c(w_{m-1} ... w_n)$$

- $\wp$ is indeed a progress measure

We have to show that the progress measure conditions are fulfilled for all vertices $v \in V$. We distinguish three cases.

Case 1: $v \in V_0 \cap W_0(\mathcal{G})$

Note that for all $w = w_0 w_1 ... w_n \in V^*$ we have that vw is consistent with σ if and only if w is consistent with σ and $w_0 = \sigma(v)$. We have that $\wp(v) \geq \wp(\sigma(v)) \oplus \Omega(v)$ by

$$\begin{aligned} & \wp(v) \\ = \ & max\{\ \overleftarrow{sh}(vw) \mid vw \in \mathrm{Pref}(\rho) \wedge \rho \in \mathrm{Plays}(\mathcal{A}, \sigma, v)\ \} \\ = \ & max\{\ sh(w_n w_{n-1} ... w_0 v) \mid vw \in \mathrm{Pref}(\rho) \wedge \rho \in \mathrm{Plays}(\mathcal{A}, \sigma, v)\ \} \\ = \ & max\{\ sh(w_n w_{n-1} ... w_0) \oplus \Omega(v) \mid w \in \mathrm{Pref}(\rho) \wedge \rho \in \mathrm{Plays}(\mathcal{A}, \sigma, \sigma(v))\ \} \\ = \ & max\{\ sh(w_n w_{n-1} ... w_0) \mid w \in \mathrm{Pref}(\rho) \wedge \rho \in \mathrm{Plays}(\mathcal{A}, \sigma, \sigma(v))\ \} \oplus \Omega(v) \\ = \ & max\{\ \overleftarrow{sh}(w) \mid w \in \mathrm{Pref}(\rho) \wedge \rho \in \mathrm{Plays}(\mathcal{A}, \sigma, \sigma(v))\ \} \oplus \Omega(v) \\ = \ & \wp(\sigma(v)) \oplus \Omega(v) \end{aligned}$$

Case 2: $v \in V_1 \cap W_0(\mathcal{G})$

Note that for all $w = w_0 w_1 ... w_n \in V^*$ we have that vw is consistent with σ if and only if w is consistent with σ and $(v, w_0) \in E$. We have that $\wp(v) \geq \wp(v') \oplus \Omega(v)$ for all $(v, v') \in E$ by

$$
\begin{aligned}
& \wp(v) \\
= \;& max\{\, \overleftarrow{sh}(vw) \mid vw \in \mathrm{Pref}(\rho) \wedge \rho \in \mathrm{Plays}(\mathcal{A},\sigma,v) \,\} \\
= \;& max\{\, sh(w_n w_{n-1} ... w_0 v) \mid vw \in \mathrm{Pref}(\rho) \wedge \rho \in \mathrm{Plays}(\mathcal{A},\sigma,v) \,\} \\
= \;& max\{\, sh(w_n w_{n-1} ... w_0) \oplus \Omega(v) \mid w \in \mathrm{Pref}(\rho) \wedge \rho \in \mathrm{Plays}(\mathcal{A},\sigma,v') \wedge (v,v') \in E \,\} \\
= \;& max\{\, max\{\, sh(w_n w_{n-1} ... w_0) \mid w \in \mathrm{Pref}(\rho) \wedge \rho \in \mathrm{Plays}(\mathcal{A},\sigma,v') \,\} \oplus \Omega(v) \mid (v,v') \in E \,\} \\
= \;& max\{\, max\{\, \overleftarrow{sh}(w) \mid w \in \mathrm{Pref}(\rho) \wedge \rho \in \mathrm{Plays}(\mathcal{A},\sigma,v') \,\} \oplus \Omega(v) \mid (v,v') \in E \,\} \\
= \;& max\{\, \wp(v') \oplus \Omega(v) \mid (v,v') \in E \,\} \\
\geq \;& \wp(v') \oplus \Omega(v) \text{ for all } (v,v') \in E
\end{aligned}
$$

Case 3: $v \in W_1(\mathcal{G})$

We have that $\wp(v) = \top$ which is the maximal element such that $\wp(v) \geq \wp(v') \oplus \Omega(V)$ for all $(v,v') \in E$.

□

Definition 2.22. Let $\mathcal{G} = (\mathcal{A}, \textsc{parity}(\Omega))$ be a parity game with $\mathcal{A} = (V, V_0, V_1, E)$ and let $\mathcal{P}_\mathcal{G} = \{\, \wp \mid \wp \colon V \to \mathrm{Sh}(\mathcal{G}) \,\}$ with $\wp, \wp' \in \mathcal{P}_\mathcal{G}$. We define $\wp$ to be *smaller or equal than* $\wp'$, denoted by $\wp \sqsubseteq \wp'$, iff

$$\forall v \in V.\ \wp(v) \leq \wp'(v)$$

We use $\wp \sqsubset \wp'$ to denote $\wp \sqsubseteq \wp' \wedge \wp \neq \wp'$.

Corollary 2.3. *Let $\mathcal{G}$ be a parity game. Then it holds that $L = (\mathcal{P}_\mathcal{G}, \sqsubseteq)$ is a complete lattice.*

Definition 2.23. Let $\mathcal{G} = (\mathcal{A}, \textsc{parity}(\Omega))$ be a parity game with $\mathcal{A} = (V, V_0, V_1, E)$. Then for every $v \in V$ we define the function $\mathrm{Lift}_v \colon \mathcal{P}_\mathcal{G} \to \mathcal{P}_\mathcal{G}$ as follows:

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

Lemma 2.8. *Let $\mathcal{G} = (\mathcal{A}, \textsc{parity}(\Omega))$ be a parity game with $\mathcal{A} = (V, V_0, V_1, E)$. Then it holds for all $v \in V$ that Lift_v is $\sqsubseteq$-monotone, i.e.*

$$\forall \wp, \wp' \in \mathcal{P}_\mathcal{G}.\ \wp \sqsubseteq \wp' \ \Rightarrow\ \mathrm{Lift}_v(\wp) \sqsubseteq \mathrm{Lift}_v(\wp')$$

Proof. By Definition 2.22 we have to show that

$$\forall v \in V.\ \forall \wp, \wp' \in \mathcal{P}_\mathcal{G}.\ \forall u \in V.\ \wp(u) \leq \wp'(u) \Rightarrow \mathrm{Lift}_v(\wp)(u) \leq \mathrm{Lift}_v(\wp')(u)$$

Accordingly, let $\wp, \wp' \in \mathcal{P}_\mathcal{G}$ and $v, u \in V$ be arbitrary.

Case 1: $u \neq v$

$\Rightarrow\ \mathrm{Lift}_v(\wp)(u) = \wp(u) \leq \wp'(u) = \mathrm{Lift}_v(\wp')(u)$

Case 2: $u = v \wedge u \in V_0$

$$
\begin{aligned}
& \mathrm{Lift}_v(\wp)(u) \\
= \;& max\{\wp(u), min\{\, \wp(v') \oplus \Omega(v) \mid (v,v') \in E \,\}\} \\
\underset{\wp(u) \leq \wp'(u)}{\leq} \;& max\{\wp'(u), min\{\, \wp(v') \oplus \Omega(v) \mid (v,v') \in E \,\}\} \\
\underset{\text{Lem. 2.6}}{\leq} \;& max\{\wp'(u), min\{\, \wp'(v') \oplus \Omega(v) \mid (v,v') \in E \,\}\} \\
= \;& \mathrm{Lift}_v(\wp)(u)
\end{aligned}
$$

Case 3: $u = v \wedge u \in V_1$

$$
\begin{array}{rl}
 & \mathrm{Lift}_v(\wp)(u) \\
= & max\{\wp(u), max\{\, \wp(v') \oplus \Omega(v) \mid (v,v') \in E \,\}\} \\
\underset{\wp(u) \leq \wp'(u)}{\leq} & max\{\wp'(u), max\{\, \wp(v') \oplus \Omega(v) \mid (v,v') \in E \,\}\} \\
\underset{\text{Lem. 2.6}}{\leq} & max\{\wp'(u), max\{\, \wp'(v') \oplus \Omega(v) \mid (v,v') \in E \,\}\} \\
= & \mathrm{Lift}_v(\wp)(u)
\end{array}
$$

□

Lemma 2.9. *Let $\mathcal{G} = (\mathcal{A}, \text{PARITY}(\Omega))$ be a parity game with $\mathcal{A} = (V, V_0, V_1, E)$. It holds that $\wp$ is a progress measure for $\mathcal{G}$ iff for all $v \in V$ it holds that $\mathrm{Lift}_v(\wp) \sqsubseteq \wp$, i.e. $\wp$ is a pre-fixed point.*

Proof. Let $\mathcal{G} = (\mathcal{A}, \text{PARITY}(\Omega))$ be some parity game with $\mathcal{A} = (V, V_0, V_1, E)$.

"⇒":

Assume that $\wp$ is a progress measure and let $v \in V$ be arbitrary. By Definition 2.22 it is sufficient to show that $\wp(u) \geq \mathrm{Lift}_v(\wp)(u)$ for all $u \in V$. We distinguish three cases.

Case 1: $v \neq u$

We have that $\mathrm{Lift}_v(\wp)(u) = \wp(u)$ and accordingly $\wp(u) \geq \wp(u)$.

Case 2: $v = u \wedge u \in V_0$

$$
\begin{array}{ll}
\Rightarrow & \exists u' \in V.\ (u,u') \in E \wedge \wp(u) \geq \wp(u') \oplus \Omega(u) \\
\Rightarrow & \wp(u) \geq min\{\, \wp(u') \oplus \Omega(u) \mid (u,u') \in E \,\} \\
\Rightarrow & \wp(u) \geq max\{\wp(u), min\{\, \wp(u') \oplus \Omega(u) \mid (u,u') \in E \,\} \\
\Rightarrow & \wp(u) \geq \mathrm{Lift}_v(\wp)(u)
\end{array}
$$

Case 3: $v = u \wedge u \in V_1$

$$
\begin{array}{ll}
\Rightarrow & \forall u' \in V.\ (u,u') \in E \Rightarrow \wp(u) \geq \wp(u') \oplus \Omega(u) \\
\Rightarrow & \wp(u) \geq max\{\, \wp(u') \oplus \Omega(u) \mid (u,u') \in E \,\} \\
\Rightarrow & \wp(u) \geq max\{\wp(u), max\{\, \wp(u') \oplus \Omega(u) \mid (u,u') \in E \,\} \\
\Rightarrow & \wp(u) \geq \mathrm{Lift}_v(\wp)(u)
\end{array}
$$

"⇐":

Let $\mathrm{Lift}_v(\wp) \sqsubseteq \wp$ for all $v \in V$. By Definition 2.22 we know that $\forall v \in V.\ \forall u \in V.\ \wp(u) \geq \mathrm{Lift}_v(\wp)(u)$. We have two show two statements:

1. $\forall v \in V_0.\ \exists v' \in V.\ (v,v') \in E \wedge \wp(v) \geq \wp(v') \oplus \Omega(v)$

   Let $v \in V_0$ be arbitrary, then for $u = v$ it holds that

   $$
   \begin{array}{ll}
   & \wp(v) \geq \mathrm{Lift}_v(\wp)(v) \\
   \Rightarrow & \wp(v) \geq max\{\wp(v), min\{\, \wp(v') \oplus \Omega(v) \mid (v,v') \in E \,\}\} \\
   \Rightarrow & \wp(v) \geq min\{\, \wp(v') \oplus \Omega(v) \mid (v,v') \in E \,\} \\
   \Rightarrow & \exists v' \in V.\ (v,v') \in E \wedge \wp(v) \geq \wp(v') \oplus \Omega(v)
   \end{array}
   $$

2. $\forall v \in V_1.\ \forall v' \in V.\ (v,v') \in E \Rightarrow \wp(v) \geq \wp(v') \oplus \Omega(v)$

   Let $v \in V_1$ be arbitrary, then for $u = v$ it holds that

   $$
   \begin{array}{ll}
   & \wp(v) \geq \mathrm{Lift}_v(\wp)(v) \\
   \Rightarrow & \wp(v) \geq max\{\wp(v), max\{\, \wp(v') \oplus \Omega(v) \mid (v,v') \in E \,\}\} \\
   \Rightarrow & \wp(v) \geq max\{\, \wp(v') \oplus \Omega(v) \mid (v,v') \in E \,\} \\
   \Rightarrow & \forall v' \in V.\ (v,v') \in E \wedge \wp(v) \geq \wp(v') \oplus \Omega(v)
   \end{array}
   $$

□

Construction 2.3. Let $\mathcal{G} = (\mathcal{A}, \text{PARITY}(\Omega))$ be a parity game with $\mathcal{A} = (V, V_0, V_1, E)$. Further let $\wp_0$ be the unique element in $\mathcal{P}_\mathcal{G}$ such that $\wp_0(v) = \vec{0}$ for all $v \in V$. The *progress measure construction* on $\mathcal{G}$ is then defined for all $n \in \mathbb{N}$ as:

$$\wp^n_\mathcal{G} = \begin{cases} \wp_0 & \text{if } n = 0 \\ \text{Lift}_v(\wp^{n-1}_\mathcal{G}) & \text{if } n > 0 \wedge \exists v \in V.\ \wp^{n-1}_\mathcal{G} \sqsubset \text{Lift}_v(\wp^{n-1}_\mathcal{G}) \\ \wp^{n-1} & \text{if } n > 0 \wedge \forall v \in V.\ \text{Lift}_v(\wp^{n-1}_\mathcal{G}) \sqsubseteq \wp^{n-1}_\mathcal{G} \end{cases}$$

$$\wp_\mathcal{G} = \max\{\, \wp^n_\mathcal{G} \mid n \in \mathbb{N} \,\}$$

Lemma 2.10. *Let $\mathcal{G} = (\mathcal{A}, \text{PARITY}(\Omega))$ be a parity game with $\mathcal{A} = (V, V_0, V_1, E)$. Then it holds that $W_0(\mathcal{G}) = \|\wp_\mathcal{G}\|$ and $W_1(\mathcal{G}) = V \setminus \|\wp_\mathcal{G}\|$.*

Proof. By Lemma 2.7 *ii* there exists a progress measure $\wp^*$ with $\|\wp^*\| = W_0(\mathcal{G})$. The following holds:

- *$\wp_\mathcal{G}$ is a least pre-fixed point of the* Lift_v*-operators*

  Directly follows from Construction 2.3, Lemma 2.8 and the Knaster–Tarski theorem.

- *$\wp^*$ is a pre-fixed point of the* Lift_v*-operators*

  Directly follows from Lemma 2.9.

We can conclude that $\wp_\mathcal{G} \sqsubseteq \wp^*$, i.e. $\wp_\mathcal{G}(v) \leq \wp^*(v)$ for all $v \in V$. Thus, by $\|\wp^*\| = W_0(\mathcal{G})$, we also have that $\wp_\mathcal{G}(v) \leq \wp^*(v) < \top$ for all $v \in W_0(\mathcal{G})$. It follows that $W_0(\mathcal{G}) \subseteq \|\wp_\mathcal{G}\|$ and together with Lemma 2.7 *i* we get $W_0(\mathcal{G}) = \|\wp_\mathcal{G}\|$. Finally by Theorem 2.5 it follows that $W_1(\mathcal{G}) = V \setminus \|\wp_\mathcal{G}\|$. □

Reconsider the example given in Figure 10. Using the progress measure algorithm we can now solve the game. An example execution is depicted in Figure 11. As we have already discussed earlier the result is indeed a progress measure such that we get for the winning regions $W_0 = \{v_3, v_6, v_7\}$ and $W_1 = \{v_0, v_1, v_2, v_4, v_5, v_8\}$.

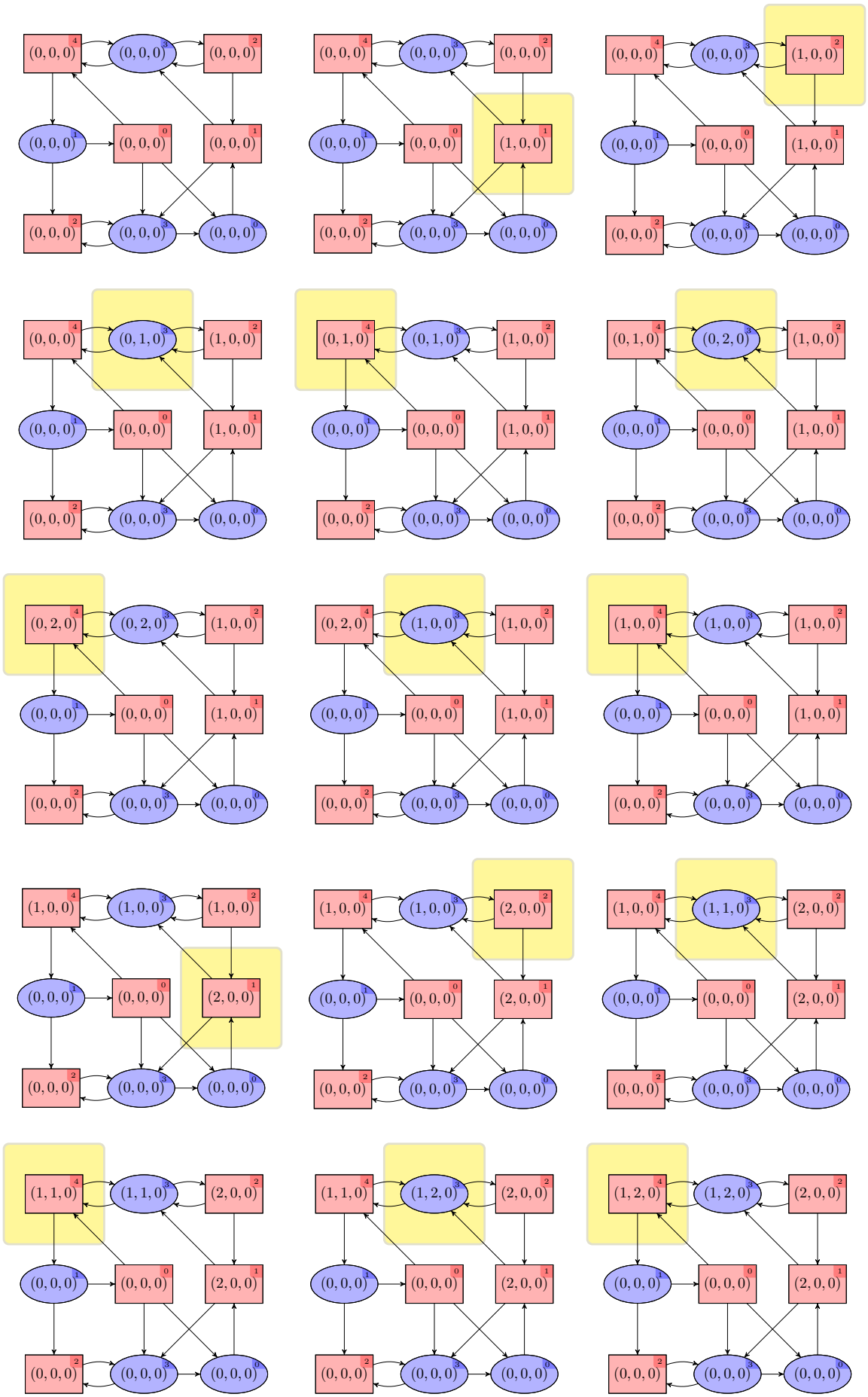
Theorem 2.6. *Every parity game $\mathcal{G} = (\mathcal{A}, \text{PARITY}(\Omega\colon V \to [k]))$ with $\mathcal{A} = (V, V_0, V_1, E)$ can be solved in $\mathcal{O}(k \cdot log|V| \cdot |E| \cdot (\frac{|V|}{\lceil \frac{1}{2}k \rceil})^{\lceil \frac{1}{2}k \rceil})$ time and $\mathcal{O}(|V| \cdot log|V| \cdot k)$ space.*

Proof. We use Construction 2.3 to solve parity games. By Lemma 2.10 we get the winning regions for both players and from the proof of Lemma 2.7 *i* it follows that we also get a positional winning strategy for Player 0. As by Lemma 2.5 parity games are self-dual we also get a winning strategy for Player 1 by solving the dual game.

In Construction 2.3 we only need to store the actual $\wp^i_\mathcal{G}$ of each iteration i. We have that $\wp^i_\mathcal{G}$ only consists of a score sheets for every vertex $v \in V$ and score sheets are tuples of length $\frac{k}{2}$. Finally we have that a score sheet only contains values of size at most $|V|$ such that they can be stored in $log|V|$. It directly follows that we need at most $\mathcal{O}(k \cdot |V| \cdot log|V|)$ space. The number of iterations is bounded by $|\text{Sh}(\mathcal{G})|$ and in each iteration we have to execute Lift_v iterations for all $v \in V$ which have to inspect each edge at most once. Reading and writing the score sheets needs at least $k \cdot log|V|$ time such that the upper time bound finally follows from

$$|\text{Sh}(\mathcal{G})| \quad \leq \quad \prod_{\substack{c \in [k], \\ \text{Par}(c) = 1}} (n_c + 1) \quad \leq \quad (\tfrac{|V|}{\lceil \frac{1}{2}k \rceil})^{\lceil \frac{1}{2}k \rceil}$$

where the last inequation follows from the fact, that the sum of all n_c exactly sums up to $|V|$ and choosing an equalized portion of vertices for each color maximizes the product. □


(0, 0, 0)
(1, 0, 0)
(0, 1, 0)
(0, 2, 0)
(2, 0, 0)
(1, 1, 0)
(1, 2, 0)

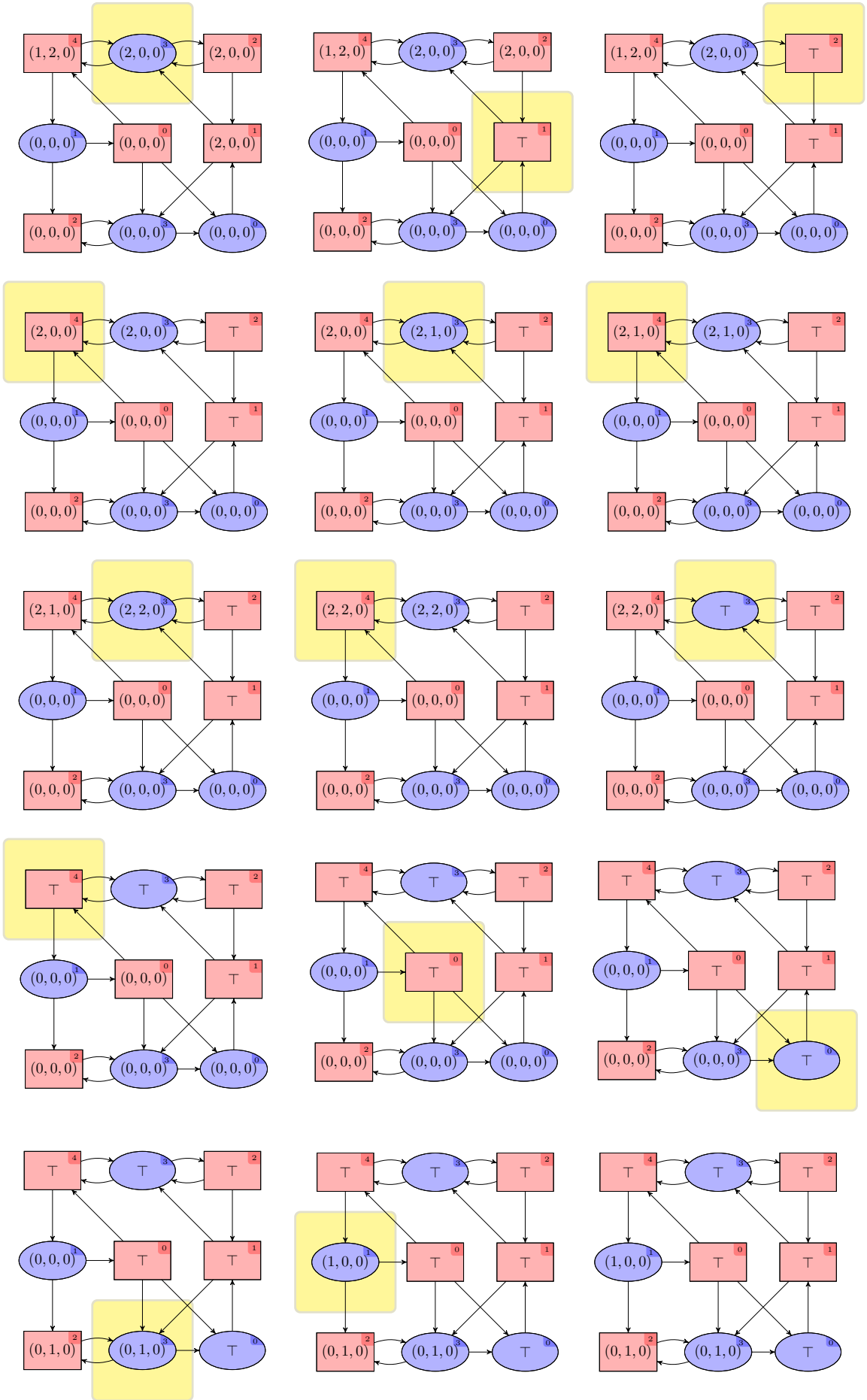


Figure 11: Progress measure algorithm for the game of Figure 10. The Lift$_v$-operations are marked.

3 Memory

3.1 Finite State Strategies

In the last chapter, we already have seen several games which were all positionally determined with uniform winning strategies. Accordingly, there was always an easy way to represent the corresponding winning strategies directly. We now want to leave this world and come to the first games that are not positionally determined any more. As a first example consider the game $\mathcal{G} = (\mathcal{A}, \text{Win})$ depicted in Figure 12. In this game, the winning condition is defined as $\text{Win} = \{\, \rho \in \text{Plays}(\mathcal{A}) \mid \text{Inf}(\rho) = \{v_0, v_1, v_2\} \,\}$ such that the goal of Player 0 is to visit every vertex infinitely often. This goal is not reachable using a positional strategy since then Player 0 would always go from v_1 to v_0 or to v_2. Accordingly, if starting in v_1 every consistent play would never reach the other vertex she is not going to. However, Player 0 still can win the game from every vertex, she only has to alternate between the two possible successors. Thus, a possible winning strategy σ for all histories $w \in V^*$ would be given as follows:

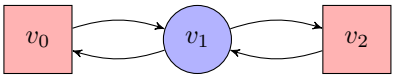


Figure 12: Arena $\mathcal{A}$ for the game $\mathcal{G} = (\mathcal{A}, \text{Win})$ with $\text{Win} = \{\, \rho \in \text{Plays}(\mathcal{A}) \mid \text{Inf}(\rho) = \{v_0, v_1, v_2\} \,\}$.

$$\sigma(wv_1) = \begin{cases} v_2 & \text{if } \text{Lst}(w) = v_0 \\ v_0 & \text{otherwise} \end{cases}$$

Consider that this is only one way to describe this strategy. An alternative representation is given by using a finite automaton, that reads a play history w as input and outputs the desired successor $\sigma(w)$. The corresponding automaton for our strategy σ can be found in Figure 13.

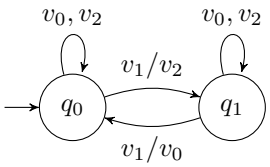


Figure 13: Automaton representing the finite state strategy σ from the example of Figure 12.

Our notation for the edge labeling of the automaton can be read as follows. A transition $\xrightarrow{v/v'}$ means that we read the input $v \in V_0$ and output the vertex v'. A transition $\xrightarrow{v}$ means that we only read $v \in V_1$ but output nothing as it is not our turn. An example run of the automaton, starting in v_1, would then be given as follows:

current state:	q_0		q_1		q_1		q_0		q_0		q_1
input/output:		v_1	$\xrightarrow{\text{OUTPUT}}$	v_2		v_1	$\xrightarrow{\text{OUTPUT}}$	v_0		v_1	$\xrightarrow{\text{OUTPUT}}$	v_2

We will now formally define our idea of such an automaton. However, we will not directly use the representation as above as it has some disadvantages for later definitions. Instead, we start with something called a memory structure that more or less represents the memory of a player. The idea is that every state in the memory structure describes a specific situation in the game where we always can answer in the same way. By that, we have a positional decision again. Not on the vertex of the game any more but on the vertex of the game together with the current memory state of our memory structure. We only have to update this state with every move in the arena correspondingly. Formally, we get the following:

Definition 3.1. A *memory structure* $\mathcal{M} = (M, \text{init}, \text{upd})$ for an arena $\mathcal{A} = (V, V_0, V_1, E)$ is a triple consisting of

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

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

Note, that we still miss a corresponding output function that describes our next move in the arena. But at first, we want to move in our memory structure multiple steps at once. Therefore, we extend the corresponding update function as follows.

Definition 3.2. Let $\mathcal{M} = (M, \text{init}, \text{upd})$ be a memory structure for an arena $\mathcal{A} = (V, V_0, V_1, E)$. We define the function $\text{upd}^* \colon V^+ \to M$ as follows:

$$\text{upd}^*(wv) = \begin{cases} \text{init}(v) & \text{if } w = \varepsilon \\ \text{upd}(\text{upd}^*(w), v) & \text{otherwise} \end{cases}$$

Informally, $\text{upd}^*(w)$ gives the memory content that has been reached after processing w. We continue by defining our output function, called the next-move function.

Definition 3.3. Let $\mathcal{A} = (V, V_0, V_1, E)$ be an arena and $\mathcal{M} = (M, \text{init}, \text{upd})$ be a memory structure for $\mathcal{A}$. A *next-move function* for Player i is a mapping $\text{nxt} \colon V_i \times M \to V$ satisfying $(v, \text{nxt}(v, m)) \in E$ for all $v \in V_i$ and $m \in M$.

Together, $\mathcal{M}$ and nxt then define a strategy σ for Player i via $\sigma(w) = \text{nxt}(\text{Lst}(w), \text{upd}^*(w))$. We also say σ is finite-state and implemented by $\mathcal{M}$. We define the size of σ to be $|M|$. This may be a bit abusive, since σ might also be implementable by smaller memory structures. However, we will only use the size of a strategy, if the memory structure, that implements it is clear from the context. Finally note, that if σ is positional, it can be implemented by a memory structure of size one, and vice versa.

3.2 Reductions

We now are interested in how we can compute such finite state strategies. Remember, that our idea for a memory structure was that we can use a positional strategy on the memory structure again. But we have to somehow connect it to the underlying arena again. Therefore, we use a cartesian product construction.

Definition 3.4. Let $\mathcal{A} = (V, V_0, V_1, E)$ be an arena and $\mathcal{M} = (M, \text{init}, \text{upd})$ be a memory structure for $\mathcal{A}$. We denote by $\mathcal{A} \times \mathcal{M}$ the *by $\mathcal{A}$ and $\mathcal{M}$ induced and expanded arena*, defined as $\mathcal{A} \times \mathcal{M} = (V \times M, V_0 \times M, V_1 \times M, E')$ with

$$((v, m), (v', m')) \in E' \quad \Leftrightarrow \quad (v, v') \in E \wedge \text{upd}(m, v') = m'$$

for all $v, v' \in V$ and $m, m' \in M$.

A play through the extended arena can then be constructed from a play through the original arena as follows.

Definition 3.5. Let $\mathcal{A} = (V, V_0, V_1, E)$ be an arena, $\mathcal{M} = (M, \text{init}, \text{upd})$ be a memory structure for $\mathcal{A}$ and $\rho \in \text{Plays}(\mathcal{A})$. We define the *extended play* $\text{ext}(\rho) = (\rho_0, m_0)(\rho_1, m_1)... \in \text{Plays}(\mathcal{A} \times \mathcal{M})$ with $m_0 = \text{init}(\rho_0)$ and m_n recursively defined as $m_n = \text{upd}(m_{n-1}, \rho_n)$ for all $n \in \mathbb{N}^+$.

Consider that it also holds that $m_n = \text{upd}^*(\rho[n])$. But how can we use this extended arena now to solve our original game. The idea is, that we extend the arena by choosing the right memory structure in such a way, that it represents a simpler game, where we can use positional strategies again. If this game is of a form that we already know we can solve it and the resulting winning strategies gives us together with the memory structure a winning strategy for the original game. Accordingly, we more or less reduce the harder game to a simpler game by choosing the right memory structure. Such a reduction can be formally expressed a follows.

Definition 3.6. Let $\mathcal{G} = (\mathcal{A}, \text{Win})$ and $\mathcal{G}' = (\mathcal{A}', \text{Win}')$ be two games and $\mathcal{M}$ be a memory structure for $\mathcal{A}$. We say $\mathcal{G}$ *is reducible to* $\mathcal{G}'$ *via memory structure* $\mathcal{M}$, denoted by $\mathcal{G} \leq_{\mathcal{M}} \mathcal{G}'$, iff

1. $\mathcal{A}' = \mathcal{A} \times \mathcal{M}$
2. $\forall \rho \in \text{Plays}(\mathcal{A}).\ \rho \in \text{Win} \ \Leftrightarrow\ \text{ext}(\rho) \in \text{Win}'$

We can finalize our idea with the following lemma.

Lemma 3.1. *Let $\mathcal{G} = (\mathcal{A}, \mathrm{Win})$ be a game with $\mathcal{A} = (V, V_0, V_1, E)$ and $V' \subseteq V$. Then, if $\mathcal{G} \leq_{\mathcal{M}} \mathcal{G}_p$, for some game $\mathcal{G}_p$, and Player i has positional winning strategy σ_p in $\mathcal{G}_p$ from $\{ (v, \mathrm{init}(v)) \mid v \in V' \}$, she also has a finite-state winning strategy σ with memory $\mathcal{M}$ in $\mathcal{G}$ from V'.*

Proof. Let $\mathcal{M} = (M, \mathrm{init}, \mathrm{upd})$. Then by Definition 3.6 we have that $\mathcal{G}_p = (\mathcal{A} \times \mathcal{M}, \mathrm{Win}_p)$ for some winning condition Win_p and with $\mathcal{A} \times \mathcal{M} = (V \times M, V_0 \times M, V_1 \times M, E')$. We define $\mathrm{nxt}(v, m) = v'$ for $\sigma_p(v, m) = (v', m')$ for all $m, m' \in M$, $v \in V_i$ and $v' \in V$. Let then σ be defined as the induced strategy by $\sigma(w) = \mathrm{nxt}(\mathrm{Lst}(w), \mathrm{upd}^*(w))$ for all $w \in V^*$.

We have to show that σ is winning from every $v \in V'$. So let $v \in V'$ and $\rho \in \mathrm{Plays}(\mathcal{A}, \sigma, v)$ be arbitrary. Further, let $\rho' = \mathrm{ext}(\rho)$, which is a play of $\mathcal{G}_p$ starting in $(v, \mathrm{init}(v))$. We first show that $\rho' \in \mathrm{Plays}(\mathcal{A} \times \mathcal{M}, \sigma_p, (v, \mathrm{init}(v)))$. To do so, we show that every prefix $(\rho_0, m_0)...(\rho_n, m_n) \in \mathrm{Pref}(\rho')$ is consistent with σ_p. The proof goes via induction over $n \in \mathbb{N}$.

Induction Base: $n = 0$

The prefix $(\rho_0, m_0) = (v, \mathrm{init}(v))$ is trivially consistent with σ_p.

Induction Hypothesis:

$\forall n \in \mathbb{N}.\ w \in \mathrm{Pref}(\rho') \wedge |w| = n + 1 \Rightarrow w$ is consistent with σ_p

Induction Step: $n > 0$

By induction hypothesis $\rho'[n-1] = (\rho_0, m_0)(\rho_1, m_1)...(\rho_{n-1}, m_{n-1})$ is consistent with σ_p. If it is not Player i's turn, i.e. $(\rho_{n-1}, m_{n-1}) \in V_{i-1} \times M$, then $\rho'[n-1](\rho_n, m_n)$ is trivially consistent with σ_p, so let $(\rho_{n-1}, m_{n-1}) \in V_i \times M$. It follows that $\rho_{n-1} \in V_i$ and correspondingly, that $\rho_n = \sigma(\rho[n-1]) = \mathrm{nxt}(\rho_{n-1}, \mathrm{upd}^*(\rho[n-1]))$. Thus, by Definition 3.1

$$\exists m \in M.\ \sigma_p(\rho_{n-1}, \mathrm{upd}^*(\rho[n-1])) = \sigma_p(\rho_{n-1}, m_{n-1}) = (\rho_n, m)$$

It remains to show that $m = m_n$. We have $m_n = \mathrm{upd}(m_{n-1}, \rho_n)$ by Definition 3.5. Finally, with Definition 3.4 we get

$$\sigma_p(\rho_{n-1}, m_{n-1}) = (\rho_n, m) \quad \Rightarrow \quad ((\rho_{n-1}, m_{n-1}), (\rho_n, m)) \in E' \quad \Rightarrow \quad m = \mathrm{upd}(m_{n-1}, \rho_n) = m_n$$

We now have that $\mathrm{ext}(\rho) \in \mathrm{Plays}(\mathcal{A} \times \mathcal{M}, \sigma_p, (v, \mathrm{init}(v))$ for every $v \in V'$. As σ_p is a winning strategy, it follows that $\mathrm{ext}(\rho) \in \mathrm{Win}_p$ and with Definition 3.6 we can conclude that $\rho \in \mathrm{Win}$. Thus, σ is winning for every $v \in V'$. □

Corollary 3.1. *Let $\mathcal{G} = (\mathcal{A}, \mathrm{Win}) \leq_{\mathcal{M}} (\mathcal{A} \times \mathcal{M}, \mathrm{Win}') = \mathcal{G}'$ for some arena $\mathcal{A} = (V, V_0, V_1, E)$. Then the following holds:*

- *$W_i(\mathcal{G}) = \{ v \in V \mid (v, \mathrm{init}(v)) \in W_i(\mathcal{G}') \}$ for both Player i.*
- *If $\mathcal{G}'$ is determined with uniform positional winning strategies then $\mathcal{G}$ is determined with uniform finite state strategies implemented by $\mathcal{M}$.*

Let us continue with some examples. Therefore, we first need a new type of game that we can reduce to a game type where we already know how to solve it. Such a new type is given by weak Muller games.

Definition 3.7. Let the weak Muller condition $\textsc{wmuller}(\mathcal{F})$ on a set $\mathcal{F} \subseteq 2^V$ for an arena $\mathcal{A} = (V, V_0, V_1, E)$ be defined as:

$$\textsc{wmuller}(\mathcal{F}) := \{ \rho \in \mathrm{Plays}(\mathcal{A}) \mid \mathrm{Occ}(\rho) \in \mathcal{F} \}$$

Then we call the game $\mathcal{G} = (\mathcal{A}, \textsc{wmuller}(\mathcal{F}))$ a *weak Muller game* with acceptance set $\mathcal{F}$.

We reduce weak Muller games to weak parity games as follows. Remember that we have defined weak parity games in Exercise 4.2.

Lemma 3.2. *Weak Muller games are reducible to weak parity games.*

Proof. Let $\mathcal{G} = (\mathcal{A}, \text{WMULLER}(\mathcal{F}))$ be a weak Muller game with $\mathcal{A} = (V, V_0, V_1, E)$. We choose a memory structure $\mathcal{M} = (M, \text{init}, \text{upd})$ with $M = 2^V$, $\text{init}(v) = \{v\}$ and $\text{upd}(m, v) = S \cup \{v\}$ for all $v \in V$ and $m \in M$. We want to show that $\mathcal{G} = (\mathcal{A}, \text{WMULLER}(\mathcal{F})) \leq_{\mathcal{M}} (\mathcal{A} \times \mathcal{M}, \text{WPARITY}(\Omega)) = \mathcal{G}'$ where Ω is defined as

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

We have that $\text{upd}^*(w) = \text{Occ}(w)$ for all $w \in V^*$. Let $\rho \in \text{Plays}(\mathcal{A})$ be arbitrary. It follows by construction that

$$\exists n \in \mathbb{N}.\ \forall m \geq n.\ \text{Occ}(\rho[m]) = \text{Occ}(\rho) \wedge \forall m' < n.\ \text{Occ}(\rho[m']) \subseteq \text{Occ}(\rho[n])$$

Now consider the play $\text{ext}(\rho) = (\rho_0, m_0)(\rho_1, m_1)... \in \text{Plays}(\mathcal{A} \times \mathcal{M})$. By the conditions above we get that

- $\forall m' < n.\ \Omega(\rho_{n'}, m_{n'}) \geq \Omega(\rho_n, m_n)$
- $\forall m \geq n.\ \Omega(\rho_{n'}, m_{n'}) = \Omega(\rho_n, m_n)$

It follows that the minimal color in $\text{ext}(\rho)$ is $\Omega(\rho_n, m_n)$ which is even if $m_n = \text{Occ}(\rho) \in \mathcal{F}$ and odd otherwise. It follows that

$$\rho \in \text{WMULLER}(\mathcal{F}) \quad \Leftrightarrow \quad \text{ext}(\rho) \in \text{PARITY}(\Omega)$$

concluding the proof. □

It follows that we can solve weak Muller games by reducing them to weak parity games. However, for this reduction we use a memory structure of exponential size such that the resulting winning strategies for the weak Muller games can be of exponential size as well. The question arises, whether we still can improve our memory structure to get smaller strategies? It turns out that this is not possible and we will now show why this is the case. Therefore, we show the following.

Lemma 3.3. *There exits a family $\mathcal{G}_n = (\mathcal{A}_n, \text{WMULLER}(\mathcal{F}_n))$ of weak muller games with $|\mathcal{F}| = 2$ and $|\mathcal{A}_n| \in \mathcal{O}(n)$ such that every $\mathcal{A}_n$ has a designated vertex v where*

- *Player 0 has a winning strategy from v, but*
- *every finite-state winning strategy for Player 0 from v has at least 2^n states.*

Proof. Consider the arena $\mathcal{A}_n$, depicted in Figure 14, with vertex set V_n. It contains n widgets made of the vertices s_j, h_j, u_j, and d_j for all $j \in \mathbb{N}$ as well as 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. We define $\mathcal{F}_n = \{V_n \setminus \{v_A\}, V_n \setminus \{v_B\}\}$, i.e. Player 0 needs to visit all vertices except for v_A or all vertices except for v_B. Notice that in the case where the play reaches the vertex c_1 twice Player 1 can win by going to v_A one time and to v_B the other time. As a consequence, $\mathcal{A}_n$ and $\mathcal{F}_n$ satisfy the requirements formulated by the lemma.

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 ever 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 for one of the set $\{v_A, v_B\}$ is visited.

Now, assume that Player 0 has a 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 $j \in [1, n]$ 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[1]. 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 game since $\mathcal{F}_n$ contains no set containing both, v_A and v_B.

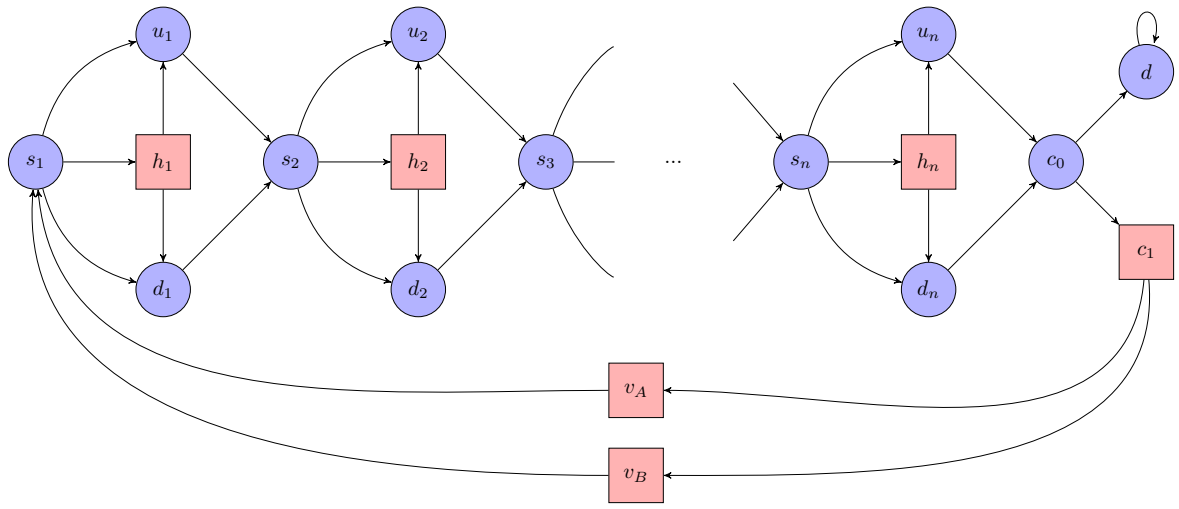


Figure 14: The arena $\mathcal{A}_n$ for the weak Muller game $\mathcal{G}_n$ of Lemma 3.3.

As a consequence, Player 0 always has to move to h_j when at vertex s_j in the second 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. $\text{upd}^*(w) = \text{upd}^*(w')$ for two different play prefixes w and w' from s_1 to c_0. Hence, from c_0 onward, the strategy will makes the same moves to extend the play prefixes w and w'. Accordingly, the play continuations of these prefixes can only differ in a choices of Player 1, like at a vertex h_j, e.g. the continuation of w can visit u_j while the continuation of w' will visit d_j. Hence, the strategy has to visit d_j after the play prefix w and u_j after the prefix w'. However, this is impossible since it prescribes the same moves for w and w'. Hence, one of the corresponding plays will be losing contradicting the fact that σ is a winning strategy for Player 0 from s_1. □

3.3 Muller Games

We now will analyze more complex games and reductions for them. We start with Muller games.

Definition 3.8. Let the Muller condition $\text{MULLER}(\mathcal{F})$ on a set $\mathcal{F} \subseteq 2^V$ for an arena $\mathcal{A} = (V, V_0, V_1, E)$ be defined as

$$\text{MULLER}(\mathcal{F}) := \{\, \rho \in \text{Plays}(\mathcal{A}) \mid \text{Inf}(\rho) \in \mathcal{F} \,\}$$

Then we call the game $\mathcal{G} = (\mathcal{A}, \text{MULLER}(\mathcal{F}))$ a *Muller game* with acceptance set $\mathcal{F}$.

As an example consider the family of Muller games $DJW_n = (\mathcal{A}_n, \text{MULLER}(\mathcal{F}_n))$ with $\mathcal{A}_n$ depicted in Figure 15 and formally defined as $\mathcal{A}_n = (V, V_0, V_1, E)$ with $V = V_0 \cup V_1$, $V_0 = \{\, v_j \mid j \in [1, n] \,\}$, $V_1 = \{\, v'_j \mid j \in [1, n] \,\}$ and $E = V_0 \times V_1 \cup V_1 \times V_0$ and $\mathcal{F}_n = \{\, F \subseteq V \mid |F \cap V_0| = max\{\, j \in [1, n] \mid v'_j \in F \cap V_1 \,\} \,\}$. In this game, Player 1 can choose to visit a non-empty set $S \subseteq V_0$ infinitely often. The goal of Player 0 is to answer to this choice by choosing a vertex $v_j \in V_1$ infinitely often, such that $j = |S|$. Further, she is not allowed to choose a vertex with higher index infinitely often.

[1] the case of d_j is analogous

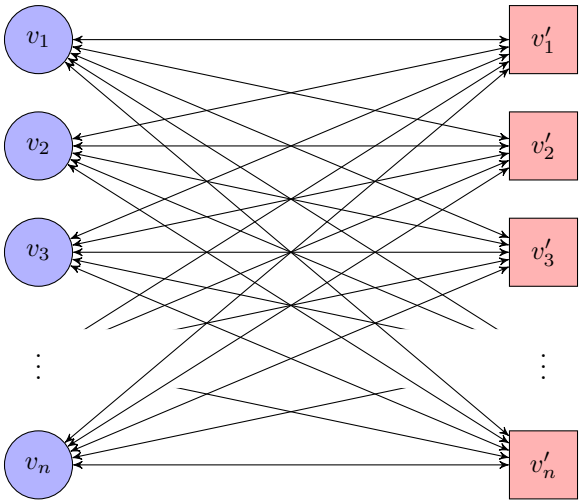


Figure 15: Graphical representation of the arenas for the family of Muller games DJW_n.

Which player wins the games DJW_n? The crucial point here is picking the right memory structure to implement a winning strategy. We claim that Player 0 has a uniform finite-state winning strategy from every vertex using memory that stores in which order the vertices from V_0 appeared for the latest time. As an example consider the game DJW_4 and the play prefix $v_1v_1v_2v_2v_1v_2v_4v_4v_4v_1v_2$. The last vertex from V_0 that appeared is v_2. The second-to-last vertex from V_0 that appeared is v_4. The third-to-last is also v_4 which had a later occurrence, so it is ignored. The only other vertex from V_0 that appeared is v_1. Hence, the memory state for this play should store the list $v_2v_4v_1$. Updating this list when visiting a vertex $v \in V_0$ is done by removing v from the list, if it appears, and putting it in front of the list again. There is no update when visiting vertices in V_1. We need one additional information in our memory to implement a winning strategy. We mark the position where we deleted v by a $\sharp$, e.g. when visiting v_2 we update $v_1\sharp\, v_2v_3v_4$ to $v_2v_1\sharp\, v_3v_4$. The $\sharp$ will be used to determine the infinity set of the plays.

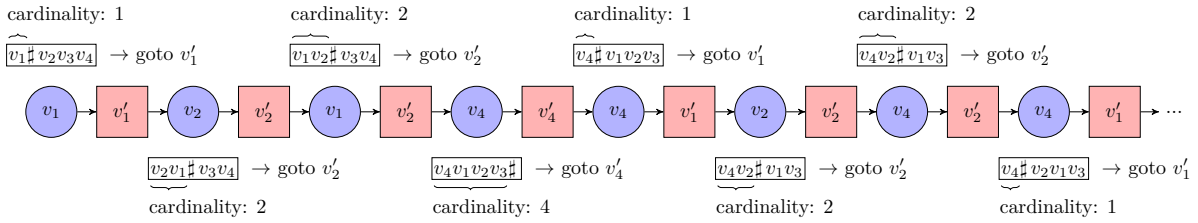


Figure 16: Example play and strategy, where the current memory state is depicted in the additional boxes.

Consider the example play of DJW_4 depicted in Figure 16. The memory states reached are depicted in the boxes above and below the vertices in V_0, 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 set of vertices to the left of $\sharp$ in the current memory state. For example, her first move leads to v'_1, since only one vertex is on the left side of $\sharp$ in the memory state $v_1\sharp\, v_2v_3v_4$. Her second move leads to v'_2 since there are two vertices on the left side of $\sharp$ in the memory state $v_2v_1\sharp\, v_3v_4$.

Now, consider an infinite play ρ that is consistent with this strategy. From some point onward, only vertices from $\mathrm{Inf}(\rho)$ are visited by ρ. After this position, only vertices from $\mathrm{Inf}(\rho) \cap V_0$ are moved to the front of the list and after some time no vertex from $V_0 \setminus \mathrm{Inf}(\rho)$ is on the left of $\sharp$. Furthermore, infinitely often exactly $\mathrm{Inf}(\rho) \cap V_0$ is to the left of $\sharp$. If this would not be the case, then some vertex would stay forever to the right of the $\sharp$, but this vertex is seen infinitely often and every visit moves it to the left of the $\sharp$. Hence, using the strategy described above, Player 0 will infinitely often move to $v'_{|\mathrm{Inf}(\rho)|}$ and only finitely often to some v'_j with $j > |\mathrm{Inf}(\rho)|$. Thus, the strategy is indeed winning for Player 0. In the following we show that a generalization of this book-keeping of the latest appearance of vertices suffices for every Muller game. To this end, we introduce the so called "latest-appearance-record" memory structure or shortly LAR.

Memory structure 3.1. Let $\mathcal{G} = (\mathcal{A}, \text{MULLER}(\mathcal{F}))$ be a Muller game with arena $\mathcal{A} = (V, V_0, V_1, E)$ and $\mathcal{F} \subseteq 2^V$. Further, let $\sharp \notin V$ be some new symbol different from all vertices of $\mathcal{A}$. An LAR over V is a word $l = v_0 v_1 ... v_m \sharp\, v_{m+1} ... v_n \in (V \cup \{\sharp\})^*$ with $m \in [n+1]$ such that

$$V = \{v_0, ..., v_n\} \wedge \forall j, j' \in [n+1].\ (j \neq j' \Rightarrow v_j \neq v_{j'})$$

We use LAR_V to denote the set of all LAR's over V. The memory structure $\mathcal{M}_{\text{LAR}}$ is then defined as $\mathcal{M}_{\text{LAR}} = (\text{LAR}_V, \text{init}, \text{upd})$ where $\text{init}\colon V \to \text{LAR}_V$ is some arbitrary function and

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

for all $l = v_0 v_1 ... v_m \sharp\, v_{m+1} ... v_n \in LAR_V$. Further, we denote by $\text{hit}(l) = \{v_0, ..., v_m\}$, the so called the "hit-set" of l.

Theorem 3.1. *Muller games are reducible to parity games.*

Proof. Let $\mathcal{G} = (\mathcal{A}, \text{MULLER}(\mathcal{F}))$ be a Muller game with arena $\mathcal{A} = (V, V_0, V_1, E)$ and $\mathcal{F} \subseteq 2^V$. We want to show that $\mathcal{G} \leq_{\mathcal{M}_{\text{LAR}}} (\mathcal{A} \times \mathcal{M}_{\text{LAR}}, \text{PARITY}(\Omega))$ for some coloring function Ω. Before we do so, we first will show some general, useful properties. Therefore, let $\rho = \rho_0 \rho_1 \rho_2 ... \in \text{Plays}(\mathcal{A})$ be arbitrary and $l = l_0 l_1 l_2 ...$ be the sequence of LARs corresponding to that play, i.e. $l_n = \text{upd}^*(\rho[n])$ for all $n \in \mathbb{N}$. Further let $S_{q,p} = \{\, \rho_j \mid j \in [q,p] \,\}$ for $p, q \in \mathbb{N}$. Then the following holds:

- $\forall q \in \mathbb{N}.\ \forall p \geq q.\ S_{q,p} = \{\, v_j \mid j \in [|S_{q,p}|] \,\}$ for $l_p = v_0 v_1 ... v_m \sharp\, v_{m+1} ... v_n$ (1)

  We show this by induction on $p - q$.

  Induction Base: $p = q$

  We have that $l_q = l_p = \text{upd}(l_{p-1}, \rho_p) = \rho_p v_1 ... v_{m-1} \sharp\, v_{m+1} ... v_n = v_0 v_1 ... v_{m-1} \sharp\, v_{m+1} ... v_n$. It follows that $S_{p,p} = \{\, \rho_j \mid j \in [p,p] \,\} = \{\rho_p\} = \{v_0\} = \{\, v_j \mid j \in [1] \,\} = \{\, v_j \mid j \in [|S_{p,p}|] \,\}$.

  Induction Hypothesis:

  $\forall q \in \mathbb{N}.\ \forall p \geq q.\ S_{q,p} = \{\, v_j \mid j \in [|S_{q,p}|] \,\}$ for $l_p = v_0 v_1 ... v_m \sharp\, v_{m+1} ... v_n$

  Induction Step: $p > q$

  Let $l_{p-1} = v'_0 ... v'_{m'} \sharp\, v'_{m'+1} ... v'_n$. By induction hypothesis we get $S_{q,p-1} = \{\, v'_j \mid j \in [|S_{q,p-1}|] \,\}$. Further, it holds that $l_p = \text{upd}(l_{p-1}, \rho_p) = \rho_p v'_0 v'_1 ... v'_{m-1} \sharp\, v'_{m+1} ... v'_n$ with $v'_m = \rho_p$. It follows that $v_j = v'_{j-1}$ for all $j \in [1, m]$ and $v_j = v'_j$ for $j \in [m+1, n]$. Further, we get:

  $$S_{q,p} = \{\, \rho_j \mid j \in [q,p] \,\} = \{\, \rho_j \mid j \in [q, p-1] \,\} \cup \{\rho_p\} = S_{q,p-1} \cup \{\rho_p\} = S_{q,p-1} \cup \{v_0\} = \{\, v'_j \mid j \in [|S_{q,p-1}|] \,\} \cup \{v_0\}$$

  It remains to show that $\{\, v'_j \mid j \in [|S_{q,p-1}|] \,\} \cup \{v_0\} = \{\, v_j \mid j \in [|S_{q,p}|] \,\}$. We distinguish two cases:

  Case 1: $m \leq |S_{q,p-1}|$

  It follows $v'_m = \rho_p \in S_{q,p-1}$ such that $|S_{q,p}| = |S_{q,p-1}|$. We get

  $$\begin{aligned} & \{\, v'_j \mid j \in [|S_{q,p-1}|] \,\} \cup \{v_0\} \\ = \ & \{\, v'_j \mid j \in [|S_{q,p}|] \,\} \cup \{v_0\} \\ = \ & \{\, v'_j \mid j \in [0, m-1] \,\} \cup \{\, v'_j \mid j \in [m, |S_{q,p}|] \,\} \cup \{v_0\} \\ = \ & \{\, v_j \mid j \in [1, m] \,\} \cup \{\, v_j \mid j \in [m, |S_{q,p}|] \,\} \cup \{v_0\} \\ = \ & \{\, v_j \mid j \in [|S_{q,p}|] \,\} \end{aligned}$$

  Case 2: $m > |S_{q,p-1}|$

  It follows $v'_m = \rho_p \notin S_{q,p-1}$ such that $|S_{q,p}| = |S_{q,p-1}| + 1$. We get

$$
\begin{aligned}
& \{ v'_j \mid j \in [|S_{q,p-1}|] \} \cup \{v_0\} \\
= \quad & \{ v'_j \mid j \in [0, |S_{q,p}| - 1] \} \cup \{v_0\} \\
= \quad & \{ v_j \mid j \in [1, |S_{q,p}|] \} \cup \{v_0\} \\
= \quad & \{ v_j \mid j \in [|S_{q,p}|] \}
\end{aligned}
$$

- There exists an $n \in \mathbb{N}$ such that $\text{hit}(l_{n'}) \subseteq \text{Inf}(\rho)$ for all $n' \geq n$ (2)

  Let $n_0 = max\{ n \in \mathbb{N} \mid \rho_n \notin \text{Inf}(\rho) \} + 1$, then $\text{Occ}(\rho_{n_0}\rho_{n_0+1}...) = \text{Inf}(\rho)$. Furthermore, let $n_1 \geq n_0$ be such that $\text{Inf}(\rho) = \{ \rho_j \mid j \in [n_0, n_1] \}$. Then by (1) the first $|\text{Inf}(\rho)|$ entries of l_{n_1} are exactly the vertices of $\text{Inf}(\rho)$.

  Now let $n_2 \geq n_1$ be such that $\text{Occ}(\rho[n_1 + 1, n_2]) = \text{Inf}(\rho)$. Then as before, by (1) the first $|\text{Inf}(\rho)|$ entries are exactly the vertices of $\text{Inf}(\rho)$. Further, $\sharp$ appears before the first element of $V \setminus \text{Inf}(\rho)$. After n_2 only vertices from $\text{Inf}(\rho)$ occur. Thus, repeating the argument of (1) all later LARs are exactly $\text{Inf}(\rho)$. Finally, we have that $\sharp$ is always at a position smaller or equal to $|\text{Inf}(\rho)| + 1$ such that $\text{hit}(l_{n'}) \subseteq \text{Inf}(\rho)$ for all $n' \geq n_2$.

- Infinitely often, it holds that $\text{hit}(l_n) = \text{Inf}(\rho)$ (3)

  Let $n \geq n_2$ be a position such that l_n has exactly $\text{Inf}(\rho)$ as the first $|\text{Inf}(\rho)|$ entries of ρ. Such a position exists as shown in (2). Now, consider the entry $v \in V$ at the last position in l_n with $v \in \text{Inf}(\rho)$. Then there exists an $n' > n$ with $\rho_{n'} = v$ where w.l.o.g. we can assume that n' is the minimal position with this property. Then by construction we have that v is still at the last entry in $l_{n'-1}$ of vertices in $\text{Inf}(\rho)$. Thus, for l_n we have that v is in the first position and $\sharp$ is at position $|\text{Inf}(\rho)| + 1$. Together with (2) it follows that $\text{hit}(l_{n'}) = \text{Inf}(\rho)$. Repeating the argument it follows by induction that there are infinitely many such positions n'.

We now are ready to define our coloring function $\Omega \colon V \times \text{LAR}_V \to \mathbb{N}$:

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

We have to show that $\rho \in \text{MULLER}(\mathcal{F})$ iff $\text{ext}(\rho) = (\rho_0, l_0)(\rho_1, l_1)... \in \text{PARITY}(\Omega)$. We have that

$$
\begin{aligned}
& \rho \in \text{MULLER}(\mathcal{F}) \\
\Leftrightarrow \quad & \text{Inf}(\rho) \in \mathcal{F} \\
\Leftrightarrow \quad & \exists F \in \mathcal{F}.\ (\forall m \in \mathbb{N}.\ \exists n \geq m.\ \text{hit}(l_n) = F) \\
& \quad \wedge (\exists m' \in \mathbb{N}.\ \forall n' \geq m'.\ \text{hit}(l_{n'}) \subseteq F) \\
\Leftrightarrow \quad & \exists F \in \mathcal{F}.\ \exists m' \in \mathbb{N}.\ \forall m \geq m'.\ \exists n \geq m.\ \text{hit}(l_n) = F \\
& \quad \Omega(\rho_n, l_n) = 2 \cdot |V| - 2 \cdot |\text{hit}(l_n)| \\
& \qquad\qquad = 2 \cdot |V| - 2 \cdot |F| \\
& \qquad\qquad \leq 2 \cdot |V| - 2 \cdot |\text{hit}(l_m)| \\
& \qquad\qquad = \Omega(\rho_m, l_m) \\
\Leftrightarrow \quad & \text{ext}(\rho) \in \text{PARITY}(\Omega)
\end{aligned}
$$

□

Corollary 3.2. *Ever Muller game $\mathcal{G} = (\mathcal{A}, \text{MULLER}(\mathcal{F}))$ is determined with uniform finite state winning strategies of size $|\mathcal{A}| \cdot |\mathcal{A}|!$ and can be solved in exponential time.*

Proof. We have to solve the parity game $\mathcal{G}' = (\mathcal{A} \times \mathcal{M}_{\text{LAR}}, \text{PARITY}(\Omega))$ with $\mathcal{A} = (V, V_0, V_1, E)$ and Ω defined as in the proof of Theorem 3.1. We have that $\mathcal{G}'$ has $n = |V| \cdot |V|!$ many vertices, $m = |E| \cdot |V| \cdot |V|!$ many edges and $d = 2 \cdot |V|$ many colors. As the positional winning strategies for both players in $\mathcal{G}'$ are a one to one representation of the corresponding finite-state winning strategy in $\mathcal{G}$ we get that these strategies have at least size $|\mathcal{A}| \cdot |\mathcal{A}|!$. Using the progress measure algorithm we finally get the exponential upper bound for solving Muller games by

$$
\mathcal{O}(|E| \cdot |V| \cdot |V|! \cdot (|V| \cdot |V|!)^{|V|}) = \mathcal{O}(|E| \cdot |V| \cdot |V|! \cdot 2^{|V| \cdot log(|V| \cdot |V|!)}) = \mathcal{O}(|E| \cdot |V| \cdot |V|! \cdot 2^{|V| \cdot (log(|V|) + \sum_{j=1}^{|V|} log(j))})
$$

□

Consider that the complexity of solving Muller games depends on the way of representing $\mathcal{F}$. We will close this chapter by a list of some possible representations of $\mathcal{F}$ and discuss the computational complexity of solving Muller games using these representations.

Explicit Representation:

The easiest way to represent $\mathcal{F}$ is by giving it explicitly as a list of all subsets.

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

Boolean Formula:

We also can encode the set as a boolean formula consisting of variables x_v for all vertices v. Every clause then represents a subset of $\mathcal{F}$ by imposing constrains to membership and non-membership properties. An example for such a formula and its encoding is given below.

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

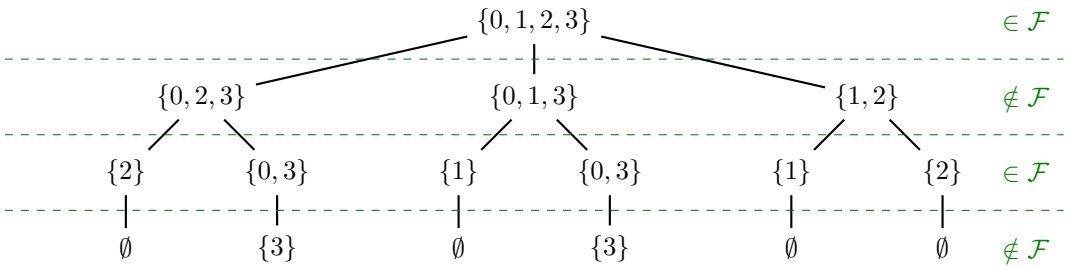
Boolean Circuit:

Another possibility is the encoding as a boolean circuit instead of a boolean formula. There is the possibility that it can be smaller than the formula as shared subformulas can use the same gates.

Tree:

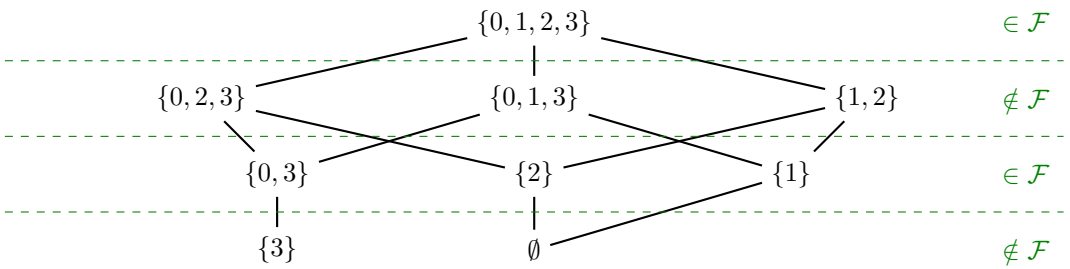
We can also represent $\mathcal{F}$ as a tree defined by the following inductive definition:

- 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 $F' \in \mathcal{F}$.



Directed Acyclic Graph:

We can improve the previous representation of a tree to a directed acyclic graph.



Color Function:

We can color each vertex v by a color function $\Omega\colon V \to [k]$ for some $k \in \mathbb{N}$, define an explicit list $\mathcal{F}' \subseteq 2^{[k]}$ and represent $\mathcal{F}$ as

$$\mathcal{F} = \{\, F \subseteq V \mid \Omega(F) \in \mathcal{F}' \,\}$$

Important Subset:

If we choose a subset $W \subseteq V$ and define $\mathcal{F}' \subseteq 2^W$ as an explicit list, then we can represent $\mathcal{F}$ as

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

Theorem 3.2.

a) Solving Muller games is in P *if $\mathcal{F}$ is given as an explicit list.*

b) Solving Muller games is in NP ∩ Co-NP *if $\mathcal{F}$ is given as a tree.*

c) Solving Muller games is Pspace*-complete if $\mathcal{F}$ is given in any of the other representations.*

3.4 Limits of Reductions

In the previous subsection, we have seen several reductions, e.g. how to reduce Muller games to parity games. A natural question concerns the limits of such reductions, i.e. can parity games be reduced to safety games? It turns out that this is impossible. A parity condition is harder than a safety condition. On an intuitive level, this hardness can be phrased as follows. In a parity game, the coloring controls which vertices have to be seen infinitely often and which ones may only be visited finitely often. On the other hand, in a safety game, the set S of safe vertices only specifies which vertices cannot be visited, not even once, which is a much weaker condition. Such informal statements will be made precise now. We will introduce the Borel hierarchy, which characterizes the complexity of languages[2] in terms of their topological complexity. Starting from very simple languages it consists of infinitely many levels of languages. Then, we will show that reductions cannot go "down" the hierarchy. A complicated language cannot be reduced to a simpler one, i.e. one that is on a lower level of the hierarchy.

We begin by defining the hierarchy, whose basic sets are the so-called open ones.

Definition 3.9. Let V be a finite set. A set $L \subseteq V^\omega$ is called *open* iff it is of the form KV^ω for some $K \subseteq V^*$.

Intuitively, an open set is a abstract reachability property. As soon as a prefix of a $\rho \in V^\omega$ is in K, ρ is in L. However note, that K might be arbitrarily complicated, i.e. the set $K = \{\, 0^p \in \mathbb{B}^* \mid p \text{ is a prime} \,\}\mathbb{B}^\omega$ is open. As expected, reachability winning conditions are open, as we have $\textsc{reach}(R) = (V^*R)V^\omega$. We now can formally define the Borel hierarchy.

Definition 3.10. Let V be a finite set. The *Borel hierarchy* on V^ω consists of levels $\mathbf{\Sigma}_n$ and $\mathbf{\Pi}_n$ for every $n \in \mathbb{N}^+$, defined recursively as follows:

- $\mathbf{\Sigma}_1 = \{\, L \subseteq V^\omega \mid L \text{ is open} \,\}$
- $\mathbf{\Pi}_n = \{\, L \subseteq V^\omega \mid V^\omega \setminus L \in \mathbf{\Sigma}_n \,\}$
- $\mathbf{\Sigma}_{n+1} = \{\, L \subseteq V^\omega \mid L = \bigcup_{j\in\mathbb{N}} L_j \text{ for infinitely many } L_j \in \mathbf{\Pi}_n \text{ with } j \in \mathbb{N} \,\}$

Consider that the languages L_j not necessarily to be different from each other.

The Borel hierarchy also has levels $\mathbf{\Sigma}_\alpha$ and $\mathbf{\Pi}_\alpha$ for countably infinite ordinals α. However, 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 $n \leq 3$. Hence, we omit the definition of the higher levels.

[2] and therefore also winning conditions for infinite games

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

Lemma 3.4. *For every $n \in \mathbb{N}^+$, we have that $\mathbf{\Sigma}_n \cup \mathbf{\Pi}_n \subseteq \mathbf{\Sigma}_{n+1} \cap \mathbf{\Pi}_{n+1}$.*

Proof. Let V be a finite set and $L \subseteq V^\omega$. Then it holds that

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

and therefore also

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

Next, we show $L \in \mathbf{\Sigma}_n \Rightarrow L \in \mathbf{\Sigma}_{n+1}$ and $L \in \mathbf{\Pi}_n \Rightarrow L \in \mathbf{\Pi}_{n+1}$ by induction over $n \in \mathbb{N}^+$.

Induction Base: $n = 1$

Let $L \in \mathbf{\Sigma}_1$ be arbitrary. We have to show that $L \in \mathbf{\Sigma}_2$, i.e. $L = \bigcup_{j \in \mathbb{N}} L_j$ for infinitely many $L_j \in \mathbf{\Pi}_1$ with $j \in \mathbb{N}$. For $w = w_0 w_1 ... w_n \in V^*$ let

$$K_w = \{\, w_0 ... w_{n'} v \mid n' < n \wedge v \neq w_{n'+1} \,\}$$

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, we have that $L = KV^\omega$ for some $K \subseteq V^*$. We show

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

which suffices to show $L \in \mathbf{\Sigma}_2$ since every $K_w V^\omega \in \mathbf{\Sigma}_1$ and accordingly every $V^\omega \setminus K_w V^\omega$ in $\mathbf{\Pi}_1$.

"⇒":

Let $v_0 v_1 v_2 ... \in L$ be arbitrary. We have that there exists a prefix $w = v_0 ... v_n$ with $w \in K$. If $w = \varepsilon$, then we have that $K_w = \emptyset$ and $V^\omega \setminus K_w V^\omega = V^\omega$. As a consequence, we have that $v_0 v_1 v_2 ... \in \bigcup_{w \in K} V^\omega \setminus K_w V^\omega$. Now, consider the case $w \neq \varepsilon$. Then, we have $v_0 v_1 v_2 ... \notin K_w V^\omega$ and therefore $v_0 v_1 v_2 ... \in V^\omega \setminus K_w V^\omega$. Finally, we get $v_0 v_1 v_2 ... \in \bigcup_{w \in K} V^\omega \setminus K_w V^\omega$.

"⇐":

Let $v_0 v_1 v_2 ... \notin L$ and $w \in K$ be arbitrary. Note that $w \neq \varepsilon$ since $\varepsilon \in K$ implies $L = KV^\omega = V^\omega$ which contradicts $v_0 v_1 v_2 ... \notin L$. Furthermore, we have $v_0 v_1 v_2 ... \in K_w V^\omega$, since w is not a prefix of $v_0 v_1 v_2 ....$ Thus, $v_0 v_1 v_2 ... \notin V^\omega \setminus K_w V^\omega$ and as this holds for every $w \in K$ we can conclude that $v_0 v_1 v_2 ... \notin \bigcup_{w \in K} V^\omega \setminus K_w V^\omega$.

It remains to consider $\mathbf{\Pi}_1$. The result directly follows from $L \in \mathbf{\Sigma}_1 \Rightarrow L \in \mathbf{\Sigma}_2$ and

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

Induction Hypothesis:

$\forall n \in \mathbb{N}^+.\ (L \in \mathbf{\Sigma}_n \Rightarrow L \in \mathbf{\Sigma}_{n+1}) \wedge (L \in \mathbf{\Pi}_n \Rightarrow L \in \mathbf{\Pi}_{n+1})$

Induction Step: $n > 1$

We have that

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

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 in induction base. We have that

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

where we again apply the result just proven for Σ_n. □

We now can place all winning conditions considered so far in the first three levels of the Borel hierarchy.

	$\mathbf{\Sigma}_1$	$\mathbf{\Pi}_1$	$\mathbf{\Sigma}_2 \cap \mathbf{\Pi}_2$	$\mathbf{\Sigma}_2$	$\mathbf{\Pi}_2$	$\mathbf{\Sigma}_3 \cap \mathbf{\Pi}_3$
$\text{REACH}(R)$	×		×	×	×	×
$\text{SAFE}(S)$		×	×	×	×	×
$\text{WPARITY}(\Omega)$			×	×	×	×
$\text{WMULLER}(\mathcal{F})$			×	×	×	×
$\text{BÜCHI}(F)$					×	×
$\text{COBÜCHI}(C)$				×		×
$\text{PARITY}(\Omega)$						×
$\text{MULLER}(\mathcal{F})$						×

Furthermore, for each type of winning condition appearing in the tabular above, there is a condition which is in the marked class, but not in a smaller class or in the complement-class[3]. For example, there is a parity condition $\text{PARITY}(\Omega)$, which is also a Muller condition, such that $\text{PARITY}(\Omega) \notin \mathbf{\Sigma}_2 \cup \mathbf{\Pi}_2$. Similarly, there is a Büchi condition $\text{BÜCHI}(F)$ such that $\text{BÜCHI}(F) \notin \mathbf{\Sigma}_1 \cup \mathbf{\Pi}_1$ and $\text{BÜCHI}(F) \notin \mathbf{\Sigma}_2$.

Next, we study a generalized notion of reductions via continuous mappings between languages. We will see later that game reductions defined in the previous subsection are a special case of them.

Definition 3.11. A function $f\colon U^\omega \to V^\omega$ is called *continuous* iff $f^{-1}(L)$ is open for every open set $L \subseteq V^\omega$.

Definition 3.12. A set $L \subseteq U^\omega$ is *Wadge-reducible* to a set $L' \subseteq V^\omega$, denoted by $L \leq L'$, iff there exists a continuous function $f\colon U^\omega \to V^\omega$ such that $f^{-1}(L') = L$.

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

Remark 3.1. *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 3.5. *Let $L \subseteq U^\omega$ and $L' \subseteq V^\omega$ for two finite sets U and V such that $L \leq L'$. Then for all $n \in \mathbb{N}^+$ holds:*

1. $L' \in \mathbf{\Sigma}_n \;\Rightarrow\; L \in \mathbf{\Sigma}_n$
2. $L' \in \mathbf{\Pi}_n \;\Rightarrow\; L \in \mathbf{\Pi}_n$

Proof. We prove both statements simultaneously by induction over $n \in \mathbb{N}^+$.

Induction Base: $n = 1$

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 we have that L is also open and correspondingly $L \in \mathbf{\Sigma}_1$.

Let $L' \in \mathbf{\Pi}_1$. We have that $V^\omega \setminus L' \in \mathbf{\Sigma}_1$ and therefore also $U^\omega \setminus L \in \mathbf{\Sigma}_1$ by the proof for $\mathbf{\Sigma}_1$ and Remark 3.1. We finally get $L \in \mathbf{\Pi}_1$.

Induction Hypothesis:

$\forall n \in \mathbb{N}^+.\ (L' \in \mathbf{\Sigma}_n \Rightarrow L \in \mathbf{\Sigma}_n) \wedge (L' \in \mathbf{\Pi}_n \Rightarrow L \in \mathbf{\Pi}_n)$

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

Induction Step: $n > 1$

Let $L' \in \mathbf{\Sigma}_n$. It follows that $L' = \bigcup_{j\in\mathbb{N}} L'_j$ with $L_j \in \mathbf{\Pi}_{n-1}$ for all $j \in \mathbb{N}$. We define $L_j = f^{-1}(L_j)$ for all $j \in \mathbb{N}$. 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. To conclude, consider that

$$x \in L \;\Leftrightarrow\; f(x) \in L' \;\Leftrightarrow\; \exists j \in \mathbb{N}.\; f(x) \in L'_j \;\Leftrightarrow\; \exists j \in \mathbb{N}.\; x \in L_j$$

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

Now, 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 3.1. Thus, as already shown, we have that $U^\omega \setminus L \in \mathbf{\Sigma}_n$ 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 II picks a word $y_n \in V^*$. After ω many rounds, the pair (x, y) with $x = x_0x_1x_2...$ and $y = y_0y_1y_2...$ determines the winner. Player II wins if and only if y is infinite and we have $x \in L$ iff $y \in L'$.

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 $\rho = x_0y_0x_1x_1...$ is consistent with σ iff $x_n = \sigma(y_0...y_{n-1})$ for all $n \in \mathbb{N}$ and it is consistent with τ iff $y_n = \tau(x_0...x_n)$ for every $n \in \mathbb{N}$. A strategy is winning, if every play that is consistent with the strategy is winning.

As an example consider the game $W(0^*1(0+1)^\omega, (0^*1)^\omega)$. 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 $y \in (0^*1)^\omega$. It follows that the strategy is indeed winning for II.

We now show that Wadge games characterize Wadge reductions:

Theorem 3.3. *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. We show both directions separately.

"⇒":

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 $x_0, ..., x_n$ and II has picked possibly empty words $y_0, ..., y_{n-1}$ and has to pick y_n. By construction of our strategy, the set $\{x_0...x_n\}U^\omega$ is partitioned into $\{\, f^{-1}(\{y_0...y_{n-1}v\}V^\omega) \mid v \in V \,\}$. If there is a v such that $\{x_0...x_n\}U^\omega \subseteq f^{-1}(\{y_0...y_{n-1}v\}V^\omega)$, then II picks $y_n = v$, otherwise II picks $y_n = \varepsilon$. Note, that this choice preserves the partition property required above.

We need to show that $y_0y_1y_2...$ is an infinite word. If this is the case, then we have $y_0y_1y_2... = f(x_0x_1x_2...)$ and therefore $x_0x_1x_2... \in L$ if and only if $y_0y_1y_2... = f(x_0x_1x_2...) \in L'$ 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 $x = x_0x_1x_2...$. Then, $\{x_0...x_n...x_{n+k}\}U^\omega$ is not a subset of $f^{-1}(\{y_0...y_{n-1}v\}V^\omega)$ for every $v \in V$ and every $K \in \mathbb{N}$. As every $f(\{y_0...y_{n-1}v\}V^\omega)$ is open and correspondingly also every $f^{-1}(\{y_0...y_{n-1}v\}V^\omega)$ we have that

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

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

"⇐":

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 $u_0u_1u_2... \in U^\omega$ and the sequence $y = y_0y_1y_2... \in V^\omega$ of words defined by $y_n = \tau(u_0...u_n)$ for all $n \in \mathbb{N}$. Then as τ is a winning strategy we have that y is infinite. Hence, we can define $f(u_0u_1u_2...) = y$. We get that $f^{-1}(L') = L$ 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 the set KV^ω for some $K \subseteq V^\omega$. We have to show that $f^{-1}(KV^\omega)$ is open. So, define

$$K' = \{\, u_0...u_n \in U^* \mid \tau(u_0)\tau(u_0u_1)...\tau(u_0...u_n) \in K \,\}$$

We show that we have that $f^{-1}(KV^\omega) = K'U^\omega$. Let $x \in f^{-1}(KV^\omega)$, i.e. $f(x) \in KV^\omega$. Accordingly, there exists a prefix $u_0...u_n$ of x satisfying $\tau(u_0)\tau(u_0u_1)...\tau(u_0...u_n) \in K$. As a consequence, $u_0...u_n \in K'$ and therefore $x \in K'U^\omega$. Now, let $x \in K'U^\omega$. Then there exits a prefix $u_0...u_n$ of x with $\tau(u_0)\tau(u_0u_1)...\tau(u_0...u_n) \in K$. We get $f(x) \in KV^\omega$. □

Finally, we can apply the general results we obtained so far to study game reductions as introduced above. We show that every game reduction is a Wadge-reduction. This especially implies that there are no game reductions "down" the Borel hierarchy, e.g. parity games cannot be reduced to safety games.

Lemma 3.6. *Let $\mathcal{G} = (\mathcal{A}, \mathrm{Win})$ and $\mathcal{G}' = (\mathcal{A} \times M, \mathrm{Win}')$ be two games with $\mathcal{G} \leq_{\mathcal{M}} \mathcal{G}'$ for some memory structure $\mathcal{M} = (M, \mathrm{init}, \mathrm{upd})$. Then, it holds that $\mathrm{Win} \leq \mathrm{Win}'$.*

Proof. We show that II has a winning strategy for the game $W(\mathrm{Win}, \mathrm{Win}')$ which we define by $\tau(v_0...v_n) = (v_n, \mathrm{upd}^*(v_0...v_n))$. Accordingly, II constructs $\mathrm{ext}(\rho)$ as response to I picking a play $\rho \in \mathrm{Plays}(\mathcal{A})$. 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 concludes $\mathrm{Win} \leq \mathrm{Win}'$. □

We conclude this subsection by Martin's determinacy theorem, which subsumes all determinacy results we proved so far. However note, that it does not yield 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 iff it is contained in some level of the Borel hierarchy[4].

Theorem 3.4. *Every infinite game $\mathcal{G} = (\mathcal{A}, \mathrm{Win})$ where* Win *is Borel is determined.*

[4] even if the level has an ordinal index

4 Infinite Arenas

Until now, we only have considered games where the underlying arena was a finite object. In this chapter, we will overcome this restriction and allow arenas to be countably infinite. Plays, strategies, winning regions, etc. are still defined as usual.

We first like to give a short overview of some differences between games with finite arenas and games with infinite arenas. First of all, we can observe new questions appearing if we view the problem from an algorithmic standpoint.

- The input to the solution algorithms is now an infinite object.
- Even positional strategies are infinite objects.
- The infinity set $\mathrm{Inf}(\rho)$ might be empty.
- Paths of bounded, finite length might not be sufficient for an appropriate attractor construction. For an example see Figure 17. Here, vertex v will never by added to any 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 we reach v' is fixed.

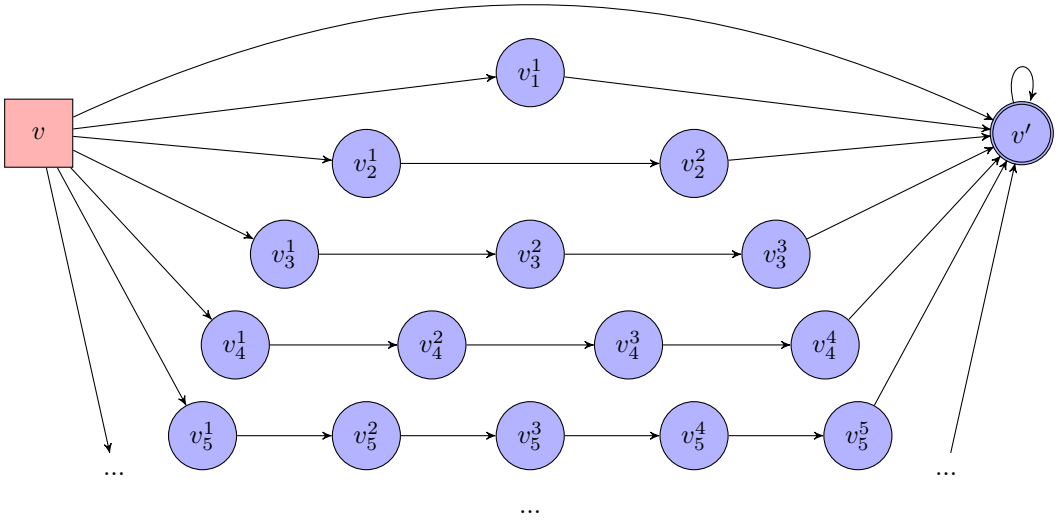


Figure 17: There is no $n \in \mathbb{N}$ such that v is in $\mathrm{Attr}_0^n(\{v'\})$.

Accordingly, to solve this game by an attractor construction we need more than infinitely many steps. We 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[5]. The good news are, that it can be shown that there is always a level at which the attractor gets stationary, as long as the underlying arena is countably infinite.

- Consider a parity game $\mathcal{G} = (\mathcal{A}, \textsc{parity}(\Omega\colon V \to [k]))$ with $k \in \mathbb{N}$. As we set an upper bound on the number of colors, we can still ensure that there is a color appearing infinitely often, even if every vertex would only appear once. For such games, one can still prove positional determinacy, using an induction on the number of colors and the adapted attractor construction. Note, that it is indeed crucial here that the codomain of Ω is finite, as otherwise the minimal color seen infinitely often may be undefined.

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

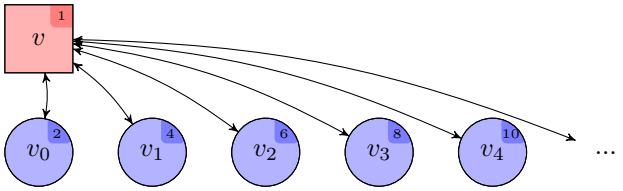


Figure 18: Max-parity game which cannot be translated into an equivalent min-parity game and where Player 1 needs a winning strategy of infinite size.

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 18. 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. Consider that we cannot represent this game as an equivalent min-parity game, as both players have positional strategies in min-parity games[6]. For further insights on this topic, we suggest to read the paper "Positional determinacy of games with infinitely many priorities" by Grädel and Walukiewicz (LMCS 2006).

4.1 Parity Pushdown Games

In the following, we want to consider a class of infinite games on infinite arenas that have a better behavior than the examples discussed so far. The idea is, that we want to have 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.

Definition 4.1. A *pushdown system* $\mathcal{P} = (Q, \Gamma, \Delta, q_{\text{in}})$ 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 $q_{\text{in}} \in Q$,

where $\Gamma_\perp$ denotes the extended stack alphabet $\Gamma_\perp = \Gamma \cup \{\perp\}$. We call $\perp$ the initial stack symbol. Further we call

$$(q, A, q', \alpha) \in \Delta \begin{cases} \text{a pop transition} & \text{if } |\alpha| = 0, \\ \text{a skip transition} & \text{if } |\alpha| = 1 \text{ and} \\ \text{a push transition} & \text{if } |\alpha| = 2. \end{cases}$$

Every pushdown system $\mathcal{P}$ has to fulfill the following criteria:

- The system is deadlock free: $\forall q \in Q.\ \forall A \in \Gamma_\perp.\ \exists q' \in Q.\ \exists \alpha \in \Gamma_\perp^{\leq 2}.\ (q, A, q', \alpha) \in \Delta$
- $\perp$ is never deleted nor written on the stack: $(q, \perp, q', \varepsilon) \notin \Delta \wedge ((q, A, q', \perp X) \in \Delta \Rightarrow AX = \perp)$

For a pushdown system $\mathcal{P} = (Q, \Gamma, \Delta, q_{\text{in}})$ we use the following terms to denote the below mentioned elements of $\mathcal{P}$.

- We call $\gamma \in \Gamma^*\{\perp\}$ a *stack content* of $\mathcal{P}$.
- We call $(q, \gamma) \in Q \times \Gamma^*\{\perp\}$ a *configuration* of $\mathcal{P}$.
- For a configuration (q, γ) of $\mathcal{P}$ we denote its *stack height* by $|(q, \gamma)| = |\gamma| - 1$.
- A transition from one configuration to another is defined as:

$$(q, \gamma) \rightarrowtriangle (q', \gamma') \quad \Leftrightarrow \quad (q, \gamma_0, q', \alpha) \in \Delta \ \wedge \ \gamma' = \alpha\gamma_1\gamma_2...\gamma_{|(q,\gamma)|}$$

[6] even with infinitely many colors

We now can define the corresponding configuration graph of a pushdown system.

Definition 4.2. Let $\mathcal{P} = (Q, \Gamma, \Delta, q_{\text{in}})$ be a pushdown system. The *induced 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
- $E = \{\, (v, v') \in V \times V \mid v \rightarrowtriangle v' \,\}$ is the set of transitions.

Consider that the induced configuration graph of some pushdown system $\mathcal{P}$ has no terminal vertices, as $\mathcal{P}$ is deadlock free. It remains to define the players on a given configuration graph and to color the vertices appropriately. We then get the desired parity game definition.

Definition 4.3. Let $\mathcal{P} = (Q, \Gamma, \Delta, q_{\text{in}})$ be a pushdown system, $Q_0 \dot\cup Q_1 = Q$ be a partition of Q and $\Omega' \colon Q \to [k]$ be a coloring function with $k \in \mathbb{N}$. Then we define the *induced 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 all $i \in \{0, 1\}$
- $\Omega((q, \gamma)) = \Omega'(q)$ for all $(q, \gamma) \in V$

An example for parity game induced by a pushdown system $\mathcal{P}$ is depicted in Figure 19. The corresponding pushdown system is given by $\mathcal{P} = (\{q_{\text{in}}, q_1, q_2\}, \{A\}, \Delta, q_{\text{in}})$ with Δ defined as the set

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

the partition $Q_1 = \{q_{in}\}$ and $Q_0 = \{q_1, q_2\}$ and the coloring function $\Omega \colon \{q_{\text{in}}, q_1, q_2\} \to [2]$ with $\Omega(v) = 0 \iff v \neq q_1$. 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 will be 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[7] and q_2 has an even color, odd colors occur only finitely often. Accordingly, Player 0 wins this game from every vertex.

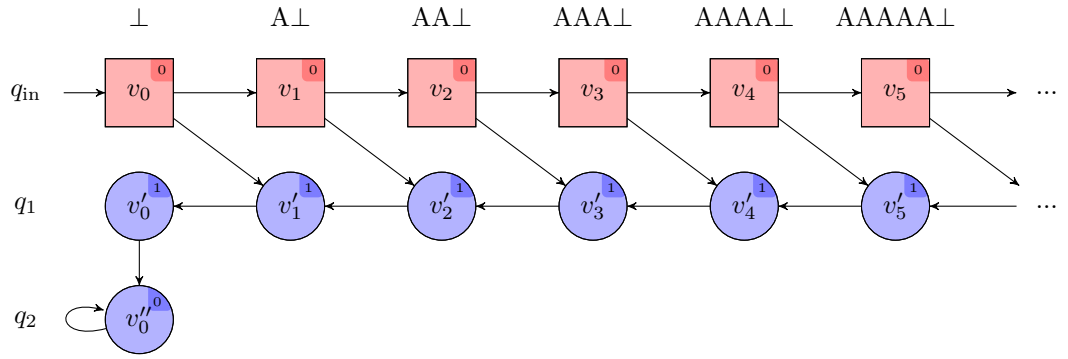


Figure 19: An example for a pushdown game potentially visiting infinitely many vertices.

We now are interested in solving such games with infinite arenas. In general we want to tackle the following problem.

Problem of solving pushdown games:

Input: Pushdown game $\mathcal{G}$, represented by a pushdown system $\mathcal{P}$, a partition $Q_0 \dot\cup Q_1$ and a coloring function Ω and some $i \in \{0, 1\}$.

Output: Winning strategy for Player i from $(q_{\text{in}}, \bot)$, if there exists one.

[7] as this are exactly the number of steps Player 1 has increased it

Pushdown Transducers

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 do not suffice, but, not surprisingly, using pushdown machines suffices.

Definition 4.4. A *pushdown transducer* $\mathcal{T} = (Q, \Gamma, \Delta, q_{\text{in}}, \Sigma_I, \Sigma_O, \lambda)$ is a tuple consisting of

- a non-empty, finite set of states Q,
- a finite stack alphabet Γ,
- a transition relation $\Delta \subseteq Q \times \Gamma_\bot \times (\Sigma_I \cup \{\varepsilon\}) \times Q \times \Gamma_\bot^{\leq 2}$,
- an initial state q_{in},
- a finite input alphabet Σ_I,
- a finite output alphabet Σ_O and
- a partial output function $\lambda \colon Q \rightharpoonup \Sigma_O$.

where $\Gamma_\bot$ is defined similar as for pushdown systems and $\varepsilon \notin \Sigma_I$.

Definition 4.5. Let $\mathcal{T} = (Q, \Gamma, \Delta, q_{\text{in}}, \Sigma_I, \Sigma_O, \lambda)$ be a pushdown transducer. We say that $\mathcal{T}$ is *deterministic* iff

$$\forall q \in Q.\ \forall A \in \Gamma_\bot.\ \forall a \in \Sigma_I.\ |\{\, (q', \alpha) \mid (q, A, a, q', \alpha) \in \Delta \vee (q, \varepsilon, a, q', \alpha) \in \Delta \,\}| \leq 1$$

Definition 4.6. Let $\mathcal{T} = (Q, \Gamma, \Delta, q_{\text{in}}, \Sigma_I, \Sigma_O, \lambda)$ be a pushdown transducer. A *run r of $\mathcal{T}$* on a finite input word $w \in \Sigma_I^*$ is defined as

$$(q_0, \gamma_0)(q_1, \gamma_1)...(q_m, \gamma_m) \in (Q \times \Gamma^*\{\bot\})^*$$

with $(q_0, \gamma_0) = (q_{in}, \bot)$ and with $(q_j, \gamma_j) \xrightarrow{a_j} (q_{j+1}, \gamma_{j+1})$ 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 $(q, \gamma) \xrightarrow{a} (q', \gamma')$ to denote that $(q, \gamma_0, a, q', x) \in \Delta$ with $\gamma' = \alpha\gamma(1)...\gamma(|\gamma| - 1)$. Consider that 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 $\text{run}(w)$.

Definition 4.7. Let $\mathcal{T} = (Q, \Gamma, \Delta, q_{\text{in}}, \Sigma_I, \Sigma_O, \lambda)$ be a deterministic pushdown transducer. Then $\mathcal{T}$ *induces* a partial function $f_\mathcal{T} \colon (\Sigma_I)^* \rightharpoonup \Sigma_O$ such that $f_\mathcal{T}(w) = \lambda(pr_0(\text{Lst}(\text{run}(w))))$ for all $w \in \Sigma_I^*$, if such a run exists.

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 that $\Sigma_I = \Sigma_O = \Delta$, 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[8]. Thus, we have to require the output transition to be executable from the last configuration of the play prefix induced by the input sequence.

[8] in case the last configuration of the play prefix is a Player i configuration

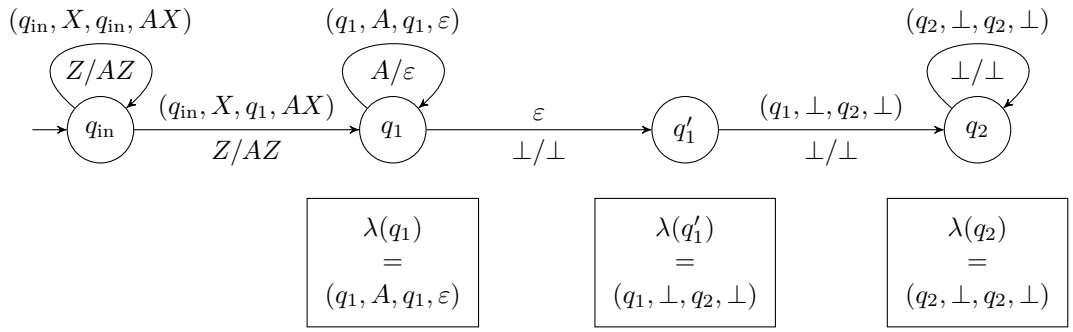


Figure 20: Pushdown transducer implementing a winning strategy for Player 0 from $(q_{\text{in}}, \perp)$ in the example pushdown system $\mathcal{P}$.

4.2 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 $c \in [k]$ a subset $P_c \subseteq Q$ of states. 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 $c \in [k]$ 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 $c \in [k]$ 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 $\mathcal{G} = (\mathcal{A}, \text{PARITY}(\Omega))$ be a pushdown game with arena $\mathcal{A} = (V, V_0, V_1, E)$ induced by $\mathcal{P} = (Q, \Gamma, \Delta, q_{\text{in}})$ with partition $Q_0 \cup Q_1$ of Q and coloring function $\Omega\colon Q \to [k]$. 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 $\mathsf{Pred} = (2^Q)^n$.

We define the game $\mathcal{G}' = (\mathcal{A}', \text{PARITY}(\Omega'))$ 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 $c, d \in [k]$ 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 $c, d \in [k]$. 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 $c, d \in [k]$. 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 $e \in [k]$ 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 $c, d \in [k]$, 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 function $\Omega' \colon V' \to [k+1]$ 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 4.1. *Let $\mathcal{G}$ be a pushdown parity game and $\mathcal{G}'$ the parity game constructed as above. We have $(q_{\text{in}}, \bot) \in W_i(\mathcal{G})$ if and only if $\mathsf{Check}[q_{\text{in}}, \bot, P^{\text{in}}, \Omega(q_{\text{in}}), \Omega(q_{\text{in}})] \in W_i(\mathcal{G})$ for $i \in \{0, 1\}$.*

Proof. We show that $(q_{\text{in}}, \bot) \in W_0(\mathcal{G})$ implies $\mathsf{Check}[q_{\text{in}}, \bot, P^{\text{in}}, \Omega(q_{\text{in}}), \Omega(q_{\text{in}})] \in W_0(\mathcal{G})$ 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 $(q_{\text{in}}, \bot)$ in $\mathcal{A}$ respectively in $\mathsf{Check}[q_{\text{in}}, \bot, P^{\text{in}}, \Omega(q_{\text{in}}), \Omega(q_{\text{in}})]$ 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 $(q_{\text{in}}, \bot)$ in $\mathcal{G}$. We need to construct a winning strategy σ' for Player 0 from $\mathsf{Check}[q_{\text{in}}, \bot, P^{\text{in}}, \Omega(q_{\text{in}}), \Omega(q_{\text{in}})]$ 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 $\mathsf{Check}[q_{\text{in}}, \bot, P^{\text{in}}, \Omega(q_{\text{in}}), \Omega(q_{\text{in}})]$.

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 $\mathsf{Check}[q_{\text{in}}, P^{\text{in}}, \Omega(q_{\text{in}}), \Omega(q_{\text{in}})]$, we only have to consider the play prefix $\mathsf{Check}[q_{\text{in}}, P^{\text{in}}, \Omega(q_{\text{in}}), \Omega(q_{\text{in}})]$ and define

$$f(\mathsf{Check}[q_{\text{in}}, P^{\text{in}}, \Omega(q_{\text{in}}), \Omega(q_{\text{in}})]) = (q_{\text{in}}, \bot) \ .$$

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 $\mathsf{Check}[q_{\text{in}}, \bot, P^{\text{in}}, \Omega(q_{\text{in}}), \Omega(q_{\text{in}})]$. 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 $\mathsf{Check}[q_{\text{in}}, \bot, P^{\text{in}}, \Omega(q_{\text{in}}), \Omega(q_{\text{in}})]$. 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 $\mathcal{T}_\sigma = (Q^\sigma, \Gamma^\sigma, \Delta^\sigma, q^\sigma_{\text{in}}, \Sigma^\sigma_I, \Sigma^\sigma_O, \lambda^\sigma)$, where

- $Q^\sigma = V'|_{\sigma'}$,
- $\Gamma^\sigma = \{v \in V'|_{\sigma'} \mid v \text{ is a } \mathsf{Claim}\text{-vertex in } \mathcal{A}'|_\sigma\}$,
- $q^\sigma_{\text{in}} = \mathsf{Check}[q_{\text{in}}, \bot, P^{\text{in}}, \Omega(q_{\text{in}}), \Omega(q_{\text{in}})]$, and
- $\Sigma^\sigma_I = \Sigma^\sigma_O = \Delta$.

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 $Z \in \Gamma^\sigma_\bot$.
- If v is a Claim-vertex and v' is a Check-vertex, then $(v, Z, \ell(v,v'), v', vZ) \in \Delta^\sigma$ for $Z \in \Gamma^\sigma_\bot$, 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 $(q_{\text{in}}, \bot)$. To show this, let $\rho = (q_0,\gamma_0)(q_1,\gamma_1)(q_2,\gamma_2)...$ be a play that is consistent with the strategy σ implemented by $\mathcal{T}$, starting in $(q_{\text{in}}, \bot)$. 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 $n_0 < n_1 < n_2 < ...$ be the ascending enumeration of ρ's stairs. We chop ρ into infinitely many pieces, leading from one stair to the next by defining $x_0 = (q_0,\gamma_0)...(q_{n_0},\gamma_{n_0})$ and

$$x_j = (q_{n_{j-1}+1},\gamma_{n_{j-1}+1})...(q_{n_j},\gamma_{n_j})\ .$$

Then, we have $\rho = x_0x_1x_2...$ and furthermore, the minimal color seen infinitely often during ρ is the same as the minimal color seen infinitely often in the sequence $c_0c_1c_2...$, 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 $c_0c_1c_2...$ 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 $\rho'_0 = \mathsf{Check}[q_{\text{in}}, \bot, P^{\text{in}}, \Omega(q_{\text{in}}), \Omega(q_{\text{in}})]$. Now, assume we have simulated ρ up to stair j, defining the play prefix $\rho'_0 ... \rho'_m$, which is consistent with σ'. We need to prolong $\rho'_0 ... \rho'_m$ 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 $\rho'_0 ... \rho'_m$ 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 $\rho'_0 ... \rho'_m$ 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 $\rho'_0 ... \rho'_m$ 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 $c_0 c_1 c_2 ...$. 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 $\mathsf{Check}[q_{\text{in}}, \bot, P^{\text{in}}, \Omega(q_{\text{in}}), \Omega(q_{\text{in}})]$, and is winning for Player 1. This contradicts the fact that σ' is winning from $\mathsf{Check}[q_{\text{in}}, \bot, P^{\text{in}}, \Omega(q_{\text{in}}), \Omega(q_{\text{in}})]$. □

Since the parity game $\mathcal{G}'$ is of exponential size and has just one more color than $\mathcal{G}$, hence can be solved in exponential time, we obtain the following complexity result.

Theorem 4.1. *Solving pushdown parity games is in* EXPTIME.

Furthermore, one can show that solving pushdown parity games is also EXPTIME-hard, and therefore EXPTIME-complete. See Walukiewicz's paper "Pushdown processes: games and model checking, Information and Computation 164, 2001" for details.

5 Rabin's Theorem

In this chapter we consider an application of infinite games to automata theory by proving Rabin's theorem, which states that satisfiability of monadic second order logic over labeled binary trees (S2S for short) is decidable. Rabin's original proof was a purely combinatorial one, which has been characterized as cumbersome and complicated. Later, a simpler proof was published, which relies on determinacy of parity games.

The overall proof technique of Rabin 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[9], 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), complement (negation in S2S), and projection (quantification in S2S).

The most involved step in this translation is showing that parity tree automata are closed under complement, 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 test 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 come 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 (an universal statement), if and only if Player 1 has a winning strategy (an existential statement). Thus, in a determined infinite game we can turn an 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. Thus, 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.

5.1 Binary Trees

We start with an appropriate definition of trees. Therefore, consider the set $\mathbb{B}^*$. Technically, this set is only an infinite set of sequences, but by viewing each occurrence of a 0 or a 1 in a sequence $w \in \mathbb{B}^*$ as a decision of choosing one of two successors, we get an object describing positions, also called addresses, in a binary tree. Accordingly, the whole set $\mathbb{B}^*$ describes the set of every such address or correspondingly the complete binary tree.

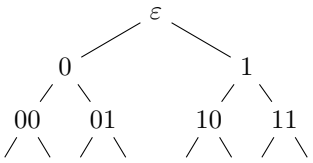


Figure 21: The complete binary tree

However, $\mathbb{B}^*$ alone is some quite boring object. Therefore, we want to extend every address by some individual labeling where we choose the corresponding labels from some predefined alphabet Σ. Correspondingly, we get a function $t\colon \mathbb{B}^* \to \Sigma$ that describes which address gets which label. This function already describes a Σ-labeled binary tree, which we refer to as a tree from now on. Summarizing, a tree is a function t with domain $\mathbb{B}^*$ and range Σ. As an example consider the tree t^e defined for all $w \in \mathbb{B}^*$ by

$$t^e(w) = \begin{cases} a & \text{if } w = 1^*0 \\ b & \text{otherwise} \end{cases}$$

[9] Most easily seen by defining an emptiness game with parity winning condition, another application of infinite games to automata theory.

A graphical representation of t^e is given in Figure 22.

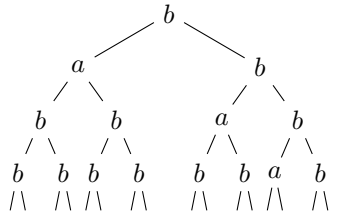


Figure 22: Graphical representation of t^e.

An alternative representation for a binary labeled tree is given as a so called relational structure. In the sequel, we will switch between both representations as necessary, but it should always be clear of the context which one we are talking about.

Definition 5.1. Given a tree $t\colon \mathbb{B}^* \to \Sigma$, we define the *structure* $\underline{t} = (\mathbb{B}^*, S_0, S_1, \preceq, (P_a)_{a\in\Sigma}, \varepsilon)$ consisting of

- the universe $\mathbb{B}^*$ of binary strings,
- the left successor relation $S_0 = \{\, (w, w0) \mid w \in \mathbb{B}^* \,\}$,
- the right successor relation $S_1 = \{\, (w, w1) \mid w \in \mathbb{B}^* \,\}$,
- the ancestor (or prefix) relation $\preceq = \{\, (w, ww') \mid w, w' \in \mathbb{B}^* \,\}$,
- a family of label assigning sets $(P_a)_{a\in\Sigma}$ with $P_a = \{\, w \in \mathbb{B}^* \mid t(w) = a \,\}$ and
- the root ε.

We close this section with some useful definitions.

Definition 5.2. Let $t\colon \mathbb{B}^* \to \Sigma$ be a tree. The *sub-tree* of t rooted in $w \in \mathbb{B}^*$ is denoted by t_w and defined by

$$\forall w' \in \mathbb{B}^*.\ t_w(w') = t(ww')$$

For an example, consider the sub-tree t^e_0 of t^e rooted in 0 in Figure 23. Other sub-trees are $t^e_1 = t^e_2 = ... = t$.

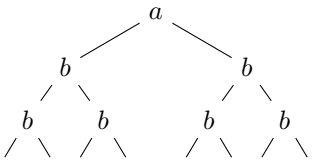


Figure 23: The tree t^e_0.

Definition 5.3. Let $t\colon \mathbb{B}^* \to \Sigma$ be a tree. An *unlabeled path* trough t is an infinite sequence $\pi = b_0b_1b_2... \in \mathbb{B}^\omega$ of binary directions, inducing a sequence of nodes

$$(\varepsilon)(b_0)(b_0b_1)(b_0b_1b_2)... \in (\mathbb{B}^*)^\omega$$

The corresponding (labeled) path through t we then denote by

$$t_{|\pi} = t(\varepsilon)t(b_0)t(b_0b_1)t(b_0b_1b_2)... \in \Sigma^\omega$$

Some paths trough t^e are for example:

$$t^e_{|0^\omega} = bab^\omega \qquad t^e_{|(10)^\omega} = bbab^\omega$$
$$t^e_{|1^\omega} = b^\omega \qquad t^e_{|(01)^\omega} = bab^\omega$$

5.2 Monadic Second-Order Logic of Two Successors

We now want to introduce a logic over such binary trees. This logic is called Monadic Second-Order Logic of two Successors, or short S2S. Some terms in this name may need some further explanation.

Second-Order:

> The term "second-order" here means that there are two types of quantification. One way is given by quantifying over objects (first-order quantification), the other way is to quantify over relations or functions (second-order quantification).

Monadic:

> The term "monadic" here refers to a restriction allowing second-order quantification only over unary relations, which are sets of objects.

Correspondingly, there are two different types of variables in S2S. First-order variables are denoted by small letter from the end of the english alphabet, e.g., $x, y, z, ...$ Second-order variables are denoted by capital letter from the end of the english alphabet, e.g., $X, Y, Z, ...$ We assume the existence of two universal sets $\mathcal{V}_1$ and $\mathcal{V}_2$ of variables containing every first-order and every second-order variable, respectively. Further, we demand that $\varepsilon \notin \mathcal{V}_1 \cup \mathcal{V}_2$ and that their intersection is empty, i.e. there cannot be a variable that is a first-order variable and a second-order variable at the same time. First-order variables are then used to describes addresses in our binary tree, second-order variables are used to describe sets of addresses. We now can define the syntax of S2S.

Definition 5.4. An *expression of S2S* is either a term, an atomic formula or a formula, which are defined recursively as follows.

- every $x \in \mathcal{V}_1$ is a term
- ε is a term

Let s, s' be terms and $X \in \mathcal{V}_2$, then

- $s = s'$ is an atomic formula
- $s \preceq s'$ is an atomic formula
- $P_a(s)$ is an atomic formula for every $a \in \Sigma$
- $S_i(s, s')$ is an atomic formula for every $i \in \{0, 1\}$
- $s \in X$ is an atomic formula

Every atomic formula is a formula. If φ and φ' are formulas, $x \in \mathcal{V}_1$ and $X \in \mathcal{V}_2$, then

- $\neg\varphi$ is a formula
- $\varphi \wedge \varphi'$ is a formula
- $\varphi \vee \varphi'$ is a formula
- $\exists x.\ \varphi$ is a formula
- $\forall x.\ \varphi$ is a formula
- $\exists X.\ \varphi$ is a formula
- $\exists X.\ \varphi$ is a formula

Note, that as we have all the boolean operators, we also have the derivable operators like $\rightarrow, \leftrightarrow, ...$ as usual. Next, we want to define the semantics of S2S. To do so, we first need to give the first-order and second-order variables our desired semantical meaning. We do this using a so-called valuation function μ.

Definition 5.5. A *variable valuation* $\mu\colon \mathcal{V}_1 \to \mathbb{B}^* \cup \mathcal{V}_2 \to 2^{\mathbb{B}^*} \cup \{\varepsilon\} \to \{\varepsilon\}$ is a function mapping every first-order variable x to $\mu(x) \in \mathbb{B}^*$ and every second-order variable X to $\mu(X) \subseteq \mathbb{B}^*$. Finally, μ maps ε to ε such that the domain of μ contains all terms.

Definition 5.6. 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

$$\mu[x \mapsto w](y) = \begin{cases} \mu(y) & \text{if } y \neq x \\ w & \text{if } y = x \end{cases}$$

Definition 5.7. 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

$$\mu[X \mapsto B](Y) = \begin{cases} \mu(Y) & \text{if } Y \neq X \\ B & \text{if } Y = X \end{cases}$$

Definition 5.8. The *semantics of S2S* are defined recursively using a satisfaction relation $\vDash$. Given a tree t and a variable valuation μ, we have

$$\begin{aligned}
t, \mu \vDash s = s' \quad &:\Leftrightarrow \quad \mu(s) = \mu(s') \\
t, \mu \vDash s \preceq s' \quad &:\Leftrightarrow \quad \mu(s) \in \mathrm{Pref}(\mu(s')) \\
t, \mu \vDash P_a(s) \quad &:\Leftrightarrow \quad t(\mu(s)) = a \\
t, \mu \vDash S_i(s, s') \quad &:\Leftrightarrow \quad \mu(s') = \mu(s)\, i \\
t, \mu \vDash x \in X \quad &:\Leftrightarrow \quad \mu(s) \in \mu(X) \\
t, \mu \vDash \neg\varphi \quad &:\Leftrightarrow \quad \text{it is not the case that } t, \mu \vDash \varphi \\
t, \mu \vDash \varphi \wedge \psi \quad &:\Leftrightarrow \quad t, \mu \vDash \varphi \text{ and } t, \mu \vDash \psi \\
t, \mu \vDash \varphi \vee \psi \quad &:\Leftrightarrow \quad t, \mu \vDash \varphi \text{ or } t, \mu \vDash \psi \\
t, \mu \vDash \exists x.\ \varphi \quad &:\Leftrightarrow \quad t, \mu[x \mapsto w] \vDash \varphi \text{ for some } w \in \mathbb{B}^* \\
t, \mu \vDash \forall x.\ \varphi \quad &:\Leftrightarrow \quad t, \mu[x \mapsto w] \vDash \varphi \text{ for all } w \in \mathbb{B}^* \\
t, \mu \vDash \exists X.\ \varphi \quad &:\Leftrightarrow \quad t, \mu[X \mapsto B] \vDash \varphi \text{ for some } B \subseteq \mathbb{B}^* \\
t, \mu \vDash \forall X.\ \varphi \quad &:\Leftrightarrow \quad t, \mu[X \mapsto B] \vDash \varphi \text{ for all } B \subseteq \mathbb{B}^*
\end{aligned}$$

Consider, that it makes a difference whether a variable occurs under the scope of a quantifier or not regarding the valuation according to μ. The following definition characterizes this difference more formally.

Definition 5.9. Let φ be an S2S formula. Then 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*. If φ has no free occurrences of variables it is called a *sentence*.

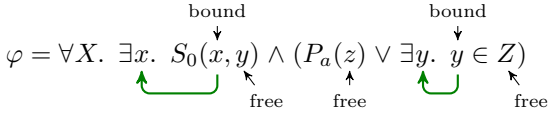


Figure 24: Example formula φ with free and bound variables.

Consider, that the satisfaction of sentences does not depend on the variable valuations. Accordingly, it suffices two write $t \vDash \varphi$ instead of $t, \mu \vDash \varphi$ and we just say t satisfies φ. We are interested in the satisfiability problem:

"Given a sentence φ, is there some t such that $t \vDash \varphi$?"

Using a language definition for φ, given by $\mathcal{L}(\varphi) = \{\, t \colon \mathbb{B}^* \to \Sigma \mid t \vDash \varphi \,\}$, we can also reformulate the problem as:

"Given a sentence φ, is $\mathcal{L}(\varphi)$ nonempty?"

In the upcoming sections, we will see how to solve this problem. We close this section by some examples for S2S formulas.

- $t^e \vDash P_b(\varepsilon)$
- $t^e \nvDash P_a(\varepsilon)$
- $t^e \vDash \exists x.\ S_1(\varepsilon, x) \wedge P_b(x)$
- $t^e \vDash \exists X.\ \textsc{path}(X) \wedge \forall x.\ (x \in X \Rightarrow P_b(x))$

  where $\textsc{path}(X)$ is defined as

$$\begin{aligned}\textsc{path}(X) = \varepsilon \in X \wedge \forall x.\ \big(x \in X \wedge \neg\exists y.\ S_0(y,x) \vee S_1(y,x)\big) \Rightarrow x = \varepsilon \wedge \\ \forall x.\ \big(x \in X \Rightarrow \exists y.\ y \in X \wedge (S_0(x,y) \vee S_1(x,y)) \wedge \\ \forall y'.\ (y' \in X \wedge (S_0(x,y') \vee S_1(x,y'))) \Rightarrow y = y'\big)\end{aligned}$$

  Saying the root is in X and the only node in X without predecessor and each node in X has a successor in X but not two.

5.3 Parity Tree Automata

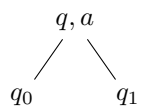
We want to show how do decide the satisfiability problem for S2S. However, as working in a logic is typically cumbersome, one first translates φ into an automaton $\mathscr{A}_\varphi$ recognizing exactly the trees satisfying φ. Then, one has to solve the emptiness problem for this automaton model. The model we consider here are parity tree automata. These automata work top-down, meaning that the initial state is assumed to be in 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 path of the run. Such an automaton is necessarily non-deterministic, as a deterministic automaton cannot even check whether there is an a-labeled node in the tree.

In this section, we introduce parity tree automata, show how to solve their emptiness problem, and prove their equivalence to S2S.

Definition 5.10. A *parity tree 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 relation $\Delta \subseteq Q \times \Sigma \times Q \times Q$ and
- a coloring function $\Omega \colon Q \to \mathbb{N}$.

We often depict a transition $\tau = (q, a, q_0, q_1)$ by



Thus, 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 5.11. 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

- $r(\varepsilon) = q_I$
- $\forall w \in \mathbb{B}^*.\ (r(w), t(w), r(w0), r(w1)) \in \Delta$

Note, that a run is a Q-labeled binary tree, i.e., all our notations developed for trees are applicable.

Definition 5.12. A run r of a parity tree automaton $\mathscr{A} = (Q, \Sigma, q_I, \Delta, \Omega)$ on a tree $t\colon \mathbb{B}^* \to \Sigma$ is *accepting* iff for every path $\pi \in \mathbb{B}^*$ the labeled path $r_{|\pi} = q_0 q_1 q_2 ... \in Q^\omega$ satisfies

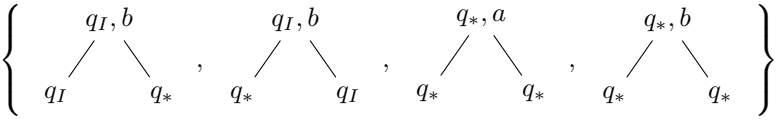
$$\mathrm{Par}(min(\mathrm{Inf}(\Omega(q_0)\Omega(q_1)\Omega(q_2)...))) = 0$$

We then can define the language of a parity tree automaton $\mathscr{A}$ as

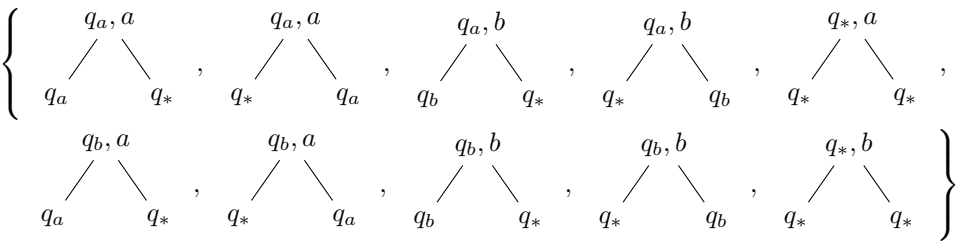
$$\mathcal{L}(\mathscr{A}) = \{\, t\colon \mathbb{B}^* \to \Sigma \mid \mathscr{A} \text{ has an accepting run on } t \,\}$$

A parity tree automaton $\mathscr{A} = (Q, \Sigma, q_I, \Delta, \Omega)$ is complete iff for every $q \in Q$ and every $q \in \Sigma$ there are $q_0, q_1 \in Q$ 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 assume all our automata to be complete, except for examples where we keep them incomplete to improve readability. To get some intuition for these new definitions, we want to construct parity tree automata recognizing the following languages of trees.

$$L_0 = \{\, t\colon \mathbb{B}^* \to \{a,b\} \mid \exists \pi.\ t_{|\pi} = b^\omega \,\} \qquad L_1 = \{\, t\colon \mathbb{B}^* \to \{a,b\} \mid \exists \pi.\ b \in \mathrm{Inf}(t_{|\pi}) \,\} \qquad L_2 = \{t^e\}$$

- We want to construct a parity tree automaton $\mathscr{A}_0$ that accepts every tree containing a path completely labeled with b's, i.e., such that $\mathcal{L}(\mathscr{A}_0) = L_0$. As parity tree automata are non-deterministic, $\mathscr{A}_0$ can guess such a path and verify that it only contains b's. For all other nodes in the tree we just need a dummy state allowing an arbitrary label. Accordingly, we can define $\mathscr{A}_0 = (Q_0, \{a,b\}, q_I, \Delta_0, \Omega_0)$ by
  - $Q_0 = \{q_I, q_*\}$
  - $\Omega_0(q_I) = \Omega_0(q_*) = 0$
  - $\Delta_0 =$ 

  Note that the automaton gets stuck if it encounters an a in state q_I. Therefore, the coloring is not needed and just having a run is enough. Further, every state has color 0.

- Now we want to construct a parity tree automaton $\mathscr{A}_1$ that accepts every tree containing a path with infinitely many b's. Similar to $\mathscr{A}_0$ we can guess the corresponding path. We only need to adjust the coloring function to control the infinite behavior. Further, we should not get stuck any more if we see an a. Our corresponding automaton $\mathscr{A}_1 = (Q_1, \{a,b\}, q_a, \Delta_1, \Omega_1)$ is then given by
  - $Q_1 = \{q_*, q_a, q_b\}$
  - $\Omega_1(q_b) = \Omega_1(q_*) = 0$ and $\Omega_1(q_a) = 1$
  - $\Delta_1 =$ 

  Accordingly, we have that on the guessed path if an a or a b is read, the next state is q_a or q_b, respectively. It follows that $\mathcal{L}(\mathscr{A}_1) = L_1$. Note, that using q_b as initial state does not change the language accepted by the automaton.

- Finally we want to construct a parity tree automaton $\mathscr{A}_2$ that only accepts our example tree t^e form the previous section. We construct $\mathscr{A}_2 = (Q_2, \{a, b\}, q_r, \Delta_2, \Omega_2)$ to get $\mathcal{L}(\mathscr{A}_2) = L_2$ as follows.
  - $Q_2 = \{q_*, q_r, q_l\}$
  - $\Omega_2(q_r) = \Omega_2(q_l) = \Omega_2(q_r) = 0$
  - $\Delta_2 = \left\{ \begin{array}{ccccc} q_r, b & & q_l, a & & q_*, b \\ \diagup\ \diagdown & , & \diagup\ \diagdown & , & \diagup\ \diagdown \\ q_l \quad q_r & & q_* \quad q_* & & q_* \quad q_* \end{array} \right\}$

We continue by showing one direction of the equivalence between S2S and parity tree automata stating that every language recognized by a parity tree automaton is also definable by some S2S sentence.

Theorem 5.1. *For every parity tree automaton $\mathscr{A}$ over some alphabet Σ there exists an S2S-sentence $\varphi_{\mathscr{A}}$ such that*

$$\forall t \colon \mathbb{B}^* \to \Sigma.\ t \in \mathcal{L}(\mathscr{A}) \Leftrightarrow t \vDash \varphi_{\mathscr{A}}$$

Proof. Let $\mathscr{A} = (Q, \Sigma, q_I, \Delta, \Omega)$ be some given parity tree automaton. We show how to construct a sentence $\varphi_{\mathscr{A}}$ that expresses that there exists an accepting run of $\mathscr{A}$ on t. Without loss of generality, let $Q = \{q_0, ..., q_{n-1}\}$ and $q_I = q_0$. We construct $\varphi_{\mathscr{A}}$ as follows:

$$\begin{aligned} \varphi_{\mathscr{A}} \quad = \quad & \exists X_0.\ \exists X_1.\ ...\exists X_{n-1}. \\ & \varepsilon \in X_0 \qquad \text{initial state at the root} \\ & \wedge\ \forall x.\ \bigvee_{j=0}^{n-1} \big(x \in X_j \wedge \bigwedge_{j' \neq j} \neg(x \in X_{j'})\big) \qquad \text{the } X_j \text{ partition the universe} \\ & \wedge\ \forall x.\ \exists x_l.\ \exists x_r.\ S_0(x, x_l) \wedge S_1(x, x_r) \wedge \qquad \text{the transition relation is satisfied} \\ & \qquad \bigvee_{(q_t, a, q_i, q_j) \in \Delta} \big(x \in X_t \wedge P_a(x) \wedge x_l \in X_i \wedge x_r \in X_j\big) \\ & \wedge\ \forall X.\ \textsc{path}(X) \Rightarrow \bigvee_{\substack{c \in \Omega(Q), \\ \mathrm{Par}(c)=0}} (\forall x.\ x \in X \Rightarrow \exists y.\ y \in X \wedge x \preceq y \wedge \bigvee_{\substack{j \in [n], \\ \Omega(q_j)=c}} y \in X_j) \wedge \\ & \qquad \text{minimal color on path } X \text{ is even} \qquad (\exists x.\ x \in X \wedge \forall y.\ y \in X \wedge x \preceq y \Rightarrow \bigvee_{\substack{j \in [n], \\ \Omega(q_j) \geq c}} y \in X_j) \end{aligned}$$

□

The backward direction turns out to be more complicated. 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.

Definition 5.13. Let $\mathscr{A} = (Q, \Sigma, q_I, \Delta, \Omega)$ be a complete parity tree automaton and $t \colon \mathbb{B}^* \to \Sigma$ be some tree. Then the parity game $\mathcal{G}(\mathscr{A}, t) = (\mathcal{A}, \textsc{parity}(\Omega'))$ with $\mathcal{A} = (V, V_0, V_1, E)$ is defined by

- $V = V_0 \cup V_1$
- $V_0 = \mathbb{B}^* \times Q$
- $V_1 = \{\, (w, \tau) \in \mathbb{B}^* \times \Delta \mid \exists q, q_0, q_1.\ \tau = (q, t(w), q_0, q_1) \,\}$
- $E = \{\, ((w, q), (w, \tau) \in V_0 \times V_1 \mid \exists q_0, q_1 \in Q.\ \tau = (q, t(w), q_0, q_1) \,\} \cup$
  $\{\, ((w, (q, t(w), q_0, q_1)), (wj, q_j)) \in V_1 \times V_0 \mid j \in \{0, 1\} \,\}$
- $\Omega'((w, q)) = \Omega(q)$ for all $(w, q) \in V_0$
- $\Omega'((w, (q, a, q_0, q_1))) = \Omega(q)$ 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 path π. During this, Player 0 has to pick transitions along this path 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 path 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 path 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 5.1. *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 $\mathcal{A} = (V, V_0, V_1, E)$.

"⇒": See Exercise 12.3.

"⇐": 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 starting in (ε, q_I), being consistent with σ and ending 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 will 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

$$p_{wj} = p_w \, (w, (q, t(w), q_0, q_1)) \, (wj, q_j)$$

for both $j \in \{0, 1\}$. Hence, $q_j = r(wj)$ 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 path of r and let $\rho \in \mathrm{Plays}(\mathcal{A}, \sigma, (\varepsilon, q_I))$ 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 path π it finally follows that also r is an accepting run. □

Before we continue proving closure under complement, we show that the acceptance game can be modified to solve the emptiness problem for parity tree automata. Note that $\mathcal{G}(\mathscr{A}, t)$ is played in an infinite arena, since it encodes the nodes of t. The main idea behind the emptiness game is that we take the acceptance game $\mathcal{G}(\mathscr{A}, t)$ and project t away which yields a finite game where Player 0 has to guess a tree t and an accepting run of $\mathscr{A}$ on t. We can define the emptiness game $\mathcal{G}(\mathscr{A})$ as follows.

Definition 5.14. Let $\mathscr{A} = (Q, \Sigma, q_I, \Delta, \Omega)$ be a complete parity tree automaton over Σ. Then the parity game $\mathcal{G}(\mathscr{A}) = (\mathcal{A}, \textsc{parity}(\Omega'))$ with $\mathcal{A} = (V, V_0, V_1, E)$ is defined by

- $V = V_0 \cup V_1$
- $V_0 = Q$
- $V_1 = \Delta$
- $E = \{ (q, \tau) \in V_0 \times V_1 \mid \exists q_0, q_1 \in Q.\ \exists a \in \Sigma.\ \tau = (q, a, q_0, q_1) \} \cup$
  $\{ ((q, a, q_0, q_1), q_j) \in V_1 \times V_0 \mid j \in \{0, 1\} \}$
- $\Omega'(q) = \Omega(q)$ for all $q \in V_0$
- $\Omega'((q, a, q_0, q_1)) = \Omega(q)$ for all $(q, a, q_0, q_1) \in V_1$

It remains to show that $\mathcal{G}(\mathscr{A})$ characterizes emptiness of $\mathscr{A}$.

Theorem 5.2. *Let $\mathscr{A}$ be a parity tree automaton over Σ. Then it holds that*

$$q_I \in W_0(\mathcal{G}(\mathscr{A})) \Leftrightarrow \mathcal{L}(\mathscr{A}) \neq \emptyset$$

Proof. Let $\mathscr{A} = (Q, \Sigma, q_I, \Delta, \Omega)$ and $\mathcal{G}(\mathscr{A}) = (\mathcal{A}, \text{PARITY}(\Omega'))$ with $\mathcal{A} = (V, Q, \Delta, E)$.

"$\Rightarrow$":

See Exercise 13.1.

"$\Leftarrow$":

Let $t \in \mathcal{L}(\mathscr{A})$ be arbitrary and r be the corresponding run of $\mathscr{A}$ on t. We define a strategy σ for Player 0 from q_I inductively by

- $\sigma(q_I) = (r(\varepsilon), t(\varepsilon), r(0), r(1))$
- $\sigma(q_I \tau_I q_1 \tau_1 ... \tau_{n-1} q_n) = (r(w), t(w), r(w0), r(w1))$

  for every prefix $q_I \tau_I q_1 \tau_1 ... \tau_{n-1} q_n$ consistent with σ and $w = b_0 ... b_{n-1}$ such that for all $j \in [n]$ there are some $a_j \in \Sigma$ and $q_j, q_j' \in Q$ giving

$$b_j = \begin{cases} 0 & \text{if } \tau_j = (q_j, a_j, q_{j+1}, q_j') \\ 1 & \text{if } \tau_j = (q_j, a_j, q_j', q_{j+1}) \end{cases}$$

Now let $\rho \in \text{Plays}(\mathcal{A}, \sigma, q_I)$ be arbitrary and $b_0 b_1 b_2 ...$ be the infinite sequence of associated directions. Consider that such a sequence always exists as σ is applicable to every $v \in \text{Occ}(\rho) \cap V_0$ by definition. Further, let $c_0 c_1 c_2 ...$ be the sequence of colors seen during ρ, where we have that $c_{2n+1} = c_{2n}$ for all $n \in \mathbb{N}$. Then the sequence $c_0 c_2 c_4 ...$ is equal to $\Omega(r(\varepsilon))\Omega(r(b_1))\Omega(r(b_1 b_2))$, i.e., the colors on the path $r_{|b_0 b_1 b_2 ...}$ of r. As this sequence satisfies the parity condition by acceptance of t it follows that also ρ satisfies the parity condition. Hence, σ is winning from q_I.- □

Thus, we have shown that the emptiness problem for parity tree automata can be reduced to solving a parity game that is polynomially-sized in the size of the automaton.

After the detour of introducing the emptiness game, 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, if $(\varepsilon, q_I) \in W_1(\mathcal{G}(\mathscr{A}, t))$. 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 if and only if the strategy encoded by s is winning for $\mathcal{G}(\mathscr{A}, t)$.

First, we encode a strategy for Player 0 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 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 $(w1, q_1)$. 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 strategy is local since 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 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 strategy tree.

Remark 5.1. *$t \notin \mathcal{L}(\mathscr{A})$ if and only if there is a winning strategy tree for t.*

Thus, we need to 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 path π and every strategy for Player 0, whether the parity condition of $\mathscr{A}$ is satisfied if Player 1 picks the path 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 parity word automaton $\mathscr{M} = (Q, \Sigma', q_I, \Delta', \Omega)$ where the set of states, the initial state, and the coloring function are as in $\mathscr{A}$ and with $\Sigma' = \mathbb{B}^\Delta \times \Sigma \times \mathbb{B}$ and the transition $(q, (f, a, b), q')$ is contained in the transition relation Δ' of the automaton $\mathscr{M}$, 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 M from state q to state q_b.

Fix some tree $t\colon \mathbb{B}^* \to \Sigma$ and a strategy tree $s\colon \mathbb{B}^* \to \mathbb{B}^\Delta$, let $\pi = b_1 b_2 b_3 \cdots \in \mathbb{B}^\omega$ be a path. We define the word $w(s, t, \pi)$ by

$$w(s,t,\pi) = (s(\varepsilon), t(\varepsilon), b_1)\,(s(b_1), t(b_1), b_2)\,(s(b_1b_2), t(b_1b_2), b_3)\,(s(b_1b_2b_3), t(b_1b_2b_3), b_4)\cdots \in (\Sigma')^\omega$$

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 path π, together with the path itself.

Using these definitions, we can characterize s being a winning strategy tree for t in terms of the automaton $\mathscr{M}$.

Lemma 5.2. *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 and assume there is a path $\pi = b_0 b_1 b_2 \cdots$ 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 define a strategy for Player 0 against the strategy for Player 1 encoded by s. Consider the resulting 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 contains a position of length j. Since w_{j+1} is obtained from w_j by appending a bit, the w_j induce an infinite path $\pi = b_0 b_1 b_2 \cdots$ 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. This completes the proof. □

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.

Theorem 5.3. *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 function 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 all $b \in \mathbb{B}$. Thus, $\mathscr{B}$ indeed simulates $\mathscr{S}$ along every path 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 5.3. *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 5.2, we have $L(s,t) \cap \mathcal{L}(\mathscr{M}) = \emptyset$ and therefore $L(s,t) \subseteq \mathcal{L}(\mathscr{S})$. Hence, for every path 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

$$(s(\varepsilon), t(\varepsilon), b_0)\,(s(b_0), t(b_0), b_1)\,(s(b_0b_1), t(b_0b_1), b_2) \cdots (s(b_0b_1\cdots b_{j-1}), t(b_0b_1\cdots b_{j-1}), b_j) \ .$$

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 path, 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 path. 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 5.2. □

Now, we can finish our argument by applying closure of parity tree automata under projection, i.e., our final automaton $\mathscr{B}'$ runs on t and guesses a strategy tree s and uses $\mathscr{B}$ to verify that s is winning for t.

Theorem 5.4. *Parity tree automata are closed under complement.*

Proof. Let $\mathscr{B}'$ recognize the projection of $\mathcal{L}(\mathscr{B})$ to the first component (see Exercise 12.2). 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 5.1, $\mathscr{B}'$ 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 first-order quantification and syntactic sugar, obtaining the logic $\mathrm{S2S}_0$. First-order quantification is simulated by second-order quantification of singletons. Thus, we need new atomic formulas that replace the atomic formulas of S2S.

Definition 5.15. Let $X, X' \in \mathcal{V}_2$ be second-order variables. The *atomic formulas of S2S$_0$* are

- $\mathrm{Sing}(X)$,
- $X \subseteq P_a$ for $a \in \Sigma$,
- $X \subseteq X'$, and
- $S_i(X, X')$ for $i \in \mathbb{B}$.

Furthermore, every atomic formula is a formula of $\mathrm{S2S}_0$. If φ and ψ are formulas and $X \in \mathcal{V}_2$, 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 X is a singleton and letter a is at the unique position contained in X, while $S_i(X, X')$ holds if $X = \{x\}$ and $X' = \{x'\}$ are singletons and x' is the i-successor of x. Formally, we define the semantics as follows.

Definition 5.16. The *semantics of S2S$_0$* are defined recursively using a satisfaction relation $\vDash$. Given a tree t and a variable valuation μ, we have

$$
\begin{aligned}
t, \mu \vDash \mathrm{Sing}(X) &\quad :\Leftrightarrow \quad |\mu(X)| = 1 \\
t, \mu \vDash X \subseteq P_a &\quad :\Leftrightarrow \quad \mu(X) = \{w\} \text{ and } t(w) = a \text{ for some } w \in \mathbb{B}^* \\
t, \mu \vDash X \subseteq X' &\quad :\Leftrightarrow \quad \mu(X) \subseteq \mu(X') \\
t, \mu \vDash S_i(X, X') &\quad :\Leftrightarrow \quad \mu(X) = \{w\} \text{ and } \mu(X') = \{wi\} \text{ for some } w \in \mathbb{B}^* \\
t, \mu \vDash \neg\varphi &\quad :\Leftrightarrow \quad \text{it is not the case that } t, \mu \vDash \varphi \\
t, \mu \vDash \varphi \vee \psi &\quad :\Leftrightarrow \quad t, \mu \vDash \varphi \text{ or } t, \mu \vDash \psi \\
t, \mu \vDash \exists X.\ \varphi &\quad :\Leftrightarrow \quad t, \mu[X \mapsto B] \vDash \varphi \text{ for some } B \subseteq \mathbb{B}^*
\end{aligned}
$$

Now, we can show that we can turn every S2S-sentence into an equivalent S2S$_0$-sentence.

Lemma 5.4. *For every S2S-sentence φ there is an S2S$_0$ sentence φ' such that $t \models \varphi \Leftrightarrow t \models \varphi'$ for every tree t.*

Proof. First, we showed in Exercise 11.2 that ε and $\preceq$ are syntactic sugar and can be replaced. 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 an 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. We define φ' by induction over the construction of S2S-formulas using the remaining atomic formulas, boolean connectives, and existential quantification.

- $(x = y)' = \mathrm{Sing}(X) \wedge \mathrm{Sing}(Y) \wedge X \subseteq Y \wedge Y \subseteq X$
- $(P_a(x))' = X \subseteq P_a$
- $(S_i(x, y))' = S_i(X, Y)$
- $(x \in Y)' = \mathrm{Sing}(X) \wedge X \subseteq Y$
- $(\neg\varphi)' = \neg(\varphi')$
- $(\varphi \vee \psi)' = \varphi' \vee \psi'$
- $(\exists x\varphi)' = \exists X \mathrm{Sing}(X) \wedge (\varphi')$
- $(\exists X\varphi)' = \exists X(\varphi')$

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 S2S$_0$. □

Thus, it suffices to show how to translate 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 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 5.5. *For every S2S$_0$-formula φ there is a parity tree automaton $\mathscr{A}_\varphi$ such that $t, \mu \models \varphi \Leftrightarrow t^\frown t_\mu \in \mathcal{L}(\mathscr{A}_\varphi)$ for every tree t and every variable valuation μ. Especially, if φ is a sentence, we have $t \models \varphi \Leftrightarrow t \in \mathcal{L}(\mathscr{A}_\varphi)$.*

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 path 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 path 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 but 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 path to a position where the component encoding X_j has a 1 in the component encoding $X_{j'}$, checks that the i-successor has a 1 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 $\mathcal{L}(\mathscr{A}_\varphi)$.
- $\mathscr{A}\varphi \vee \psi$ is the automaton recognizing $\mathcal{L}(\mathscr{A}_\varphi) \cup \mathcal{L}(\mathscr{A}_\psi)$.
- $\mathscr{A}_{\exists X_j \varphi}$ is the automaton recognizing the projection of $\mathcal{L}(\mathscr{A}_\varphi)$ to all components but the one encoding X_j. □