Solving Infinite Games with Bounds

Von der Fakultät für Mathematik, Informatik und Naturwissenschaften der RWTH Aachen University zur Erlangung des akademischen Grades eines Doktors der Naturwissenschaften genehmigte Dissertation

vorgelegt von

Diplom-Informatiker
Martin Zimmermann
aus Köln

Berichter: Universitätsprofessor Dr. Dr.h.c. Wolfgang Thomas
Universitätsprofessor Dr. Jean-François Raskin

Tag der mündlichen Prüfung: 27. Februar 2012

Zusammenfassung

Wir untersuchen die Existenz von optimalen Strategien in unendlichen Spielen auf Graphen, sowie die Komplexität die nötig ist, um diese zu berechnen und zu implementieren.

Parametrisierte lineare temporale Logiken sind Erweiterungen von „Linear Temporal Logic“ (LTL) mit temporalen Operatoren, die mit variablen Zeitschranken versehen werden können. Solche Logiken wurden als „PLTL“ von Alur et al. und als „PROMPT-LTL“ von Kupferman et al. für das Model-Checking entwickelt. Wir zeigen, dass in doppelt-exponentieller Laufzeit entschieden werden kann, ob ein Spieler ein Spiel mit PLTL-Gewinnbedingung bezüglich mindestens einer, unendlich vieler, oder aller Variablenbelegungen gewinnt. Diese Probleme sind also nicht schwerer als das Lösen von LTL-Spielen. Weiterhin stellen wir einen Algorithmus mit dreifach-exponentieller Laufzeit vor, der optimale Variablenbelegungen bestimmt, bezüglich derer ein Spieler gewinnt. Schließlich zeigen wir doppelt-exponentielle obere und untere Schranken für die Werte von optimalen Variablenbelegungen.

In Muller-Spielen messen wir die Qualität einer Gewinnstrategie mit McNaughtons Score-Werten. Wir konstruieren Gewinnstrategien, die alle Score-Werte des verlierenden Spielers mit zwei beschränken und zeigen, dass der Wert zwei optimal ist. Damit verringern wir die bisher beste obere Schranke $n!$ in einem Muller-Spiel mit n Knoten von McNaughton. Unter Benutzung dieser Strategien zeigen wir, wie aus einen Muller-Spiel ein Safetyspiel konstruiert werden kann, dessen Lösung es ermöglicht, beide Gewinnregionen und eine Gewinnstrategie für einen Spieler im ursprünglichen Muller-Spiel zu bestimmen. Dadurch erhalten wir eine neue Antiketten-basierte Speicherstruktur und die erste Definition von permissiven Strategien für Muller-Spiele. Zusätzlich verallgemeinern wir diese Konstruktion, indem wir einen neuen Begriff einer Spielreduktion von beliebigen Spielen auf Safety-Spiele einführen, und zeigen, dass dieser auf viele andere Gewinnbedingungen anwendbar ist.

Abstract

We investigate the existence and the complexity of computing and implementing optimal winning strategies for graph games of infinite duration.

Parameterized linear temporal logics are extensions of Linear Temporal Logic (LTL) by temporal operators equipped with variables for time bounds. In model-checking, such specifications were introduced as "PLTL" by Alur et al. and as "PROMPT-LTL" by Kupferman et al. We show how to determine in doubly-exponential time, whether a player wins a game with PLTL winning condition with respect to some, infinitely many, or all variable valuations. Hence, these problems are not harder than solving LTL games. Furthermore, we present an algorithm with triply-exponential running time to determine optimal variable valuations that allow a player to win a game. Finally, we give doubly-exponential upper and lower bounds on the values of optimal variable valuations.

In Muller games, we measure the quality of a winning strategy using McNaughton's scoring functions. We construct winning strategies that bound the losing player's scores by two and show this to be optimal. This improves the previous best upper bound of $n!$ in a game with n vertices, obtained by McNaughton. Using these strategies, we show how to transform a Muller game into a safety game whose solution allows to determine the winning regions of the Muller game and to compute a finite-state winning strategy for one player. This yields a novel antichain-based memory structure and the first definition of permissive strategies for Muller games. Moreover, we generalize our construction by presenting a new type of game reduction from infinite games to safety games and show its applicability to several other winning conditions.

Contents

List of Figures

Chapter 1

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. The emergence of such 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 [Chu57, Chu63]: 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.

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 [BL69], 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.

Hence, 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. Commonly used formalisms include linear temporal logics and acceptance conditions from the theory of automata on infinite words. These were the focus of intensive research that classified the computational complexity of the solution problem for almost all types of winning conditions and resulted in (typically) tight upper and lower bounds on the size of winning strategies. A notable and important exception concerns the complexity of solving parity games: the problem is known to be in $\mathbf{NP} \cap \mathbf{Co\text{-}NP}$ and subexponential algorithms were found, but it is open whether parity games can be solved in polynomial time.

To make synthesis applicable in practice, a good compromise between the expressiveness of winning conditions on the one hand and the resulting computational complexity of solving the game and the size of winning strategies on the other hand has to be found. Typically, reachability, safety,

Büchi-, and co-Büchi games, which can be solved in polynomial time and have small winning strategies, are too weak to express important properties, while expressive formalisms like Linear Temporal Logic (LTL) or request-response conditions are considered infeasible and require prohibitively large finite-state winning strategies. Only recently, the first tools to solve games with LTL winning conditions were published [JB06, FJR11, Ehl11, BBF+] and showed promising results. Further developments might show the perception of LTL synthesis being infeasible wrong and make LTL synthesis, in spite of its high worst-case complexity, applicable in practice.

In this work, we focus on a third facet of synthesis besides expressiveness and complexity: the quality of a controller. For many winning conditions there exist one or more natural preference orders on winning strategies. For example, for a request-response condition of the form "every request of a resource has to be granted eventually", we might be interested in a strategy that answers requests as soon as possible or in minimizing the average waiting time between a request and a response. Computing (approximatively) optimal winning strategies with respect to a quality measure is another challenge that has to be met in order to make synthesis viable in practical applications.

This aspect is not addressed by classical solution algorithms, which determine a winning strategy with no (explicit) regard to optimality according to quality measures. Furthermore, it could be the case that there are trade-offs between different quality measures, which would require the application of different algorithms depending on the measure under consideration. The situation changed in the last years, when more and more attention was being paid to synthesizing optimal winning strategies. This includes the use of weighted automata to measure the quality of plays and strategies [BCHJ09, CHJS11, CH+11] and work on request-response games [HTW08] and their extensions [Zim09].

We consider two types of winning conditions, each equipped with a natural quality measure that allows to investigate the existence of optimal (with respect to the measure) winning strategies and the complexity of finding and implementing them. We begin with an extension of LTL with parameterized operators which allow to formulate explicit timing constraints in a specification. Here, we are interested in minimizing or maximizing these bounds. Our work on this high-level specification language is complemented by work on Muller games, a low-level, but still expressive and challenging automata-theoretic winning condition. Here, we use scoring functions for Muller games and aim to minimize the losing player's scores.

Synthesis from Parametric Linear Temporal Logics

The first extension of LTL by operators for explicit timing constraints was Metric Temporal Logic [Koy90, AH93] which allows to restrict the scope of temporal operators with intervals of natural numbers. For example, the for-

mula $\mathbf{F}_{(a,b)}\varphi$ is satisfied at a position n, if φ holds at least once between the positions $n + a$ and $n + b$. This allows to verify whether a given bound is guaranteed, but finding one is cumbersome and its value depends on the granularity of the model. To overcome these shortcomings, Alur et al. introduced Parametric Linear Temporal Logic [AETP01] with parametric bounds of the form $\mathbf{F}_{\leq x}$ (the parameterized eventually operator) and $\mathbf{G}_{\leq y}$ (the parameterized always operator), where x and y are variables. Thus, satisfaction is defined with respect to a variable valuation α mapping variables to natural numbers: $\mathbf{F}_{\leq x}\varphi$ is satisfied with respect to α, if φ holds at least once during the next $\alpha(x)$ steps. Dually, $\mathbf{G}_{\leq y}\varphi$ is satisfied with respect to α, if φ holds throughout the next $\alpha(y)$ positions. This formalism allows to ask whether there *exists* a variable valuation such that a given PLTL formula is satisfied and even ask for optimal variable valuations: for parameterized eventually operators we are interested in minimizing the variable values while we want to maximize the values for parameterized always operators. This allows to specify that we want requests to be served as soon as possible or that we want to maximize the uptime of a system.

Alur et al. showed that the analogues of LTL model-checking – determining whether a transition system satisfies a PLTL formula with respect to some, infinitely many, or all variable valuations – are **Pspace**-complete, which is also the complexity of LTL model-checking. Thus, adding parameterized operators to LTL does not increase the computational complexity of the model-checking problem. For two important fragments of PLTL they even showed how to solve optimization problems: if a formula contains only parameterized eventually operators or only parameterized always operators, then optimal variable valuations (according to several natural notions of optimality) can be computed in polynomial space.

Later, Kupferman et al. [KPV09] considered the fragment of PLTL containing no parameterized always' (called PROMPT–LTL in their work), gave an alternative model-checking algorithm, and solved the assume-guarantee model-checking and the realizability problem for PROMPT–LTL.

We lift these results to infinite games by showing that determining whether a given player wins a game with PLTL winning condition with respect to some, infinitely many, or all variable valuations is **2Exptime**-complete, which matches the complexity of solving games with LTL winning conditions. For the optimization problems for infinite games with winning conditions in the fragments described above, we give an algorithm that determines optimal variable valuations in triply-exponential time. Hence, there is an exponential gap between the lower bound obtained by the **2Exptime**-hardness of solving LTL games and our algorithm. We complement this with doubly-exponential upper and lower bounds on the values of optimal valuations.

The results on the decision and optimization problems for PLTL were published in [Zim11] while the lower bounds on optimal variable valuations are unpublished joint work with Christof Löding.

Optimal Strategies for Muller Games

A Muller game consists of graph with vertex set V and a partition $(\mathcal{F}_0, \mathcal{F}_1)$ of the power set of V. Player i wins a play if the set of vertices that are visited infinitely often by the play is in $\mathcal{F}_i$. Therefore, Muller games generalize every other type of game in which the winner of a play only depends on the set of vertices visited infinitely often, e.g., Büchi, co-Büchi, parity, Rabin, and Streett games. The complexity of solving Muller games depends on how succinct the partition $(\mathcal{F}_0, \mathcal{F}_1)$ is encoded, but is **Pspace**-complete for concise encodings [HD05], and in **NP** [DJW97] respectively in **P** [Hor08] for (more or less) explicit encodings. Muller games can either be solved by direct algorithms [McN93, Zie98, Hor08] or by reductions to parity games. As a consequence of these reductions, Muller games are determined with finite-state strategies of size $n!$, where n is the size of the game graph, and there are matching lower bounds [DJW97].

While investigating the interest of Muller games for "casual living-room recreation" [McN00], McNaughton introduced scoring functions to describe the progress a player is making towards winning a play. The score for a set F of vertices measures how often F has been visited completely since the last visit of a vertex that is not in F. If such a vertex is visited, then the score is reset to zero. Player i wins a play if and only if there is an $F \in \mathcal{F}_i$ such that the score for F tends to infinity while being reset only finitely often. Hence, the winning player of a play can be characterized by the evolution of the scores during the play.

McNaughton used these functions to propose a finite-duration variant of Muller games to be played by humans: the players construct a play until some score for a set F reaches a predefined threshold score. At this point, the play is stopped and Player i with $F \in \mathcal{F}_i$ wins this finite play. By proving the existence of strategies for the winning player of the Muller game that bound her opponent's scores for each set F by $|F|!$ – which are winning for both games – McNaughton showed that Player i wins the Muller game from a vertex v if and only if she wins the finite-duration game with threshold $|F|!+1$ from v.

In this work, we improve the bound $|F|!$ by showing that the losing player's score can even be bounded by two, no matter how large the graph is and how complicated the partition $(\mathcal{F}_0, \mathcal{F}_1)$ is. This shows that McNaughton's threshold scores can lowered from $|F|! + 1$ to three while retaining the fact that from a given vertex, the same player wins both games. We complement this by showing that the losing player can enforce of a score of two in some games, which proves our results to be optimal in this sense. Furthermore, we provide tight upper and lower bounds on the length of plays that avoid a score of $|F|! + 1$ respectively three.

Using the existence of winning strategies that bound the losing player's scores by two, we show how to determine the winning regions of a Muller

game and a finite-state winning strategy for one player by solving a safety game. This extends previous work on reductions from co-Büchi and parity games to safety games having the same properties. As a byproduct of our construction, we obtain a new memory structure for Muller games and a natural generalization of permissive strategies from parity games to Muller games. In a parity game, a non-deterministic strategy is permissive if it subsumes the behavior of every positional (non-deterministic) winning strategy. We show that this definition can be rephrased in terms of scoring functions for parity games and then generalize this new definition to Muller games.

Finally, we present a new notion of reduction from arbitrary games to safety games which encompasses all reductions mentioned above and present several other applications. Unlike classical game reductions, which are limited by topological properties of the winning plays, our reduction overcomes these obstacles. However, a price has to be paid: our reductions only yield the winning regions and a winning strategy for one player, while classical reductions yield both winning regions and strategies for both players. This is made up for by the fact that safety games can be solved by a simple, linear time algorithm, whereas, e.g., solving Muller games by classical game reductions requires to solve a parity game, a task for which only superpolynomial algorithms are known.

The existence of strategies bounding the losing player's scores by two was obtained in collaboration with John Fearnley [FZ12] while the reduction from Muller to safety games was developed in collaboration with Daniel Neider and Roman Rabinovich [NRZ11].

Outline

This work is organized as follows. In Chapter 2, we formalize notions like games, strategies, and game reductions, and conclude by listing some results about finite games. In Chapter 3, we present our results on games with PLTL winning conditions. We define the syntax and semantics and state some basic properties of PLTL and of games with winning conditions in PLTL in Section 3.1. Then, we solve the decision problems in Section 3.2 and the optimization problems in Section 3.3. In Chapter 4, we recap McNaughton's work on scoring functions in Section 4.1 and prove the existence of strategies that bound the losing player's score by two in Section 4.2. Then, in Chapter 5, after a short introduction to the Borel hierarchy in Section 5.1, we show how to reduce parity and Muller games to safety games using scoring functions in Section 5.2 and Section 5.3, respectively. We conclude this chapter by a new notion of game reduction that encompasses these constructions in Section 5.4. Finally, in Chapter 6, we give a conclusion and list some open questions and pointers to further research.

Acknowledgments

I want to thank my advisor Wolfgang Thomas for his constant support and for giving me advice and guidance whenever I needed it. Furthermore, he brought McNaughton's work on playing infinite games in finite time to my attention and encouraged me to work on this question, which led to the results presented in Chapter 4 and thereby a substantial part of this thesis.

Furthermore, I want to thank the external reviewer Jean-François Raskin for readily taking on this task and Joost-Pieter Katoen and Thomas Seidl for completing my thesis committee.

Many of the results presented here were obtained in collaborations: I want to thank my co-authors John Fearnley, Daniel Neider, and Roman Rabinovich, and the following people who directly contributed to this thesis: Namit Chaturvedi for proving the lower bound presented in the appendix, Christof Löding for a discussion that led to Theorem 3.43 (and to a whiteboard falling on my forehead), and Roman Rabinovich for a fruitful train ride after the GAMES Workshop 2010 during which we found the Muller game presented in Theorem 4.16 and for coining the term "blinking semantics".

The results presented in Chapter 4 are the outcome of a visit in Warwick. I am grateful to Marcin Jurdziński for inviting me to visit him and John.

Furthermore, I want to thank Wladimir Fridman for innumerable discussions and for drawing large graphs way beyond the call of duty. Finally, Namit Chaturvedi, Wladimir Fridman, Daniel Neider, Jörg Olschewski, Stefan Repke, and Roman Rabinovich earned my gratitude for proof-reading parts of this thesis.

Finally, I want to thank my wife Nadine Wacker for still challenging me every day and still just being the way you are. Good times lie ahead of us. But no basketball scores this time, stupid lockout.

Chapter 2

Preliminaries

This chapter is devoted to presenting definitions and notations used throughout this work. After some preliminaries and after fixing our notation for automata on finite and infinite words, we give a thorough introduction to infinite games comprising winning conditions, strategies, and finite-state strategies, as well as game reductions. We conclude by listing some known results about infinite games which are used in the remainder of this work.

2.1 Basic Definitions

The set of non-negative integers is denoted by $\mathbb{N}$. We denote the parity of an integer n by $\mathrm{Par}(n)$, i.e., $\mathrm{Par}(n) = 0$ if n is even, and $\mathrm{Par}(n) = 1$ if n is odd. For $n \in \mathbb{N}$, let $[n] = \{0, \ldots, n-1\}$; especially, $[0] = \emptyset$. The cardinality of a set S is denoted by $|S|$, and its power set by 2^S.

An alphabet Σ is a non-empty, finite set of letters or symbols. The set of finite words over Σ is denoted by Σ^*, the set of non-empty words by Σ^+, and the empty word by ε. Furthermore, the set of ω-words over Σ is denoted by Σ^ω. A language[1] is either a subset of Σ^* or a subset of Σ^ω.

The length of a finite word w is denoted by $|w|$. Given a finite or infinite word w, we denote its n-th letter by w_n, where the first letter of a non-empty word is w_0. Furthermore, we also use the notation $\mathrm{Lst}(w)$ to denote the last letter of a non-empty finite word w.

Concatenation of a finite word w and a (finite or infinite) word w' is denoted by ww'. Similarly, concatenation of a language L of finite words and a language L' of (finite or infinite) words is denoted by $L \cdot L'$. The prefix relation on words is denoted by $\sqsubseteq$. The set of prefixes of a (finite or infinite) word w is denoted by $\mathrm{Pref}(w)$. If $w = xy$, then $x^{-1}w = y$ and $wy^{-1} = x$.

The occurrence set $\mathrm{Occ}(w)$ of a finite or infinite word w over an alpha-

[1] When we speak of a language, it is always clear from the context whether it contains finite or infinite words.

bet Σ is defined by

$$\mathrm{Occ}(w) = \{a \in \Sigma \mid \text{there exists an } n \text{ such that } w_n = a\} \ ,$$

and the infinity set $\mathrm{Inf}(w)$ of an infinite word w is defined by

$$\mathrm{Inf}(w) = \{a \in \Sigma \mid \text{there exist infinitely many } n \text{ such that } w_n = a\} \ .$$

The following complexity classes of decision problems appear in this work. We refrain from giving formal definitions (see, e.g., [Pap94]), since we only use them to classify the complexity of solving the games we consider here.

- **P**: deterministic polynomial time.
- **UP**: unambiguous[2] non-deterministic polynomial time.
- **NP**: non-deterministic polynomial time.
- **Pspace**: polynomial space.
- **2Exptime**: deterministic doubly-exponential time, i.e., running time bounded by $2^{2^{p(n)}}$ for some polynomial p.

Furthermore, **Co-UP** and **Co-NP** denote the complement classes of **UP** and **NP**, respectively. For example, **Co-NP** contains all decision problems whose complement problem is in **NP**. The following inclusions hold: **P** $\subseteq$ **UP** $\subseteq$ **NP** $\subseteq$ **Pspace**, **P** $\subseteq$ **Co-UP** $\subseteq$ **Co-NP** $\subseteq$ **Pspace**, and **Pspace** $\subseteq$ **2Exptime**.

2.2 Automata

In this section, we introduce our notation for automata on finite (Subsection 2.2.1) and infinite (Subsection 2.2.2) words. Furthermore, we define unambiguity and non-confluence of ω-automata, which play an important role when we determinize them in a later chapter.

2.2.1 Automata on Finite Words

A (non-deterministic) finite automaton $\mathfrak{A} = (Q, \Sigma, Q_0, \Delta, F)$ consists of a finite set Q of states, an alphabet Σ, a set $Q_0 \subseteq Q$ of initial states, a transition relation $\Delta \subseteq Q \times \Sigma \times Q$, and a set $F \subseteq Q$ of final states. The automaton is deterministic, if $|Q_0| = 1$ and if for every $q \in Q$ and every $a \in \Sigma$ there is a unique $q' \in Q$ such that $(q, a, q') \in \Delta$. In this case, we

[2] A Turing machine is unambiguous if it has at most one accepting computation on every input.

denote $Q_0 = \{q_0\}$ by q_0 and Δ as function $\delta\colon Q \times \Sigma \to Q$. The size of $\mathfrak{A}$, denoted by $|\mathfrak{A}|$, is the number $|Q|$ of states of $\mathfrak{A}$.

A run of $\mathfrak{A}$ on a word $w \in \Sigma^*$ is a sequence of states $q_0 \cdots q_{|w|}$ such that $q_0 \in Q_0$ and $(q_n, w_n, q_{n+1}) \in \Delta$ for every n in the range $0 \leq n < |w|$; it is accepting if $q_{|w|} \in F$. A deterministic automaton has a unique run on every w. A word w is accepted by $\mathfrak{A}$, if there exists an accepting run of $\mathfrak{A}$ on w. The language recognized by $\mathfrak{A}$ is $L(\mathfrak{A}) = \{w \in \Sigma^* \mid \mathfrak{A} \text{ accepts } w\}$.

2.2.2 Automata on Infinite Words

In this subsection, we give a general definition of ω-automata which encompasses all types of ω-automata that appear in the literature. However, according to our definition, such an ω-automaton is an infinite object, since we specify the set of accepting runs explicitly. This is useful here, since it allows us to state definitions and properties as general as possible. Later, we introduce (generalized) Büchi and parity automata as special cases and easily obtain some useful facts about them by applying the results about general ω-automata.

A (non-deterministic) ω-automaton[3] $\mathfrak{A} = (Q, \Sigma, Q_0, \Delta, \mathrm{Acpt})$ consists of a finite set Q of states, an alphabet Σ, a set $Q_0 \subseteq Q$ of initial states, a transition relation $\Delta \subseteq Q \times \Sigma \times Q$, and a set $\mathrm{Acpt} \subseteq Q^\omega$ of accepting runs. Again, $\mathfrak{A}$ is deterministic, if $|Q_0| = 1$ and if for every $q \in Q$ and every $a \in \Sigma$ there is a unique $q' \in Q$ such that $(q, a, q') \in \Delta$. In this case, we denote $Q_0 = \{q_0\}$ by q_0 and Δ as function $\delta\colon Q \times \Sigma \to Q$. The size of $\mathfrak{A}$, denoted by $|\mathfrak{A}|$, is the number $|Q|$ of states of $\mathfrak{A}$.

A run of $\mathfrak{A}$ on an ω-word $w \in \Sigma^\omega$ is an infinite sequence ζ of states satisfying $\zeta_0 \in Q_0$ and $(\zeta_n, w_n, \zeta_{n+1}) \in \Delta$ for every $n \in \mathbb{N}$; it is accepting if $\zeta \in \mathrm{Acpt}$. A deterministic ω-automaton has a unique run on every w. An ω-word w is accepted by $\mathfrak{A}$ if there exists an accepting run ζ of $\mathfrak{A}$ on w. The language recognized by $\mathfrak{A}$ is defined as $L(\mathfrak{A}) = \{w \in \Sigma^\omega \mid \mathfrak{A} \text{ accepts } w\}$.

An ω-automaton $\mathfrak{A} = (Q, \Sigma, Q_0, \Delta, \mathrm{Acpt})$ is strong[4] if we have for all runs ζ, ζ' of $\mathfrak{A}$: if $\mathrm{Inf}(\zeta) = \mathrm{Inf}(\zeta')$, then $\zeta \in \mathrm{Acpt}$ if and only if $\zeta' \in \mathrm{Acpt}$, i.e., acceptance depends only on the infinity set of a run. A state q of an ω-automaton is productive, if there is at least one accepting run ζ with $q \in \mathrm{Occ}(\zeta)$. A state that is not productive is unproductive.

Remark 2.1. Let $\mathfrak{A} = (Q, \Sigma, Q_0, \Delta, \mathrm{Acpt})$ be an ω-automaton, let P be its set of productive states, and let $\mathfrak{A}' = (P, \Sigma, Q_0 \cap P, \Delta \cap P \times \Sigma \times P, \mathrm{Acpt} \cap P^\omega)$ be the automaton obtained from $\mathfrak{A}$ by removing all unproductive states. Then, $L(\mathfrak{A}) = L(\mathfrak{A}')$.

[3] For notational convenience, we use the same notation for automata on finite and infinite words. It is always clear from the context which type of automaton is meant.

[4] As opposed to weak automata, where acceptance depends only on the occurrence set of a run.

An ω-automaton is unambiguous if every ω-word has at most one accepting run. Furthermore, it is non-confluent if it satisfies the following property for every ω-word w, every pair of runs ζ and ζ' of $\mathfrak{A}$ on w, and every n: if $\zeta_n = \zeta'_n$, then $\zeta_m = \zeta'_m$ for every $m < n$. Intuitively, if there is a nondeterministic choice that splits a run prefix into two, then these runs can never join at a later position. A useful property of a non-confluent automaton is that it has at most $|Q|$ different run prefixes on a prefix $w \in \Sigma^*$ of an ω-word, each of which can be identified uniquely by the last state of the run.

Lemma 2.2. *Let $\mathfrak{A}$ be a strong automaton without unproductive states. If $\mathfrak{A}$ is unambiguous, then it is non-confluent.*

Proof. Towards a contradiction, assume that $\mathfrak{A}$ is unambiguous but not non-confluent. Then, there exists an ω-word w and runs $q_0q_1q_2\cdots$ and $q'_0q'_1q'_2\cdots$ of $\mathfrak{A}$ on w such that $q_m \neq q'_m$ and $q_n = q'_n$ for indices $m < n$. As q_n is productive, there exists an accepting run ζ on some ω-word x such that $q_n \in \mathrm{Occ}(\zeta)$. We denote the suffix of ζ that begins with the first occurrence of q_n by ζ', and we denote the suffix of x that is read during ζ' by x'. Consider the ω-word $w_0 \cdots w_{n-1}x'$: it has two distinct runs $q_0 \cdots q_{n-1}\zeta'$ and $q'_0 \cdots q'_{n-1}\zeta'$ which are both accepting, since they have the same infinity set as ζ. This yields the desired contradiction, since $\mathfrak{A}$ is unambiguous. □

In the remainder of this work, we consider the following types of ω-automata, each of which defines the set Acpt of accepting runs implicitly using a finite description. A Büchi automaton $(Q, \Sigma, Q_0, \Delta, F)$ has a set $F \subseteq Q$ of accepting states. A run ζ is accepting, if a state from F occurs infinitely often in ζ, i.e.,

$$\mathrm{Acpt} = \{\zeta \in Q^\omega \mid \mathrm{Inf}(\zeta) \cap F \neq \emptyset\} \ .$$

A generalized Büchi automaton $(Q, \Sigma, Q_0, \Delta, \mathcal{F})$ has a family $\mathcal{F} \subseteq 2^Q$ of sets of accepting states. A run ζ is accepting, if from every $F \in \mathcal{F}$ a state occurs infinitely often in ζ, i.e.,

$$\mathrm{Acpt} = \{\zeta \in Q^\omega \mid \mathrm{Inf}(\zeta) \cap F \neq \emptyset \text{ for every } F \in \mathcal{F}\} \ .$$

A parity automaton $(Q, \Sigma, Q_0, \Delta, \Omega)$ has a priority function $\Omega\colon Q \to [k]$ for some $k \in \mathbb{N}$. The value $\Omega(q)$ is the priority of the state q. A run ζ is accepting, if the maximal priority that occurs infinitely often in ζ is even, i.e.,

$$\mathrm{Acpt} = \{\zeta \in Q^\omega \mid \max(\Omega(\mathrm{Inf}(\zeta))) \text{ is even}\} \ .$$

By definition, Büchi, generalized Büchi, and parity automata are strong. Hence, we obtain a simple corollary of Lemma 2.2.

Corollary 2.3. *Let $\mathfrak{A}$ be a Büchi, generalized Büchi, or parity automaton without unproductive states. If $\mathfrak{A}$ is unambiguous, then it is non-confluent.*

For all three types of ω-automata a simple reachability analysis suffices to determine the set of productive states. Hence, we can always assume our automata to only have productive states.

2.3 Infinite Games

In this work, we consider turn-based, two-player, zero-sum games of perfect information and infinite duration played on finite graphs. To this end, a graph is equipped with a partition of its set of vertices into the positions of Player 0 and the positions of Player 1. To start a play, a token is placed at some vertex and the players build an infinite path through the graph according to the following rule: whenever the token is at a position of Player i, she moves the token along an outgoing edge to an adjacent vertex. The winning condition of the game determines the winner of each infinite path.

For pronominal convenience [McN00], we assume Player 0 and the indetermined Player $i \in \{0, 1\}$ to be female, and we assume Player 1 and Player $1 - i \in \{0, 1\}$ to be male.

2.3.1 Arenas and Games

We begin by introducing our notation for arenas, the graphs in which the games are played. Then, we present several types of winning conditions that appear in this work.

An arena $\mathcal{A} = (V, V_0, V_1, E)$ consists of a finite directed graph (V, E), a set $V_0 \subseteq V$ of vertices of Player 0 (drawn as elliptical vertices), and the set $V_1 = V \setminus V_0$ of vertices of Player 1 (drawn as rectangular vertices). To avoid the nuisance of dealing with plays ending prematurely in a dead end, we assume every vertex to have at least one outgoing edge. The size $|\mathcal{A}|$ of $\mathcal{A}$ is the cardinality of V. A set $X \subseteq V$ induces the subarena

$$\mathcal{A}[X] = (V \cap X, V_0 \cap X, V_1 \cap X, E \cap (X \times X)) \ ,$$

if every vertex in X has at least one successor in X. An arena is solitary for Player i if $V_{1-i} = \emptyset$. A play in $\mathcal{A}$ starting in $v \in V$ is an infinite sequence ρ of vertices with $\rho_0 = v$ and $(\rho_n, \rho_{n+1}) \in E$ for all $n \in \mathbb{N}$. We illustrate these definitions in the following example which runs through the whole section.

Example 2.4. Figure 2.1 shows an arena $\mathcal{A}$. It is Player 0's turn at the vertices 0 and 2 while it is Player 1's turn at the vertices 1, 3, and 4. The set $\{0, 2\}$ induces a subarena which is solitary for Player 0. However, the set $\{1, 3, 4\}$ does not induce a subarena, since the vertex 4 has no successor in this set. The path $320(142)^\omega$ is a play in $\mathcal{A}$ starting in vertex 3. $\diamond$

After we have explained the interaction between the players we need to specify how the winner of a play is determined. A game $\mathcal{G} = (\mathcal{A}, \mathrm{Win})$

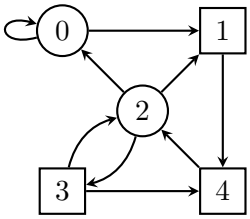

Figure 2.1: The arena $\mathcal{A}$ for the running example in Section 2.3

consists of an arena $\mathcal{A}$ with vertex set V and a set $\text{Win} \subseteq V^\omega$ of winning plays for Player 0. The set of winning plays for Player 1 is $V^\omega \setminus \text{Win}$.

At the moment, a game is an infinite object, thus it is not suited to serve as an input to an algorithm. In the remainder of this work, we deal with the following types of games, each defining the set of winning plays implicitly by some finite object. Let $\mathcal{G} = (\mathcal{A}, \text{Win})$ be a game with vertex set V. We say that $\mathcal{G}$ is a reachability game if there exists a set $F \subseteq V$ such that

$$\text{Win} = \{\rho \in V^\omega \mid \text{Occ}(\rho) \cap F \neq \emptyset\} \ ,$$

i.e., a play in a reachability game is winning for Player 0 if a vertex from F occurs. We say that $\mathcal{G}$ is a safety game if there exists a set $F \subseteq V$ such that

$$\text{Win} = \{\rho \in V^\omega \mid \text{Occ}(\rho) \subseteq F\} \ ,$$

i.e., a play in a safety game is winning for Player 0 if only vertices from F occur. We say that $\mathcal{G}$ is a Büchi game if there exists a set $F \subseteq V$ such that

$$\text{Win} = \{\rho \in V^\omega \mid \text{Inf}(\rho) \cap F \neq \emptyset\} \ ,$$

i.e., a play in a Büchi game is winning for Player 0 if a vertex from F occurs infinitely often. Finally, we say that $\mathcal{G}$ is a co-Büchi game if there exists a set $F \subseteq V$ such that

$$\text{Win} = \{\rho \in V^\omega \mid \text{Inf}(\rho) \subseteq F\} \ ,$$

i.e., a play in a co-Büchi game is winning for Player 0 if only vertices from F occur infinitely often. Reachability, safety, Büchi, and co-Büchi games are all denoted by $(\mathcal{A}, F)$.

Furthermore, we say that $\mathcal{G}$ is a parity game if there exists a priority function $\Omega \colon V \to [k]$ for some $k \in \mathbb{N}$ such that

$$\text{Win} = \{\rho \in V^\omega \mid \max(\Omega(\text{Inf}(\rho))) \text{ is even}\} \ .$$

The value $\Omega(v)$ is the priority of the vertex v. Player 0 wins a play if and only if the parity of the maximal priority that occurs infinitely often is 0. Thus, Player 1 wins a play ρ if $\max(\Omega(\text{Inf}(\rho)))$ is odd. Parity games are denoted by $(\mathcal{A}, \Omega)$.

A loop of $\mathcal{A}$ is a non-empty strongly connected set C of vertices, i.e., for all vertices $v, v' \in C$, there is a path from v to v' that only visits vertices in C. $\mathcal{G}$ is a Muller game if there exists a family $\mathcal{F}_0$ of loops of $\mathcal{A}$ such that

$$\text{Win} = \{\rho \in V^\omega \mid \text{Inf}(\rho) \in \mathcal{F}_0\} \ ,$$

i.e., a play is winning for Player 0 in $\mathcal{G}$ if its infinity set is in $\mathcal{F}_0$. Muller games are denoted by $\mathcal{G} = (\mathcal{A}, \mathcal{F}_0, \mathcal{F}_1)$ where $\mathcal{F}_1$ contains all loops of the arena that are not in $\mathcal{F}_0$. Since the infinity set of a play is always a loop of $\mathcal{A}$, a play ρ is winning for Player 1 if $\text{Inf}(\rho) \in \mathcal{F}_1$.

In the literature, there appear at least two other definitions of Muller games. The first one uses a coloring $\Omega\colon V \to [k]$ of the vertices and a partition $(\mathcal{F}_0, \mathcal{F}_1)$ of $2^{[k]}$. In this version, Player i wins a play ρ if and only if $\Omega(\text{Inf}(\rho)) \in \mathcal{F}_i$. The other definition uses a subset $W \subseteq V$ and a partition $(\mathcal{F}_0, \mathcal{F}_1)$ of 2^W. In this version, Player i wins a play ρ if and only if $(\text{Inf}(\rho) \cap W) \in \mathcal{F}_i$. It is easy to verify that if a game can be expressed as Muller game in one of the three definitions, then also in both other versions. For reasons that become apparent in Chapter 4, we prefer the first one.

According to our definition, every Büchi or co-Büchi game is a parity game (label the vertices in F by 2 and every other vertex by 1 for Büchi games, and label the vertices in F by 0 and every other vertex by 1 for co-Büchi games) and every parity game is a Muller game. Also, the representation of parity games introduced above is not necessarily unique, i.e., there is more than one way to label the vertices with priorities that defines a certain set Win. Our way of introducing these games, which might seem unnecessarily complicated at the moment, turns out to be useful in Section 5.1, where we classify the sets Win in the Borel hierarchy. For the time being, we do not distinguish between a game and its representation.

Reachability and safety games are duals of each other in the sense that Player 1's objective in a reachability game is to avoid F (a safety condition) and Player 1's objective in a safety game is to reach $V \setminus F$ (a reachability condition). The same holds true for Büchi and co-Büchi games: to win a play in a Büchi game, Player 1 has to visit F only finitely often (a co-Büchi condition) respectively to win in a co-Büchi game he has to visit $V \setminus F$ infinitely often (a Büchi condition). In this regard, parity and Muller games are self-dual as evident from the definitions above.

The size of a reachability, safety, Büchi, co-Büchi, or parity game is the number of its vertices. The situation for Muller games is more intricate since the winning condition $(\mathcal{F}_0, \mathcal{F}_1)$ can be encoded in various ways. The succinctness of these encodings influences the algorithmic properties of Muller games. We briefly discuss this at the end of Subsection 2.3.4.

We close this subsection by giving some example games in the arena $\mathcal{A}$ of Example 2.4.

Example 2.5. The play $421420(142)^\omega$ is winning for Player 0 in the reachability game $(\mathcal{A}, \{0\})$, since it visits the vertex 0 once. The play $(214)^\omega$ is winning for Player 1, since it never visits 0. Hence, it is winning for Player 0 in the safety game $(\mathcal{A}, \{1, 2, 3, 4\})$. ◇

Example 2.6. Consider the parity game $(\mathcal{A}, \Omega)$ with $\Omega(v) = v$ for each vertex v. The play $21420(142)^\omega$ is winning for Player 0, since the maximal priority that occurs infinitely often, which is 4, is even. The play $(32)^\omega$ is winning for Player 1 since the maximal priority that occurs infinitely often is odd. Hence, it is winning for Player 0 in the parity game $(\mathcal{A}, \Omega)$ with $\Omega'(v) = \Omega(v) + 1 = v + 1$. ◇

Example 2.7. Let $\mathcal{G} = (\mathcal{A}, \mathcal{F}_0, \mathcal{F}_1)$ be a Muller game with $\mathcal{F}_0 = \{\{1, 2, 3, 4\}\}$ and where $\mathcal{F}_1$ contains every other loop of $\mathcal{A}$. The play $000(142342)^\omega$ is winning for Player 0 since exactly the vertices 1, 2, 3, and 4 are visited infinitely often. The play $(32)^\omega$ is winning for Player 1 since its infinity set is in $\mathcal{F}_1$. Hence, it is winning for Player 0 in the Muller game $(\mathcal{A}, \mathcal{F}_1, \mathcal{F}_0)$. ◇

2.3.2 Strategies

What makes games interesting is their strategic nature: a move may depend on the history of the play, and has consequences for the future of the play. In an infinite game, a strategy is a mapping that determines the next move of Player i depending on the play prefix constructed so far. It is a winning strategy if every play that is played according to the strategy is won by Player i. Furthermore, if games are used to synthesize controllers for reactive systems, it is a winning strategy that is turned into a controller. For these reasons, the notion of strategy lies at the heart of game theory.

Fix an arena $\mathcal{A} = (V, V_0, V_1, E)$. A strategy for Player i is a mapping $\sigma\colon V^*V_i \to V$ satisfying $(\rho_n, \sigma(\rho_0 \cdots \rho_n)) \in E$ for every play prefix $\rho_0 \cdots \rho_n$, i.e., the next move the strategy prescribes is a legal one. The set of all strategies for Player i for $\mathcal{A}$ is denoted by $\Pi_i^{\mathcal{A}}$. If the arena we consider is clear from the context, we omit the superscript $\mathcal{A}$ and just write Π_i. We say that σ is positional if we have $\sigma(\rho_0 \cdots \rho_n) = \sigma(\rho_n)$ for every $\rho_0 \cdots \rho_n \in V^*V_i$. Since the value of such a strategy only depends on the last vertex of the play prefix, we also denote positional strategies as mappings $\sigma\colon V_i \to V$.

A play ρ is consistent with a strategy σ for Player i if $\rho_{n+1} = \sigma(\rho_0 \cdots \rho_n)$ for every n with $\rho_n \in V_i$. Sometimes, we also use the notion of consistency for play prefixes: such a prefix $\rho_0 \cdots \rho_m$ is consistent with σ if $\rho_{n+1} = \sigma(\rho_0 \cdots \rho_n)$ for every $n < m$ with $\rho_n \in V_i$. If we fix an initial vertex v and strategies $\sigma \in \Pi_i$ and $\tau \in \Pi_{1-i}$ for the players, there is a unique play $\rho(v, \sigma, \tau) = v_0 v_1 v_2 \cdots$ that is consistent with both strategies and starts in v, i.e., we have $v_0 = v$ and

$$v_{n+1} = \begin{cases} \sigma(v_0 \cdots v_n) & \text{if } v_n \in V_i, \\ \tau(v_0 \cdots v_n) & \text{if } v_n \in V_{1-i}. \end{cases}$$

Given a vertex v and a strategy σ for Player i in $\mathcal{A}$, we denote the set of plays of $\mathcal{A}$ starting in v that are consistent with σ by $\mathrm{Beh}_{\mathcal{A}}(v, \sigma)$; for σ and $W \subseteq V$ we define $\mathrm{Beh}_{\mathcal{A}}(W, \sigma) = \bigcup_{v \in W} \mathrm{Beh}_{\mathcal{A}}(v, \sigma)$. If $\mathcal{A}$ is clear from the context, we drop the subscript for the sake of readability and just write $\mathrm{Beh}(v, \sigma)$ and $\mathrm{Beh}(W, \sigma)$.

We illustrate these definitions by two example strategies for the arena $\mathcal{A}$ of Example 2.4.

Example 2.8. The strategy $\sigma_1 \in \Pi_0^{\mathcal{A}}$ defined by $\sigma_1(w0) = 0$ and $\sigma_1(w2) = 0$ is positional. We have $\mathrm{Beh}_{\mathcal{A}}(\{2, 3, 4\}, \sigma_1) = \{20^\omega, 320^\omega, 3420^\omega, 420^\omega\}$. ◇

Example 2.9. Let $\sigma_2 \in \Pi_0^{\mathcal{A}}$ be defined by

$$\sigma_2(w) = \begin{cases} 3 & \text{if } w \text{ ends with } 142, \\ 1 & \text{if } w \text{ ends with } 0,\ 2,\ 32,\ 42, \text{ or } 342. \end{cases}$$

The strategy alternates at vertex 2 between moving to 1 and moving to 3. The play $\rho = 014(23421423214)^\omega$ is consistent with σ_2. We have $\rho = \rho(0, \sigma_2, \tau)$ where τ is the strategy for Player 1 that alternates between moving to 2 and to 4 when at vertex 3, starting with moving to 4. Note that all other vertices in V_1 have only one successor. Hence, the strategy has no choice but to prescribe a move to this successor. Neither of these strategies is positional. ◇

Consider a game $\mathcal{G} = (\mathcal{A}, \mathrm{Win})$ with vertex set V. A strategy σ for Player i is a winning strategy for her from a set of vertices $W \subseteq V$ if every play starting in W that is consistent with σ is winning for Player i, i.e., $\mathrm{Beh}(W, \sigma) \subseteq \mathrm{Win}$ if $i = 0$, and $\mathrm{Beh}(W, \sigma) \subseteq V^\omega \setminus \mathrm{Win}$ if $i = 1$. We say that σ is a winning strategy for Player i from a vertex v if it is winning from $\{v\}$. The winning region $W_i(\mathcal{G})$ of Player i contains all vertices of $\mathcal{A}$ from which she has a winning strategy – but not necessarily the same from every vertex. However, if she has a winning strategy σ that is winning from $W_i(\mathcal{G})$, then we say that σ is a uniform winning strategy.

Remark 2.10. Let $\mathcal{G}$ be a game. Then, $W_0(\mathcal{G}) \cap W_1(\mathcal{G}) = \emptyset$.

On the other hand, we say that $\mathcal{G}$ is determined if $V = W_0(\mathcal{G}) \cup W_1(\mathcal{G})$, i.e., from each vertex one of the players has a winning strategy. Furthermore, we say that a game is determined with positional strategies if it is determined and both players have positional winning strategies from each vertex of their winning region. Solving a game amounts to determining the winning regions and corresponding winning strategies for both players.

Example 2.11. We show that the strategy σ_1 defined in Example 2.8 is a uniform winning strategy for Player 0 from every vertex of the reachability game $(\mathcal{A}, \{0\})$: if the play starts at any vertex but 0 and 2, then Player 1

has no choice but to move the token to vertex 2 in at most two steps. Then, from vertex 2, σ_1 prescribes to move the token to 0. Hence, every play that is consistent with σ_1 visits 0 and we have $W_0(\mathcal{A}, \{0\}) = \{0, 1, 2, 3, 4\}$. Thus, the game is positionally determined. ◇

Example 2.12. The strategy σ_2 defined in Example 2.9 is a uniform winning strategy for Player 0 from every vertex of the Muller game $\mathcal{G}$ defined in Example 2.7. Remember that it is Player 0's objective to enforce the infinity set $\{1, 2, 3, 4\}$. By always alternating between moving to 1 and 3, when the token is at vertex 2, she achieves her objective: from vertex 1 and 3, Player 1 has no choice but to move the token (directly or indirectly) to 2. Also, when the token is at vertex 1, he cannot avoid a visit to 4. Finally, Player 0 moves the token from 0 to 1 if the play starts there. Hence, every play that is consistent with σ visits 2 infinitely often, and therefore also 1 (and thus 4) and 3 infinitely often, but 0 at most once. Hence, the infinity set of such a play is $\{1, 2, 3, 4\}$. Hence, we have $W_0(\mathcal{G}) = \{0, 1, 2, 3, 4\}$ and conclude that $\mathcal{G}$ is determined. ◇

2.3.3 Finite-state Strategies and Game Reductions

As defined above, a strategy may take into account the whole play prefix constructed so far when it determines a successor to move the token to. Thus, a strategy is in general an infinite object. In contrast, a positional strategy is a finite object. However, positional strategies are often too weak to be winning, e.g., we have seen in Example 2.12 that σ_2 is a winning strategy for Player 0 in the Muller game $\mathcal{G}$ defined in Example 2.7, but it is easy to see that she has no positional winning strategy. But still, σ_2 does not need to know the whole play prefix, a finite abstraction suffices: the value of the strategy at vertex 2 depends only on whether vertex 1 or vertex 3 was seen last among the two. This idea is formalized by finite-state strategies which are introduced in the following. In the second part of this subsection, we define game reductions using the machinery of finite-state strategies. Such a reduction allows to solve a game with "complicated" winning condition by solving a game with a "simpler" winning condition in a larger arena.

A memory structure $\mathcal{M} = (M, \text{init}, \text{upd})$ for an arena (V, V_0, V_1, E) consists 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 update function can be extended to $\text{upd}^*\colon V^+ \to M$ by $\text{upd}^*(\rho_0) = \text{init}(\rho_0)$ and

$$\text{upd}^*(\rho_0 \dots \rho_n \rho_{n+1}) = \text{upd}(\text{upd}^*(\rho_0 \dots \rho_n), \rho_{n+1}) \ .$$

A next-move function for Player i is a function $\text{nxt}\colon V_i \times M \to V$ that satisfies $(v, \text{nxt}(v, m)) \in E$ for every $v \in V_i$ and every $m \in M$. It implements a strategy σ for Player i with memory $\mathcal{M}$ via

$$\sigma(\rho_0 \dots \rho_n) = \text{nxt}(\rho_n, \text{upd}^*(\rho_0 \dots \rho_n)) \ .$$

The size of $\mathcal{M}$ (and, slightly abusive, σ) is $|M|$. A strategy is called finite-state if it can be implemented with a memory structure and some next-move function.

Remark 2.13. A strategy σ is positional if and only if it can be implemented with a single memory state.

We say that a game is determined with finite-state strategies, if the game is determined and both players have a finite-state winning strategy from each vertex of their winning region.

Example 2.14. We show that the strategy σ_2 defined in Example 2.9 is finite-state: let $M = \{m_1, m_3\}$, $\mathrm{init}(v) = m_3$ for every v, and

$$\mathrm{upd}(m, v) = \begin{cases} m_3 & \text{if } v = 3, \\ m_1 & \text{if } v = 1, \\ m & \text{otherwise.} \end{cases}$$

The memory structure stores whether 1 or 3 was visited last between the two of them. Hence, we define the next-move function by

$$\mathrm{nxt}(v, m) = \begin{cases} 3 & \text{if } (v, m) = (2, m_1), \\ 1 & \text{if } (v, m) = (2, m_3), \\ 1 & \text{if } v = 0. \end{cases}$$

Simple calculations show that the strategy σ implemented by $(M, \mathrm{init}, \mathrm{upd})$ and nxt is indeed equal to σ_2. Thus, $\mathcal{G}_2$ is determined with finite-state strategies. ◇

Next, we introduce game reductions. An arena $\mathcal{A} = (V, V_0, V_1, E)$ and a memory structure $\mathcal{M} = (M, \mathrm{init}, \mathrm{upd})$ for $\mathcal{A}$ induce the extended arena

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

where $((s, m), (s', m')) \in E'$ if and only if $(s, s') \in E$ and $\mathrm{upd}(m, s') = m'$. For every $\rho \in V^\omega$, we define $\rho' \in (V \times M)^\omega$ by

$$\rho' = (\rho_0, m_0)(\rho_1, m_1)(\rho_2, m_2) \cdots$$

with $m_0 = \mathrm{init}(\rho_0)$ and $m_{n+1} = \mathrm{upd}(m_n, \rho_{n+1})$, i.e., $m_n = \mathrm{upd}^*(\rho_0 \cdots \rho_n)$. If ρ is a play in $\mathcal{A}$, then ρ' is a play in $\mathcal{A} \times \mathcal{M}$, which we refer to as the extended play of ρ.

A game $\mathcal{G} = (\mathcal{A}, \mathrm{Win})$ is reducible to a game $\mathcal{G}' = (\mathcal{A}', \mathrm{Win}')$ via $\mathcal{M}$, written $\mathcal{G} \leq_{\mathcal{M}} \mathcal{G}'$, if $\mathcal{A}' = \mathcal{A} \times \mathcal{M}$, and if $\rho \in \mathrm{Win}$ if and only if $\rho' \in \mathrm{Win}'$ for every $\rho \in V^\omega$. In this situation, a play ρ in $\mathcal{G}$ is won by the player who wins the extended play ρ' in $\mathcal{G}'$.

Lemma 2.15. *Let $\mathcal{M} = (M, \text{init}, \text{upd})$, let $\mathcal{G}$ be a game with vertex set V, and let $W \subseteq V$. If $\mathcal{G} \leq_{\mathcal{M}} \mathcal{G}'$ and if Player i has a positional winning strategy for $\mathcal{G}'$ from $\{(v, \text{init}(v)) \mid v \in W\}$, then she also has a finite-state winning strategy with memory $\mathcal{M}$ for $\mathcal{G}$ from W.*

Proof. Let $\sigma' \colon V_i \times M \to V \times M$ be a positional winning strategy for Player i for $\mathcal{G}'$ from $\{(v, \text{init}(v)) \mid v \in W\}$. We define a next-move function $\text{nxt} \colon V_i \times M \to V$ by $\text{nxt}(v, m) = v'$ if $\sigma'(v, m) = (v', m')$. We have $(v, v') \in E$ as required, since $((v, m), (v', m')) \in E'$ implies the former claim by definition of E'. We denote the strategy for $\mathcal{G}$ implemented by $\mathcal{M}$ and nxt by σ. It remains to show that σ is a winning strategy for Player i for $\mathcal{G}$ from W.

Consider a play ρ that starts in W and is consistent with σ. We show that its extended play $\rho' = (\rho_0, m_0)(\rho_1, m_1)(\rho_2, m_2) \cdots$ in $\mathcal{A} \times \mathcal{M}$ is consistent with σ'. This suffices, since ρ is winning for Player i in $\mathcal{G}$ if and only if ρ' is winning for Player i in $\mathcal{G}'$. The latter proposition is then true, since σ' is a winning strategy for Player i for $\mathcal{G}'$ from $(\rho_0, \text{init}(\rho_0)) = (\rho_0, m_0)$.

So, let $(\rho_n, m_n) \in V_i'$ and $\sigma'(\rho_n, m_n) = (v, m)$. We have to show $(v, m) = (\rho_{n+1}, m_{n+1})$. We have

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

By definition of ρ', we have $m_n = \text{upd}^*(\rho_0 \cdots \rho_n)$, i.e., $\rho_{n+1} = \text{nxt}(\rho_n, m_n)$. Since nxt is defined by σ', we obtain $v = \rho_{n+1}$. Finally, by definition of E', we obtain $m = \text{upd}(m_n, \rho_{n+1}) = m_{n+1}$. Thus, ρ' is consistent with σ'. □

Corollary 2.16. *Let $\mathcal{G}$ be a game that is reducible to a positionally determined game via a memory structure $\mathcal{M}$. Then, $\mathcal{G}$ is determined with finite-state strategies implemented by $\mathcal{M}$.*

We conclude this subsection by discussing the tight connection between memory structures that implement winning strategies and deterministic ω-automata that recognize the set of winning plays of Player 0. We state the following result for the parity condition, but similar results can be obtained for every other acceptance and winning conditions, respectively.

Lemma 2.17. *Let $\mathcal{G} = (\mathcal{A}, \text{Win})$ be a game and let $\mathfrak{A}$ be a deterministic parity automaton such that $L(\mathfrak{A}) = \text{Win}$. Then, $\mathcal{G}$ can be reduced to a parity game via a memory structure of size $|\mathfrak{A}|$.*

Proof. Let $\mathfrak{A} = (Q, V, q_0, \delta, \Omega)$ and define $\mathcal{M} = (M, \text{init}, \text{upd})$ by $M = Q$, $\text{init}(v) = \delta(q_0, v)$ and $\text{upd}(m, v) = \delta(m, v)$. We show $\mathcal{G} \leq_{\mathcal{M}} \mathcal{G}'$, where $\mathcal{G}' = (\mathcal{A} \times \mathcal{M}, \Omega')$ with $\Omega'(v, m) = \Omega(m)$. The condition on the arena of $\mathcal{G}'$ is met and we just have to show that $\rho \in \text{Win}$ if and only if the maximal priority that appears infinitely often in $\rho' = (\rho_0, m_0)(\rho_1, m_1)(\rho_2, m_2) \cdots$ is even. By construction, the sequence $m_0 m_1 m_2 \cdots$ is the unique run of $\mathfrak{A}$ on ρ and we have

$$\max(\Omega(\text{Inf}(m_0 m_1 m_2 \cdots))) = \max(\Omega'(\text{Inf}(\rho'))) \ .$$

Hence, $\rho \in \mathrm{Win} = L(\mathfrak{A})$ if and only if $\max(\Omega'(\mathrm{Inf}(\rho')))$ is even. □

A converse of Lemma 2.17 holds as well: if $(\mathcal{A}, \mathrm{Win}) \leq_{\mathcal{M}} \mathcal{G}'$, then $\mathcal{A} \times \mathcal{M}$ can be turned into a deterministic ω-automaton $\mathfrak{A}$ of the same size and with the same acceptance condition as the winning condition of $\mathcal{G}'$ such that the language of $\mathfrak{A}$ contains exactly the plays of $\mathcal{A}$ that are winning for Player 0.

2.3.4 Some Results about Infinite Games

We conclude this introductory chapter with some results about the games mentioned above.

We begin by defining attractors and attractor strategies, which are important concepts that are the basis of many solution algorithms for infinite games. Let $\mathcal{A} = (V, V_0, V_1, E)$ be an arena and let $X \subseteq V$ induce a subarena. The attractor for Player i of a set $F \subseteq V$ in X is $\mathrm{Attr}_i^X(F) = \bigcup_{n=0}^{|V|} A_n$, where $A_0 = F \cap X$ and

$$\begin{aligned} A_{n+1} = A_n \cup \{v \in V_i \cap X \mid \exists v' \in A_n \text{ such that } (v, v') \in E\} \\ \cup \{v \in V_{1-i} \cap X \mid \forall v' \in X \text{ with } (v, v') \in E : v' \in A_n\} \ . \end{aligned}$$

The attractor contains exactly the vertices of $\mathcal{A}[X]$ from which Player i can enforce a visit to F in the subarena induced by X. If $X = V$, we omit the superscript and write $\mathrm{Attr}_i(F)$ for short. Using backwards breadth-first search, attractors can be computed in linear time in the number of edges of the arena $\mathcal{A}[X]$ [NRY96].

A set $X \subseteq V$ is a trap for Player i in $\mathcal{A}$ if all outgoing edges of the vertices in $V_i \cap X$ lead to X and at least one successor of every vertex in $V_{1-i} \cap X$ is in X. By definition, every trap induces a subarena. Player $1-i$ can ensure that a play starting in X is confined to X by always picking a successor that is in X. The following lemma formalizes the properties of attractors and traps discussed above.

Lemma 2.18. *Let $\mathcal{A}$ be an arena with vertex set V, let $F \subseteq V$, and let $X \subseteq V$ induce a subarena of $\mathcal{A}$. We denote $\mathrm{Attr}_i^X(F)$ by A.*

i. Player i has a positional strategy $\sigma \in \Pi_i^{\mathcal{A}[X]}$ such that each play in $\mathrm{Beh}_{\mathcal{A}[X]}(A, \sigma)$ visits F.

ii. Player $1-i$ has a positional strategy $\tau \in \Pi_{1-i}^{\mathcal{A}[X]}$ such that each play in $\mathrm{Beh}_{\mathcal{A}[X]}(X \setminus A, \tau)$ never visits F. Furthermore, $X \setminus A$ induces a subarena of $\mathcal{A}[X]$ and is a trap for Player i in $\mathcal{A}[X]$.

Let us stress that these properties only hold in the subarena $\mathcal{A}[X]$, but not necessarily in the original arena $\mathcal{A}$ (see, e.g., Example 2.19).

A strategy for Player i as in i.) is called attractor strategy and proceeds by decreasing the distance to F in each move. Let $v \in A$: if $v \in F$ there is

nothing to show. Otherwise, let n be the iteration in which v is added to A, i.e., $v \in A_n \setminus A_{n-1}$. If $v \in V_i$, there is some successor at a smaller level $A_{n'}$ for some $n' < n$ to which the strategy moves the token. On the other hand, if $v \in V_{1-i}$, then all successors in X are at a smaller level. Hence, by always moving to a smaller level, Player i ensures a visit to F. Furthermore, this ensures that each vertex in A is visited at most once before a vertex in F is reached.

The strategy for Player $1-i$ as in ii.) is obtained by keeping the token in $X \setminus A$. Every vertex v of Player $1-i$ in $X \setminus A$ has a successor in $X \setminus A$ (if not, then v would have been added to the attractor) and all successors of a vertex v of Player i in $X \setminus A$ are in $X \setminus A$ or not even in X (again, if not, v would have been added to the attractor). Since, $F \cap X \subseteq A$, a play in the subarena induced by X that never leaves $X \setminus A$ never visits F.

Example 2.19. One last time consider the arena $\mathcal{A}$ defined in Example 2.4 and let $F = \{0\}$ and $X = \{0, 2, 4\}$ (note that X induces a subarena). Then, we have $\mathrm{Attr}_1^X(F) = X$. Player 1 can enforce a visit to $\{0\}$ in $\mathcal{A}[X]$ from every vertex by moving from 4 to 2, which has only one successor in $\mathcal{A}[X]$, namely 0. However, in the original arena $\mathcal{A}$, Player 1 cannot enforce a visit to 0 from 4, since Player 0 can always move from vertex 2 to 3 or to 1.

On the other hand, for $X = \{0, 2, 3\}$ (which again induces a subarena), we have that

$$V \setminus \mathrm{Attr}_1^X(\{0\}) = X \setminus \{0\} = \{2, 3\}$$

is a trap for Player 1 in the arena induced by X. However, $\{2, 3\}$ is not a trap in the original arena $\mathcal{A}$, since Player 1 can move the token from 3 to 4, thereby escaping the set $\{2, 3\}$. ◇

Applying Lemma 2.18 with $V = X$ solves reachability and safety games.

Theorem 2.20. *Reachability and safety games are determined with uniform positional strategies and can be solved in linear time (in the number of edges of the arena).*

Hence, the winning region of Player 0 in a safety game is a trap for Player 1 due to Remark 2.18(ii). For games in which the winner of a play only depends on the vertices visited infinitely often, both winning regions are traps. Since Büchi, co-Büchi, and parity conditions can be translated into Muller conditions, we only remark the following.

Remark 2.21. In every Muller game $\mathcal{G}$, $W_i(\mathcal{G})$ is a trap for Player $1-i$.

Now, let us turn our attention to solution algorithms for parity games.

Theorem 2.22. *Parity games are determined with uniform positional strategies [EJ91, Mos91] and can be solved in time $\mathcal{O}(m(n/d)^{\lceil d/2 \rceil})$, where n, m, and d are the number of vertices, edges, and priorities of the arena [Jur00].*

Solving parity games is in $\mathbf{UP} \cap \mathbf{Co\text{-}UP}$ [Jur98], and whether parity games can be solved in polynomial time is one of the most interesting open questions in the theory of infinite games due to its intriguing status in terms of computational complexity (see above), and due to the fact that model-checking the modal μ-calculus [Koz83] and solving parity games are polynomially interreducible [EJS01]. Depending on the number of priorities in proportion to the number of vertices, there are algorithms that solve parity games faster than the bound stated in Theorem 2.22 (e.g., see [Sch07, JPZ08] and the references therein). However, the result stated above suffices for our purposes.

Since Büchi and co-Büchi conditions can be expressed by parity conditions with only two priorities, we obtain positional determinacy of these games as corollary of Theorem 2.22.

Corollary 2.23. *Büchi and co-Büchi games are determined with uniform positional strategies and can be solved in polynomial time.*

Finally, we consider Muller games.

Theorem 2.24. *Muller games are determined with uniform finite-state strategies of size $n!$, where n is the number of vertices of the arena.*

This result can be shown by a game reduction to parity games using either latest appearance records [GH82] or by directly constructing a winning strategy of this size [DJW97]. Furthermore, there are matching lower bounds on the memory size needed to win a Muller game [DJW97].

The computational complexity of determining the winning regions of Muller games depends very much on the encoding of the winning condition $(\mathcal{F}_0, \mathcal{F}_1)$. The more succinctly it is encoded the harder it is to solve the game. The problem is in $\mathbf{P}$, if $\mathcal{F}_0$ (or $\mathcal{F}_1$) is given as explicit list of sets [Hor08]; it is in $\mathbf{NP} \cap \mathbf{Co\text{-}NP}$, if $(\mathcal{F}_0, \mathcal{F}_1)$ is encoded by a Zielonka-tree [DJW97] (see Subsection 4.2.1); and it is $\textsc{Pspace}$-complete if $(\mathcal{F}_0, \mathcal{F}_1)$ is encoded by one of the following five formalisms [HD05]: a boolean formula with variables V (also called Emerson-Lei games [EL85]), a boolean circuit over the input variables V, a Zielonka-DAG (a more compact representation of the Zielonka-tree), an explicit list of sets of colors or an explicit list of subsets of W, which itself is a subset of V. The latter two encodings refer to the alternative definitions of Muller games mentioned in Subsection 2.3.1.

Chapter 3

Synthesis from Parametric Linear Time Specifications

A crucial aspect of automated verification and synthesis is the choice of a specification formalism; a decision which is subject to several conflicting objectives. On the one hand, the formalism should be expressive enough to specify desirable properties of reactive systems, but at the same time simple enough to be employed by practitioners without formal training in automata theory or logics. Furthermore, the formalism should have *nice* algorithmic properties such as a feasible model-checking problem. In practice, Linear Temporal Logic (LTL) has emerged as a good compromise: it is expressively equivalent to first-order logic (with order-relation) over words [Kam68], its model-checking problem is **PSPACE**-complete [SC85], and it has a compact, variable-free syntax and intuitive semantics: for example, the specification "every request q is answered by a response p" is expressed by the formula

$$\mathbf{G}(q \rightarrow \mathbf{F}p) \ .$$

However, LTL lacks capabilities to express timing constraints, e.g., the request-response condition is satisfied even if the response time doubles with each request. Similarly, when synthesizing a controller for the request-response specification, we prefer an implementation that answers every request as soon as possible, but there is no guarantee that such an optimal controller is computed when solving a game with this winning condition.

The simplest way to enrich LTL with timing constraints is to add the operator $\mathbf{F}_{\leq k}$, where $k \in \mathbb{N}$ is an arbitrary, but fixed constant, with the expected semantics: the formula $\mathbf{F}_{\leq k}\varphi$ is satisfied, if φ holds at least once within the next k steps. Koymans [Koy90] and Alur and Henzinger [AH93] investigated generalizations of this approach in the form of logics with temporal operators bounded by constant intervals of natural numbers. This allows to infer some quantitative information about a system: the formula

$$\mathbf{G}(q \rightarrow \mathbf{F}_{\leq k}p)$$

is satisfied if every request is answered within k steps. But finding the right bound k is not practicable: it is generally not known beforehand and depends on the granularity of the model of the system. On the other hand, adding $\mathbf{F}_{\leq k}$ does not increase the expressiveness of LTL, as it can be expressed by a disjunction of nested next-operators.

To overcome these shortcomings, several parameterized temporal logics [AETP01, KPV09, GTN10] were introduced for the verification of closed systems: here, constant bounds on the temporal operators are replaced by parametric bounds. In this formalism, we can ask whether there *exists* a bound on the response time, as opposed to asking whether some fixed k is a bound. Furthermore, we can ask for optimal bounds.

We are mainly concerned with Parametric Linear Temporal Logic (PLTL) introduced by Alur et al. [AETP01], which adds the operators $\mathbf{F}_{\leq x}$ and $\mathbf{G}_{\leq y}$ to LTL. In PLTL, the request-response specification is expressed by

$$\mathbf{G}(q \rightarrow \mathbf{F}_{\leq x} p) \ ,$$

stating that every request is answered within the next x steps, where x is a variable. Hence, satisfaction of a formula is defined with respect to a variable valuation α mapping variables to natural numbers: $\mathbf{F}_{\leq x}\varphi$ holds, if φ is satisfied at least once within the next $\alpha(x)$ steps, while $\mathbf{G}_{\leq y}\varphi$ holds, if φ is satisfied for the next $\alpha(y)$ steps.

The model-checking problem for a parameterized temporal logic is typically no harder than the model-checking problem for the unparameterized fragment, e.g., Alur et al. showed that deciding whether a transition system satisfies a PLTL formula with respect to some, infinitely many, or all variable valuations is **Pspace**-complete [AETP01], as is LTL model-checking [SC85]. Furthermore, for two interesting fragments of PLTL and several notions of optimality, they showed that optimal variable valuations for which a formula is satisfied by a given transition system can be determined in polynomial space as well.

In this chapter, we consider infinite games with winning conditions in PLTL and lift the results on model-checking parameterized specifications to synthesis from parameterized specifications. We show that determining whether a player wins a PLTL game with respect to some, infinitely many, or all variable valuations is **2Exptime**-complete, as is determining the winner of an LTL game [PR89b]. Again, we observe the same phenomenon as in model-checking: the addition of parameterized operators does not increase the computational complexity of the decision problems. Afterwards, we give an algorithm with triply-exponential running time to compute optimal winning strategies in the two fragments of PLTL already considered by Alur et al. for model-checking. We complement this with doubly-exponential upper and lower bounds on values of optimal variable valuations for games in these fragments.

3.1 Parametric Linear Temporal Logics

In this section, we introduce two parameterized temporal logics we use to specify winning conditions for infinite games. The main focus is on Parametric Linear Temporal Logic (PLTL) as introduced above.

Our definition of the syntax of PLTL differs in several aspects from the original definition of Alur et al.: we add the release operator to have a dual operator for the until operator, but disregard several parameterized operators that were shown to be syntactic sugar. Similarly, we disallow constant bounds since they do not add expressiveness. Finally, we do not use two disjoint sets of variables – one for parameterized eventually operators and the other one for parameterized always operator – as it is done in the original definition. Instead, we use only one set and later restrict ourselves by defining a notion of well-formedness, which exactly simulates the use of two sets of variables. Each of these differences is discussed in detail below.

Additionally, we consider PROMPT–LTL, which can be seen as the fragment of PLTL in which only eventually operators may be parameterized, all by the same variable. For the sake of simplicity, we use this characterization to define PROMPT–LTL. We briefly explain the original definition below.

In Subsection 3.1.1 we introduce the syntax and in Subsection 3.1.2 the semantics of PLTL and define LTL and PROMPT–LTL as fragments of PLTL. Then, in Subsection 3.1.3 we discuss some properties of PLTL such as negation of formulae and the monotonicity of parameterized operators. Finally, in Subsection 3.1.4, we define games with winning conditions in PLTL and state some basic results.

3.1.1 Syntax of PLTL

In this subsection, we introduce the syntax of Parametric Linear Temporal Logic (PLTL). The logic is obtained by adding parameterized eventually operators $\mathbf{F}_{\leq z}$ and parameterized always operators $\mathbf{G}_{\leq z}$ to LTL in negation normal form.

Formally, let $\mathcal{V}$ be an infinite set of variables and let us fix a finite[5] set P of atomic propositions which we use to build our formulae and to label arenas in which we evaluate them. The formulae of PLTL are given by the grammar

$$\varphi ::= p \mid \neg p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \mathbf{X}\varphi \mid \varphi \,\mathbf{U}\, \varphi \mid \varphi \,\mathbf{R}\, \varphi \mid \mathbf{F}_{\leq z}\varphi \mid \mathbf{G}_{\leq z}\varphi \ ,$$

where $p \in P$ and $z \in \mathcal{V}$. We use the derived operators $\mathbf{tt} := p \vee \neg p$ and $\mathbf{ff} := p \wedge \neg p$ for some fixed $p \in P$, $\mathbf{F}\varphi := \mathbf{tt}\,\mathbf{U}\,\varphi$, and $\mathbf{G}\varphi := \mathbf{ff}\,\mathbf{R}\,\varphi$. Furthermore, we use $\varphi \rightarrow \psi$ as shorthand for $\neg\varphi \vee \psi$, where we have to require the antecedent φ to be a (negated) atomic proposition and identify $\neg\neg p$ with

[5] We require P to be finite so that its power set is an alphabet. This greatly simplifies our notation and exposition when we translate formulae into automata, but is not essential.

p. We assume negation to bind stronger than every other connective and operator, which allows us to omit some brackets.

In the original work on PLTL [AETP01], the operators $\mathbf{U}_{\leq x}$, $\mathbf{F}_{>y}$, $\mathbf{G}_{>x}$, and $\mathbf{U}_{>y}$ as well as constant bounds of the form $\mathbf{F}_{\leq k}$, $\mathbf{G}_{\leq k}$, etc. for $k \in \mathbb{N}$ are also allowed. However, since these operators do not add expressiveness (see Lemma 3.2 and Lemma 3.8) we treat them as derived operators instead of adding them as primitive operators.

The set of subformulae of a PLTL formula φ, denoted by $\mathrm{cl}(\varphi)$, is defined inductively by

- $\mathrm{cl}(p) = \{p\}$ and $\mathrm{cl}(\neg p) = \{\neg p\}$,
- $\mathrm{cl}(\varphi \wedge \psi) = \{\varphi \wedge \psi\} \cup \mathrm{cl}(\varphi) \cup \mathrm{cl}(\psi)$,
- $\mathrm{cl}(\varphi \vee \psi) = \{\varphi \vee \psi\} \cup \mathrm{cl}(\varphi) \cup \mathrm{cl}(\psi)$,
- $\mathrm{cl}(\mathbf{X}\varphi) = \{\mathbf{X}\varphi\} \cup \mathrm{cl}(\varphi)$,
- $\mathrm{cl}(\varphi\, \mathbf{U}\, \psi) = \{\varphi\, \mathbf{U}\, \psi\} \cup \mathrm{cl}(\varphi) \cup \mathrm{cl}(\psi)$,
- $\mathrm{cl}(\varphi\, \mathbf{R}\, \psi) = \{\varphi\, \mathbf{R}\, \psi\} \cup \mathrm{cl}(\varphi) \cup \mathrm{cl}(\psi)$,
- $\mathrm{cl}(\mathbf{F}_{\leq z}\varphi) = \{\mathbf{F}_{\leq z}\varphi\} \cup \mathrm{cl}(\varphi)$, and
- $\mathrm{cl}(\mathbf{G}_{\leq z}\varphi) = \{\mathbf{G}_{\leq z}\varphi\} \cup \mathrm{cl}(\varphi)$.

We define the size of a PLTL formula φ, denoted by $|\varphi|$, to be the cardinality of $\mathrm{cl}(\varphi)$. Furthermore, we define

$$\mathrm{var}_{\mathbf{F}}(\varphi) = \{z \in \mathcal{V} \mid \mathbf{F}_{\leq z}\psi \in \mathrm{cl}(\varphi)\}$$

to be the set of variables parameterizing eventually operators in φ,

$$\mathrm{var}_{\mathbf{G}}(\varphi) = \{z \in \mathcal{V} \mid \mathbf{G}_{\leq z}\psi \in \mathrm{cl}(\varphi)\}$$

to be the set of variables parameterizing always operators in φ, and we define

$$\mathrm{var}(\varphi) = \mathrm{var}_{\mathbf{F}}(\varphi) \cup \mathrm{var}_{\mathbf{G}}(\varphi)$$

to be the set of variables occurring in φ. From now on, we denote variables in $\mathrm{var}_{\mathbf{F}}(\varphi)$ by x and variables in $\mathrm{var}_{\mathbf{G}}(\varphi)$ by y. A formula φ is variable-free, if $\mathrm{var}(\varphi) = \emptyset$.

3.1.2 Semantics of PLTL

In this subsection, we present the semantics of PLTL, explain the differences to the original definition of PLTL, and introduce LTL and PROMPT–LTL as fragments of PLTL.

In order to evaluate PLTL formulae, we need to specify the values of the variables appearing as bounds in the parameterized operators. To this end, we define a variable valuation to be a mapping $\alpha\colon \mathcal{V} \to \mathbb{N}$. Now, we can define the model relation between an ω-word $w \in \left(2^P\right)^\omega$, a position $n \in \mathbb{N}$ of w, a variable valuation $\alpha\colon \mathcal{V} \to \mathbb{N}$, and a PLTL formula as follows:

- $(w, n, \alpha) \models p$ if and only if $p \in w_n$,
- $(w, n, \alpha) \models \neg p$ if and only if $p \notin w_n$,
- $(w, n, \alpha) \models \varphi \wedge \psi$ if and only if $(w, n, \alpha) \models \varphi$ and $(w, n, \alpha) \models \psi$,
- $(w, n, \alpha) \models \varphi \vee \psi$ if and only if $(w, n, \alpha) \models \varphi$ or $(w, n, \alpha) \models \psi$,
- $(w, n, \alpha) \models \mathbf{X}\varphi$ if and only if $(w, n+1, \alpha) \models \varphi$,
- $(w, n, \alpha) \models \varphi\, \mathbf{U}\, \psi$ if and only if there exists a $k \geq 0$ such that $(w, n+k, \alpha) \models \psi$ and $(w, n+j, \alpha) \models \varphi$ for every j in the range $0 \leq j < k$,
- $(w, n, \alpha) \models \varphi \mathbf{R} \psi$ if and only if for every $k \geq 0$: either $(w, n+k, \alpha) \models \psi$ or there exists a j in the range $0 \leq j < k$ such that $(w, n+j, \alpha) \models \varphi$,
- $(w, n, \alpha) \models \mathbf{F}_{\leq x}\varphi$ if and only if there exists a j in the range $0 \leq j \leq \alpha(x)$ such that $(w, n+j, \alpha) \models \varphi$, and
- $(w, n, \alpha) \models \mathbf{G}_{\leq y}\varphi$ if and only if for every j in the range $0 \leq j \leq \alpha(y)$: $(w, n+j, \alpha) \models \varphi$.

For the sake of brevity, we write $(w, \alpha) \models \varphi$ instead of $(w, 0, \alpha) \models \varphi$ and say that w is a model of φ with respect to α. Furthermore, as the satisfaction of a variable-free formula φ is independent of the variable valuation α, we omit α and write $(w, n) \models \varphi$ instead of $(w, n, \alpha) \models \varphi$ and accordingly $w \models \varphi$ instead of $(w, 0) \models \varphi$.

Example 3.1. Let $w_1 = (\{q\}\emptyset\emptyset\emptyset\{p\})^\omega$ and

$$w_2 = \prod_{j \geq 0} (\{q\}\emptyset^j\{p\}) = \{q\}\{p\}\{q\}\emptyset\{p\}\{q\}\emptyset\emptyset\{p\}\{q\}\emptyset\emptyset\emptyset\{p\}\cdots \ .$$

The formula $\varphi_1 = \mathbf{GF}p$ is satisfied if p holds true infinitely often. Hence, we have $w_1 \models \varphi_1$ and $w_2 \models \varphi_1$. The formula $\varphi_2 = \mathbf{G}(q \rightarrow \mathbf{F}p)$ requires every position at which q holds to be followed by a position at which p holds. We have $w_1 \models \varphi_2$ and $w_2 \models \varphi_2$. Finally, consider the formula $\varphi_3 = \mathbf{G}(q \rightarrow \mathbf{F}_{\leq x}p)$ obtained by parameterizing the eventually operator in φ_2.

It is satisfied with respect to a variable valuation α if for every position at which q holds one of the next $\alpha(x)+1$ positions (including the current one) satisfies p. Hence, we have $(w_1, \alpha) \models \varphi_3$ provided $\alpha(x) \geq 4$, but we have $(w_2, \alpha) \not\models \varphi_3$ for every α. $\diamond$

As mentioned above, the original work on PLTL also introduced the operators $\mathbf{U}_{\leq x}$, $\mathbf{F}_{>y}$, $\mathbf{G}_{>x}$, and $\mathbf{U}_{>y}$ with the following semantics.

- $(w, n, \alpha) \models \varphi \, \mathbf{U}_{\leq x} \, \psi$ if and only if there exists a k in the range $0 \leq k \leq \alpha(x)$ such that $(w, n+k, \alpha) \models \psi$ and $(w, n+j, \alpha) \models \varphi$ for every j in the range $0 \leq j < k$,
- $(w, n, \alpha) \models \mathbf{F}_{>y}\varphi$ if and only if there exists a $j > \alpha(y)$ such that $(w, n+j, \alpha) \models \varphi$,
- $(w, n, \alpha) \models \mathbf{G}_{>x}\varphi$ if and only if for every $j > \alpha(x)$: $(w, n+j, \alpha) \models \varphi$, and
- $(w, n, \alpha) \models \varphi \, \mathbf{U}_{>y} \, \psi$ if and only if there exists a $k > \alpha(y)$ such that $(w, n+k, \alpha) \models \psi$ and $(w, n+j, \alpha) \models \varphi$ for every j in the range $0 \leq j < k$.

We ignore these parameterized operators, since they can be expressed using $\mathbf{F}_{\leq x}$ and $\mathbf{G}_{\leq y}$, at the cost of a linear increase of the formula's size.

Lemma 3.2 ([AETP01])**.** *The following equivalences hold.*

i. $(w, n, \alpha) \models \varphi \, \mathbf{U}_{\leq x} \, \psi$ *if and only if* $(w, n, \alpha) \models (\varphi \, \mathbf{U} \, \psi) \wedge \mathbf{F}_{\leq x}\psi$.

ii. $(w, n, \alpha) \models \mathbf{F}_{>y}\varphi$ *if and only if* $(w, n, \alpha) \models \mathbf{G}_{\leq y}\mathbf{F}\mathbf{X}\varphi$.

iii. $(w, n, \alpha) \models \mathbf{G}_{>x}\varphi$ *if and only if* $(w, n, \alpha) \models \mathbf{F}_{\leq x}\mathbf{G}\mathbf{X}\varphi$.

iv. $(w, n, \alpha) \models \varphi \, \mathbf{U}_{>y} \, \psi$ *if and only if* $(w, n, \alpha) \models \mathbf{G}_{\leq y} \, (\varphi \wedge \mathbf{X} \, (\varphi \, \mathbf{U} \, \psi))$.

Furthermore, the original definition of PLTL included constant bounds of the form $\mathbf{F}_{\leq k}$, $\mathbf{G}_{\leq k}$, etc. for $k \in \mathbb{N}$ with the expected semantics, although the main reason to introduce parameterized operators was the inadequacy of the constant bounds. Thus, we ignore them too, as they do not add expressiveness (see the proof of Lemma 3.8). However, if the bounds are encoded in binary, the logic with constant bounds is exponentially more succinct than PLTL as defined here. Nevertheless, it turns out that this gap in succinctness does not affect the complexity of solving games with winning conditions in these logics. This can be shown by combining the alternating color technique presented in Subsection 3.2.1 and the construction of parity automata for PLTL formulae presented in Subsection 3.3.2.

As usual for parameterized temporal logics, the use of variables in parameterized operators has to be restricted to avoid undecidability of the

satisfiability problem. Our definition of a PLTL-formula is too general as it allows to parameterize an eventually operator and an always operator by the same variable: consider the formula

$$\psi = \mathbf{X}\mathbf{F}_{\leq z} p \wedge \mathbf{G}_{\leq z} \neg p \ ,$$

which expresses that the proposition p holds within the next $\alpha(z) + 1$ positions, but not in the next $\alpha(z)$ positions, i.e., p holds for the first time exactly $\alpha(z) + 1$ positions after the current one. This allows to divide an infinite word into infinitely many blocks of the same length using the formula $p \wedge \mathbf{G}(p \rightarrow \mathbf{X}\psi)$. These blocks allow to encode the terminating run of a two-counter machine [SS63]. To this end, the correct updates of the state of the machine can be enforced by an LTL formula while the correct updates of the counters are enforced using the parameterized operators. Hence, for a given two-counter machine we can construct a PLTL formula φ that has a model w with respect to some variable valuation α if and only if the machine terminates when started in the initial configuration. Here, $\alpha(z)$ has to be the maximal value one of the counters assumes during the computation of the machine. Hence, the question whether a PLTL formula (in the current definition) has a model with respect to some variable valuation is undecidable. A detailed proof[6] can be found in [AETP01]. We just note the following consequence that we use below. It is obtained by replacing the variable parameterizing the eventually operator in ψ by x and replacing the variable parameterizing the always operator by y.

Corollary 3.3 ([AETP01])**.** *The following problem is undecidable: given a* PLTL *formula* φ *with* $\mathrm{var}_{\mathbf{F}}(\varphi) = \{x\}$ *and* $\mathrm{var}_{\mathbf{G}}(\varphi) = \{y\}$*, is there a* $w \in \left(2^P\right)^\omega$ *and a variable valuation* α *such that* $\alpha(x) = \alpha(y)$ *and* $(w, \alpha) \models \varphi$*?*

For the reason just discussed, the original definition of PLTL uses two disjoint sets of variables: one to parameterize eventually operators and the other one to parameterize always operators. Alur et al. showed that satisfiability and validity of PLTL as well as PLTL model-checking are decidable when separating the variables for eventually and always operators. However, Corollary 3.3 is still valid in this setting.

Unlike in the original definition of PLTL, we choose to use a single set of variables for reasons that become apparent in Subsection 3.1.3. Hence, we have to restrict ourselves to formulae that do not use a variable to parameterize both an eventually and an always operator to simulate the original definition.

Definition 3.4. A PLTL formula φ is well-formed, if $\mathrm{var}_{\mathbf{F}}(\varphi) \cap \mathrm{var}_{\mathbf{G}}(\varphi) = \emptyset$.

[6] This proof uses the operator $\mathbf{U}_{=x}$ with the expected semantics. However, it can easily be adapted to use $\mathbf{F}_{\leq z}$ and $\mathbf{G}_{\leq z}$ as done here.

Several linear temporal logics are syntactic fragments of PLTL. First of all, linear temporal logic (LTL) is the fragment containing the formulae without parameterized operators. Second, PROMPT–LTL of Kupferman et al. [KPV09] can be seen as the fragment containing the formulae without parameterized always operators and a single variable for the parameterized eventually operators. Furthermore, we consider two fragments of PLTL introduced in [AETP01]: the fragment of formulae without parameterized always operators and the fragment of formulae without parameterized eventually operators.

Let φ be a PLTL formula.

- φ is an LTL formula, if $\mathrm{var}(\varphi) = \emptyset$.
- φ is a PROMPT–LTL formula, if $\mathrm{var}_{\mathbf{G}}(\varphi) = \emptyset$ and $|\mathrm{var}_{\mathbf{F}}(\varphi)| \leq 1$, i.e., φ contains no parameterized always operator and all parameterized eventually operators have the same variable.
- φ is a $\mathrm{PLTL}_{\mathbf{F}}$ formula, if $\mathrm{var}_{\mathbf{G}}(\varphi) = \emptyset$, i.e., φ contains no parameterized always operators.
- φ is a $\mathrm{PLTL}_{\mathbf{G}}$ formula, if $\mathrm{var}_{\mathbf{F}}(\varphi) = \emptyset$, i.e., φ contains no parameterized eventually operators.
- φ is a unipolar formula, if it is either a $\mathrm{PLTL}_{\mathbf{F}}$ or a $\mathrm{PLTL}_{\mathbf{G}}$ formula.

Since a PROMPT–LTL formula contains at most one variable x, there is no need to name it explicitly and a variable valuation α can be replaced by the value $k = \alpha(x)$. Hence, the original definition of PROMPT–LTL [KPV09] uses the unary operator $\mathbf{F}_{\mathbf{P}}$ (called prompt-eventually) instead of $\mathbf{F}_{\leq x}$ and the semantics is defined with respect to a non-negative integer k:

- $(w, n, k) \models \mathbf{F}_{\mathbf{P}}\varphi$ if and only if there is a j in the range $0 \leq j \leq k$ such that $(w, n + j, k) \models \varphi$.

The semantics of all other connectives and operators is independent of k and defined as for PLTL. Our definition of PROMPT–LTL in terms of $\mathbf{F}_{\leq x}$ as syntactic fragment is equivalent to the original definition.

By definition, every LTL formula is a PROMPT–LTL formula and a $\mathrm{PLTL}_{\mathbf{G}}$ formula, and every PROMPT–LTL formula is a $\mathrm{PLTL}_{\mathbf{F}}$ formula. Another simple consequence of the definitions of these fragments is that they only contain well-formed formulae. Since the unipolar formulae subsume all other fragments we just remark the following.

Remark 3.5. Every unipolar formula is well-formed.

3.1.3 Properties of PLTL

In this subsection we present some basic properties of PLTL that we utilize throughout this chapter.

We begin by discussing the role of negation: remember that formulae of PLTL are defined in negation normal form. This restriction is closely related to the need for separating variables parameterizing eventually operators and variables parameterizing always operators. A negation flips the polarity of an operator: $(w, n, \alpha) \models \neg \mathbf{F}_{\leq x} \varphi$ is equivalent to $(w, n, \alpha) \models \mathbf{G}_{\leq x} \neg \varphi$ and $(w, n, \alpha) \models \neg \mathbf{G}_{\leq y} \varphi$ is equivalent to $(w, n, \alpha) \models \mathbf{F}_{\leq y} \neg \varphi$. Hence, allowing arbitrary negation would complicate the notion of well-formed formulae. However, we have just seen how to push a negation over a parameterized temporal operator: $\mathbf{F}_{\leq z}$ and $\mathbf{G}_{\leq z}$ are dual. Hence, it is possible to define the negation of a formula by pushing the negation to the atomic propositions. Here, it is crucial that we add the release-operator to have a dual to the until-operator.

Lemma 3.6. *For every* PLTL *formula* φ *there exists an* PLTL *formula* φ' *with* $(w, n, \alpha) \models \varphi$ *if and only if* $(w, n, \alpha) \not\models \varphi'$. *Furthermore,* φ' *is constructible in linear time (in* $|\varphi|$*).*

Proof. The proof relies on the duality of the pairs $(p, \neg p)$, $(\wedge, \vee)$, $(\mathbf{X}, \mathbf{X})$ $(\mathbf{U}, \mathbf{R})$, and $(\mathbf{F}_{\leq z}, \mathbf{G}_{\leq z})$. We define φ' inductively by

- $p' = \neg p$ and $(\neg p)' = p$,
- $(\varphi \wedge \psi)' = \varphi' \vee \psi'$,
- $(\varphi \vee \psi)' = \varphi' \wedge \psi'$,
- $(\mathbf{X}\varphi)' = \mathbf{X}\varphi'$,
- $(\varphi \,\mathbf{U}\, \psi)' = \varphi' \,\mathbf{R}\, \psi'$,
- $(\varphi \,\mathbf{R}\, \psi)' = \varphi' \,\mathbf{U}\, \psi'$,
- $(\mathbf{F}_{\leq x} \varphi)' = \mathbf{G}_{\leq x} \varphi'$, and
- $(\mathbf{G}_{\leq y} \varphi)' = \mathbf{F}_{\leq y} \varphi'$.

Using these rewriting rules, φ' can be constructed in linear time in the size of φ. It remains to prove that we have $(w, n, \alpha) \models \varphi$ if and only if $(w, n, \alpha) \not\models \varphi'$. This is done by structural induction over the construction of φ. The cases of atomic formulae, boolean connectives, and unparameterized temporal operators are straightforward applications of the induction hypothesis and duality of these connectives and operators. Thus, we only

consider the case of parameterized temporal operators. We have

$$
\begin{aligned}
& (w, n, \alpha) \models \mathbf{F}_{\leq x}\varphi \\
\overset{\text{Def.}}{\iff} & \text{there is a } j \text{ in the range } 0 \leq j \leq \alpha(x) \text{ s. t. } (w, n+j, \alpha) \models \varphi \\
\overset{\text{I.H.}}{\iff} & \text{there is a } j \text{ in the range } 0 \leq j \leq \alpha(x) \text{ s. t. } (w, n+j, \alpha) \not\models \varphi' \\
\iff & \text{not for every } j \text{ in the range } 0 \leq j \leq \alpha(x)\text{: } (w, n+j, \alpha) \models \varphi' \\
\overset{\text{Def.}}{\iff} & (w, n, \alpha) \not\models \mathbf{G}_{\leq x}\varphi' \\
\overset{\text{Def.}}{\iff} & (w, n, \alpha) \not\models (\mathbf{F}_{\leq x}\varphi)'
\end{aligned}
$$

and dually

$$
\begin{aligned}
& (w, n, \alpha) \models \mathbf{G}_{\leq y}\varphi \\
\overset{\text{Def.}}{\iff} & \text{for every } j \text{ in the range } 0 \leq j \leq \alpha(y)\text{: } (w, n+j, \alpha) \models \varphi \\
\overset{\text{I.H.}}{\iff} & \text{for every } j \text{ in the range } 0 \leq j \leq \alpha(y)\text{: } (w, n+j, \alpha) \not\models \varphi' \\
\iff & \text{there is no } j \text{ in the range } 0 \leq j \leq \alpha(y) \text{ s. t. } (w, n+j, \alpha) \models \varphi' \\
\overset{\text{Def.}}{\iff} & (w, n, \alpha) \not\models \mathbf{F}_{\leq y}\varphi' \\
\overset{\text{Def.}}{\iff} & (w, n, \alpha) \not\models (\mathbf{G}_{\leq y}\varphi)' \ . \qquad \square
\end{aligned}
$$

Here it becomes apparent why we prefer to use a single set of variables and consider only well-formed formulae instead of using two disjoint sets of variables, one for parameterized eventually operators and one for parameterized always operators: the definition of well-formedness is independent of variable names, which allows us to leave the variable names unchanged when pushing negations over parameterized operators. If we would use two sets, we would need to use a name from the other set of variables when pushing a negation over a parameterized operator. This would require notational overhead when working with the negation of a formula.

In the following, we freely use $\neg\varphi$ as shorthand for the formula φ' as in Lemma 3.6. A formula φ and its "negation" φ' satisfy the following useful properties: most importantly, the negation of a well-formed formula is well-formed as well.

Remark 3.7. Let φ be a PLTL formula and let $\neg\varphi$ be its negation as defined in the proof of Lemma 3.6.

i. $\neg\neg\varphi = \varphi$, i.e., $(w, n, \alpha) \models \varphi$ if and only if $(w, n, \alpha) \models \neg\neg\varphi$.

ii. $|\neg\varphi| = |\varphi|$.

iii. $\mathrm{var}_{\mathbf{F}}(\neg\varphi) = \mathrm{var}_{\mathbf{G}}(\varphi)$ and $\mathrm{var}_{\mathbf{G}}(\neg\varphi) = \mathrm{var}_{\mathbf{F}}(\varphi)$.

iv. If φ is well-formed, then so is $\neg\varphi$.

v. If φ is an LTL formula, then so is $\neg\varphi$.

vi. If φ is a $\text{PLTL}_{\mathbf{F}}$ formula, then $\neg\varphi$ is a $\text{PLTL}_{\mathbf{G}}$ formula.

vii. If φ is a $\text{PLTL}_{\mathbf{G}}$ formula, then $\neg\varphi$ is a $\text{PLTL}_{\mathbf{F}}$ formula.

Note that the first statement means that double negation is a syntactic equivalence, not only a semantic one. The first three statements are simple consequences of the definition of φ' and the last four statements are applications of the third one.

When we consider a PLTL formula with respect to a fixed variable valuation, all bounds are determined, e.g., $\mathbf{F}_{\leq x}\varphi$ expresses that φ is satisfied at least once within the next $\alpha(x)$ steps. Since α is fixed, this property can be expressed by an LTL formula using a disjunction of nested next operators, i.e., without the use of parameterized operators. Dually, $\mathbf{G}_{\leq y}\varphi$ can be expressed using a conjunction of nested next operators, if α is fixed.

Lemma 3.8. *For every* PLTL *formula* φ *and every valuation* α*, there exists an effectively constructible* LTL *formula* φ_α *of size* $|\varphi| \cdot \mathcal{O}(\max_{z\in\text{var}(\varphi)} \alpha(z))$ *such that* $(w, n, \alpha) \models \varphi$ *if and only if* $(w, n) \models \varphi_\alpha$*.*

Proof. Given a PLTL formula ψ and $k \in \mathbb{N}$, we define the formula $\mathbf{X}_\vee^k\psi$ inductively by $\mathbf{X}_\vee^0\psi = \psi$ and $\mathbf{X}_\vee^{k+1}\psi = \psi \vee \mathbf{X}(\mathbf{X}_\vee^k\psi)$. It is satisfied at a position n, if ψ holds at a position $n + j$ for some j in the range $0 \leq j \leq k$. Similarly, we define the formula $\mathbf{X}_\wedge^k\psi$ inductively by $\mathbf{X}_\wedge^0\psi = \psi$ and $\mathbf{X}_\wedge^{k+1}\psi = \psi \wedge \mathbf{X}(\mathbf{X}_\wedge^k\psi)$. It is satisfied at a position n, if ψ holds at every position $n+j$ for j in the range $0 \leq j \leq k$.

Now, fix some variable valuation α. We define φ_α by inductively replacing every parameterized subformula $\mathbf{F}_{\leq x}\psi$ by $\mathbf{X}_\vee^{\alpha(x)}\psi$ and $\mathbf{G}_{\leq y}\psi$ by $\mathbf{X}_\wedge^{\alpha(y)}\psi$, respectively. Formally, we define

- $(p)_\alpha = p$ and $(\neg p)_\alpha = \neg p$,
- $(\varphi \wedge \psi)_\alpha = \varphi_\alpha \wedge \psi_\alpha$,
- $(\varphi \vee \psi)_\alpha = \varphi_\alpha \vee \psi_\alpha$,
- $(\mathbf{X}\varphi)_\alpha = \mathbf{X}\varphi_\alpha$,
- $(\varphi\, \mathbf{U}\, \psi)_\alpha = \varphi_\alpha\, \mathbf{U}\, \psi_\alpha$,
- $(\varphi\, \mathbf{R}\, \psi)_\alpha = \varphi_\alpha\, \mathbf{R}\, \psi_\alpha$
- $(\mathbf{F}_{\leq x}\varphi)_\alpha = \mathbf{X}_\vee^{\alpha(x)}\varphi_\alpha$, and
- $(\mathbf{G}_{\leq y}\varphi)_\alpha = \mathbf{X}_\wedge^{\alpha(y)}\varphi_\alpha$.

Since we have $|\mathbf{X}^k_\vee\psi| = |\mathbf{X}^k_\wedge\psi| = |\psi| + \mathcal{O}(k)$, the size of φ_α is bounded by $|\varphi| \cdot \mathcal{O}\left(\max_{z\in\mathrm{var}(\varphi)} \alpha(z)\right)$. Now, it remains to show $(w, n, \alpha) \models \varphi$ if and only if $(w, n) \models \varphi_\alpha$ by structural induction over the construction of φ. The only non-trivial cases are the parameterized operators. We have

$$
\begin{aligned}
& (w, n, \alpha) \models \mathbf{F}_{\leq x}\varphi \\
\overset{\text{Def.}}{\iff} & \text{there is a } j \text{ in the range } 0 \leq j \leq \alpha(x) \text{ s. t. } (w, n+j, \alpha) \models \varphi \\
\overset{\text{I.H.}}{\iff} & \text{there is a } j \text{ in the range } 0 \leq j \leq \alpha(x) \text{ s. t. } (w, n+j) \models \varphi_\alpha \\
\iff & (w, n) \models \mathbf{X}^{\alpha(x)}_\vee \varphi_\alpha \\
\overset{\text{Def.}}{\iff} & (w, n) \models (\mathbf{F}_{\leq x}\varphi)_\alpha
\end{aligned}
$$

and

$$
\begin{aligned}
& (w, n, \alpha) \models \mathbf{G}_{\leq y}\varphi \\
\overset{\text{Def.}}{\iff} & \text{for all } j \text{ in the range } 0 \leq j \leq \alpha(x)\text{: } (w, n+j, \alpha) \models \varphi \\
\overset{\text{I.H.}}{\iff} & \text{for all } j \text{ in the range } 0 \leq j \leq \alpha(x)\text{: } (w, n+j) \models \varphi_\alpha \\
\iff & (w, n) \models \mathbf{X}^{\alpha(y)}_\wedge \varphi_\alpha \\
\overset{\text{Def.}}{\iff} & (w, n) \models (\mathbf{G}_{\leq y}\varphi)_\alpha \ . \qquad \square
\end{aligned}
$$

Due to Lemma 3.8, we do not consider a fixed variable valuation when defining games with winning conditions in PLTL, but ask whether Player 0 can win a game with respect to some, infinitely many, or all variable valuations or ask for optimal variable valuations.

A simple, but very useful property of PLTL is the monotonicity of the parameterized operators. If φ is satisfied at least once within in the next k steps, then it is also satisfied at least once within in the next k' steps, provided we have $k' > k$. Dually, if φ is satisfied during each of the next k steps, then also during the next k' steps, provided $k' < k$. Hence, the set of variable valuations α such that a formula φ is satisfied by w with respect to α is upwards-closed for variables $x \in \mathrm{var}_{\mathbf{F}}(\varphi)$ and downwards-closed for variables $y \in \mathrm{var}_{\mathbf{G}}(\varphi)$.

Lemma 3.9 ([AETP01])**.** *Let φ be a* PLTL *formula and let α and β be variable valuations satisfying $\beta(x) \geq \alpha(x)$ for every $x \in \mathrm{var}_{\mathbf{F}}(\varphi)$ and $\beta(y) \leq \alpha(y)$ for every $y \in \mathrm{var}_{\mathbf{G}}(\varphi)$. If $(w, n, \alpha) \models \varphi$, then $(w, n, \beta) \models \varphi$.*

Finally, it is well-known that LTL formulae can be translated into ω-automata. The translation into Büchi automata is the basis of LTL model-checking while the translation into deterministic automata is the basis of solving LTL games. Since we are concerned with games, we only state the second translation (see also Subsection 3.3.2), which can be obtained by

combining a translation from LTL to (generalized) Büchi automata [VW94] (see also [BK08] and the references therein) and a determinization procedure for Büchi automata to deterministic automata [Saf88, MS95, Pit07, MS08].

Theorem 3.10. *For every* LTL *formula* φ*, there exists an effectively constructible deterministic parity automaton* $\mathfrak{A}$ *of size* $2^{2^{\mathcal{O}(|\varphi|)}}$ *with* $\mathcal{O}(2^{|\varphi|})$ *priorities such that* $L(\mathfrak{A}) = \{w \in \left(2^P\right)^\omega \mid w \models \varphi\}$.

3.1.4 PLTL Games

After we have introduced PLTL, we are now able to define infinite games with winning conditions in PLTL in this subsection. Then, we discuss some basic properties of these games.

To evaluate the winning condition – which is a PLTL formula – on a play, we have to label the arena with atomic propositions from the set P we use to build our formulae. Thus, a labeled arena $\mathcal{A} = (V, V_0, V_1, E, \ell)$ consists of an arena (V, V_0, V_1, E) as defined in Subsection 2.3.1 and a labeling function $\ell\colon V \to 2^P$. In figures, we denote the labeling of a vertex v by a set of propositions above or below v, where we omit empty labels. The trace of a play ρ is $\mathrm{tr}(\rho) = \ell(\rho_0)\ell(\rho_1)\ell(\rho_2)\cdots$. In this chapter, we mainly deal with labeled arenas. To keep things simple, we refer to them as arenas, too, as long as no confusion can arise. Furthermore, it is also convenient to consider a designated initial vertex in which every play starts.

Definition 3.11. A PLTL game $\mathcal{G} = (\mathcal{A}, v_0, \varphi)$ consists of a (labeled) arena $\mathcal{A} = (V, V_0, V_1, E, \ell)$, an initial vertex $v_0 \in V$, and a well-formed[7] PLTL formula φ.

The size of $\mathcal{G}$, denoted by $|\mathcal{G}|$, is defined as $|\mathcal{G}| = |\mathcal{A}| + |\varphi|$. LTL, PROMPT–LTL, $\mathrm{PLTL}_{\mathbf{F}}$, $\mathrm{PLTL}_{\mathbf{G}}$, and unipolar games are defined by restricting the winning conditions to LTL, PROMPT–LTL, $\mathrm{PLTL}_{\mathbf{F}}$, $\mathrm{PLTL}_{\mathbf{G}}$, and unipolar formulae, respectively.

A play in $(\mathcal{A}, v_0, \varphi)$ is an infinite path through $\mathcal{A}$ starting in v_0. To define the notions of winning a play and of winning strategies, we need to evaluate φ on a trace or a set of traces. Thus, both notions are defined with respect to a variable valuation which allows the evaluation. We say that Player 0 wins a play ρ with respect to a variable valuation α if $(\mathrm{tr}(\rho), \alpha) \models \varphi$, otherwise Player 1 wins ρ with respect to α. A strategy σ for Player i is a winning strategy for her with respect to α if every play that is consistent with σ is won by Player i with respect to α. If Player i has such a winning strategy, then we say that she wins $\mathcal{G}$ with respect to α. Again, winning an LTL game $\mathcal{G}$ is independent of α, hence we are justified to say that Player i wins $\mathcal{G}$.

[7] The need for the restriction to well-formed formulae is discussed in Subsection 3.1.2.

Definition 3.12. Let $\mathcal{G}$ be a PLTL game. The set $\mathcal{W}_i(\mathcal{G})$ of winning variable valuations for Player i is

$$\mathcal{W}_i(\mathcal{G}) = \{\alpha \mid \text{Player } i \text{ wins } \mathcal{G} \text{ with respect to } \alpha\} \ .$$

If $\mathcal{G}$ is an LTL game, then $\mathcal{W}_i(\mathcal{G})$ contains either every variable valuation – which is the case if Player i wins $\mathcal{G}$ – or it is empty – which is the case if Player $1 - i$ wins $\mathcal{G}$.

We illustrate these definitions with two example games in the arena $\mathcal{A}$ depicted in Figure 3.1. Here, the propositions q_0 and q_1 represent requests of some resources and p_0 and p_1 represent the corresponding responses. At vertex v_0, Player 1 can choose to request one or both of q_0 and q_1, and at vertex v_5, Player 0 can respond to at most one of the requests. Furthermore, at vertex v_4, Player 1 can choose to delay the play as long as he wants to, even preventing Player 1 from responding to requests. To avoid this kind of behavior, all our example winning conditions have a fairness condition which guarantees a win for Player 0, if the play stays in v_4 ad infinitum.

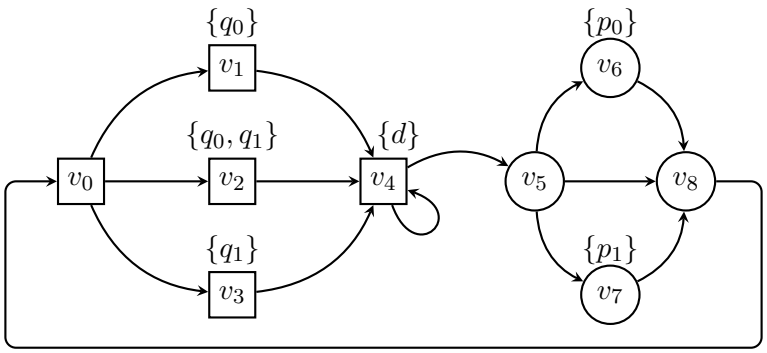

Figure 3.1: The arena $\mathcal{A}$ for Example 3.13 and Example 3.14

Example 3.13. Player 0 wins the LTL game $(\mathcal{A}, v_0, \varphi_1)$ with

$$\varphi_1 = \mathbf{FG}d \vee \bigwedge_{i \in \{0,1\}} \mathbf{G}(q_i \rightarrow \mathbf{F}p_i) \ .$$

In this game, Player 0 wins a play if it eventually stays in v_4 ad infinitum or if she is able to answer every request. One possible winning strategy proceeds by alternatingly moving to v_6 and v_7 when the token is at v_5, thereby answering every request eventually, if the play does not end up staying in v_4 ad infinitum. ◇

Example 3.14. Now, consider the formula

$$\varphi_2 = \mathbf{FG}d \vee \bigwedge_{i \in \{0,1\}} \mathbf{G}(q_i \rightarrow \mathbf{F}_{\leq x_i} p_i)$$

obtained by parameterizing the eventually operators of φ_1. To win the $\text{PLTL}_{\mathbf{F}}$ game $\mathcal{G}_2 = (\mathcal{A}, v_0, \varphi_2)$ with respect to a variable valuation α, Player 0 has to answer every request q_i within $\alpha(x_i)$ positions. We claim $\mathcal{W}_0(\mathcal{G}_2) = \emptyset$: fix some variable valuation α and let $k = \max\{2, \alpha(x_0), \alpha(x_1)\}$ and consider the play in which Player 1 activates both requests by moving to v_2 and then uses the self-loop at vertex v_4 $(k-2)$ times and then moves to v_5. Then, both requests q_i are not answered within $\alpha(x_i)$ positions. For the remainder of the play, Player 1 just has to make sure to leave v_4 infinitely often. This strategy is winning against every strategy of Player 0. Hence, we have $\alpha \in \mathcal{W}_1(\mathcal{G}_2)$. $\diamond$

LTL games were – in a more general framework – investigated by Pnueli and Rosner, who showed them to be **2Exptime**-complete. Their results hold in the setting of graph-based games, too, and serves as the yardstick we measure our results about PLTL games against.

Theorem 3.15 ([PR89a, PR89b, Ros91])**.** LTL *games are determined with uniform finite-state strategies of size* $2^{2^{\mathcal{O}(|\varphi|)}}$ *and determining the winner of an* LTL *game is* **2Exptime***-complete.*

2Exptime-membership can be proven by applying Lemma 2.17 to reduce an LTL game to a parity game using a deterministic parity automaton $\mathfrak{A}$ as in Theorem 3.10. The automaton $\mathfrak{A}$ is of doubly-exponential size and has exponentially many priorities (both measured in $|\varphi|$). Hence, the parity game obtained in the reduction can be solved in doubly-exponential time due to Theorem 2.22.

Furthermore, Alur et al. [ATM03, AT04] refined the work of Pnueli and Rosner and determined the complexity of solving games with winning conditions in several fragments of LTL obtained by restricting the set of boolean connectives and temporal operators used to built formulae.

A simple consequence of Lemma 3.8 is that a PLTL game with winning condition φ with respect to a fixed variable valuation α is nothing more than an LTL game with winning condition φ_α. This is formally stated in the next lemma. Hence, we are not interested in games with a fixed valuation but ask whether a player can win with respect to some, infinitely many, or all variable valuations or want to compute optimal valuations.

Lemma 3.16. *Let* $\mathcal{G} = (\mathcal{A}, v_0, \varphi)$ *be a* PLTL *game and let* α *be a variable valuation. We have* $\alpha \in \mathcal{W}_i(\mathcal{G})$ *if and only if Player* i *wins the* LTL *game* $(\mathcal{A}, v_0, \varphi_\alpha)$*.*

We obtain a determinacy result for PLTL games as a corollary of the previous lemma.

Corollary 3.17. *Let* $\mathcal{G}$ *be a* PLTL *game and let* α *be a variable valuation. Then, one of the players has a finite-state winning strategy for* $\mathcal{G}$ *with respect to* α*.*

We have already seen that a PLTL formula φ can be negated such that $(w, n, \alpha) \models \neg\varphi$ if and only if $(w, n, \alpha) \not\models \varphi$. Hence, we can dualize PLTL games. This is especially useful when considering unipolar games, since the negation of a $\text{PLTL}_{\mathbf{F}}$ formula is a $\text{PLTL}_{\mathbf{G}}$ formula and vice versa. Hence, we can solve many problems by only considering one type of unipolar game.

Definition 3.18. Let $\mathcal{A} = (V, V_0, V_1, E, \ell)$ be an arena and $\mathcal{G} = (\mathcal{A}, v_0, \varphi)$ be a PLTL game. Then, $\overline{\mathcal{A}} = (V, V_1, V_0, E, \ell)$ is the dual arena of $\mathcal{A}$, and $\overline{\mathcal{G}} = (\overline{\mathcal{A}}, v_0, \neg\varphi)$ is the dual game of $\mathcal{G}$.

Since the negation of a well-formed formula is well-formed, too, the dual game satisfies the definition of a PLTL game. Furthermore, the dual of a $\text{PLTL}_{\mathbf{F}}$ game is a $\text{PLTL}_{\mathbf{G}}$ game and vice versa. Both facts are direct consequences of Remark 3.7. Let α be a variable valuation: since we negate the winning condition and swap the player's positions, we have $\alpha \in \mathcal{W}_i(\mathcal{G})$ if and only if $\alpha \in \mathcal{W}_{1-i}(\overline{\mathcal{G}})$. Our first results are a consequence of Corollary 3.17 and of the previous observation, respectively.

Lemma 3.19. *Let $\mathcal{G}$ be a* PLTL *game.*

i. $\mathcal{W}_i(\mathcal{G})$ is the complement of $\mathcal{W}_{1-i}(\mathcal{G})$.

ii. $\mathcal{W}_i(\mathcal{G}) = \mathcal{W}_{1-i}(\overline{\mathcal{G}})$.

Finally, Lemma 3.9 can easily be lifted to games.

Lemma 3.20. *Let $\mathcal{G} = (\mathcal{A}, v_0, \varphi)$ be a* PLTL *game.*

i. Let α and β be variable valuations satisfying $\beta(x) \geq \alpha(x)$ for every $x \in \text{var}_{\mathbf{F}}(\varphi)$ and $\beta(y) \leq \alpha(y)$ for every $y \in \text{var}_{\mathbf{G}}(\varphi)$. If $\alpha \in \mathcal{W}_0(\mathcal{G})$, then $\beta \in \mathcal{W}_0(\mathcal{G})$.

ii. Let α and β be variable valuations satisfying $\beta(x) \leq \alpha(x)$ for every $x \in \text{var}_{\mathbf{F}}(\varphi)$ and $\beta(y) \geq \alpha(y)$ for every $y \in \text{var}_{\mathbf{G}}(\varphi)$. If $\alpha \in \mathcal{W}_1(\mathcal{G})$, then $\beta \in \mathcal{W}_1(\mathcal{G})$.

Hence, $\mathcal{W}_0(\mathcal{G})$ is upwards-closed for variables parameterizing eventually operators and downwards-closed for variables parameterizing always operators. For a unipolar game $\mathcal{G}$, this means that $\mathcal{W}_i(\mathcal{G})$ has a very simple structure and can be represented finitely as a semilinear set. After we have proved this, we discuss the general case of non-unipolar games.

In the following, we assume $\text{var}(\varphi) = \{z_0, \ldots, z_{k-1}\}$ and identify a variable valuation α (restricted to the variables occurring in φ) by a vector $\mathbf{a} = (\alpha(z_0), \ldots, \alpha(z_{k-1})) \in \mathbb{N}^k$. Accordingly, we think of $\mathcal{W}_i(\mathcal{G})$ as subset of $\mathbb{N}^k$, if the winning condition of $\mathcal{G}$ has k variables. To state our results, we need some additional notation.

Given two vectors $\mathbf{a} = (a_0, \ldots, a_{k-1}) \in \mathbb{N}^k$ and $\mathbf{b} = (b_0, \ldots b_{k-1}) \in \mathbb{N}^k$, and a scalar $s \in \mathbb{N}$, we denote the componentwise addition of $\mathbf{a}$ and $\mathbf{b}$ by

$$\mathbf{a} + \mathbf{b} = (a_0 + b_0, \ldots, a_{k-1} + b_{k-1})$$

and the scalar multiplication of s and $\mathbf{a}$ by

$$s \cdot \mathbf{a} = (s \cdot a_0, \ldots, s \cdot a_{k-1}) \ .$$

Moreover, we compare vectors componentwise, i.e., we have $\mathbf{a} \leq \mathbf{b}$ if $a_j \leq b_j$ for every $j \in [k]$. The upwards-closure of the vector $\mathbf{a}$ is the set $\mathbf{a}{\uparrow} = \{\mathbf{b} \in \mathbb{N}^k \mid \mathbf{a} \leq \mathbf{b}\}$. A set $A \subseteq \mathbb{N}^k$ is upwards-closed, if $\mathbf{a} \in A$ implies $\mathbf{a}{\uparrow} \subseteq A$. A set is downwards-closed, if it is the complement of an upwards-closed set[8]. A simple corollary of Lemma 3.20 is that the sets of winning valuations for a player in a unipolar game are closed. Due to duality, we state the result for Player 0 only.

Corollary 3.21.

i. Let $\mathcal{G}$ be a $\mathrm{PLTL}_{\mathbf{F}}$ game. Then, $\mathcal{W}_0(\mathcal{G})$ is upwards-closed.

ii. Let $\mathcal{G}$ be a $\mathrm{PLTL}_{\mathbf{G}}$ game. Then, $\mathcal{W}_0(\mathcal{G})$ is downwards-closed.

A set $A \subseteq \mathbb{N}^k$ is linear if there exist vectors $\mathbf{a_0}, \mathbf{a_1}, \ldots, \mathbf{a_n}$ such that

$$A = \{\mathbf{a} \in \mathbb{N}^k \mid \mathbf{a} = \mathbf{a_0} + \sum_{j=1}^{n} s_j \cdot \mathbf{a_j} \text{ for some scalars } s_1, \ldots, s_n \in \mathbb{N}\} \ ,$$

and a subset of $\mathbb{N}^k$ is semilinear, if it is a finite union of linear sets. It is easy to see that the upwards-closure $\mathbf{a}{\uparrow}$ of a vector $\mathbf{a} \in \mathbb{N}^k$ is linear. A semilinear set can be represented finitely by the generators of its constituting linear sets or by a Presburger formula [Pre29], a first-order formula in which addition emerges as the only operator.

Lemma 3.22. *Every upwards- or downwards-closed subset of $\mathbb{N}^k$ is semilinear.*

Proof. We show that an upwards-closed set A is semilinear. This suffices, since every downwards-closed set is the complement of an upwards-closed set and since the semilinear sets are closed under complementation [GS64].

Let $A' \subseteq A$ be the set of $\leq$-minimal elements of A, which is finite due to Dickson's Lemma [Dic13]. Then, we have $A = \bigcup_{\mathbf{a} \in A'} \mathbf{a}{\uparrow}$. As every upwards-closure is linear, A is semilinear. □

[8] Alternatively, one could require the downwards-closure of each element to be contained in the set, where the downwards-closure of a vector is defined analogously to the upwards-closure.

Our result is now a simple consequence of Corollary 3.21 and Lemma 3.22.

Theorem 3.23. *Let $\mathcal{G}$ be a unipolar* PLTL *game. Then, $\mathcal{W}_i(\mathcal{G})$ is semilinear.*

The exact structural complexity of the set of winning valuations for a non-unipolar PLTL game is open. This is even the case for PLTL model-checking, i.e., solitary games [AETP01]. Furthermore, the undecidability results for formulae that are not well-formed imply that if $\mathcal{W}_0(\mathcal{G})$ were semilinear, then a finite representation cannot be effectively computable: remember that it is undecidable, whether a PLTL formula has a model with respect to a variable valuation α such that $\alpha(x) = \alpha(y)$ for some variables $x \in \mathrm{var}_{\mathbf{F}}(\varphi)$ and $y \in \mathrm{var}_{\mathbf{G}}(\varphi)$. Since this question can easily be answered for linear sets represented by its generators using linear equations (and therefore also for semilinear sets), there is no algorithm that computes a finite representation of $\mathcal{W}_0(\mathcal{G})$ as semilinear set.

Nevertheless, using the monotonicity of the parameterized always operators and the alternating color technique presented in Subsection 3.2.2 we can compute projections of $\mathcal{W}_i(\mathcal{G})$ to the coordinates representing variables from $\mathrm{var}_{\mathbf{F}}(\varphi)$ and $\mathrm{var}_{\mathbf{G}}(\varphi)$, respectively. Here, we only explain the first case. For the second case, we need additional tools and refer to the end of Subsection 3.2.2 for the time being.

Let $\mathcal{G} = (\mathcal{A}, v_0, \varphi)$ be a PLTL game such that $\mathrm{var}_{\mathbf{F}}(\varphi) = \{z_0, \ldots, z_{k-1}\}$ and $\mathrm{var}_{\mathbf{G}}(\varphi) = \{z_k, \ldots, z_{k+k'-1}\}$. Now, let $\varphi_{\mathbf{F}}$ be the formula obtained by inductively replacing every subformula $\mathbf{G}_{\leq y}\psi$ by ψ and consider the $\mathrm{PLTL}_{\mathbf{F}}$ game $\mathcal{G}' = (\mathcal{A}, v_0, \varphi_{\mathbf{F}})$. Due to downwards-closure, we have that $(n_1, \ldots, n_{k-1}) \in \mathcal{W}_0(\mathcal{G}')$ if and only if $(n_1, \ldots, n_{k-1}, n_k, \ldots, n_{k+k'-1}) \in \mathcal{W}_0(\mathcal{G})$ for some $n_k, \ldots, n_{k+k'-1} \in \mathbb{N}$.

Note that the dual construction for the coordinates representing variables from $\mathrm{var}_{\mathbf{G}}(\varphi)$ does not work: the next example shows that replacing parameterized eventually operators by unparameterized eventually operators does not yield the projection. Using the tools developed in the next section we give a correct construction.

Example 3.24. Consider the solitary PLTL game $\mathcal{G} = (\mathcal{A}, v_0, \varphi)$ where $\mathcal{A}$ is depicted in Figure 3.2 and $\varphi = (\mathbf{FG}p \vee \mathbf{GF}_{\leq x}q) \wedge \mathbf{G}_{\leq y}\mathbf{tt}$.

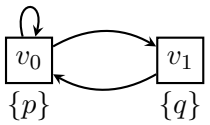

Figure 3.2: The arena $\mathcal{A}$ for Example 3.24

Since the second conjunct is satisfied with respect to every variable valuation, Player 0 wins with respect to a variable valuation α, if p is satisfied

continuously from some position onwards or if q is satisfied at least once in every infix of length $\alpha(x) + 1$. Thus, the play $\prod_{j \geq 0}((v_0)^j v_1)$ is winning for Player 1 with respect to every variable valuation, since its trace contains arbitrary long infixes in which q does not hold, but $\neg p$ holds infinitely often. Hence, we have $\mathcal{W}_0(\mathcal{G}) = \emptyset$.

Now, consider the game $\mathcal{G}' = (\mathcal{A}, v_0, \varphi')$ obtained by replacing the parameterized eventually operator by an unparameterized one, i.e., φ' is the $\text{PLTL}_{\mathbf{G}}$ formula $(\mathbf{FG}p \vee \mathbf{GF}q) \wedge \mathbf{G}_{\leq y}\mathbf{tt}$. We claim that $\mathcal{W}_0(\mathcal{G}')$ contains every variable valuation. We have already seen that the second conjunct is satisfied with respect to every variable valuation. Thus, we only have to consider the first conjunct, which is variable-free. Consider a play in $\mathcal{A}$: either it visits v_1 infinitely often, which means its trace satisfies $\mathbf{GF}q$, or it stays from some point onwards in v_0, which means its trace satisfies $\mathbf{FG}p$. Hence, every play is winning for Player 0 with respect to every variable valuation. Thus, $\mathcal{W}_0(\mathcal{G}')$ is not the projection of $\mathcal{W}_0(\mathcal{G})$ to the coordinate representing y. $\diamond$

Let us conclude this introductory subsection about PLTL games by discussing how to adapt game reductions to initialized games of the form $\mathcal{G} = (\mathcal{A}, v_0, \text{Win})$, where Win is the set of winning plays for Player 0. We say that $\mathcal{G}$ is reducible to $\mathcal{G}' = (\mathcal{A}', v_0', \text{Win}')$ via memory $\mathcal{M} = (M, \text{init}, \text{upd})$, again written $\mathcal{G} \leq_{\mathcal{M}} \mathcal{G}'$, if $\mathcal{A}'$ is the extended arena[9] $\mathcal{A} \times \mathcal{M}$, if the initial vertex[10] v_0' is $(v_0, \text{init}(v_0))$, and if every play in $\mathcal{A}$ (which starts in v_0) is won by the same player who wins the extended play in $\mathcal{A}'$ (which starts in v_0'). We obtain the correctness of this construction as corollary of Lemma 2.15.

Corollary 3.25. *Let $\mathcal{G}$ be an initialized game. If $\mathcal{G} \leq_{\mathcal{M}} \mathcal{G}'$ and Player i has a positional winning strategy for $\mathcal{G}'$, then she also has a finite-state winning strategy with memory $\mathcal{M}$ for $\mathcal{G}$.*

Furthermore, Lemma 2.17 can be formulated for initialized games as well.

Corollary 3.26. *Let $\mathcal{G} = (\mathcal{A}, v_0, \text{Win})$ and let $\mathfrak{A}$ be a deterministic parity automaton such that $L(\mathfrak{A}) = \text{Win}$. Then, $\mathcal{G}$ can be reduced to a parity game via a memory structure of size $|\mathfrak{A}|$.*

3.2 Solving Games with PLTL Winning Conditions

In this section, we show how to solve PLTL games. Since we consider initialized games, solving them only requires to determine the winner from the initial vertex and a corresponding winning strategy. However, winning a

[9] When defining the product of a labeled arena and a memory structure, we ignore the labeling function and use the definition presented in Subsection 2.3.3.

[10] Since a play always begins in the initial vertex we could replace the initialization function by an initial memory state m_0. To avoid notational overhead, we choose to stick to the function, even if it is only applied to v_0.

PLTL game (and being a winning strategy) is defined with respect to a variable valuation. Hence, solving a PLTL game $\mathcal{G}$ refers to properties of the sets $\mathcal{W}_i(\mathcal{G})$ of winning valuations. The membership question for a variable valuation α asks whether a given player wins the game with respect to α. However, this question is not very interesting since we have seen that a PLTL game with respect to a fixed variable valuation is equivalent to an LTL game. Hence, we are interested in the emptiness, finiteness, and universality problem for $\mathcal{W}_i(\mathcal{G})$, i.e., the question whether Player i can win $\mathcal{G}$ with respect some, infinitely many, or all variable valuations. In this section, we show that all three problems are not harder than solving LTL games. Formally, we are interested in the following decision problems.

- Membership: given a PLTL game $\mathcal{G}$, $i \in \{0,1\}$, and a variable valuation α, does $\alpha \in \mathcal{W}_i(\mathcal{G})$ hold?
- Emptiness: given a PLTL game $\mathcal{G}$ and $i \in \{0,1\}$, is $\mathcal{W}_i(\mathcal{G})$ empty?
- Finiteness: given a PLTL game $\mathcal{G}$ and $i \in \{0,1\}$, is $\mathcal{W}_i(\mathcal{G})$ finite?
- Universality: given a PLTL game $\mathcal{G}$ and $i \in \{0,1\}$, does $\mathcal{W}_i(\mathcal{G})$ contain every variable valuation?

As already defined earlier, the size of a game is the sum of the sizes of its arena and its winning condition. Furthermore, we encode variable valuations in binary (and restrict them to variables occurring in the winning condition). Hence, we measure the running time of algorithms for the membership problem in

$$|\mathcal{G}| + \sum_{z \in \mathrm{var}(\varphi)} \lceil \log_2(\alpha(z) + 1) \rceil$$

and the running time of algorithms for the latter three problems in $|\mathcal{G}|$.

Our first result is a simple consequence of Lemma 3.16 and Theorem 3.15: a PLTL formula φ with respect to a fixed variable valuation α is equivalent to an LTL formula φ_α. Hence, to determine membership of α in $\mathcal{W}_i(\mathcal{G})$ it suffices to solve the LTL game with winning condition φ_α. Since the size of φ_α is exponential in the binary representation of α and linear in the size of φ, the algorithm sketched here has triply-exponential running time. We obtain an algorithm with doubly-exponential running time when we compute optimal winning strategies in Section 3.3. For the time being, we just remark that the membership problem is trivially **2Exptime**-hard. Let $\mathcal{G}$ be an LTL game: then, $\mathcal{W}_0(\mathcal{G})$ contains the empty variable valuation if and only if Player 0 wins $\mathcal{G}$.

Remark 3.27. The membership problem for PLTL is **2Exptime**-hard.

In the remainder of this section we consider the latter three problems. In Subsection 3.2.1, we extend the alternating color technique of Kupferman et al. [KPV09] for PROMPT–LTL to $\text{PLTL}_{\mathbf{F}}$. Then, in Subsection 3.2.2, we apply this technique to solve the emptiness problem for $\text{PLTL}_{\mathbf{F}}$ games in doubly-exponential time by a reduction to solving LTL games. Finally, in Subsection 3.2.3, we prove that this result and the monotonicity of PLTL suffice to solve the latter three problems for full PLTL in doubly-exponential time as well.

3.2.1 Digression: The Alternating Color Technique

Kupferman et al. introduced PROMPT–LTL and solved several of its decision problems – among them model-checking, assume-guarantee model-checking, and the realizability problem – by using their alternating color technique [KPV09]. Although the technique in its original formulation is only applicable to PROMPT–LTL formulae it is easy to see that the restriction to a single variable is not necessary. Furthermore, it turns out to be useful to abandon the restriction when we consider the optimization problems for PLTL games in Section 3.3. Hence, we state the technique here in a more general version than it was presented in the original work on PROMPT–LTL. We discuss the extent of these generalizations after the proof of Lemma 3.28, which states the correctness of the extended technique.

Intuitively, the alternating color technique allows to replace a parameterized eventually operator by an LTL formula: consider a PROMPT–LTL formula φ and let p be an atomic proposition that does not appear in φ. We say that a position n of a trace w is green, if p holds at it, otherwise we say that n is red. Hence, p induces a decomposition of w into (maximal) monochromatic blocks. Now, we relativize φ by inductively replacing a subformula $\mathbf{F}_{\leq x}\psi$ by an LTL formula specifying that ψ is satisfied within one color change, i.e., ψ is either satisfied in the current or in the following block. If the distance between color changes is bounded, then the original PROMPT–LTL formula is satisfied with respect to some variable valuation, if the relativized LTL formula is satisfied. Dually, if the blocks are not shorter than the value of α for the variable occurring in φ, then the relativized LTL formula is satisfied, if the original PROMPT–LTL formula is satisfied with respect to α. Hence, the problem of finding a bound for the parameterized eventually operators is reducible to the problem of satisfying an LTL formula while ensuring a bound on the color changes; the alternating color technique simplifies the dependencies between parameterized subformulae.

After introducing the alternating color technique formally, we end this subsection by presenting an application to the PROMPT–LTL realizability problem due to Kupferman et al. [KPV09]. Then, in the following subsection, we extend the solution of the realizability problem to the emptiness problem for graph-based $\text{PLTL}_{\mathbf{F}}$ games and subsequently use this result to solve the

emptiness, finiteness, and universality problem for PLTL games.

Let $p \notin P$ be a fixed fresh proposition. An ω-word $w' \in \left(2^{P \cup \{p\}}\right)^\omega$ is a p-coloring of $w \in \left(2^P\right)^\omega$ if $w'_n \cap P = w_n$, i.e., w_n and w'_n coincide on all propositions in P. The additional proposition p can be thought of as the color of w'_n: we say that a position n is green if $p \in w'_n$, and say that it is red if $p \notin w'_n$. Furthermore, we say that the color changes at position n, if $n = 0$ or if the colors of w'_{n-1} and w'_n are not equal. In this situation, we say that n is a change point A p-block is a maximal monochromatic infix $w'_m \cdots w'_n$ of w', i.e., the color changes at m and $n+1$ but not in between. Let $k \geq 1$: we say that w' is k-spaced, if the color changes infinitely often and each p-block has length at least k; we say that w' is k-bounded, if each p-block has length at most k (which implies that the color changes infinitely often).

Now, we introduce the relativization that replaces a parameterized eventually operator $\mathbf{F}_{\leq x}\psi$ by an LTL formula requiring ψ to be satisfied within one color change. Then, we only need to bound the distance between color changes while satisfying an LTL formula, which turns out to be much simpler. To be as general as possible, we allow to remove only a subset of the parameterized eventually operators. Given a PLTL formula φ and $X \subseteq \mathrm{var}_{\mathbf{F}}(\varphi)$, let φ_X denote the relativized formula obtained by inductively replacing every subformula $\mathbf{F}_{\leq x}\psi$ with $x \notin X$ by

$$(p \rightarrow (p \,\mathbf{U}\, (\neg p \,\mathbf{U}\, \psi_X))) \wedge (\neg p \rightarrow (\neg p \,\mathbf{U}\, (p \,\mathbf{U}\, \psi_X))) \ .$$

We have $\mathrm{var}_{\mathbf{F}}(\varphi_X) = X$ (i.e., X denotes the variables that are not replaced), $\mathrm{var}_{\mathbf{G}}(\varphi_X) = \mathrm{var}_{\mathbf{G}}(\varphi)$, and $|\varphi_X| \in \mathcal{O}(|\varphi|)$. Furthermore, the formula $alt_p = \mathbf{GF}p \wedge \mathbf{GF}\neg p$ is satisfied if the colors change infinitely often. Finally, consider the formula $\varphi_X \wedge alt_p$. It is satisfied by w with respect to a variable valuation α, if the following holds:

- The color changes infinitely often.
- Every subformula $\mathbf{F}_{\leq x}\psi$ with $x \notin X$ is satisfied within one color change.
- The parameterized eventually operators with variables $x \in X$ – which are not replaced in φ_X – are satisfied within the bounds specified by α.
- The parameterized always operators are satisfied with respect to the bounds specified by α.

The application of the alternating color technique allows to remove some variables from $\mathrm{var}_{\mathbf{F}}(\varphi)$. Next, we show that φ and φ_X are "equivalent" on ω-words which are bounded and spaced at the same time. Our correctness lemma differs from the original one presented in [KPV09], since we generalize the technique to PLTL formulae and allow to replace just some parameterized eventually operators. However, the proof itself is similar to the original one.

Lemma 3.28 (cf. Lemma 2.1 of [KPV09]). *Let φ be a* PLTL *formula, let $X \subseteq \mathrm{var}_{\mathbf{F}}(\varphi)$, and let $w \in \left(2^P\right)^\omega$.*

i. If $(w, \alpha) \models \varphi$, then $(w', \alpha) \models \varphi_X \wedge alt_p$ for every k-spaced p-coloring w' of w, where $k = \max_{x \in \mathrm{var}_{\mathbf{F}}(\varphi) \setminus X} \alpha(x)$.

ii. Let $k \in \mathbb{N}$. If w' is a k-bounded p-coloring of w with $(w', \alpha) \models \varphi_X$, then $(w, \beta) \models \varphi$, where $\beta(z) = \begin{cases} 2k & \text{if } z \in \mathrm{var}_{\mathbf{F}}(\varphi) \setminus X, \\ \alpha(z) & \text{otherwise.} \end{cases}$

Proof. i.) Let w' be a k-spaced p-coloring of w. By definition, the formula alt_p is satisfied by every k-spaced p-coloring. Hence, it remains to show $(w, n, \alpha) \models \varphi$ implies $(w', n, \alpha) \models \varphi_X$ by structural induction over the construction of φ.

The cases of atomic formulae, boolean connectives, unparameterized temporal operators, and $\mathbf{F}_{\leq x}$ with $x \in X$ as well as $\mathbf{G}_{\leq y}$ are straightforward applications of the induction hypothesis, since we only replace parameterized eventually operators with variable $x \notin X$. So, suppose we have $(w, n, \alpha) \models \mathbf{F}_{\leq x}\psi$ with $x \notin X$. We have to show

$$(w', n, \alpha) \models (p \rightarrow (p \,\mathbf{U}\, (\neg p \,\mathbf{U}\, \psi_X))) \wedge (\neg p \rightarrow (\neg p \,\mathbf{U}\, (p \,\mathbf{U}\, \psi_X))) \ , \quad (3.1)$$

where ψ_X is the relativization of ψ. Assume $p \in w'_n$; the case $p \notin w'_n$ is analogous. Then, we just have to show that the first conjunct is satisfied. Since $(w, n, \alpha) \models \mathbf{F}_{\leq x}\psi$, there is a j in the range $0 \leq j \leq \alpha(x)$ such that $(w, n + j, \alpha) \models \psi$ and thus $(w', n + j, \alpha) \models \psi_X$ by induction hypothesis. Since $k \geq \alpha(x)$ and since w' is k-spaced, there is at most one change point in between n and $n + j$ in w'. If there is no such change point, then $(w', n, \alpha) \models p \rightarrow (p \,\mathbf{U}\, \psi_X)$ and thus also $(w', n, \alpha) \models p \rightarrow (p \,\mathbf{U}\, (\neg p \,\mathbf{U}\, \psi_X))$. Hence, we have shown (3.1). If there is a change point at a position $n + c$ with $n < n + c \leq n + j$, then $(w', n + c, \alpha) \models \neg p \,\mathbf{U}\, \psi_X$ and therefore $(w', n, \alpha) \models p \rightarrow p \,\mathbf{U}\, (\neg p \,\mathbf{U}\, \psi_X)$, which again suffices to show (3.1).

ii.) Let w' be a k-bounded p-coloring of w. We show by induction over the construction of φ that $(w', n, \alpha) \models \varphi_X$ implies $(w, n, \beta) \models \varphi$. Again, the cases of atomic formulae, boolean connectives, unparameterized temporal operators, and $\mathbf{F}_{\leq x}$ with $x \in X$ as well as $\mathbf{G}_{\leq y}$ are straightforward applications of the induction hypothesis. So, consider the case $\mathbf{F}_{\leq x}\psi$ with $x \notin X$. Then, we have $(w', n, \alpha) \models (\mathbf{F}_{\leq x}\psi)_X$, i.e.,

$$(w', n, \alpha) \models (p \rightarrow (p \,\mathbf{U}\, (\neg p \,\mathbf{U}\, \psi_X))) \wedge (\neg p \rightarrow (\neg p \,\mathbf{U}\, (p \,\mathbf{U}\, \psi_X))) \ .$$

We have to show $(w, n, \beta) \models \mathbf{F}_{\leq x}\psi$. Assume $p \in w'_n$; the case $p \notin w'_n$ is again analogous. Then, we have $(w', n, \alpha) \models p \,\mathbf{U}\, (\neg p \,\mathbf{U}\, \psi_X))$. As w' is k-bounded, there are change points $n + c$ and $n + c + c'$ with $0 \leq c, c' \leq k$. Hence, there has to be a $j \leq c + c' \leq 2k$ such that $(w', n + j, \alpha) \models \psi_X$. Applying the induction hypothesis, we obtain $(w, n + j, \beta) \models \psi$ and thus $(w, n, \beta) \models \mathbf{F}_{\leq x}\psi$ due to $j \leq 2k = \beta(x)$. □

The original alternating color technique as introduced in [KPV09] is defined for PROMPT–LTL and replaces all parameterized operators at once, i.e., the case $X = \emptyset$ in our formulation. By reflecting on the previous proof it becomes clear that our extensions did not complicate it in any way: they can all be proven to be correct by applications of the induction hypothesis.

We conclude by discussing the application of the alternating color technique to the PROMPT–LTL realizability problem. An adaption of this proof allows us to solve the emptiness problem for $\text{PLTL}_{\mathbf{F}}$ games (which is closely related to the $\text{PLTL}_{\mathbf{F}}$ realizability problem) and then all decision problems.

A realizability problem for a logic is concerned with determining the winner of an abstract two-player game without underlying arena, i.e., the game consists only of a formula φ specifying the winning plays for one of the players. In this game, two players 0 and 1 alternatingly pick propositions, Player 0 from a set I of inputs and Player 1 from a set O of outputs. Hence, by picking in each round n a set $i_n \subseteq I$ respectively $o_n \subseteq O$ they construct an infinite word $\rho = (i_0 \cup o_0)(i_1 \cup o_1)(i_2 \cup o_2) \cdots$. Consider a PROMPT–LTL winning condition φ over the atomic propositions $I \cup O$. The play ρ is winning for Player 1 with respect to a variable valuation α if $(\rho, \alpha) \models \varphi$. A strategy for Player 1 for such a game is a mapping $\sigma\colon \left(2^I\right)^* \to 2^O$ and $(i_0 \cup o_0)(i_1 \cup o_1)(i_2 \cup o_2) \cdots$ is consistent with σ, if $o_n = \sigma(i_0 \cdots i_n)$ for every n. The realizability problem for PROMPT–LTL asks, whether there exists a variable valuation α and a strategy for Player 1 such that each consistent play is winning for him with respect to α. In this case we say that φ is realizable. Note that this formulation is already very close to the emptiness problem for PROMPT–LTL games as defined above.

Kupferman et al. [KPV09] applied the alternating color technique to show that a PROMPT–LTL formula φ with $\text{var}(\varphi) = \{x\}$ is realizable if and only if the LTL formula $\varphi_\emptyset \wedge \mathit{alt}_p$ is realizable. The crucial insight is that a finite-state strategy – which suffices to realize $\varphi_\emptyset \wedge \mathit{alt}_p$ – only produces k-bounded plays, where k only depends on the size of the finite-state strategy, which in turn only depends on the size of $\varphi_\emptyset \wedge \mathit{alt}_p$. Due to Lemma 3.28(ii), such a strategy also realizes φ with respect to a variable valuation that maps x to $2k$. For the other direction, Lemma 3.28(i) shows that a strategy realizing φ with respect to some variable valuation α can be turned into a strategy realizing $\varphi_\emptyset \wedge \mathit{alt}_p$ by outputting an $\alpha(x)$-spaced p-coloring of the original output.

As LTL realizability is **2Exptime**-hard [PR89b, Ros91] and as $\varphi_\emptyset \wedge \mathit{alt}_p$ is only linearly larger than φ, it follows that PROMPT–LTL realizability is in **2Exptime** as well. Furthermore, as LTL is a fragment of PROMPT–LTL, the PROMPT–LTL realizability problem is trivially **2Exptime**-hard as well. Thus, adding parameterized eventually operators to LTL does not increase the asymptotic complexity of the realizability problem.

3.2.2 Solving the Emptiness Problem for $\text{PLTL}_{\mathbf{F}}$ Games

The proof of **2Exptime**-membership of the PROMPT–LTL realizability problem can be adapted to solve the emptiness problem for $\text{PLTL}_{\mathbf{F}}$ games: instead of applying the original alternating color technique we use the extended one presented in the previous subsection. The proof presented here is similar to the one of Kupferman et al. for the PROMPT–LTL realizability problem [KPV09], but the presentation is more involved, since we consider graph-based games[11]. Most importantly, we have to allow Player 0 to produce p-colorings of plays. Since she has to be able to change the color while its not her turn, we have to add choice vertices to the arena that allow her to produce change points at any time. However, adding the choice vertices requires to ignore them when evaluating a formula to determine the winner of a play. Thus, we introduce blinking semantics for infinite games: under this semantics, only every other vertex contributes to the trace of a play. So, to show that the emptiness problem for $\text{PLTL}_{\mathbf{F}}$ games can be solved in doubly-exponential time, we add the choice vertices to the arena and apply the alternating color technique to obtain a polynomially larger LTL game under blinking semantics. As a corollary, we derive an upper bound on the values of a variable valuation that is winning for Player 0, provided there exists one at all.

We also use the construction presented here for the solution of the PLTL optimization problems and to construct the projection of $\mathcal{W}_0(\mathcal{G})$ to coordinates representing variables parameterizing always operators. Therefore, we present a more general reduction than we would need if we just want to prove the following result.

Theorem 3.29. *The emptiness problem for* $\text{PLTL}_{\mathbf{F}}$ *games is in* **2Exptime**.

As already hinted at above, we proceed as follows: given a $\text{PLTL}_{\mathbf{F}}$ game $\mathcal{G}$ we apply the alternating color technique for PLTL by constructing an LTL game $\mathcal{G}'$ under blinking semantics with $|\mathcal{G}'| \in \mathcal{O}(|\mathcal{G}|^2)$ such that $\mathcal{W}_0(\mathcal{G}) \neq \emptyset$ if and only if Player 0 wins $\mathcal{G}'$.

Lemma 3.28 shows how to replace (on suitably bounded and spaced p-colorings) a parameterized eventually operator by an LTL formula, while still ensuring a bound on the satisfaction of the parameterized operator. So, we begin by transforming the original arena $\mathcal{A}$ into an arena $\mathcal{A}_b$ in which Player 0 produces p-colorings of the plays of the original arena, i.e., $\mathcal{A}_b$ will consist of two disjoint copies of $\mathcal{A}$, one labeled by p, the other one not. Assume a play is in vertex v in one component. Then, the player whose turn it is at v chooses a successor v' of v and Player 0 picks a component. The play then continues in this component's vertex v'. We split this into two sequential moves: first, the player whose turn it is chooses a successor

[11] Alternatively, one could encode the arena into the winning condition φ, which also entails some technical difficulties.

and then Player 0 chooses the component. Thus, we have to introduce a new choice vertex for every edge of $\mathcal{A}$ which allows Player 0 to choose the component.

Formally, given an arena $\mathcal{A} = (V, V_0, V_1, E, \ell)$, we define the extended arena $\mathcal{A}_b = (V', V_0', V_1', E', \ell')$ by

- $V' = V \times \{0,1\} \cup E$,
- $V_0' = V_0 \times \{0,1\} \cup E$,
- $V_1' = V_1 \times \{0,1\}$,
- $E' = \{((v,0),e), ((v,1),e), (e,(v',0)), (e,(v',1)) \mid e = (v,v') \in E\}$, and
- $\ell'(e) = \emptyset$ for every $e \in E$ and $\ell'(v,b) = \begin{cases} \ell(v) \cup \{p\} & \text{if } b = 0, \\ \ell(v) & \text{if } b = 1. \end{cases}$

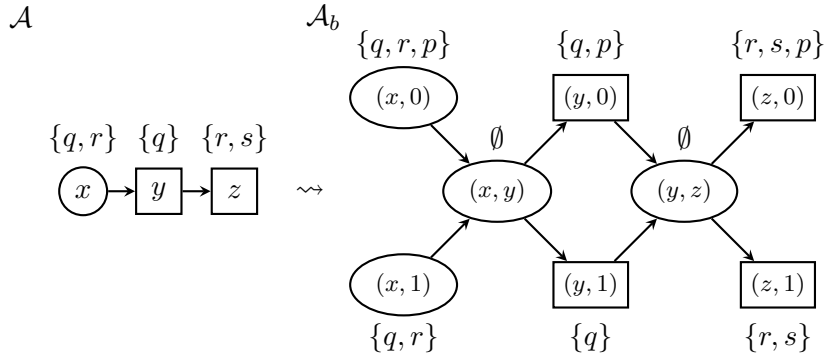

Figure 3.3: The construction of $\mathcal{A}_b$, (x,y) and (y,z) are choice vertices

For every vertex v of $\mathcal{A}_b$ we have two copies $(v,0)$ and $(v,1)$ in $\mathcal{A}$ and every edge e in $\mathcal{A}$ is turned into a choice vertex named e in $\mathcal{A}_b$. The construction is illustrated in Figure 3.3: instead of moving the token from v to v' in $\mathcal{A}$, the token in $\mathcal{A}_b$ is moved from (v,b) for some $b \in \{0,1\}$ to the choice vertex (v,v') from which Player 0 then can either move the token to $(v',0)$ or to $(v',1)$. Thus, a path through $\mathcal{A}_b$ has the form $(\rho_0, b_0)e_0(\rho_1, b_1)e_1(\rho_2, b_2)\cdots$ where $\rho_0\rho_1\rho_2\cdots$ is a path through $\mathcal{A}$, $e_n = (\rho_n, \rho_{n+1})$, and the b_n are in $\{0,1\}$. Also, we have $|\mathcal{A}_b| \in \mathcal{O}(|\mathcal{A}|^2)$.

The definition of $\mathcal{A}_b$ necessitates a modification of the game's semantics: only every other vertex is significant when it comes to determining the winner of a play in $\mathcal{A}_b$, the choice vertices have to be ignored. This motivates blinking semantics for PLTL games. Let $\mathcal{G} = (\mathcal{A}, v_0, \varphi)$ be a PLTL game and let $\rho = \rho_0\rho_1\rho_2\cdots$ be a play. Player 0 wins ρ under blinking semantics with respect to α, if $(\mathrm{tr}(\rho_0\rho_2\rho_4\cdots), \alpha) \models \varphi$. Analogously, Player 1 wins ρ under blinking semantics with respect to α, if $(\mathrm{tr}(\rho_0\rho_2\rho_4\cdots), \alpha) \not\models \varphi$. The

notions of winning strategies and winning $\mathcal{G}$ under blinking semantics with respect to α are defined as for games with standard semantics.

Alternatively, we could avoid using blinking semantics by rewriting an LTL formula φ to a formula φ_b such that $w_0 w_1 w_2 \cdots \models \psi$ if and only if $w_0\{o\}w_1\{o\}w_2\{o\}\cdots \models \varphi_b$ for some fresh proposition o. For unparameterized operators, this is straightforward since we mark the odd positions by o. For example, for atomic propositions p and q, $\mathbf{X}p$ is rewritten to $\mathbf{XX}p$ and $p\,\mathbf{U}\,q$ is rewritten to $(p \vee o)\,\mathbf{U}\,(q \wedge \mathbf{X}o)$. However, to extend this transformation to parameterized operators, we have to adjust the variable valuations as well. If $(w_0 w_1 w_2 \cdots, \alpha) \models \mathbf{F}_{\leq x} p$, then we can only conclude $(w_0\{o\}w_1\{o\}w_2\{o\}\cdots, \beta) \models \mathbf{F}_{\leq x} p$ if β satisfies $\beta(x) \geq 2\alpha(x)$. Furthermore, variable valuations mapping x to an odd value are useless in the padded words, since the bounds are always satisfied by an even number of steps. To save ourselves from dealing with these nuisances, we accept the notational overhead of dealing with two semantics for PLTL games.

Finite-state determinacy of LTL games under blinking semantics (and thus also finite-state determinacy of PLTL games under blinking semantics with respect to a fixed variable valuation) can be proven analogously to the case for LTL games under standard semantics.

Lemma 3.30. LTL *games under blinking semantics are determined with uniform finite-state strategies of size* $2^{2^{\mathcal{O}(|\varphi|)}}$ *and determining the winner is* **2ExpTime**-*complete.*

Proof. Let $\mathcal{G} = (\mathcal{A}, v_0, \varphi)$ be an LTL game under blinking semantics and let $\mathfrak{A} = (Q, 2^P, q_0, \delta, \Omega)$ be a deterministic parity automaton that recognizes the language of models of φ as in Theorem 3.10. By introducing new states we can turn $\mathfrak{A}$ into a deterministic parity automaton $\mathfrak{A}'$ recognizing the language

$$\{w \in \left(2^P\right)^\omega \mid w_0 w_2 w_4 \cdots \models \varphi\} \ .$$

Formally, we define $\mathfrak{A}' = (Q \times \{0,1\}, 2^P, (q_0, 0), \delta', \Omega')$ where

$$\delta'((q,i),a) = \begin{cases} (\delta(q,a),1) & \text{if } i = 0, \\ (q,0) & \text{if } i = 1, \end{cases}$$

and $\Omega'(q,i) = \Omega(q)$ for $i \in \{0,1\}$. We have $|\mathfrak{A}'| \in 2^{2^{\mathcal{O}(|\varphi|)}}$.

The automaton $\mathfrak{A}'$ recognizes exactly those traces that satisfy φ under blinking semantics. Hence, we can apply Corollary 3.26 to reduce $\mathcal{G}$ to a parity game via a memory structure of size $|\mathfrak{A}'|$. Thus, we have shown that there is a finite-state winning strategy of doubly-exponential size for $\mathcal{G}$. Furthermore, the parity game can be solved in doubly-exponential time in the size of $\mathcal{G}$ due to Theorem 2.22, which also solves the original LTL game under blinking semantics.

Finally, **2ExpTime**-hardness follows directly from **2ExpTime**-hardness of solving LTL games: an LTL game under standard semantics can easily be

turned into an equivalent LTL game under blinking semantics by subdividing every edge of the arena by adding a new vertex. □

Since a PLTL game $\mathcal{G} = (\mathcal{A}, v_0, \varphi)$ under blinking semantics with respect to a fixed variable valuation α is nothing more than an LTL game with winning condition φ_α (see Lemma 3.16) in the same arena, Lemma 3.30 yields the following determinacy result for PLTL games under blinking semantics with respect to a fixed variable valuation.

Corollary 3.31. PLTL *games under blinking semantics with respect to a fixed variable valuation α are determined with finite-state strategies.*

Now, we can state the "equivalence" of a $\text{PLTL}_\mathbf{F}$ game $(\mathcal{A}, v_0, \varphi)$ and its counterpart in $\mathcal{A}_b$ with blinking semantics obtained by replacing every parameterized eventually operator. Analogously to the **2Exptime**-membership proof of the PROMPT–LTL realizability problem, the following proof relies on the existence of finite-state winning strategies which necessarily produce only k-bounded plays for some fixed k, if alt_p is part of the winning condition. As already mentioned in the beginning of this subsection, we state a more general result for full PLTL. This turns out to be useful when we solve optimization problems for unipolar PLTL games in Section 3.3 and when we want to determine the projection of $\mathcal{W}_0(\mathcal{G})$.

Lemma 3.32. *Let $\mathcal{G} = (\mathcal{A}, v_0, \varphi)$ be a* PLTL *game, let $X \subseteq \text{var}_\mathbf{F}(\varphi)$, and let $\mathcal{G}' = (\mathcal{A}_b, (v_0, 0), \varphi_X \wedge alt_p)$.*

i. If Player 0 wins $\mathcal{G}$ with respect to a variable valuation α, then she also wins $\mathcal{G}'$ under blinking semantics with respect to α.

ii. If Player 0 wins $\mathcal{G}'$ under blinking semantics with respect to a variable valuation α, then there exists a variable valuation β with $\beta(z) = \alpha(z)$ for every $x \in X \cup \text{var}_\mathbf{G}(\varphi)$ such that she wins $\mathcal{G}$ with respect to β.

Before we prove the lemma, let us mention that this suffices to prove Theorem 3.29: let $\mathcal{G} = (\mathcal{A}, v_0, \varphi)$ be a $\text{PLTL}_\mathbf{F}$ game and consider the case $X = \emptyset$. Then, φ_X is an LTL formula and Player 0 wins $\mathcal{G}$ with respect to some variable valuation α if and only if she wins the LTL game $\mathcal{G}' = (\mathcal{A}_b, (v_0, 0), \varphi_X \wedge alt_p)$ under blinking semantics. As $\mathcal{G}'$ is only polynomially larger than $\mathcal{G}$ and as LTL games under blinking semantics can be solved in doubly-exponential time, we have shown **2Exptime**-membership of the emptiness problem for $\mathcal{W}_0(\mathcal{G})$, if $\mathcal{G}$ is a $\text{PLTL}_\mathbf{F}$ game.

Now, let us turn to the proof of Lemma 3.32.

Proof. i.) Let σ be a winning strategy for Player 0 for $\mathcal{G}$ with respect to α and define $k = \max_{x \in \text{var}_\mathbf{F}(\varphi) \setminus X} \alpha(x)$. We turn σ into a strategy σ' for $\mathcal{G}'$ that mimics the behavior of σ at vertices (v, b) and colors the play in alternating p-blocks of length k at the choice vertices. Hence, the trace of

the resulting play (without choice vertices) in $\mathcal{A}_b$ is a k-spaced p-coloring of the trace of a play that is consistent with σ, which allows us to apply Lemma 3.28(i) to show that σ' is winning under blinking semantics with respect to α. Formally, we define

$$\sigma'((\rho_0, b_0)(\rho_0, \rho_1) \cdots (\rho_{n-1}, \rho_n)(\rho_n, b_n)) = (\rho_n, \sigma(\rho_0 \cdots \rho_n))$$

if $(\rho_n, b_n) \in V_0'$, which implies $\rho_n \in V_0$. Thus, at a non-choice vertex, Player 0 mimics the behavior of σ. At choice vertices, she alternates between the two copies of the arena every k steps, i.e.,

$$\sigma'((\rho_0, b_0)(\rho_0, \rho_1) \cdots (\rho_n, b_n)(\rho_n, \rho_{n+1})) = \begin{cases} (\rho_{n+1}, 0) & \text{if } n \bmod 2k < k, \\ (\rho_{n+1}, 1) & \text{if } n \bmod 2k \geq k. \end{cases}$$

Let $\rho = \rho_0\rho_1\rho_2 \cdots$ be a play in $\mathcal{A}_b$ that is consistent with σ' and let

$$\rho' = \rho_0\rho_2\rho_4 \cdots = (v_0, b_0)(v_1, b_1)(v_2, b_2) \cdots \ .$$

By definition of σ', the sequence $v_0v_1v_2 \cdots$ is a play in $\mathcal{A}$ that is consistent with σ and thus winning for Player 0 with respect to α, i.e., we have $(\mathrm{tr}(v_0v_1v_2 \cdots), \alpha) \models \varphi$. Furthermore, $\mathrm{tr}(\rho')$ is a k-spaced p-coloring of $\mathrm{tr}(v_0v_1v_2 \cdots)$. Hence, $(\mathrm{tr}(\rho'), \alpha) \models \varphi_X \wedge \mathit{alt}_p$ due to Lemma 3.28(i). Thus, σ' is a winning strategy for $(\mathcal{A}_b, (v_0, 0), \varphi_X \wedge \mathit{alt}_p)$ under blinking semantics with respect to α.

ii.) Assume that Player 0 wins $(\mathcal{A}_b, (v_0, 0), \varphi_X \wedge \mathit{alt}_p)$ under blinking semantics with respect to α. Then, due to Corollary 3.31, she also has a finite-state winning strategy σ' implemented by some memory structure $\mathcal{M}' = (M', \mathrm{init}', \mathrm{upd}')$ and some next-move function nxt'. Since such a finite-state strategy only generates plays that are k-bounded p-colorings, for some k that only depends on $|M|$ and $|\mathcal{A}_b|$, Lemma 3.28(ii) is applicable to the plays that are consistent with σ'. We construct a strategy σ for $\mathcal{G}$ by simulating σ' such that each play in $\mathcal{A}$ that is consistent with σ has a k-bounded p-coloring in $\mathcal{A}_b$ that is consistent with σ'. This suffices to show that σ is winning with respect to a variable valuation β as required.

Let us sketch the intuition behind the simulation. Assume the token in $\mathcal{A}$ is in the initial vertex v_0 and the token in $\mathcal{A}_b$ in the initial vertex $(v_0, 0)$ as well. If it is Player 0's turn in v_0, then it is also her turn in $(v_0, 0)$ and σ' prescribes to move the token to some choice vertex (v_0, v_1). Since all choice vertices belong to Player 0, there is some bit b_1 such that σ prescribes to move to (v_1, b_1). She mimics these moves in $\mathcal{A}$ by moving to v_1 directly. On the other hand, if it is Player 1's turn at v_0, he moves to some successor v_1. This move is simulated in $\mathcal{A}_b$ by letting him move the token from $(v_0, 0)$ to (v_0, v_1). At this choice vertex, σ' again prescribes Player 0 to move the token to a vertex (v_1, b_1) for some bit b_1. Hence, the token in $\mathcal{A}$ is at vertex v_1 and the token in $\mathcal{A}_b$ at vertex (v_1, b_1) for some $b_1 \in \{0, 1\}$. Thus, we simulate

a move in $\mathcal{A}$ by two moves in $\mathcal{A}_b$ and if its Player 0's turn, we mimic σ' to pick a successor in $\mathcal{A}$.

Since σ' is implemented by $\mathcal{M}'$, it suffices to keep track of the last vertex of the simulated play – which is never a choice vertex – and the memory state for the simulated play. Hence, we transform $\mathcal{M}'$ into a memory structure $\mathcal{M} = (M, \text{init}, \text{upd})$ for $\mathcal{A}$ with $M = (V \times \{0,1\}) \times M'$,

$$\text{init}(v) = ((v,0), \text{init}'(v,0)) \ ,$$

and

$$\text{upd}(((v,b),m),v') = (\text{nxt}'(e,m'), \text{upd}'(m', \text{nxt}'(e,m')))$$

where $e = (v,v')$ and $m' = \text{upd}'(m,e)$.

Let w be a play prefix of a play in $\mathcal{A}$. The memory state $\text{upd}^*(w) = ((v,b),m)$ encodes the following information: the simulated play w' in $\mathcal{A}_b$ ends in (v,b), where $v = \text{Lst}(w)$, and we have $\text{upd}^*(w') = m$. Hence, it contains all information necessary to apply the next-move function nxt' to mimic σ'. Let us explain the simulation in some more detail: the initialization function maps v_0 to the initial vertex of $\mathcal{G}'$ and the initial memory state for this vertex. Since we only consider initialized games, all other vertices can be ignored when defining the initialization function. Now, suppose the players have constructed a play prefix w with $\text{upd}^*(w) = ((v,b),m)$ and the token is moved to v'. The new memory state $\text{upd}^*(wv') = \text{upd}(((v,b),m),v')$ is obtained as follows. In the simulated play in $\mathcal{A}_b$, the token is moved from (v,b) to the choice vertex $e = (v,v')$ and the memory is updated to $m' = \text{upd}'(m,e)$. Then, σ' prescribes to move the token to $\text{nxt}(e,m')$. This is the first component of the updated memory state as defined above. Furthermore, when moving the token to $\text{nxt}'(e,m')$, the memory $\mathcal{M}'$ is updated from m' to $\text{upd}'(m', \text{nxt}'(e,m'))$, which is the second component of the updated memory state for $\mathcal{A}$.

Finally, we define a next-move function $\text{nxt}\colon V_0 \times M \to V$ for Player 0 in $\mathcal{A}$ by

$$\text{nxt}(v,((v',b),m)) = \begin{cases} v'' & \text{if } v = v' \text{ and } \text{nxt}'((v',b),m) = (v',v''), \\ \overline{v} & \text{otherwise, for some } \overline{v} \in V \text{ with } (v,\overline{v}) \in E. \end{cases}$$

By definition of $\mathcal{M}$, the second case of the definition is never invoked, since $\text{upd}^*(wv) = ((v',b),m)$ always satisfies $v = v'$. As explained above, the next-move function for $\mathcal{A}$ mimics the behavior of σ' for the simulated play in $\mathcal{A}_b$, which is sufficiently specified by its last vertex (v,b) and its memory state m.

It remains to show that the strategy σ implemented by $\mathcal{M}$ and nxt is indeed a winning strategy for Player 0 for $(\mathcal{A}, v_0, \varphi)$ with respect to a variable valuation β that coincides with α on all variables in $X \cup \text{var}_{\mathbf{G}}(\varphi)$. We begin by relating plays consistent with σ to plays that are consistent with σ', i.e., we prove that the simulation is working correctly.

Lemma 3.33. *Let $\rho_0\rho_1\rho_2\cdots$ be a play in $\mathcal{A}$ that is consistent with σ. Then, there exist bits $b_0, b_1, b_2, \cdots$ such that $(\rho_0, b_0)(\rho_0, \rho_1)(\rho_1, b_1)(\rho_1, \rho_2)(\rho_2, b_2)\cdots$ is a play in $\mathcal{A}_b$ that is consistent with σ'.*

Proof. To prove the claim by induction over $\rho_0 \cdots \rho_n$, we need to strengthen the statement by adding the following condition: if

$$\mathrm{upd}'^*((\rho_0, b_0)(\rho_0, \rho_1)(\rho_1, b_1)\cdots(\rho_{n-1}, \rho_n)(\rho_n, b_n)) = m \ ,$$

then $\mathrm{upd}^*(\rho_0 \cdots \rho_n) = ((\rho_n, b_n), m)$.

As a play in $\mathcal{A}$ always starts in v_0, both claims are true for $\rho_0 = v_0$ with $b_0 = 0$, since the initial vertex of $\mathcal{A}_b$ is $(v_0, 0)$ and

$$\mathrm{upd}^*(v_0) = \mathrm{init}(v_0) = ((v_0, 0), \mathrm{init}'(v_0, 0)) = ((v_0, 0), \mathrm{upd}'^*(v_0, 0)) \ .$$

Now, let $\rho_0 \cdots \rho_n\rho_{n+1}$ be a play prefix in $\mathcal{A}$ that is consistent with σ. The induction hypothesis gives us bits $b_0, \ldots, b_n$ such that

$$(\rho_0, b_0)(\rho_0, \rho_1)(\rho_1, b_1)\cdots(\rho_{n-1}, \rho_n)(\rho_n, b_n)$$

is a play in $\mathcal{A}_b$ that is consistent with σ' and

$$\mathrm{upd}^*(\rho_0 \cdots \rho_n) = ((\rho_n, b_n), m) \ , \tag{3.2}$$

where $m = \mathrm{upd}'^*((\rho_0, b_0)(\rho_0, \rho_1)(\rho_1, b_1)\cdots(\rho_{n-1}, \rho_n)(\rho_n, b_n))$. We consider two cases depending on whose turn it is at ρ_n.

If $\rho_n \in V_0$, which implies $(\rho_n, b_n) \in V_0'$, let

$$\mathrm{nxt}'((\rho_n, b_n), m) = (\rho_n, v) = e \ , \tag{3.3}$$

$m' := \mathrm{upd}'(m, e)$, and $(v, b) = \mathrm{nxt}'(e, m')$. Then,

$$(\rho_0, b_0)(\rho_0, \rho_1)(\rho_1, b_1)\cdots(\rho_{n-1}, \rho_n)(\rho_n, b_n)e(v, b)$$

is consistent with σ', that is we define $b_{n+1} = b$. We have to show $v = \rho_{n+1}$: as $\rho_0 \cdots \rho_n\rho_{n+1}$ is consistent with σ, we have

$$\rho_{n+1} = \mathrm{nxt}(\rho_n, \mathrm{upd}^*(\rho_0 \cdots \rho_n)) = \mathrm{nxt}(\rho_n, ((\rho_n, b_n), m))$$

by (3.2). By definition of the next-move function, $\mathrm{nxt}(\rho_n, ((\rho_n, b_n), m)) = \rho_{n+1}$ implies $\mathrm{nxt}'((\rho_n, b_n), m) = (\rho_n, \rho_{n+1})$. Applying (3.3) yields $v = \rho_{n+1}$. It remains to prove $\mathrm{upd}(((\rho_n, b_n), m), \rho_{n+1}) = ((\rho_{n+1}, b_{n+1}), m'')$, where we write m'' for $\mathrm{upd}'(m', (\rho_{n+1}, b_{n+1}))$. We have

$$\begin{aligned}
&\mathrm{upd}(((\rho_n, b_n), m), \rho_{n+1}) \\
=&(\mathrm{nxt}'(e, \mathrm{upd}'(m, e)), \mathrm{upd}'(\mathrm{upd}'(m, e), \mathrm{nxt}'(e, \mathrm{upd}'(m, e)))) \\
=&(\mathrm{nxt}'(e, m'), \mathrm{upd}'(m', \mathrm{nxt}'(e, m'))) \\
=&((v, b), \mathrm{upd}'(m', (v, b))) \\
=&((\rho_{n+1}, b_{n+1}), m'') \ .
\end{aligned}$$

On the other hand, if $\rho_n \in V_1$, let $e = (\rho_n, \rho_{n+1})$, $m' = \mathrm{upd}'(m, e)$, and $(\rho_{n+1}, b) = \mathrm{nxt}'(e, m')$. Then,

$$(\rho_0, b_0)(\rho_0, \rho_1)(\rho_1, b_1) \cdots (\rho_{n-1}, \rho_n)(\rho_n, b_n)e(\rho_{n+1}, b)$$

is consistent with σ', that is we define $b_{n+1} := b$.

It remains to show $\mathrm{upd}(((\rho_n, b_n), m), \rho_{n+1}) = ((\rho_{n+1}, b_{n+1}), m'')$ where $m'' = \mathrm{upd}'(m', (\rho_{n+1}, b_{n+1}))$. We skip the proof here, since the reasoning is analogous to the one for $\rho_n \in V_0$. □

Now, we are able to finish the proof of Lemma 3.32(ii). Let $\rho = \rho_0\rho_1\rho_2 \cdots$ be a play that is consistent with σ. Applying the lemma we have just proved yields bits $b_0, b_1, b_2 \cdots \in \{0, 1\}$ such that the play

$$\rho' = (\rho_0, b_0)(\rho_0, \rho_1)(\rho_1, b_1)(\rho_1, \rho_2)(\rho_2, b_2) \cdots$$

is consistent with σ'. Hence, the trace of $\rho'' = (\rho_0, b_0)(\rho_1, b_1)(\rho_2, b_2) \cdots$ satisfies $\varphi_X \wedge \mathit{alt}_p$ with respect to α. We show that $\mathrm{tr}(\rho'')$ is k-bounded, where $k = |V| \cdot |M| + 1$. This suffices to finish the proof: let $\beta(x) = 2k$ for $x \in \mathrm{var}_{\mathbf{F}}(\varphi) \setminus X$ and $\beta(z) = \alpha(z)$ for $z \in X \cup \mathrm{var}_{\mathbf{G}}(\varphi)$. Then, we can apply Lemma 3.28(ii) and obtain $(\mathrm{tr}(\rho), 0, \beta) \models \varphi$, as $\mathrm{tr}(\rho'')$ is a k-bounded p-coloring of $\mathrm{tr}(\rho)$. Hence, σ is indeed a winning strategy for Player 0 for $(\mathcal{A}, v_0, \varphi)$ with respect to β.

Towards a contradiction assume that ρ'' is not k-bounded. Then, there exist consecutive change points i and j such that $j - i \geq k + 1$. Then, there also exist $i \leq i' < j' < j$ such that $\rho_{i'} = \rho_{j'}$ and

$$\mathrm{upd}'^*((\rho_0, b_0) \cdots (\rho_{i'}, b_{i'})) = \mathrm{upd}'^*((\rho_0, b_0) \cdots (\rho_{j'}, b_{j'})) \ ,$$

i.e., the last vertices of both play prefixes are equal and the memory states after both play prefixes are equal, too. Hence, the play

$$\rho^* = (\rho_0, b_0) \cdots (\rho_{i'-1}, b_{i'-1})\big[(\rho_{i'}, b_{i'}) \cdots (\rho_{j'-1}, b_{j'-1})(\rho_{j'-1}, \rho_{j'})\big]^\omega \ ,$$

obtained by traversing the cycle between $(\rho_{i'}, b_{i'})$ and $(\rho_{j'}, b_{j'})$ infinitely often, is consistent with σ', since the memory states reached at the beginning and the end of the loop are the same. Remember that the bits do not change between i and j. Thus, ρ^* has only finitely many change points and does not satisfy alt_p under blinking semantics. This contradicts the fact that σ' is a winning strategy for $(\mathcal{A}_b, (v_0, 0), \varphi_X \wedge \mathit{alt}_p)$ under blinking semantics with respect to α. □

Let $\mathcal{G} = (\mathcal{A}, v_0, \varphi)$ be a PLTL game. Lemma 3.30 allows us to bound the size of a finite-state winning strategy for $(\mathcal{A}_b, (v_0, 0), \varphi_\emptyset \wedge \mathit{alt}_p)$, which in turn bounds the values of a variable valuation that is winning for Player 0 for the game $\mathcal{G}$.

Theorem 3.34. *Let $\mathcal{G}$ be a* $\text{PLTL}_{\mathbf{F}}$ *game. If $\mathcal{W}_0(\mathcal{G}) \neq \emptyset$, then there exists a $k \in 2^{2^{\mathcal{O}(|\mathcal{G}|)}}$ such that Player 0 wins $\mathcal{G}$ with respect to the variable valuation that maps every variable to k.*

We shall see in Subsection 3.3.3 that the doubly-exponential upper bound is asymptotically optimal. The previous theorem can also be formulated in terms of $\text{PLTL}_{\mathbf{G}}$ games: let $\mathcal{G} = (\mathcal{A}, v_0, \varphi)$ be a $\text{PLTL}_{\mathbf{G}}$ game. There exists a $k \in 2^{2^{\mathcal{O}(|\mathcal{G}|)}}$ such that if Player 0 wins $\mathcal{G}$ with respect to the variable valuation mapping every variable to k, then she wins $\mathcal{G}$ with respect to every variable valuation. This can be proven by considering the dual game of $\mathcal{G}$, which is a $\text{PLTL}_{\mathbf{F}}$ game, and applying Theorem 3.34.

Let us conclude by briefly discussing how to apply the alternating color technique to construct the projection of $\mathcal{W}_0(\mathcal{G})$ to the variables in $\text{var}_{\mathbf{G}}(\varphi)$ to show that the projection is a semilinear set. Recall that we consider a PLTL game $\mathcal{G}$ with winning condition φ with $\text{var}_{\mathbf{F}}(\varphi) = \{z_0, \dots z_{k-1}\}$ and $\text{var}_{\mathbf{G}}(\varphi) = \{z_k, \dots, z_{k+k'-1}\}$, and that we view a variable valuation α as vector in $\mathbb{N}^{k+k'}$ whose j-th component is $\alpha(z_j)$. Now, let $\varphi_\emptyset$ be the formula obtained by inductively replacing every subformula $\mathbf{F}_{\leq x}\psi$ as described in Subsection 3.2.1 and consider the $\text{PLTL}_{\mathbf{G}}$ game $\mathcal{G}' = (\mathcal{A}_b, (v_0, 0), \varphi_\emptyset \wedge alt_p)$ under blinking semantics. Lemma 3.32 shows that Player 0 wins $\mathcal{G}'$ under blinking semantics with respect to the variable valuation $(n_k, \dots, n_{k+k'-1})$ if and only if $(n_1, \dots, n_{k-1}, n_k, \dots, n_{k+k'-1}) \in \mathcal{W}_0(\mathcal{G})$ for some $n_0, \dots, n_{k-1} \in \mathbb{N}$. Since the monotonicity properties hold for games under blinking semantics as well, we can apply Theorem 3.23 to show that the projection of $\mathcal{W}_0(\mathcal{G})$ to the variables in $\text{var}_{\mathbf{G}}(\varphi)$ is semilinear.

3.2.3 Solving PLTL Games

The doubly-exponential time algorithm for the emptiness problem for $\text{PLTL}_{\mathbf{F}}$ games allows us to solve the emptiness, finiteness, and the universality problem for full PLTL as well. All three algorithms rely on the monotonicity of the parameterized operators and on reductions to the emptiness problem for $\text{PLTL}_{\mathbf{F}}$ games.

Theorem 3.35. *The emptiness, the finiteness, and the universality problem for* PLTL *games are* **2Exptime***-complete.*

Proof. We begin by showing **2Exptime**-membership for all three problems. Let $\mathcal{G} = (\mathcal{A}, v_0, \varphi)$ be a PLTL game. Due to duality (see Lemma 3.19(ii)), it suffices to consider $i = 0$.

Emptiness of $\mathcal{W}_0(\mathcal{G})$: Let $\varphi_{\mathbf{F}}$ be the formula obtained from φ by inductively replacing every subformula $\mathbf{G}_{\leq y}\psi$ by ψ, and let $\mathcal{G}_{\mathbf{F}} = (\mathcal{A}, v_0, \varphi_{\mathbf{F}})$, which is a $\text{PLTL}_{\mathbf{F}}$ game. Applying the monotonicity of $\mathbf{G}_{\leq y}$, we obtain that $\mathcal{W}_0(\mathcal{G})$ is empty if and only if $\mathcal{W}_0(\mathcal{G}_{\mathbf{F}})$ is empty.

The latter problem can be decided in doubly-exponential time by Theorem 3.29. Hence, the emptiness of $\mathcal{W}_0(\mathcal{G})$ can be decided in doubly-exponential time as well, since we have $|\varphi_{\mathbf{F}}| \leq |\varphi|$.

Universality of $\mathcal{W}_0(\mathcal{G})$: Applying first complementarity and then duality as stated in Lemma 3.19, we have that $\mathcal{W}_0(\mathcal{G})$ is universal if and only if $\mathcal{W}_1(\mathcal{G})$ is empty, which is the case if and only if $\mathcal{W}_0(\overline{\mathcal{G}})$ is empty. The latter problem is decidable in doubly-exponential time as shown above.

Finiteness of $\mathcal{W}_0(\mathcal{G})$: If $\mathrm{var}_{\mathbf{F}}(\varphi) \neq \emptyset$, then $\mathcal{W}_0(\mathcal{G})$ is infinite, if and only if it is non-empty, due to monotonicity of $\mathbf{F}_{\leq x}$. The emptiness of $\mathcal{W}_0(\mathcal{G})$ can be decided in doubly-exponential time as discussed above.

If $\mathrm{var}_{\mathbf{F}}(\varphi) = \emptyset$, then $\mathcal{G}$ is a $\mathrm{PLTL}_{\mathbf{G}}$ game. We assume that φ has at least one parameterized temporal operator, since the problem is trivial otherwise. Then, the set $\mathcal{W}_0(\mathcal{G})$ is infinite if and only if there is a variable $y \in \mathrm{var}_{\mathbf{G}}(\varphi)$ that is mapped to infinitely many values by the valuations in $\mathcal{W}_0(\mathcal{G})$. By downwards-closure, we can assume that all other variables are mapped to zero. Furthermore, y is mapped to infinitely many values if and only if it is mapped to all possible values, again by downwards-closure. To combine this, we define φ_y to be the formula obtained from φ by inductively replacing every subformula $\mathbf{G}_{\leq y'}\psi$ for $y' \neq y$ by ψ and define $\mathcal{G}_y = (\mathcal{A}, v_0, \varphi_y)$. Then, $\mathcal{W}_0(\mathcal{G})$ is infinite, if and only if there exists some variable $y \in \mathrm{var}(\varphi)$ such that $\mathcal{W}_0(\mathcal{G}_y)$ is universal. So, deciding whether $\mathcal{W}_0(\mathcal{G})$ is infinite can be done in doubly-exponential time by solving $|\mathrm{var}(\varphi)|$ many universality problems for $\mathrm{PLTL}_{\mathbf{G}}$ games, which were discussed above.

It remains to show **2ExpTime**-hardness of all three problems, which is a simple consequence of the **2ExpTime**-hardness of determining the winner of an LTL game. Let $\mathcal{G} = (\mathcal{A}, v_0, \varphi)$ be an LTL game. The following statements are equivalent.

i. Player 0 wins $\mathcal{G}$.

ii. $\mathcal{W}_0(\mathcal{G})$ is non-empty.

iii. $\mathcal{W}_0(\mathcal{G})$ is universal.

iv. $\mathcal{W}_1(\mathcal{G})$ is finite.

The equivalence of the first two statements is by definition of $\mathcal{W}_0(\mathcal{G})$, the equivalence of the second and third statement is due to $\mathrm{var}(\varphi) = \emptyset$, and the equivalence of the last two statements is due to complementarity of $\mathcal{W}_0(\mathcal{G})$ and $\mathcal{W}_1(\mathcal{G})$ and the fact that φ is variable-free. Hence, all three problems are indeed **2ExpTime**-hard. □

All but the finiteness problem for $\mathrm{PLTL}_{\mathbf{G}}$ games only require the solution of a single LTL game under blinking semantics. Furthermore, all these LTL games are only polynomially larger than the original game.

3.3 Optimal Strategies for Games with PLTL Winning conditions

The algorithms for the emptiness, finiteness, and universality problem rely on the monotonicity of the parameterized operators, e.g., to check emptiness all parameterized always operator $\mathbf{G}_{\leq y}\psi$ are replaced by ψ, thereby trivializing the operator. It is more natural to view synthesis of PLTL specifications as optimization problem: which is the *best* variable valuation α such that Player 0 can win $\mathcal{G}$ with respect to α? For unipolar games, we consider two natural quality measures for a valuation α in a game with winning condition φ: the maximal parameter $\max_{z \in \mathrm{var}(\varphi)} \alpha(z)$ and the minimal parameter $\min_{z \in \mathrm{var}(\varphi)} \alpha(z)$. For a $\mathrm{PLTL}_{\mathbf{F}}$ game, Player 0 tries to minimize the waiting times. Hence, we are interested in minimizing the minimal or maximal parameter. Dually, for $\mathrm{PLTL}_{\mathbf{G}}$ games, we are interested in maximizing the quality measures. The remaining problems, i.e., maximizing the waiting times in a $\mathrm{PLTL}_{\mathbf{F}}$ game and minimizing the satisfaction time in a $\mathrm{PLTL}_{\mathbf{G}}$ game, are trivial due to upwards- respectively downwards-closure of the set of winning valuations.

The reason for only considering unipolar games is the undecidability result of Alur et al. formulated in Corollary 3.3: it is undecidable, for a given formula φ with $x \in \mathrm{var}_{\mathbf{F}}(\varphi)$ and $y \in \mathrm{var}_{\mathbf{G}}(\varphi)$, to determine whether there exists a $w \in (2^P)^\omega$ and a variable valuation α satisfying $\alpha(x) = \alpha(y)$ and $(w, \alpha) \models \varphi$. Furthermore, by construction of the formula φ in the reduction, we know that we have $\alpha(x) \geq \alpha(y)$. Now, assume we want to determine an optimal variable valuation, i.e., one that minimizes x and maximizes y. The best we can hope for is a variable valuation satisfying $\alpha(x) = \alpha(y)$. Thus, determining the maximal or minimal parameter value of an optimal valuation is computationally infeasible: if we could determine this value, then we could plug it into both x and y and check whether the resulting LTL formula is satisfiable. This is the case if and only if there is a w and a variable valuation α satisfying $\alpha(x) = \alpha(y)$ and $(w, \alpha) \models \varphi$.

The main result of this section states that all optimization problems for unipolar games can be solved in triply-exponential time[12]. Furthermore, in Subsection 3.3.3 we prove a doubly-exponential lower bound on the value of an optimal variable valuation in a unipolar PLTL game, thereby showing that the doubly-exponential upper bounds obtained in Subsection 3.2.1 are (almost) tight.

3.3.1 PLTL Optimization Problems

In this subsection, we introduce the PLTL optimization problems and present an algorithm with triply-exponential running time for solving them. As a

[12] In [Zim11], it is erroneously claimed that these problems can even be solved in doubly-exponential time. We explain the bug in the proof on page 73.

first step, we show that it suffices to consider games with winning conditions in PROMPT–LTL, and then we show how to solve PROMPT–LTL optimization problems in triply-exponential time by reductions to parity games. The reduction relies on the existence of "small" automata that recognize the language of traces that satisfy a formula with respect to a fixed variable valuation.

In the remainder of this section, we require all our winning conditions to have at least one parameterized operator, since the optimization problems are trivial otherwise. Again, we only consider Player 0, as one can dualize the game to obtain similar results for Player 1.

Theorem 3.36. *Let $\mathcal{G}_{\mathbf{F}}$ be a* $\mathrm{PLTL}_{\mathbf{F}}$ *game with winning condition $\varphi_{\mathbf{F}}$ and let $\mathcal{G}_{\mathbf{G}}$ be a* $\mathrm{PLTL}_{\mathbf{G}}$ *game with winning condition $\varphi_{\mathbf{G}}$. The following values (and winning strategies realizing them) can be computed in triply-exponential time.*

i. $\min_{\alpha \in \mathcal{W}_0(\mathcal{G}_{\mathbf{F}})} \min_{x \in \mathrm{var}(\varphi_{\mathbf{F}})} \alpha(x)$.

ii. $\min_{\alpha \in \mathcal{W}_0(\mathcal{G}_{\mathbf{F}})} \max_{x \in \mathrm{var}(\varphi_{\mathbf{F}})} \alpha(x)$.

iii. $\max_{\alpha \in \mathcal{W}_0(\mathcal{G}_{\mathbf{G}})} \max_{y \in \mathrm{var}(\varphi_{\mathbf{G}})} \alpha(y)$.

iv. $\max_{\alpha \in \mathcal{W}_0(\mathcal{G}_{\mathbf{G}})} \min_{y \in \mathrm{var}(\varphi_{\mathbf{G}})} \alpha(y)$.

A special case of the $\mathrm{PLTL}_{\mathbf{F}}$ optimization problems is the PROMPT–LTL optimization problem. Due to our non-triviality requirement, the winning condition in a PROMPT–LTL game has exactly one variable x. Hence, the inner maximization or minimization becomes trivial and the problem asks to determine $\min_{\alpha \in \mathcal{W}_0(\mathcal{G})} \alpha(x)$ and a winning strategy for Player 0 realizing this value. Dually, in a $\mathrm{PLTL}_{\mathbf{G}}$ optimization problem with a single variable y, the inner maximization or minimization becomes trivial and the problem asks to determine $\max_{\alpha \in \mathcal{W}_0(\mathcal{G})} \alpha(y)$ and a winning strategy for Player 0 realizing this value.

Due to duality, there is a tight connection between PROMPT–LTL optimization problems and $\mathrm{PLTL}_{\mathbf{G}}$ optimization problems with a single variable: let $\mathcal{G}$ be a $\mathrm{PLTL}_{\mathbf{G}}$ game with winning condition φ with $\mathrm{var}(\varphi) = \{y\}$. Then, we have

$$\max_{\alpha \in \mathcal{W}_0(\mathcal{G})} \alpha(y) = \max_{\alpha \in \mathcal{W}_1(\overline{\mathcal{G}})} \alpha(y) = \min_{\alpha \in \mathcal{W}_0(\overline{\mathcal{G}})} \alpha(y) + 1 \ ,$$

due to the closure properties and Lemma 3.19. Here, $\overline{\mathcal{G}}$ is a PROMPT–LTL game. Thus, to compute the optimal variable valuation in a $\mathrm{PLTL}_{\mathbf{G}}$ game with a single variable valuation, it suffices to solve a PROMPT–LTL optimization problem. We defer the computation of strategies realizing the optimal values in both types of games to the end of this subsection.

Let us begin by outlining the proof idea. First, we show that all four optimization problems can be reduced to PROMPT–LTL optimization problems. In this case, Corollary 3.44 yields a doubly-exponential (in the size of the game) upper bound on an optimal variable valuation. Then, using binary search, we determine the optimal value. To do this, we show that we can solve the membership problem for PLTL games in triply-exponential time in the size of the game, provided the variable valuation we are querying for is doubly-exponentially (again, in the size of the game) bounded. This is achieved by a reduction to a parity game using a deterministic automaton that recognizes the models of the winning condition with respect to the variable valuation we are interested in. As a corollary, we obtain a **2EXPTIME** algorithm for the membership problem for arbitrary formulae. Note that α is part of the input for this problem. This explains why we save one exponential compared to the algorithm mentioned above, whose input consists only of a game.

We begin by showing that all four problems mentioned in Theorem 3.36 can be reduced to optimization problems with a single variable. As we have shown above how to translate an optimization problem for a $\text{PLTL}_{\mathbf{G}}$ game with a single variable into a PROMPT–LTL optimization problem, it suffices to solve PROMPT–LTL optimization problems. The latter three reductions are simple applications of the monotonicity of the parameterized operators, while the first one requires an application of the alternating color technique.

i.) For each $x \in \text{var}(\varphi_{\mathbf{F}})$ we apply the alternating color technique (see Subsection 3.2.1) to construct the projection of $\mathcal{W}_0(\mathcal{G}_{\mathbf{F}})$ to the values of x: let $\mathcal{G}_{\mathbf{F}} = (\mathcal{A}, v_0, \varphi_{\mathbf{F}})$ and define $\mathcal{G}_x = (\mathcal{A}_b, (v_0, 0), (\varphi_{\mathbf{F}})_{\{x\}} \wedge alt_p)$ where $\mathcal{A}_b$ and the winning condition of $\mathcal{G}_x$ are defined as in Subsection 3.2.1. Applying both directions of Lemma 3.32 yields

$$\min_{\alpha \in \mathcal{W}_0(\mathcal{G}_{\mathbf{F}})} \min_{x \in \text{var}(\varphi_{\mathbf{F}})} \alpha(x) = \min_{x \in \text{var}(\varphi)} \min\{\alpha(x) \mid \text{Player 0 wins } \mathcal{G}_x \text{ under blinking semantics w.r.t. } \alpha\} \ .$$

Since $\text{var}((\varphi_{\mathbf{F}})_{\{x\}}) = \{x\}$, we have reduced the minimization problem to $|\text{var}(\varphi_{\mathbf{F}})|$ many PROMPT–LTL optimization problems, albeit under blinking semantics. We discuss the necessary adaptions to our proof, which is for the non-blinking case, below.

Furthermore, a strategy realizing the optimum on the right-hand side can be turned into a strategy realizing the optimum on the left-hand side using the construction presented in the proof of Lemma 3.32(ii) which turns a strategy for the expanded arena $\mathcal{A}_b$ into a strategy for the original arena $\mathcal{A}$.

ii.) This problem can directly be reduced to a PROMPT–LTL optimization problem: let $\varphi'_{\mathbf{F}}$ be the PROMPT–LTL formula obtained from $\varphi_{\mathbf{F}}$ by renaming each $x \in \text{var}(\varphi_{\mathbf{F}})$ to z and let $\mathcal{G}' = (\mathcal{A}, v_0, \varphi'_{\mathbf{F}})$, where $\mathcal{A}$ and v_0

are the arena and the initial vertex of $\mathcal{G}_{\mathbf{F}}$. Then,

$$\min_{\alpha \in \mathcal{W}_0(\mathcal{G}_{\mathbf{F}})} \max_{x \in \mathrm{var}(\varphi_{\mathbf{F}})} \alpha(x) = \min_{\alpha \in \mathcal{W}_0(\mathcal{G}')} \alpha(z) \ ,$$

due to upwards-closure of $\mathcal{W}_0(\mathcal{G}_{\mathbf{F}})$, and a strategy realizing the optimum on the right-hand side also realizes the optimum on the left-hand side.

iii.) For every $y \in \mathrm{var}(\varphi_{\mathbf{G}})$ let φ_y be obtained from $\varphi_{\mathbf{G}}$ by replacing every subformula $\mathbf{G}_{\leq y'}\psi$ with $y' \neq y$ by ψ and let $\mathcal{G}_y = (\mathcal{A}, v_0, \varphi_y)$, where $\mathcal{A}$ and v_0 are the arena and the initial vertex of $\mathcal{G}_{\mathbf{G}}$. Then, we have

$$\max_{\alpha \in \mathcal{W}_0(\mathcal{G}_{\mathbf{G}})} \max_{y \in \mathrm{var}(\varphi_{\mathbf{G}})} \alpha(y) = \max_{y \in \mathrm{var}(\varphi_{\mathbf{G}})} \max_{\alpha \in \mathcal{W}_0(\mathcal{G}_y)} \alpha(y) \ ,$$

due to downwards-closure of $\mathcal{W}_0(\mathcal{G}_{\mathbf{G}})$, and a strategy realizing the optimum on the right-hand side also realizes the optimum on the left-hand side.

iv.) Let $\varphi'_{\mathbf{G}}$ be obtained from $\varphi_{\mathbf{G}}$ by renaming every variable in $\varphi_{\mathbf{G}}$ to z and let $\mathcal{G}' = (\mathcal{A}, v_0, \varphi'_{\mathbf{G}})$, where $\mathcal{A}$ and v_0 are the arena and the initial vertex of $\mathcal{G}_{\mathbf{G}}$. Then,

$$\max_{\alpha \in \mathcal{W}_0(\mathcal{G}_{\mathbf{G}})} \min_{y \in \mathrm{var}(\varphi_{\mathbf{G}})} \alpha(y) = \max_{\alpha \in \mathcal{W}_0(\mathcal{G}')} \alpha(z) \ ,$$

due to downwards-closure of $\mathcal{W}_0(\mathcal{G}_{\mathbf{G}})$ and a strategy realizing the optimum on the right-hand side also realizes the optimum on the left-hand side.

All reductions increase the size of the arena at most quadratically and the size of the winning condition at most linearly. Furthermore, to minimize the minimal parameter value in a $\mathrm{PLTL}_{\mathbf{F}}$ game and to maximize the maximal parameter value in a $\mathrm{PLTL}_{\mathbf{G}}$ game, we have to solve $|\mathrm{var}(\varphi)|$ many PROMPT–LTL optimization problems (for the other two problems just one) to solve the original unipolar optimization problem with winning condition φ. Thus, due to the duality of unipolar optimization problems with a single variable discussed above, it remains to show that a PROMPT–LTL optimization problem can be solved in triply-exponential time.

So, let $\mathcal{G} = (\mathcal{A}, v_0, \varphi)$ be a PROMPT–LTL game with $\mathrm{var}(\varphi) = \{x\}$. If $\mathcal{W}_0(\mathcal{G}) \neq \emptyset$ (which can be checked in doubly-exponential time), then Theorem 3.34 yields a $k \in 2^{2^{\mathcal{O}(|\mathcal{G}|)}}$ such that $\min_{\alpha \in \mathcal{W}_0(\mathcal{G})} \alpha(x) \leq k$. Hence, we have a doubly-exponential upper bound on an optimal variable valuation.

In the following, we denote by α_n the variable valuation mapping x to n and every other variable to zero. Since φ only contains the variable x, the smallest $n < k$ such that $\alpha_n \in \mathcal{W}_0(\mathcal{G})$ is equal to $\min_{\alpha \in \mathcal{W}_0(\mathcal{G})} \alpha(x)$. As the number of such valuations α_n is doubly-exponential in $|\mathcal{G}|$, it suffices to show that $\alpha_n \in \mathcal{W}_0(\mathcal{G})$ can be decided in triply-exponential time in the size of $\mathcal{G}$, provided that $n < k$. This is achieved by a game reduction to a parity game.

Fix a variable valuation α_n and let $\mathfrak{P}_{\varphi,\alpha_n} = (Q, 2^P, q_0, \delta, \Omega)$ be a deterministic parity automaton recognizing the language

$$L(\mathfrak{P}_{\varphi,\alpha_n}) = \{w \in \left(2^P\right)^\omega \mid (w, \alpha_n) \models \varphi\} \ .$$

Note that the language is uniquely determined by φ and the value $\alpha_n(x) = n$, where x is the variable appearing in φ. Now, consider the parity game $\mathcal{G}_n = (\mathcal{A} \times \mathcal{M}, (v_0, \mathrm{init}(v_0)), \Omega')$ as defined in the proof of Lemma 2.17, i.e., $\mathcal{M} = (Q, \mathrm{init}, \delta)$ with $\mathrm{init}(v) = \delta(q_0, v)$ and $\Omega'(v, q) = \Omega(q)$. We have $\alpha_n \in \mathcal{W}_0(\mathcal{G})$ if and only if Player 0 wins $\mathcal{G}_n$ due to Corollary 3.26. However, to meet our time bounds, $\mathfrak{P}_{\varphi,\alpha_n}$ has to be of (at most) triply-exponential size (in $|\mathcal{G}|$) with (at most) doubly-exponentially many priorities, provided that we have $\alpha_n(x) < k$. If this is the case, then we can solve the parity game $\mathcal{G}_n$ in triply-exponential time due to Theorem 2.22. Furthermore, a winning strategy for the parity game associated to the minimal n can be turned in triply-exponential time into a finite-state winning strategy for the PROMPT–LTL game $\mathcal{G}$ which realizes the optimal value. This strategy is implemented by a memory induced by the automaton $\mathfrak{P}_{\varphi,\alpha_n}$ as explained above.

If the PROMPT–LTL game $\mathcal{G}$ has blinking semantics, then we have to adapt the construction slightly: instead of using an automaton $\mathfrak{P}_{\varphi,\alpha_n}$ for the language $\{w \in \left(2^P\right)^\omega \mid (w, \alpha_n) \models \varphi\}$ in the reduction, we turn $\mathfrak{P}_{\varphi,\alpha_n}$ into a deterministic parity automaton $\mathfrak{P}'_{\varphi,\alpha_n}$ that recognizes the language $\{w \in \left(2^P\right)^\omega \mid (w_0 w_2 w_4 \cdots, \alpha_n) \models \varphi\}$, which doubles the size, but does not change the number of priorities. Again, we denote by $\mathcal{G}'_n$ the parity game obtained in the reduction via memory induced by $\mathfrak{P}'_{\varphi,\alpha_n}$. Then, Player 0 wins $\mathcal{G}$ with respect to α_n under blinking semantics if and only if she wins $\mathcal{G}'_n$, which can again be determined in triply-exponential time.

Thus, the main step in the proof of Theorem 3.36 is to construct an automaton that has the following properties.

Lemma 3.37. *Let $n \leq k$. We can construct in triply-exponential time a deterministic parity automaton $\mathfrak{P}_{\varphi,\alpha_n}$ recognizing $\{w \in \left(2^P\right)^\omega \mid (w, \alpha_n) \models \varphi\}$ such that*

$$|\mathfrak{P}_{\varphi,\alpha_n}| \in 2^{2^{2^{\mathcal{O}(|\mathcal{G}|)}}}$$

and $\mathfrak{P}_{\varphi,\alpha_n}$ has at most $2^{2^{\mathcal{O}(|\mathcal{G}|)}}$ many priorities.

If we were able to prove an exponential upper bound on the value of a variable valuation in $\mathcal{W}_0(\mathcal{G})$ for a PROMPT–LTL game $\mathcal{G}$, then our technique would yield a doubly-exponential time algorithm for the optimization problems. However, there is a doubly-exponential lower bound which we present in Subsection 3.3.3.

Before we spend the next subsection proving the existence of an automaton as claimed in Lemma 3.37, let us show that these automata implement winning strategies realizing the optimal values. Due to the reductions, which allow to transfer optimal strategies, we only have to consider games with a single variable.

For a PROMPT–LTL game, we determine the optimal variable valuation α_n by reductions to parity games. The automaton $\mathfrak{P}_{\varphi,\alpha_n}$ for this opti-

mum can be turned into a memory structure which implements a finite-state winning strategy for Player 0 for $\mathcal{G}$ with respect to α_n.

Now, consider a $\text{PLTL}_{\mathbf{G}}$ game $\mathcal{G}$ whose winning condition φ has a single variable and remember that we reduced the problem to a PROMPT–LTL optimization problem via

$$\max_{\alpha \in \mathcal{W}_0(\mathcal{G})} \alpha(y) = \max_{\alpha \in \mathcal{W}_1(\overline{\mathcal{G}})} \alpha(y) = \min_{\alpha \in \mathcal{W}_0(\overline{\mathcal{G}})} \alpha(y) + 1 \ .$$

Hence, after we have determined the optimal variable valuation α_n for $\overline{\mathcal{G}}$, we construct the automaton $\mathfrak{P}_{\varphi,\alpha_{n-1}}$. This automaton can be turned into a memory structure that implements a winning strategy for $\mathcal{G}$ with respect to the variable valuation α_{n-1}, which is optimal.

3.3.2 Translating PLTL into Small Automata

In this subsection, we show how to construct a deterministic parity automaton recognizing the language $\{w \in \left(2^P\right)^\omega \mid (w, \alpha) \models \varphi\}$ for a given pair comprising an LTL formula φ and a variable valuation α. To solve the optimization problems it suffices to consider the case of PROMPT–LTL formulae. However, our construction is general enough to deal with full PLTL, which yields a **2Exptime** algorithm for the membership problem for PLTL.

When constructing the automaton, we have to make sure that we meet the requirements formulated in Lemma 3.37. Recall that φ with respect to a fixed variable valuation α is equivalent to an LTL formula, which we denote by φ_α (cf. Lemma 3.8). Hence, to construct an automaton recognizing $\{w \in (2^P)^\omega \mid (w, \alpha) \models \varphi\}$ we could translate the LTL formula φ_α into a deterministic automaton. However, this is typically done by a two-step translation via non-deterministic Büchi automata, where each step involves an exponential blow-up. Hence, the naive approach of constructing a deterministic parity automaton for the LTL formula φ_{α_n} yields an automaton that recognizes the desired language, but is too large when n is close to k, i.e., doubly exponential in $|\mathcal{G}|$. The problem arises from the fact that φ_{α_n} uses a disjunction of nested next-operators of depth n to be able to count up to n. This (in the worst case doubly-exponential) *counter* is hardwired into the formula φ_{α_n} and thus leads to a quadruply-exponential blowup (again, in the worst case) when turning φ_{α_n} into a deterministic parity automaton, since turning LTL formulae into deterministic parity automata necessarily incurs a doubly-exponential blowup [KR10]. We save one exponential compared to the naive approach by adding the counters only after we have turned the formula into a non-deterministic Büchi automaton. Thus, the Büchi automaton is at most of doubly-exponential size and yields a triply-exponential deterministic automaton.

The construction is presented in three steps: we begin by translating a PLTL formula and a variable valuation into a generalized Büchi automaton.

By taking some care in the construction, we are able to obtain an unambiguous automaton, which is typical for automata obtained from translating formulae of linear temporal logics. Then, in the second step, we use a standard construction to turn a generalized Büchi automaton into a Büchi automaton while preserving unambiguity. Finally, we can apply the determinization procedure of Morgenstern and Schneider [MS08, Mor10] for non-confluent Büchi automata to obtain a deterministic parity automaton with the desired properties in the third step. In the last step, we use Lemma 2.2: every unambiguous automaton without unproductive states is non-confluent.

From PLTL to Generalized Büchi Automata

We begin by constructing a generalized Büchi automaton $\mathfrak{A}_{\varphi,\alpha}$ recognizing the language $\{w \in \left(2^P\right)^\omega \mid (w,\alpha) \models \varphi\}$ for a given PLTL formula φ and a variable valuation α. The automaton guesses for each position of w which subformulae of φ are satisfied with respect to α at this position and verifies these guesses while processing w. Since there is only one way to guess right, the automaton is unambiguous. Our construction for the first step is the adaption of the tableaux construction for Metric Temporal Logic [Koy90, AH93] to PLTL. This logic is defined by adding the operators $\mathbf{U}_I$ (and a corresponding past temporal operator) to LTL, where I is an arbitrary interval of $\mathbb{N}$ whose end-points are integer constants, with the expected semantics. Since we are in this subsection interested in a PLTL formula with respect to a fixed variable valuation, our problem refers to constant bounds as well, and could therefore be expressed in MTL.

Let φ be a PLTL formula and recall that $\mathrm{cl}(\varphi)$ denotes the set of subformulae of φ. A set $B \subseteq \mathrm{cl}(\varphi)$ is consistent, if the following conditions are satisfied:

(C1) For all $p, \neg p \in \mathrm{cl}(\varphi)$: $p \in B$ if and only if $\neg p \notin B$.

(C2) For all $\psi_1 \wedge \psi_2 \in \mathrm{cl}(\varphi)$: $\psi_1 \wedge \psi_2 \in B$ if and only if $\psi_1 \in B$ and $\psi_2 \in B$.

(C3) For all $\psi_1 \vee \psi_2 \in \mathrm{cl}(\varphi)$: $\psi_1 \vee \psi_2 \in B$ if and only if $\psi_1 \in B$ or $\psi_2 \in B$.

(C4) For all $\psi_1 \,\mathbf{U}\, \psi_2 \in \mathrm{cl}(\varphi)$: $\psi_2 \in B$ implies $\psi_1 \,\mathbf{U}\, \psi_2 \in B$.

(C5) For all $\psi_1 \,\mathbf{R}\, \psi_2 \in \mathrm{cl}(\varphi)$: $\psi_1, \psi_2 \in B$ implies $\psi_1 \,\mathbf{R}\, \psi_2 \in B$.

(C6) For all $\mathbf{F}_{\leq x}\psi_1 \in \mathrm{cl}(\varphi)$: $\psi_1 \in B$ implies $\mathbf{F}_{\leq x}\psi_1 \in B$.

(C7) For all $\mathbf{G}_{\leq y}\psi_1 \in \mathrm{cl}(\varphi)$: $\mathbf{G}_{\leq y}\psi_1 \in B$ implies $\psi_1 \in B$.

These conditions capture the local properties of the semantics of PLTL. The non-local properties are captured by the transition relation of the automaton we are about to define. The set of consistent subsets is denoted by $\mathcal{C}(\varphi)$.

Let us denote the set of parameterized subformulae of φ by $\mathrm{cl}_p(\varphi)$. The states of our automaton are pairs (B, c) where $B \in \mathcal{C}(\varphi)$ and $c\colon \mathrm{cl}_p(\varphi) \to \mathbb{N} \cup \{\bot\}$. The counter $c(\mathbf{F}_{\leq x}\psi_1)$ is used to verify that every position at which $\mathbf{F}_{\leq x}\psi_1$ is guessed to be satisfied, is followed within $\alpha(x)$ steps by a position at which ψ_1 is guessed to be satisfied. Dually, $c(\mathbf{G}_{\leq y}\psi_1)$ is used to verify that at the next $\alpha(y)$ positions ψ_1 is guessed to be satisfied, whenever $\mathbf{G}_{\leq y}\psi_1$ is guessed to be satisfied. The value $\bot$ denotes that a counter is inactive. A pair (B, c) is consistent, if the following properties are satisfied:

(C8) For all $\mathbf{F}_{\leq x}\psi_1 \in \mathrm{cl}(\varphi)$: $\psi_1 \in B$ if and only if $c(\mathbf{F}_{\leq x}\psi_1) = 0$.

(C9) For all $\mathbf{F}_{\leq x}\psi_1 \in \mathrm{cl}(\varphi)$: $\mathbf{F}_{\leq x}\psi_1 \in B$ if and only if $c(\mathbf{F}_{\leq x}\psi_1) \neq \bot$.

(C10) For all $\mathbf{G}_{\leq y}\psi_1 \in \mathrm{cl}(\varphi)$: $\mathbf{G}_{\leq y}\psi_1 \in B$ if and only if $c(\mathbf{G}_{\leq y}\psi_1) = \alpha(y)$.

(C11) For all $\mathbf{G}_{\leq y}\psi_1 \in \mathrm{cl}(\varphi)$: $\psi_1 \in B$ if and only if $c(\mathbf{G}_{\leq y}\psi_1) \neq \bot$.

These conditions capture the relation between a parameterized subformula and its associated counter: (C8) requires the counter for the formula $\mathbf{F}_{\leq x}\psi_1$ to be zero if and only if the formula ψ_1 is guessed to be satisfied while (C9) requires the counter to be active if and only if the formula $\mathbf{F}_{\leq x}\psi_1$ is guessed to be satisfied. In this situation, the counter will be decremented in each step until it reaches value zero. At such a position, ψ_1 has to be guessed to be satisfied due to the first condition. The requirements on the counters for parameterized always operator are dual: if $\mathbf{G}_{\leq y}\psi_1$ is guessed to be satisfied, then the counter has to have value $\alpha(y)$ and is decremented in each step until it reaches value zero. Furthermore, the formula ψ_1 has to be guessed to be satisfied at every position at which the counter is active. This ensures that ψ_1 is satisfied for $\alpha(y)$ consecutive positions. Decrementing the counters is implemented in the transition relation.

Finally, given a variable valuation α, we say that a pair (B, c) is α-bounded, if we have:

(C12) For all $\mathbf{F}_{\leq x}\psi_1 \in \mathrm{cl}(\varphi)$: $c(\mathbf{F}_{\leq x}\psi_1) \neq \bot$ implies $c(\mathbf{F}_{\leq x}\psi_1) \leq \alpha(x)$.

(C13) For all $\mathbf{G}_{\leq y}\psi_1 \in \mathrm{cl}(\varphi)$: $c(\mathbf{G}_{\leq y}\psi_1) \neq \bot$ implies $c(\mathbf{G}_{\leq y}\psi_1) \leq \alpha(y)$.

These conditions bound the counters associated to a subformula with parameter z by $\alpha(z)$.

We are now ready to construct a generalized Büchi automaton recognizing the language $\{w \in \left(2^P\right)^\omega \mid (w, \alpha) \models \varphi\}$. As already mentioned above, the states are pairs (B, c) used to guess which subformulae of φ are satisfied at a position of the input. The automaton has to verify that these guesses are valid ones. The local aspects of the semantics for the unparameterized operators are taken care of by the conditions (C1) up to (C5) while the local aspects of parameterized operators are taken care of by the conditions (C6) up to (C13). Finally, the global aspects of both types of operators are taken

care of by the transition relation and the acceptance condition. This automaton is unambiguous, since there is only one way to guess the truth values of the subformulae correctly, every incorrect guess leads to a rejecting run.

Construction 1. Given a PLTL formula φ and a variable valuation α, we define the generalized Büchi automaton $\mathfrak{A}_{\varphi,\alpha} = (Q, 2^P, Q_0, \Delta, \mathcal{F})$ with the following components.

- Q is the set of pairs (B, c), where $B \in \mathcal{C}(\varphi)$ and $c\colon \mathrm{cl}_p(\varphi) \to \mathbb{N} \cup \{\bot\}$, such that (B, c) satisfies (C1) up to (C13).
- $Q_0 = \{(B, c) \in Q \mid \varphi \in B\}$.
- $((B, c), a, (B', c')) \in \Delta$ if and only if

  (T1) $B \cap P = a$,
  (T2) $\mathbf{X}\psi_1 \in B$ if and only if $\psi_1 \in B'$,
  (T3) $\psi_1 \,\mathbf{U}\, \psi_2 \in B$ if and only if $\psi_2 \in B$ or ($\psi_1 \in B$ and $\psi_1 \,\mathbf{U}\, \psi_2 \in B'$),
  (T4) $\psi_1 \,\mathbf{R}\, \psi_2 \in B$ if and only if $\psi_2 \in B$ and ($\psi_1 \in B$ or $\psi_1 \,\mathbf{R}\, \psi_2 \in B'$),
  (T5) if $\alpha(x) > 0$ and $c(\mathbf{F}_{\leq x}\psi_1) = \bot$, then $c'(\mathbf{F}_{\leq x}\psi_1) \in \{\alpha(x), \bot\}$,
  (T6) if $\alpha(x) > 0$ and $c(\mathbf{F}_{\leq x}\psi_1) > 0$, then $c'(\mathbf{F}_{\leq x}\psi_1) = c(\mathbf{F}_{\leq x}\psi_1) - 1$,
  (T7) if $\alpha(y) > 0$ and $c(\mathbf{G}_{\leq y}\psi_1) = 0$, then $c'(\mathbf{G}_{\leq y}\psi_1) = \bot$,
  (T8) if $\alpha(y) > 0$ and $0 < c(\mathbf{G}_{\leq y}\psi_1) < \alpha(y)$, then $c'(\mathbf{G}_{\leq y}\psi_1) = c(\mathbf{G}_{\leq y}\psi_1) - 1$, and
  (T9) if $\alpha(y) > 0$ and $c(\mathbf{G}_{\leq y}\psi_1) = \alpha(y)$, then $\alpha(y) - 1 \leq c'(\mathbf{G}_{\leq y}\psi_1) \leq \alpha(y)$.

- $\mathcal{F} = \mathcal{F}_{\mathbf{U}} \cup \mathcal{F}_{\mathbf{R}}$ where
  - $\mathcal{F}_{\mathbf{U}} = \{F_{\psi_1 \mathbf{U} \psi_2} \mid \psi_1 \,\mathbf{U}\, \psi_2 \in \mathrm{cl}(\varphi)\}$ with $F_{\psi_1 \mathbf{U} \psi_2} = \{(B, c) \in Q \mid \psi_1 \,\mathbf{U}\, \psi_2 \notin B \text{ or } \psi_2 \in B\}$, and
  - $\mathcal{F}_{\mathbf{R}} = \{F_{\psi_1 \mathbf{R} \psi_2} \mid \psi_1 \,\mathbf{R}\, \psi_2 \in \mathrm{cl}(\varphi)\}$ with $F_{\psi_1 \mathbf{R} \psi_2} = \{(B, c) \in Q \mid \psi_1 \,\mathbf{R}\, \psi_2 \in B \text{ or } \psi_2 \notin B\}$.

Let us explain the definition of Δ: the conditions (T1) up to (T4) are standard for LTL and reflect the semantics of these operators. Hence, we focus on the latter conditions for the parameterized operators. So, consider a formula $\mathbf{F}_{\leq x}\psi_1$ with $\alpha(x) > 0$. If the counter for this subformula is inactive, then the counter is either also inactive at the next position, or it is started with value $\alpha(x)$ (which means $\mathbf{F}_{\leq x}\psi_1$ is guessed to be satisfied). In the second case, ψ_1 has to be guessed true within $\alpha(x)$ steps. This is captured by (T5). If the counter is active, but not zero, then it is decremented in the next step, which is captured by (T6). Finally, if the counter is zero (which

is equivalent to ψ_1 is guessed to be satisfied at the current position, due to (C8)), then it is zero in the next step (ψ_1 is guessed to be satisfied in the next step as well), inactive, or can be restart with any value k (meaning that ψ_1 is guessed to be satisfied for the next time in exactly k) positions. Hence, there is no requirement on the counter in this case, any value is allowed. Note that we require a counter to start with value $\alpha(x)$ after it is $\bot$ in the previous step, i.e., $\mathbf{F}_{\leq x}\psi_1$ has to be guessed to be satisfied as soon as possible. This property is crucial to obtain an unambiguous automaton.

The conditions for formulae $\mathbf{G}_{\leq y}\psi_1$ are dual: as long as $\mathbf{G}_{\leq y}\psi_1$ is guessed to be satisfied, $c(\mathbf{G}_{\leq y}\psi_1)$ has value $\alpha(y)$ due to (C10). Beginning at the first position where $\mathbf{G}_{\leq y}\psi_1$ is no longer guessed to hold, the counter has to be decremented in each step (due to (T8) and (T9)) and checks that the next $\alpha(y)$ positions satisfy ψ_1 due to (C11). Since the counter has to be inactive after it has reached value zero (due to (T7)), the automaton cannot start the decrement phase too early or too late.

The requirements on c and c' in the definition are only phrased for parameterized formulae with variable z such that $\alpha(z) > 0$. This is because $\mathbf{F}_{\leq x}\psi_1$ and $\mathbf{G}_{\leq y}\psi_1$ are both equivalent to ψ_1 if we have $\alpha(x) = 0$ or $\alpha(y) = 0$, respectively. This is modeled by the fact that we have $c(\mathbf{F}_{\leq x}\psi_1) \in \{0, \bot\}$ for such a formula. Hence, the consistency properties (C8) and (C9) make sure that we have $\mathbf{F}_{\leq x}\psi_1 \in B$ if and only if $\psi_1 \in B$. The reasoning for $\mathbf{G}_{\leq y}\psi_1$ is the same, but applies (C10) and (C11).

Example 3.38. Consider the subformulae $\mathbf{F}_{\leq x}p$ and $\mathbf{G}_{\leq y}q$ of some formula φ and the variable valuation α with $\alpha(x) = 2$ and $\alpha(y) = 3$. Table 3.4 shows how the counters evolve during a run of the Büchi automaton $\mathfrak{A}_{\varphi,\alpha}$.

First, consider the parameterized eventually operator. The formula $\mathbf{F}_{\leq x}p$ is in B if and only if the associated counter value is active, in which case it is decremented in each step until it reaches value zero, which is exactly the case when p holds. Afterwards, it is initialized with value k, if p holds for the next time in exactly k positions; or it is initialized with $\bot$, if p does not hold within the next $\alpha(x)$ positions. Note that the same holds true at the first position of the run.

Now, consider the parameterized always operator. The formula $\mathbf{G}_{\leq y}q$ is in B if and only if the associated counter value has value $\alpha(y)$. Furthermore, if the counter is active, then q has to be satisfied at this position. Finally, the counter has value $\alpha(y)$ until it decremented to value 0. In this case, q must not be satisfied at the next position. The counter is initialized with value k at the first position of the run and after an inactive period, where k is the maximal value smaller or equal to $\alpha(y)$, such that q holds throughout the next k positions. ◇

Now, we show that the automaton $\mathfrak{A}_{\varphi,\alpha}$ recognizes the right language, is unambiguous, and bound its size.

w	$\{q\}$	$\{p,q\}$	$\emptyset$	$\{p,q\}$	$\{q\}$	$\{q\}$	$\{q\}$	$\{p,q\}$	$\emptyset^\omega$
$p \in B$	n	y	n	y	n	n	n	y	n
$\mathbf{F}_{\leq x} p \in B$	y	y	y	y	n	y	y	y	n
$c(\mathbf{F}_{\leq x} p)$	1	0	1	0	$\bot$	2	1	0	$\bot$
$q \in B$	y	y	n	y	y	y	y	y	n
$\mathbf{G}_{\leq y} q \in B$	n	n	n	y	y	n	n	n	n
$c(\mathbf{G}_{\leq y} q)$	1	0	$\bot$	3	3	2	1	0	$\bot$

Figure 3.4: A run of the automaton $\mathfrak{A}_{\varphi,\alpha}$

Lemma 3.39. *Let φ be a* PLTL *formula, let α be a variable valuation, and let $\mathfrak{A}_{\varphi,\alpha}$ be the automaton obtained in Construction 1. Then,*

i. $L(\mathfrak{A}_{\varphi,\alpha}) = \{w \in \left(2^P\right)^\omega \mid (w,\alpha) \models \varphi\}$,

ii. $\mathfrak{A}_{\varphi,\alpha}$ *is unambiguous, and*

iii. $|\mathfrak{A}_{\varphi,\alpha}| \leq 2^{|\varphi|} \cdot \left(\max_{z \in \mathrm{var}(\varphi)} \alpha(z) + 2\right)^{|\varphi|}$ *and* $|\mathcal{F}| < |\varphi|$.

Proof. i.) First, we show $L(\mathfrak{A}_{\varphi,\alpha}) \subseteq \{w \in \left(2^P\right)^\omega \mid (w,\alpha) \models \varphi\}$. Thus, let $(B_0,c_0)(B_1,c_1)(B_2,c_2)\cdots$ be an accepting run of $\mathfrak{A}_{\varphi,\alpha}$ on w. We show by structural induction over the construction of φ that $\psi \in B_n$ if and only if $(w,n,\alpha) \models \psi$. This suffices to show $(w,\alpha) \models \varphi$, since we have $\varphi \in B_0$ by definition of Q_0.

- $\psi = p \in P$: we have $p \in B_n$ if and only if $p \in w_n$ (by (T1)) if and only if $(w,n,\alpha) \models p$.
- $\psi = \neg p$ for some $p \in P$: analogously to the case $\psi = p$.
- $\psi = \psi_1 \wedge \psi_2$: we have $\psi_1 \wedge \psi_2 \in B_n$ if and only if $\psi_1 \in B_n$ and $\psi_2 \in B_n$ (by (C2)) if and only if $(w,n,\alpha) \models \psi_1$ and $(w,n,\alpha) \models \psi_2$ (by induction hypothesis) if and only if $(w,n,\alpha) \models \psi$.
- $\psi = \psi_1 \vee \psi_2$: analogously to the case of conjunction.
- $\psi = \mathbf{X}\psi_1$: we have $\mathbf{X}\psi_1 \in B_n$ if and only if $\psi_1 \in B_{n+1}$ (by (T2)) if and only if $(w,n+1,\alpha) \models \psi_1$ (by induction hypothesis) if and only if $(w,n,\alpha) \models \mathbf{X}\psi_1$.
- $\psi = \psi_1 \,\mathbf{U}\, \psi_2$: let $\psi \in B_n$. Then, we have either $\psi_2 \in B_n$ or $\psi_1 \in B_n$ and $\psi_1 \,\mathbf{U}\, \psi_2 \in B_{n+1}$, due to (T3). Iterating the second case either yields a position $n' \geq n$ such that $\psi_2 \in B_{n'}$ and $\psi_1 \in B_j$ for every j in the range $n \leq j < n'$, or it yields $\psi_1 \in B_m$ and $\psi_1 \,\mathbf{U}\, \psi_2 \in B_m$ for every $m \geq n$. In this case, we use the fact that the run is accepting, i.e., there is a position $n' > n$ such that $B_{n'} \in F_{\psi_1 \mathbf{U} \psi_2}$. Then, we have

$\psi_2 \in B_{n'}$. Thus, we can apply the induction hypothesis in every case to obtain $(w, n', \alpha) \models \psi_2$ and $(w, j, \alpha) \models \psi_1$ for every $n \leq j < n'$. Hence, $(w, n, \alpha) \models \psi_1 \mathbf{U} \psi_2$.

Now, let $(w, n, \alpha) \models \psi_1 \mathbf{U} \psi_2$, i.e., there is a position n' such that $(w, n', \alpha) \models \psi_2$ and $(w, j, \alpha) \models \psi_1$ for every j in the range $n \leq j < n'$. The induction hypothesis and consistency condition (C4) yield $\psi_1 \mathbf{U} \psi_2 \in B_{n'}$ and applying (T3) repeatedly gives $\psi_1 \mathbf{U} \psi_2 \in B_j$ for every j in the range $n \leq j < n'$.

- $\psi = \psi_1 \mathbf{R} \psi_2$: note that we have $(w, n, \alpha) \models \psi_1 \mathbf{R} \psi_2$ if and only if either there exists an $n' \geq n$ such that $(w, n', \alpha) \models \psi_1$ and $(w, j, \alpha) \models \psi_2$ for every j in the range $n \leq j \leq n'$, or if $(w, m, \alpha) \models \psi_2$ for every $m \geq n$. We use this formulation here.

  Let $\psi \in B_n$. Then, we have $\psi_2 \in B_n$ and either $\psi_1 \in B_n$ or $\psi_1 \mathbf{R} \psi_2 \in B_{n+1}$ due to (T4). Iterating the second case yields either a position $n' \geq n$ such that $\psi_1 \in B_{n'}$ and $\psi_2 \in B_j$ for every j in the range $n \leq j \leq n'$, or it yields $\psi_2 \in B_m$ for every $m \geq n$. Applying the induction hypothesis in the first case yields $(w, n', \alpha) \models \psi_1$ and $(w, j, \alpha) \models \psi_2$ for every j in the range $n \leq j \leq n'$, i.e., $(w, n, \alpha) \models \psi_1 \mathbf{R} \psi_2$. In the second case, we obtain $(w, m, \alpha) \models \psi_2$ for every $m \geq n$, i.e., $(w, n, \alpha) \models \psi_1 \mathbf{R} \psi_2$.

  Now, let $(w, n, \alpha) \models \psi_1 \mathbf{R} \psi_2$, i.e., either there exists an $n' \geq n$ such that $(w, n', \alpha) \models \psi_1$ and $(w, j, \alpha) \models \psi_2$ for every j in the range $n \leq j \leq n'$, or $(w, m, \alpha) \models \psi_2$ for every $m \geq n$. In the first case, the induction hypothesis and (C5) yield $\psi_1 \mathbf{R} \psi_2 \in B_{n'}$ and applying (T4) repeatedly shows $\psi_1 \mathbf{R} \psi_2 \in B_j$ for every j in the range $n \leq j \leq n'$. In the second case, there is a minimal position $m \geq n$ such that $B_m \in F_{\psi_1 \mathbf{R} \psi_2}$, as the run is accepting. Hence, we have $\psi_1 \mathbf{R} \psi_2 \in B_m$. Thus, applying the induction hypothesis and (T4) repeatedly shows $\psi_1 \mathbf{R} \psi_2 \in B_j$ for every j in the range $n \leq j \leq n'$. Hence, we conclude $\psi_1 \mathbf{R} \psi_2 \in B_n$ in both cases.

- $\psi = \mathbf{F}_{\leq x} \psi_1$: let $\psi \in B_n$. Then, we have $c_n(\psi) = k \neq \bot$ by (C9) and $k \leq \alpha(x)$ by α-boundedness. We claim $(w, n + k, \alpha) \models \psi_1$: applying (T6) inductively shows that we have $c_{n+j}(\psi) = k - j$ for every j in the range $0 \leq j \leq k$. Hence, we have $c_{n+k}(\psi) = 0$ and therefore $\psi_1 \in B_{n+k}$ by (C8). Applying the induction hypothesis yields $(w, n + k, \alpha) \models \psi_1$ as claimed and therefore $(w, n, \alpha) \models \psi$.

  Now, let $(w, n, \alpha) \models \psi$ and let k be minimal in the range $0 \leq k \leq \alpha(x)$ with $(w, n + k, \alpha) \models \psi_1$. Then, we have $c_{n+k}(\psi) = 0$ by the induction hypothesis and (C8) and due to (T6) and the minimality of k, we have $c_{n+(k-j)}(\psi) = j$ for every j in the range $0 \leq j \leq k$. Hence, $c_n(\psi) = k \leq \alpha(x)$, which implies $\psi \in B_n$ due to (C9).

- $\psi = \mathbf{G}_{\leq y}\psi_1$: let $\psi \in B_n$. Then, we have $c_n(\psi) = \alpha(y)$ due to (C10). Applying (T8) and (T9) inductively shows that we have $c_{n+j}(\psi) \geq \alpha(y) - j \geq 0$ for every j in the range $0 \leq j \leq \alpha(y)$. Hence, (C11) implies $\psi_1 \in B_{n+j}$ for every such j and applying the induction hypothesis yields $(w, n+j, \alpha) \models \psi_1$ for every j in the range $0 \leq j \leq \alpha(y)$. Hence, we have $(w, n, \alpha) \models \psi$.

  Now, let $(w, n, \alpha) \models \psi$, i.e., we have $(w, n+j, \alpha) \models \psi_1$ for every j in the range $0 \leq j \leq \alpha(y)$. We have $c_{n+\alpha(y)}(\psi) \neq \bot$ due to the induction hypothesis and (C11). Furthermore, (T8) and (T9) imply $c_{n+\alpha(y)-j}(\psi) \geq j$ for every j in the range $0 \leq j \leq \alpha(y)$, as the counter values cannot be $\bot$ due to (C11). Thus, $c_n(\psi) = \alpha(y)$, and therefore $\psi \in B_n$ due to (C10).

This finished the first inclusion. Now, let us prove the second inclusion $\{w \in \left(2^P\right)^\omega \mid (w, \alpha) \models \varphi\} \subseteq L(\mathfrak{A}_{\varphi,\alpha})$. Let $(w, \alpha) \models \varphi$ and define for each n

$$B_n = \{\psi \in \mathrm{cl}(\varphi) \mid (w, n, \alpha) \models \psi\} \ ,$$

to be the set of subformulae that are satisfied at position n with respect to α. Now, we need to define the counter c_n for each n: for every $\mathbf{F}_{\leq x}\psi_1 \in \mathrm{cl}_p(\varphi)$ let

$$c_n(\mathbf{F}_{\leq x}\psi_1) = \min\{k \mid 0 \leq k \leq \alpha(x) \text{ and } (w, n+k, \alpha) \models \psi_1\} \ ,$$

where we set $\min \emptyset = \bot$, be the minimal waiting times for the parameterized eventually operators at position n. Dually, for every $\mathbf{G}_{\leq y}\psi_1 \in \mathrm{cl}_p(\varphi)$ let

$$c_n(\mathbf{G}_{\leq y}\psi_1) = \max\{k \mid 0 \leq k \leq \alpha(y) \text{ and } (w, n+j, \alpha) \models \psi_1 \text{ for every } j \leq k\} \ ,$$

where we set $\max \emptyset = \bot$, be the maximal satisfaction times for the parameterized always operators at position n.

We claim that $(B_0, c_0)(B_1, c_1)(B_2, c_2)\cdots$ is an accepting run of $\mathfrak{A}_{\varphi,\alpha}$. The semantics of PLTL guarantee that each set B_n is consistent, i.e., the local requirements (C1) up to (C7) are satisfied, and the c_n are α-bounded by definition, i.e., the conditions (C12) and (C13) are satisfied. It remains to show that the consistency requirements (C8) up to (C11) are met.

- We have $\psi_1 \in B_n$ if and only if $(w, n, \alpha) \models \psi_1$, which is the case if and only if $c_n(\mathbf{F}_{\leq x}\psi_1) = 0$, i.e., (C8) is satisfied.
- We have $\mathbf{F}_{\leq x}\psi_1 \in B_n$ if and only if $(w, n, \alpha) \models \mathbf{F}_{\leq x}\psi_1$ if and only if $c_n(\mathbf{F}_{\leq x}\psi_1) \neq \bot$, i.e., (C9) is satisfied.
- We have $\mathbf{G}_{\leq y}\psi_1 \in B_n$ if and only if $(w, n, \alpha) \models \mathbf{G}_{\leq y}\psi_1$ if and only if $c_n(\mathbf{G}_{\leq y}\psi_1) = \alpha(y)$, i.e., (C10) is satisfied.

- We have $\psi_1 \in B_n$ if and only if $(w, n, \alpha) \models \psi_1$, which is the case if and only if $c_n(\mathbf{G}_{\leq y}\psi_1) \neq \bot$, i.e., (C11) is satisfied.

Hence, each pair (B_n, c_n) is a state of the automaton and we have $(B_0, c_0) \in Q_0$ due to $(w, 0, \alpha) \models \varphi$. Next, we show that the sequence is a run on w, i.e., $((B_n, c_n), w_n, (B_{n+1}, c_{n+1})) \in \Delta$ for every $n \in \mathbb{N}$. The conditions (T1) up to (T4) are satisfied due to the semantics of PLTL. So, let us consider the requirements on the parameterized operators:

- Let $c_n(\mathbf{F}_{\leq x}\psi_1) = \bot$, i.e., there is no k in the range $0 \leq k \leq \alpha(x)$ such that $(w, n+k, \alpha) \models \psi_1$. Now, assume we have $c_{n+1}(\mathbf{F}_{\leq x}\psi_1) \notin \{\alpha(x), \bot\}$, i.e., $c_{n+1}(\mathbf{F}_{\leq x}\psi_1) = k' < \alpha(x)$. Then, $(w, n+1+k', \alpha) \models \psi_1$, which is a contradiction, since we have $n+1+k' \leq \alpha(x)$. Thus, we have shown (T5).

- Let $c_n(\mathbf{F}_{\leq x}\psi_1) = k > 0$, i.e., k is minimal with $(w, n+k, \alpha) \models \psi_1$. Then, we must have $c_{n+1}(\mathbf{F}_{\leq x}\psi_1) = k - 1$ by definition, i.e., $c_{n+1}(\mathbf{F}_{\leq x}\psi_1) = c_n(\mathbf{F}_{\leq x}\psi_1) - 1$. Thus, we have shown (T6).

- Let $c_n(\mathbf{G}_{\leq y}\psi_1) = 0$. Then, we have $(w, n, \alpha) \models \psi_1$, but $(w, n+1, \alpha) \not\models \psi_1$. Hence, we have $\psi_1 \notin B_{n+1}$ and thus $c_{n+1}(\mathbf{G}_{\leq y}\psi_1) = \bot$ by (C11). Hence, (T7) is satisfied.

- Let $0 < c_n(\mathbf{G}_{\leq y}\psi_1) = k < \alpha(y)$, i.e., we have $(w, n+j, \alpha(y)) \models \psi_1$ for every j in the range $0 \leq j \leq k$, but $(w, n+k+1, \alpha(y)) \not\models \psi_1$. Then, we have $c_{n+1}(\mathbf{G}_{\leq y}\psi_1) = k-1$ by the maximality of the position $k > 0$. Hence, $c_{n+1}(\mathbf{G}_{\leq y}\psi_1) = c_n(\mathbf{G}_{\leq y}\psi_1) - 1$ and we have shown (T8).

- Let $c(\mathbf{G}_{\leq y}\psi_1) = \alpha(y)$, i.e., we have $(w, n+j, \alpha) \models \psi_1$ for every j in the range $0 \leq j \leq \alpha(y)$. Now, consider the position $n+\alpha(y)+1$. If we have $(w, n+\alpha(y)+1, \alpha) \models \psi_1$, then we have $c_{n+1}(\mathbf{G}_{\leq y}\psi_1) = \alpha(y)$. On the other hand, if $(w, n+j+1, \alpha) \not\models \psi_1$, then we have $c_{n+1}(\mathbf{G}_{\leq y}\psi_1) = \alpha(y) - 1$. Thus, we have shown (T9).

Hence, $(B_0, c_0)(B_1, c_1)(B_2, c_2)\cdots$ is a run of $\mathfrak{A}_{\varphi,\alpha}$. Furthermore, due to condition (T1), it is a run on w. It remains to show that the run is accepting. So, consider a set $F_{\psi_1 \mathbf{U} \psi_2} \in \mathcal{F}$ and assume that it is visited only finitely often, i.e., there exists a position n such that for every $n' \geq n$ we have $\psi_1 \mathbf{U} \psi_2 \in B_{n'}$ and $\psi_2 \notin B_{n'}$. By definition of the sets B_n, we have $(w, n', \alpha) \models \psi_1 \mathbf{U} \psi_2$ and $(w, n', \alpha) \not\models \psi_2$ for every $n' \geq n$. This contradicts the semantics of the until-operator, which guarantees a position $m \geq n$ such that $(w, m, \alpha) \models \psi_2$, if $(w, n, \alpha) \models \psi_1 \mathbf{U} \psi_2$. Now, assume that some $F_{\psi_1 \mathbf{R} \psi_2}$ is visited only finitely often, i.e., there exists a position n such that for every $n' \geq n$ we have $\psi_1 \mathbf{R} \psi_2 \notin B_{n'}$ and $\psi_2 \in B_{n'}$. Again, we have $(w, n', \alpha) \not\models \psi_1 \mathbf{R} \psi_2$ and $(w, n', \alpha) \models \psi_2$ for every $n' \geq n$. This contradicts the semantics of

the release-operator, which states $(w, n, \alpha) \models \psi_1 \mathbf{R} \psi_2$, if ψ_2 holds at every position $n' \geq n$.

Hence, $(B_0, c_0)(B_1, c_1)(B_2, c_2) \cdots$ is an accepting run of $\mathfrak{A}_{\varphi,\alpha}$ and we conclude $w \in L(\mathfrak{A}_{\varphi,\alpha})$.

ii.) Assume $\mathfrak{A}_{\varphi,\alpha}$ has two accepting runs $(B_0, c_0)(B_1, c_1)(B_2, c_2) \cdots$ and $(B'_0, c'_0)(B'_1, c'_1)(B'_2, c'_2) \cdots$ on a word w, i.e., there is a position n such that $B_n \neq B'_n$ or $c_n \neq c'_n$. We have shown above that we have $\psi \in B_n$ if and only if $(w, n, \alpha) \models \psi$. Since the same holds true for the sets B'_n, we conclude $B_n = B'_n$ for every n. This leaves us with $c_n \neq c'_n$. The consistency requirements (C9) and (C11) and the fact $B_n = B'_n$ imply $c_n(\psi) = \bot$ if and only if $c'_n(\psi) = \bot$ for every parameterized formula $\psi \in \mathrm{cl}_p(\varphi)$. Hence, we must have $c_n(\psi) = k \neq k' = c'_n(\psi)$ for some $\psi \in \mathrm{cl}_p(\varphi)$.

First, we consider the case $\psi = \mathbf{F}_{\leq x}\psi_1$, which implies $0 \leq k, k' \leq \alpha(x)$. We assume without loss of generality $k < k'$. Then, applying (T6) inductively yields $c_{n+j}(\mathbf{F}_{\leq x}\psi_1) = k - j$ for every j in the range $0 \leq j \leq k$ and similarly $c'_{n+j}(\mathbf{F}_{\leq x}\psi_1) = k' - j$ for every j in the range $0 \leq j \leq k'$. Hence, we have $c_{n+k}(\mathbf{F}_{\leq x}\psi_1) = 0$, which implies $\psi_1 \in B_{n+k}$ due to (C8). On the other hand, we have $c'_{n+k}(\mathbf{F}_{\leq x}\psi_1) = k' - k > 0$, which implies $\psi_1 \notin B_{n+k}$, again due to (C8). This yields the desired contradiction, since we have $B_{n+k} = B'_{n+k}$.

Now, we consider the case $\psi = \mathbf{G}_{\leq y}\psi_1$, which implies $0 \leq k, k' \leq \alpha(y)$. Again, we assume without loss of generality $k < k'$. Then, applying (T6) inductively yields $c_{n+j}(\mathbf{G}_{\leq y}\psi_1) = k - j$ for every j in the range $0 \leq j \leq k$, since we have $k < \alpha(y)$. Hence, we have $c_{n+k}(\mathbf{G}_{\leq y}\psi_1) = 0$, which implies $c_{n+k+1}(\mathbf{G}_{\leq y}\psi_1) = \bot$ due to (T7). Thus, we have $\psi_1 \notin B_{n+k+1}$ due to (C11).

Similarly, we have $c'_{n+j}(\mathbf{G}_{\leq y}\psi_1) \geq k' - j$ for every j in the range $0 \leq j \leq k'$ due to (T8) and (T9). Thus, $c'_{n+k}(\mathbf{G}_{\leq y}\psi_1) \geq k' - k > 0$. Condition (T8) implies $c_{n+k+1}(\mathbf{G}_{\leq y}\psi_1) \geq 0$, which in turn implies $\psi_1 \in B'_{n+k+1}$ due to (C11). Again, we have derived a contradiction, due to $B_{n+k+1} = B'_{n+k+1}$.

iii.) The number of consistent subsets of φ is bounded by $2^{|\varphi|}$ and we have $c(\mathbf{F}_{\leq x}\psi_1) \in \{0, \ldots, \alpha(x)\} \cup \{\bot\}$ and $c(\mathbf{G}_{\leq y}\psi_1) \in \{0, \ldots, \alpha(y)\} \cup \{\bot\}$, respectively, if c is α-bounded. This proves the upper bound on the size of $\mathfrak{A}_{\varphi,\alpha}$, since we have $|\mathrm{cl}_p(\varphi)| \leq |\varphi|$. Finally, we have $|\mathcal{F}| < |\varphi| = |\mathrm{cl}(\varphi)|$, since $\mathrm{cl}(\varphi)$ contains at least one (negated) atomic proposition. □

As already mentioned above, in [Zim11] it is claimed that the unipolar PLTL optimization problems could be solved in doubly-exponential time as well. The overall proof strategy there is the same as here: construct a small deterministic parity automaton recognizing the models of a PROMPT–LTL formula with respect to a fixed variable valuation. In [Zim11], the counters $c(\mathbf{F}_{\leq x}\psi_1)$ are not added to the generalized Büchi automaton (as we have just done here), but the automaton without counters is determinized and then the counters are added to the deterministic parity automaton. To this end, every subformula $\mathbf{F}_{\leq x}\psi_1$ is recursively replaced by $\mathbf{F}\psi_1$ and the re-

sulting LTL formula is then translated into an unambiguous Büchi automaton $\mathfrak{A}$ using Construction 1 without the second component of the states. To simulate the effect of α on the satisfaction of φ, an additional requirement on accepting runs of $\mathfrak{A}$ is formulated:

> whenever a subformula $\mathbf{F}\psi_1$ obtained by removing a parameter is guessed to be satisfied at position n, ψ_1 has to be satisfied within $\alpha(x)$ steps.

Since the automaton is unambiguous, and can therefore be assumed to be non-confluent, it satisfies the following property: for every state q, a finite word w has at most one run ending in q. Thus, the counters in the deterministic parity automaton have to track at most one run for each state, which bounds the size of the deterministic parity automaton doubly-exponentially, provided the counter values are bounded doubly-exponentially.

However, the requirement formulated above is too strong: to see this, consider the formula $p \vee \mathbf{F}_{\leq x} q$. If p holds at the first position, then the formula is satisfied with respect to every variable valuation, no matter whether q holds within $\alpha(x)$ steps or not. However, even if q holds after more than $\alpha(x)$ steps for the first time in a word w, then the only accepting run of $\mathfrak{A}$ (rightfully) guesses $\mathbf{F}q$ to be satisfied at the first position of w. Hence, the run does not satisfy our additional requirement. The problem is that $(w, n, \alpha) \models \mathbf{F}q$ implies $(w, n-1, \alpha) \models \mathbf{F}q$, a property that has to be reflected in the definition of the transition relation in order to obtain an unambiguous automaton. For a parameterized eventually operator, $(w, n, \alpha) \models \mathbf{F}_{\leq x} q$ does not imply $(w, n-1, \alpha) \models \mathbf{F}_{\leq x} q$. To model this, one could let the automaton guess the (minimal) positions n from which onwards $\mathbf{F}_{\leq x} q$ is satisfied. However, this makes the automaton ambiguous. This means that an exponential number of runs has to be tracked when the automaton is determinized. Each of these runs could have a different configuration of the counter values which have to be added to the deterministic parity automaton. This yields a deterministic automaton of triply-exponential size as well.

From Generalized Büchi Automata to Büchi Automata

Now, we use a standard construction (see, e.g., [BK08]) to turn a generalized Büchi automaton into a Büchi automaton while preserving its language and its unambiguity. The Büchi automaton uses a cyclic counter to ensure that each $F \in \mathcal{F}$ is visited infinitely often and accepts if the counter changes its value infinitely often. We assume without loss of generality that the generalized Büchi automaton has a non-empty family $\mathcal{F}$ of accepting sets. If this is not the case, then every run is accepting and we can make every state accepting to obtain an equivalent Büchi automaton, without having to change the transition structure of the automaton.

Construction 2. Let $\mathfrak{A} = (Q, \Sigma, Q_0, \Delta, \{F_1, \ldots, F_k\})$ with $k > 0$ be a generalized Büchi automaton. We construct the Büchi automaton $\mathfrak{A}' = (Q', \Sigma, q_0', \Delta', F')$ by

- $Q' = Q \times \{0, 1, \ldots, k\}$,
- $Q_0' = Q_0 \times \{1\}$,
- if $(q, a, q') \in \Delta$, then Δ' contains the following transition:
  - $((q, j), a, (q', j)) \in \Delta'$, if $j > 0$ and $q' \notin F_j$,
  - $((q, j), a, (q', j + 1 \bmod (k + 1))) \in \Delta'$, if $j > 0$ and $q' \in F_j$, and
  - $((q, 0), a, (q', 1)) \in \Delta'$.
- $F' = Q \times \{0\}$.

A state (q, j) with $j > 0$ signals that the automaton is waiting for a visit to F_j, while a state $(q, 0)$ signals that every F_j was visited since the last occurrence of a final state of $\mathfrak{A}'$. Such a state is left immediately. Note that the value of the index j in the second component of the n-th state (q, j) of a run of $\mathfrak{A}'$ depends only on the state q and the $(n-1)$-st index.

Lemma 3.40. *Let $\mathfrak{A} = (Q, \Sigma, Q_0, \Delta, \{F_1, \ldots, F_k\})$ with $k > 0$ be a generalized Büchi automaton and let $\mathfrak{A}'$ be the Büchi automaton obtained by applying Construction 2 to $\mathfrak{A}$.*

i. $L(\mathfrak{A}) = L(\mathfrak{A}')$.

ii. If $\mathfrak{A}$ is unambiguous, then $\mathfrak{A}'$ is unambiguous.

iii. $|\mathfrak{A}'| = |\mathfrak{A}| \cdot (k + 1)$

Proof. i.) Let $q_0 q_1 q_2 \cdots$ be an accepting run of $\mathfrak{A}$ on an ω-word w. It induces a unique run $(q_0, j_0)(q_1, j_1)(q_2, j_2) \cdots$ of $\mathfrak{A}'$ on w, as the second component deterministically depends on the first component. As $q_0 q_1 q_2 \cdots$ is accepting, every $F_j \in \mathcal{F}$ is visited infinitely often. Hence, the second component j_n cycles through its domain infinitely often, thereby being equal to zero infinitely often. Hence, $(q_0, j_0)(q_1, j_1)(q_2, j_2) \cdots$ is an accepting run of $\mathfrak{A}'$ on w.

Conversely, let $(q_0, j_0)(q_1, j_1)(q_2, j_2) \cdots$ be an accepting run of $\mathfrak{A}'$ on an ω-word w. Then, the projection $q_0 q_1 q_2 \cdots$ is a run of $\mathfrak{A}$ on w, which is accepting as every $F_j \in \mathcal{F}$ is visited infinitely often, as $j_0 j_1 j_2 \cdots$ does contain every $j \in \{1, \ldots, k\}$ infinitely often.

ii.) Let $q = (q_0, j_0)(q_1, j_1)(q_2, j_2) \cdots$ and $q' = (q_0', j_0')(q_1', j_1')(q_2', j_2') \cdots$ be two accepting runs of $\mathfrak{A}'$ on a word w. Then, the runs $q_0 q_1 q_2 \cdots$ and $q_0' q_1' q_2' \cdots$ of $\mathfrak{A}$ are also accepting runs on w. As $\mathfrak{A}$ is unambiguous, we

have $q_0q_1q_2\cdots = q'_0q'_1q'_2\cdots$. As the indices j_n respectively j'_n only depend on the first components of a run, we get $(q_0,j_0)(q_1,j_1)(q_2,j_2)\cdots = (q'_0,j'_0)(q'_1,j'_1)(q'_2,j'_2)\cdots$. Hence, $\mathfrak{A}'$ is unambiguous.

iii.) Clear. □

From Büchi Automata to Deterministic Parity Automata

Given a PLTL formula φ and a variable valuation α, we have shown how to construct an unambiguous Büchi automaton of size

$$|\varphi| \cdot 2^{|\varphi|} \cdot \left(\max_{z \in \mathrm{var}(\varphi)} \alpha(z) + 2 \right)^{|\varphi|}$$

recognizing the language $\{w \in \left(2^P\right)^\omega \mid (w,\alpha) \models \varphi\}$. Due to Lemma 2.2, we can assume this automaton to be non-confluent as well. In the next and last step of the construction, we determinize the Büchi automaton into a parity automaton. In order to meet the requirements formulated in Lemma 3.37 to solve the optimization problems in triply-exponential time, the determinization procedure has to turn a Büchi automaton with n states into a deterministic parity automaton of exponential size with linearly many priorities. We apply the procedure of Morgenstern and Schneider [MS08, Mor10], which turns a non-confluent Büchi automaton with n states into a deterministic parity automaton with $2^{\mathcal{O}(n^2)}$ states and $2n+1$ priorities, and obtain the following result.

Theorem 3.41. *Let φ be a* PLTL *formula and let α be a variable valuation. We denote $\max_{z \in \mathrm{var}(\varphi)} \alpha(z) + 2$ by m. There exists a deterministic parity automaton $\mathfrak{P}_{\varphi,\alpha}$ of size*

$$|\mathfrak{P}_{\varphi,\alpha}| \leq 2^{\mathcal{O}(|\varphi|^2 \cdot (2m)^{2|\varphi|})}$$

with $2 \cdot |\varphi| \cdot (2m)^{|\varphi|} + 1$ priorities and $L(\mathfrak{P}_{\varphi,\alpha}) = \{w \in \left(2^P\right)^\omega \mid (w,\alpha) \models \varphi\}$.

Note that this theorem proves Lemma 3.37, since we can construct the automaton $\mathfrak{P}_{\varphi,\alpha}$ in triply-exponential time.

Furthermore, the previous theorem allows us to give an upper bound on the complexity of the membership problem for PLTL games. The problem asks, given a PLTL game $\mathcal{G} = (\mathcal{A}, v_0, \varphi)$, $i \in \{0,1\}$, and a variable valuation α, to determine whether $\alpha \in \mathcal{W}^i_{\mathcal{G}}$ holds. Remember that we measure the size of a variable valuation α for φ by

$$|\alpha| = \sum_{x \in \mathrm{var}(\varphi)} \lceil \log_2(\alpha(x)+1) \rceil \ .$$

Hence, we have $\max_{z \in \mathrm{var}(\varphi)} \alpha(z) + 2 \in 2^{\mathcal{O}(|\alpha|)}$ and therefore

$$|\mathfrak{P}_{\varphi,\alpha}| \leq 2^{2^{\mathcal{O}((|\varphi|+|\alpha|)^2)}} \ ,$$

while the number of priorities of $\mathfrak{P}_{\varphi,\alpha}$ is bounded by

$$2^{\mathcal{O}((|\varphi|+|\alpha|)^2)} \ .$$

Thus, the membership problem can be reduced to the problem of solving a parity game $\mathcal{G}$ of size $|\mathcal{A}| \cdot |\mathfrak{P}_{\varphi,\alpha}|$ whose number of priorities is equal to the number of priorities of $\mathfrak{P}_{\varphi,\alpha}$. This game can be constructed and solved in doubly-exponential time due to Theorem 2.22. As we have already shown that the membership problem is **2Exptime**-hard, we obtain the following result.

Theorem 3.42. *The membership problem for* PLTL *games is* **2Exptime***-complete.*

3.3.3 Lower Bounds on Optimal Variable Valuations

A consequence of our algorithm for the $\text{PLTL}_{\mathbf{F}}$ emptiness problem is a $2^{2^{\mathcal{O}(|\mathcal{G}|)}}$ upper bound on optimal variable valuations which allow Player 0 to win a $\text{PLTL}_{\mathbf{F}}$ game $\mathcal{G}$. In this section, we present a $2^{2^{\sqrt{|\mathcal{G}|}}}$ lower bound.

Theorem 3.43. *For every $n \geq 1$, there exists a* PROMPT–LTL *game $\mathcal{G}_n$ with winning condition φ_n with $|\mathcal{G}_n| \in \mathcal{O}(n^2)$ and $\text{var}(\varphi_n) = \{x\}$ such that $\mathcal{W}_0(\mathcal{G}_n) \neq \emptyset$, but Player 1 wins $\mathcal{G}_n$ with respect to every variable valuation α such that $\alpha(x) \leq 2^{2^n}$.*

The idea of our proof is to encode a binary counter $d_0, d_1, d_2, \ldots$ with range $[2^{2^n}]$ and require Player 0 to satisfy some obligation expressed by a parameterized eventually operator, but only after the counter has reached value $2^{2^n}-1$. However, such a counter requires 2^n bits and exponentially long formulae to check the desired behavior. The latter claim follows from the fact that LTL formulae can be translated into Büchi automata of exponential size. To remedy this, we make use of the interaction between the players: Player 1 is in charge of maintaining the counter d and Player 0 has to check whether he increments the counter correctly. If he does not, she may fulfill her obligation earlier. Hence, by always incrementing correctly, he is able to prevent Player 0 from satisfying the parameterized eventually operator in less than 2^{2^n} steps, but not longer than that.

Each value d_ℓ is encoded by 2^n bits. To enable the winning condition to check the faulty increment claimed by Player 0, Player 1 has to precede every bit of d_ℓ by a binary representation of its position $c_j \in [2^n]$, which is of length n. Hence, the correct evolution of the c_j can be checked by an LTL formula of size $\mathcal{O}(n^2)$. Then, Player 0 marks the position of a single bit of some $d_{\ell+1}$ witnessing the faulty increase of d_ℓ to $d_{\ell+1}$. Using the addresses c_j the winning condition can verify whether the claim is correct or not. This idea is formalized in the following.

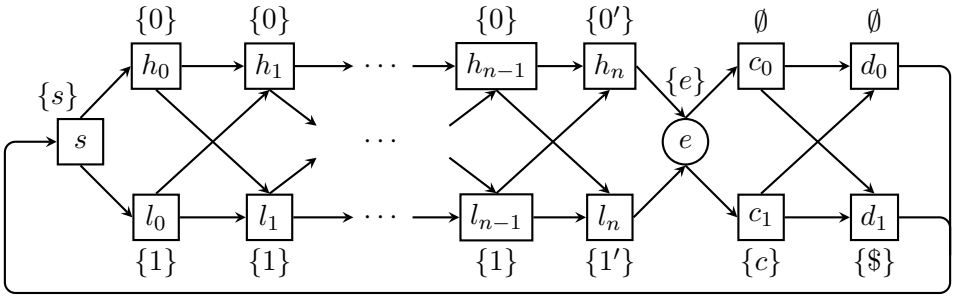

Figure 3.5: The arena $\mathcal{A}_n$ for Theorem 3.43

We begin the proof of Theorem 3.43 by fixing an $n \geq 1$. The arena $\mathcal{A}_n$ is depicted in Figure 3.5.

A play in this arena proceeds as follows: beginning in the initial vertex d_1 the token is moved to the vertex s. From here onwards, Player 1 has to specify n bits and one primed bit by moving through the vertices h_m or l_m for $0 \leq m \leq n$ and then has to move the token to vertex e. From here, Player 0 can choose to satisfy c or choose to not satisfy c. Afterwards, Player 1 can choose to satisfy \$ or choose to not satisfy \$ and then has to move the token back to s. Then, the process of specifying the bits and whether c and \$ hold is repeated. Hence, the trace $w = \mathrm{tr}(\rho)$ of a play has the form

$$\begin{aligned} w = &\{\$\}\{s\}\{b_0^0\}\cdots\{b_{n-1}^0\}\{b_n^0\}\{e\}C_0D_0 \\ &\{s\}\{b_0^1\}\cdots\{b_{n-1}^1\}\{b_n^1\}\{e\}C_1D_1 \\ &\{s\}\{b_0^2\}\cdots\{b_{n-1}^2\}\{b_n^2\}\{e\}C_2D_2\cdots \end{aligned}$$

where each b_m^j with m in the range $0 \leq m \leq n-1$ is either 0 or 1, each b_n^j is either $0'$ or $1'$, each C_j is either $\{c\}$ or $\emptyset$, and each D_j is either $\{\$\}$ or $\emptyset$. We interpret the sequence $b_0^j \cdots b_{n-1}^j$ as (big endian[13]) binary encoding of a number $c_j \in \{0, \ldots, 2^n - 1\}$. Note that we do not use the primed bits b_n^j to define c_j. These are the bits for the counter d and are discussed below.

To give the winning condition φ_n, we introduce some simplifying notation. For $j \in \mathbb{N}$, we define

$$\mathbf{X}^j = \underbrace{\mathbf{X}\cdots\mathbf{X}}_{j \text{ times}} .$$

Hence, we have $(w, n, \alpha) \models \mathbf{X}^j\varphi$ if and only if $(w, n+j, \alpha) \models \varphi$, i.e., φ holds j positions later. For a *word* of propositions $p_0 \cdots p_k \in P^+$, we define the formula $\gamma_{p_0\cdots p_k}$ inductively by $\gamma_{p_0} = p_0$ and for $k \geq 1$ by

$$\gamma_{p_0\cdots p_k} = p_0 \wedge \mathbf{X}\gamma_{p_1\cdots p_k} .$$

[13] That is, b_0 is the most significant bit.

We have $(w, n, \alpha) \models \gamma_{p_0 \cdots p_k}$ if and only if for each j in the range $0 \leq j \leq k$ the proposition p_j holds at position $n + j$.

We continue by defining some LTL formulae and by discussing the form of the plays of $\mathcal{A}_n$ that satisfy them. Here, we use the implication $\delta \rightarrow \eta$ as shorthand for the formula $\neg\delta \vee \eta$, provided that δ is an LTL formula. In this case, $\neg\delta$ is equivalent to an LTL formula (see Remark 3.7. Building on this, we use $\delta \leftrightarrow \eta$ as shorthand for $(\delta \rightarrow \eta) \wedge (\eta \rightarrow \delta)$, provided that both δ and η are LTL formulae.

We begin by expressing some requirements on the bits produced by Player 1 during a play. If w is a model of the formula $\psi_1 = \mathbf{GF}\$$, then there are infinitely many j such that $D_j = \{\$\}$. In this case, we interpret the primed bits between the j-th and the $(j+1)$-st occurrence of $\{\$\}$ as (big endian) binary encoding of a natural number d_ℓ. At the moment, we cannot bound the size of these numbers, since there is no bound on the distance between the occurrence of the dollars. Such a bound is enforced by the conjunction of the next three formulae, which require the numbers c_j between two occurrences of a dollar to implement a binary counter. The formula $\psi_2 = \mathbf{G}\,(\$ \rightarrow \mathbf{X}\gamma_{s0^n})$ formalizes the initialization of the counter: after each dollar, the next number c_j is zero. Dually, the formula $\psi_3 = \mathbf{G}\left(\gamma_{s1^n} \rightarrow \mathbf{X}^{n+4}\$\right)$ formalizes the reset of the counter: if a number c_j is equal to $2^n - 1$, then it is followed by a dollar (which has to be followed by a zero due to φ_2). Finally, we have to implement the increment of the counter: if a number c_j is strictly smaller than $2^n - 1$, then we have $c_{j+1} = c_j + 1$. This is formalized by the formula

$$\psi_4 = \mathbf{G}\,((s \wedge \neg\gamma_{s1^n}) \rightarrow \mathbf{X}\psi_{\text{inc}})$$

where

$$\begin{aligned}\psi_{\text{inc}} = \bigwedge_{j=0}^{n-1} & \mathbf{X}^j(\gamma_{1^{n-j}} \rightarrow \mathbf{X}^{n+5}0) \wedge \\ & \mathbf{X}^j(\gamma_{01^{n-(j+1)}} \rightarrow \mathbf{X}^{n+5}1) \wedge \\ & \mathbf{X}^j((\neg\gamma_{1^{n-j}} \wedge \neg\gamma_{01^{n-(j+1)}}) \rightarrow (1 \leftrightarrow \mathbf{X}^{n+5}1))\ .\end{aligned}$$

Here, we again use the negation of an LTL as shorthand to simplify our presentation. Let us quickly recall how a (big-endian) binary number is incremented: going from right to left, each one is flipped to a zero until the first zero occurs, which is flipped to a one. All bits to the left of the first zero (from the right) are left unchanged. This is expressed by the formula ψ_{inc}: if there is a one at a position and only ones to the right, then the next number has a zero at this position. If a position is the first zero (from the right), then the corresponding bit of the next number is a one. Any other number is unchanged, which happens exactly if there is a zero to the right of this bit.

If we have $w \models \psi_1 \wedge \psi_2 \wedge \psi_3 \wedge \psi_4$, then infinitely many dollars occur in w and there are exactly 2^n primed bits between each adjacent pair of dollars. Hence, we have $0 \leq d_\ell \leq 2^{2^n} - 1$ for every ℓ.

All previous formulae express requirements on Player 1's behavior since he is in charge of producing the c_j and d_ℓ. Now, we turn our attention to Player 0. Her only task is to decide whether to move to c_0 or to c_1, thereby producing a position at which the proposition c holds or does not hold. The formula $\psi_5 = \mathbf{G}(c \rightarrow \mathbf{XG}\neg c)$ is satisfied, if there is at most one position at which c holds. As last formula, consider $\psi_6 = \mathbf{F}\psi_6'$ where

$$\begin{aligned}\psi_6' = {} & s \wedge (\neg\$ \,\mathbf{U}\, (\$ \wedge \neg\$ \,\mathbf{U}\, c)) \wedge \\ & \mathbf{X}(\bigwedge_{j=0}^{n-1} (\mathbf{X}^j 0 \leftrightarrow \mathbf{F}(0 \wedge \mathbf{X}^{(n-j)+2} c))) \wedge \\ & (\psi_{\max} \vee \psi_{\text{faulty}-\text{inc}})\end{aligned}$$

where

$$\psi_{\max} = \neg\$ \,\mathbf{U}\, (\$ \wedge (\neg 0' \,\mathbf{U}\, \$))$$

and

$$\begin{aligned}\psi_{\text{faulty}-\text{inc}} = {} & \big[(\neg 0' \,\mathbf{U}\, \$) && \rightarrow \mathbf{F}\big(1' \wedge \mathbf{X}^2 c\big)\big] \wedge \\ & \big[\big((\neg 0' \wedge \neg 1') \,\mathbf{U}\, \big(0' \wedge (\neg 0' \,\mathbf{U}\, \$)\big)\big) && \rightarrow \mathbf{F}\big(0' \wedge \mathbf{X}^2 c\big)\big] \wedge \\ & \big[\big((\neg 0' \wedge \neg 1') \,\mathbf{U}\, \big(0' \wedge (\neg\$ \,\mathbf{U}\, 0')\big)\big) && \rightarrow \mathbf{F}\big(1' \wedge \mathbf{X}^2 c\big)\big] \wedge \\ & \big[\big((\neg 0' \wedge \neg 1') \,\mathbf{U}\, \big(1' \wedge (\neg\$ \,\mathbf{U}\, 0')\big)\big) && \rightarrow \mathbf{F}\big(0' \wedge \mathbf{X}^2 c\big)\big]\end{aligned} .$$

Let us dissect the formula ψ_6: assume we have $(w, \alpha) \models \psi_6$, i.e., there exists a position m such that $(w, m, \alpha) \models \psi_6'$. At this position, s holds true. Hence, the next n positions encode a number c_j. Furthermore, after the next dollar, c holds true at least once before the next dollar occurs. If we assume ψ_5 to be satisfied by w, then this is the only c occurring in w. This c is preceded by another sequence of bits which encode a number $c_{j'}$. The next subformula of ψ_6' requires these values to be equal: here, we use the fact that at these positions either 0 or 1 holds, but not both at the same time. Hence, the next primed bit after the position n is the c_j-th bit of some number d_ℓ and the primed bit two positions before the position at which c holds is the c_j-th bit of $d_{\ell+1}$. The final disjunction is satisfied, if these primed bits witness $d_{\ell+1} \neq d_\ell + 1$ (by the disjunct $\psi_{\text{faulty}-\text{inc}}$) or if we have $d_{\ell+1} = 2^{2^n} - 1$ (by the disjunct $\psi_{\max}$): $\psi_{\max}$ is satisfied, if there is no primed zero between the next two dollars, which implies $d_{\ell+1} = 2^{2^n} - 1$, since we still assume $\psi_1 \wedge \psi_2 \wedge \psi_3 \wedge \psi_4$ to be satisfied. Now consider the conjuncts of $\psi_{\text{faulty}-\text{inc}}$: the first one is satisfied if the bits right of (including) the c_j-th one of d_ℓ are all ones, but the c_j-th bit of $d_{\ell+1}$ is not flipped to zero. The second one

is satisfied if the bits right of (excluding) the c_j-th one of d_ℓ, which is zero, are all ones, but the c_j-th bit of $d_{\ell+1}$ is not flipped to one. The last two formulae are symmetric, thus we only explain the third one: it is satisfied, if the c_j-th bit of d_ℓ is a zero and is followed by another zero before the next dollar occurs, and the c_j-th bit of $d_{\ell+1}$ is flipped. Thus, ψ'_6 is indeed satisfied at a position m if the next primed bit and the primed bit before the (only) occurrence of c witness $d_{\ell+1} \neq d_\ell + 1$ or if we have $d_{\ell+1} = 2^{2^n} - 1$.

Let us wrap things up and prove Theorem 3.43.

Proof. Consider the game $\mathcal{G}_n = (\mathcal{A}_n, d_1, \varphi_n)$ with

$$\varphi_n = (\psi_1 \wedge \psi_2 \wedge \psi_3 \wedge \psi_4) \rightarrow (\psi_5 \wedge \psi_6 \wedge \mathbf{F}_{\leq x} c) \ .$$

The arena has $2n + 8$ vertices and the size of φ_n is quadratic in n.

Next, we show that $\mathcal{W}_0(\mathcal{G}_n)$ is non-empty. Let w be the trace of a play of $\mathcal{G}_n$. If it does not satisfy $\psi_1 \wedge \psi_2 \wedge \psi_3 \wedge \psi_4$, then it is winning for Player 0. So, assume we have $w \models \psi_1 \wedge \psi_2 \wedge \psi_3 \wedge \psi_4$. Then, w has the form as described above: the c_j's count from 0 to $2^n - 1$ and the numbers d_ℓ are in the range $0 \leq d_\ell \leq 2^{2^n} - 1$. In this situation, Player 0 has to ensure that c holds exactly once, at a position as described above: either after $d_\ell = 2^{2^n} - 1$ or after a primed bit that witnesses a faulty increase by Player 1. Player 0 is always able to find such a position since Player 1 can produce at most 2^{2^n} numbers d_ℓ without introducing a faulty increment. Hence, Player 0 wins $\mathcal{G}_n$ with respect to some α. On the other hand, by always incrementing the d_ℓ correctly until they reach $2^{2^n} - 1$, Player 1 is able to win $\mathcal{G}_n$ with respect to (at least) every α such that $\alpha(x) \leq n \cdot 2^n \cdot 2^{2^n}$, since there are 2^{2^n} values for d, each having 2^n bits which are encoded by one round through the arena, each of which visits more than n vertices. □

Recall that Theorem 3.34 can also be stated in terms of $\mathrm{PLTL}_\mathbf{G}$ games: for every $\mathrm{PLTL}_\mathbf{G}$ game $\mathcal{G}$, there is a $k \in 2^{2^{\mathcal{O}(|\mathcal{G}|)}}$ such that if Player 0 wins $\mathcal{G}$ with respect to the variable valuation mapping every variable to k, then $\mathcal{W}_0(\mathcal{G})$ is actually universal. Due to duality, this bound must be asymptotically tight as well: the following corollary is obtained by formulating Theorem 3.43 for the $\mathrm{PLTL}_\mathbf{G}$ game $\overline{\mathcal{G}_n}$.

Corollary 3.44. *For every $n \geq 1$, there exists a $\mathrm{PLTL}_\mathbf{G}$ game $\mathcal{G}_n$ with winning condition φ_n with $|\mathcal{G}_n| \in \mathcal{O}(n^2)$ and $\mathrm{var}(\varphi_n) = \{y\}$ such that $\mathcal{W}_0(\mathcal{G}_n)$ is not universal, but Player 0 wins $\mathcal{G}_n$ with respect to every variable valuation α with $\alpha(y) \leq 2^{2^n}$.*

The doubly-exponential lower bound on an optimal variable valuation in a PROMPT–LTL game presented in this subsection show that we cannot prove the optimization problems for unipolar formulae to be solvable in doubly-exponential time using the technique presented in Section 3.3.

3.4 Summary of Results

We have shown the membership, emptiness, finiteness, and universality problem for PLTL games to be **2Exptime**-complete. Thus, these problems are not harder than solving LTL games without parameterized operators. Furthermore, all but the finiteness problem for $\text{PLTL}_{\mathbf{G}}$ games can be reduced to solving a single LTL game. These results are in line with results on model-checking PLTL [AETP01] and model-checking parameterized real-time logics [GTN10]: on both cases, the addition of parameterized operators does not increase the asymptotic complexity of solving the model-checking problem.

This has to be contrasted with the status of the PLTL optimization problems for which we presented an algorithm with triply-exponential running time. The optimization problems for PLTL model-checking can be solved in polynomial space, hence are not harder than LTL model-checking. It is open whether the optimization problems for games can also be solved in doubly-exponential time. We have complemented our algorithm for these problems by a doubly-exponential lower bound on the value of an optimal variable valuation for a unipolar game.

Furthermore, we have shown that the sets of winning valuations for both players in a unipolar game are semilinear. The same applies to the projection of the sets of winning valuations in a non-unipolar game to the variables in $\text{var}_{\mathbf{F}}(\varphi)$ or $\text{var}_{\mathbf{G}}(\varphi)$, respectively. The structural complexity of the full set of valuations is open, as it is already the case for PLTL model-checking.

Finally, by combining the alternating color technique and the construction of "small" automata for PLTL formulae with respect to a fixed variable valuation, one can even show that adding operators like $\mathbf{F}_{\leq k}$ and $\mathbf{G}_{\leq k}$, where $k \in \mathbb{N}$ is an arbitrary, but constant bound (encoded in binary), does not increase the complexity of the decision and optimization problems.

Chapter 4

Playing Muller Games in Finite Time

How can an infinite game be played in finite time? The winner of a single play is typically only determined after ω steps, e.g., the winner of a play in a Muller game depends on the infinity set of the play. In the following, we change the rules of the game by defining a criterion that stops a play after a bounded number of steps and then declares a winner. Here, we allow a finite play to be declared winning for Player i although there might be continuations of this play that are winning for Player $1 - i$ in the original infinite game. Thus, we cannot hope to prove an equivalence between these games on the level of plays[14], but we can ask for an equivalence on the level of the existence of winning strategies: a criterion is sound, if Player i has a winning strategy from a vertex v in the infinite game if and only if she has a winning strategy from v in the finite-duration version. Thus, a sound criterion allows us to determine the winning region of an infinite game by solving a finite game.

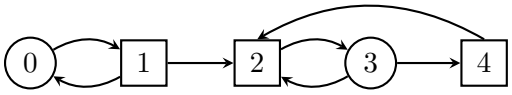

Figure 4.1: The arena $\mathcal{A}$ for the introductory example in Chapter 4

As an introductory example, consider the reachability game $(\mathcal{A}, F)$ where $\mathcal{A}$ is depicted in Figure 4.1 and $F = \{4\}$. Recall that Player 0 wins a play in a reachability game if and only if at least one vertex from F is visited. The following criterion is sound for reachability games: stop a play after $|\mathcal{A}|$

[14]For Muller games (and many other games), the existence of such an equivalence is even ruled out by the fact that the set of winning plays in a Muller game is on a higher level of the Borel hierarchy than the one in a safety game. We discuss this in depth in Chapter 5.

steps and declare Player 0 to be the winner if and only if a vertex from F is visited. In the example, the play 12342 is winning for Player 0, while the play 12323 is winning for Player 1, although Player 0 could move from the last vertex to vertex 4 in the next step. However, she could have done so already when the play visited vertex 3 for the first time. By always moving from 3 to 4 she wins the finite-duration game from the set $\{2, 3, 4\}$, which is exactly her winning region in the original reachability game. The existence of positional attractor strategies, which visit each vertex in the attractor at most once before reaching F, and the existence of trap strategies, which prevent the token from ever reaching F when starting in the complement of the attractor, show this criterion to be sound for reachability games.

McNaughton considered the problem of playing infinite games in finite time to make them suitable for "casual living room recreation" [McN00]. As human players cannot play infinitely long, he envisions a referee (a criterion in our parlance) that stops a play at a certain time and declares a winner. Since it is straightforward to give a sound criterion for positionally determined games[15], McNaughton considered Muller games. To this end, he defined for every set of vertices F a scoring function Sc_F which keeps track of the number of times the set F is visited entirely since the last visit of a vertex that is not in F. To illustrate the scores of a play prefix, consider the arena $\mathcal{A}$ in Figure 4.1. The score for the set $\{2, 3, 4\}$ of the play prefix 010123423234 is two, as it is visited twice since the last occurrence of a vertex that is not in $\{2, 3, 4\}$ (i.e., with the infixes 234 and 23234). After the prefix 0101, the score for the set $\{0, 1\}$ is two, but it is reset to zero by the occurrence of vertex 2. In an infinite play, the set of vertices seen infinitely often is the unique set F such that Sc_F tends to infinity after being reset to zero only a finite number of times. Hence, the winner of a play in a Muller game can be characterized by the evolution of the scores, albeit still only after ω steps.

McNaughton proved the following criterion to be sound [McN00]: stop a play after a score of $|F|! + 1$ for some set F is reached for the first time, and declare the winner to be the Player i such that $F \in \mathcal{F}_i$. However, it can take a large number of steps for a play to reach a score of $|F|! + 1$, as scores may increase slowly or be reset to zero.

In this chapter, we improve McNaughton's result by showing that stopping a play as soon as a score of three is reached for the first time is sound. To prove this, we construct a winning strategy that bounds the losing player's scores by two. This is complemented by examples showing that both results are optimal and by tight upper and lower bounds on the play length. Finally, we discuss some obstacles one encounters when trying to play infinite games in infinite arenas in finite time.

[15] Stop a play as soon as a vertex is visited for the second time and declare the winner of the finite play to be the winner of the infinite play induced by the cycle.

4.1 Scoring Functions

In this section, we formally introduce scoring functions for Muller games, which were originally defined by McNaughton to make infinite games playable in finite time. For every set of vertices F, the score of a play prefix w is the number of times F has been visited entirely since w visited for the last time a vertex that is not in F. An occurrence of such a vertex is referred to as a reset of the score for F. We extend the scores by the concept of an accumulator, which can be seen as the fractional part of a score: in order to increase the score for a set F, all vertices in F have to be visited at least once. The accumulator of the set F measures the progress that has been made towards the next increase by accumulating the vertices of F that were visited since the last increase or reset, depending on which happened later. McNaughton implicitly used accumulators to compute the scores inductively, but was only interested in the scores of a play. This is in contrast to our work which relies on accumulators as well, since they give a much finer description of a play prefix.

McNaughton defined the score of a play prefix w by a maximization over decompositions of w and gave a greedy recursive algorithm to compute scores on the fly. This algorithm can easily be turned into an inductive definition. Since it conveys more intuition and allows to introduce scores and accumulators at the same time, we prefer to begin with this definition and then show its equivalence to the original definition.

Let us fix some set V of vertices throughout this section. For every $F \subseteq V$ we inductively define the functions $\mathrm{Sc}_F \colon V^* \to \mathbb{N}$ and $\mathrm{Acc}_F \colon V^* \to 2^F$ as follows: the score for the empty set is always zero and its accumulator is always empty, i.e., we define $\mathrm{Sc}_\emptyset(w) = 0$ and $\mathrm{Acc}_\emptyset(w) = \emptyset$ for every $w \in V^+$.

Now, we consider non-empty sets. For the empty word, all scores are zero and all accumulators are empty. Now, consider a word wv with $w \in V^*$ and $v \in V$. If v is not in F, then the score and the accumulator for F are reset, i.e., for $v \notin F$ we define $\mathrm{Sc}_F(wv) = 0$ and $\mathrm{Acc}_F(wv) = \emptyset$. If v is in F, then we have to consider two subcases: if the accumulator $\mathrm{Acc}_F(w)$ is not equal to $F \setminus \{v\}$ (i.e., some vertex $v' \neq v$ with $v' \in F$ has not occurred since the last score increase or reset), then the score is left unchanged, but v is added to the accumulator (in which it might already be contained): if $v \in F$ and $\mathrm{Acc}_F(w) \neq (F \setminus \{v\})$, then $\mathrm{Sc}_F(wv) = \mathrm{Sc}_F(w)$ and $\mathrm{Acc}_F(wv) = \mathrm{Acc}_F(w) \cup \{v\}$.

On the other hand, if the accumulator $\mathrm{Acc}_F(w)$ contains all elements but v, then all elements of F occur at least once since the last score increase or since the last reset. In this situation, the score is increased and the accumulator is set to $\emptyset$ accordingly: if $v \in F$ and $\mathrm{Acc}_F(w) = (F \setminus \{v\})$, then $\mathrm{Sc}_F(wv) = \mathrm{Sc}_F(w) + 1$ and $\mathrm{Acc}_F(wv) = \emptyset$. We summarize the previous explanations in the following definition.

Definition 4.1. For every $F \subseteq V$ we inductively define the scoring function $\mathrm{Sc}_F \colon V^* \to \mathbb{N}$ and the accumulator function $\mathrm{Acc}_F \colon V^* \to 2^F$ as follows: $\mathrm{Sc}_F(\varepsilon) = 0$ and $\mathrm{Acc}_F(\varepsilon) = \emptyset$ and for $w \in V^*$ and $v \in V$

$$\mathrm{Sc}_F(wv) = \begin{cases} 0 & \text{if } F = \emptyset, \\ 0 & \text{if } F \neq \emptyset \text{ and } v \notin F, \\ \mathrm{Sc}_F(w) & \text{if } v \in F \text{ and } \mathrm{Acc}_F(w) \neq (F \setminus \{v\}), \\ \mathrm{Sc}_F(w) + 1 & \text{if } v \in F \text{ and } \mathrm{Acc}_F(w) = (F \setminus \{v\}), \end{cases}$$

and

$$\mathrm{Acc}_F(wv) = \begin{cases} \emptyset & \text{if } F = \emptyset, \\ \emptyset & \text{if } F \neq \emptyset \text{ and } v \notin F, \\ \mathrm{Acc}_F(w) \cup \{v\} & \text{if } v \in F \text{ and } \mathrm{Acc}_F(w) \neq (F \setminus \{v\}), \\ \emptyset & \text{if } v \in F \text{ and } \mathrm{Acc}_F(w) = (F \setminus \{v\}). \end{cases}$$

By definition, the score for the empty set is always zero. This allows us to formulate the following remark which states that the sets F that can reach a score of two are exactly the loops of the arena, i.e., potential infinity sets that determine the winner of a play.

Remark 4.2. Let $\mathcal{A}$ be an arena and let F a subset of its vertices. There exists a play prefix w in $\mathcal{A}$ with $\mathrm{Sc}_F(w) \geq 2$ if and only if F is a loop in $\mathcal{A}$.

Finally, for every family of sets $\mathcal{F} \subseteq 2^V$ we define the maximum score function which maps a (finite or infinite) word w to the highest score that is reached during w for a set in $\mathcal{F}$, and ∞, if arbitrarily large scores are reached. By convention, we set $\mathrm{MaxSc}_\emptyset(w) = 0$ for every finite or infinite word w.

Definition 4.3. Let $\mathcal{F} \subseteq 2^V$. We define $\mathrm{MaxSc}_\mathcal{F} \colon V^* \cup V^\omega \to \mathbb{N} \cup \{\infty\}$ by

$$\mathrm{MaxSc}_\mathcal{F}(w) = \max_{F \in \mathcal{F}} \sup_{w' \sqsubseteq w} \mathrm{Sc}_F(w') \ .$$

We illustrate these definitions in the following example.

Example 4.4. Consider $w = 12210122$ and $F = \{1, 2\}$. We have $\mathrm{Sc}_F(w) = 1$, because 122 is the longest suffix of w that is contained in F, and the entire set $\{1, 2\}$ occurs once during this suffix. We have $\mathrm{Acc}_F(w) = \{2\}$, because only vertex 2 occurs after the score for F increased to 1. On the other hand, we have $\mathrm{MaxSc}_{\{F\}}(w) = 2$ because the prefix $w' = 1221$ of w has $\mathrm{Sc}_F(w') = 2$. By visiting the vertex 0 the score for F is reset to 0, e.g., we have $\mathrm{Sc}_F(12210) = 0$. ◇

The evolution of the scores during a play allows to determine the winner of an infinite play ρ. First, let us consider the score for the set $\mathrm{Inf}(\rho)$. There is a position n of ρ such that only vertices in $\mathrm{Inf}(\rho)$ are visited after n. Hence, the score for $\mathrm{Inf}(\rho)$ is never reset after position n. This means that the score for $\mathrm{Inf}(\rho)$ tends to infinity, since each vertex in $\mathrm{Inf}(\rho)$ is visited again and again. On the other hand, for a set $F \neq \mathrm{Inf}(\rho)$, we distinguish two cases. If there is a vertex $v \in \mathrm{Inf}(\rho) \setminus F$, then the score for F is reset to 0 infinitely often. On the other hand, a vertex $v \in F \setminus \mathrm{Inf}(\rho)$ is occurs only finitely often, say k times. Thus, the score for F is bounded by k throughout ρ. Hence, we can characterize the infinity set of a play by the limit inferior of the scores.

Remark 4.5. Let $\rho \in V^\omega$. There is a unique set $F \subseteq V$ such that

$$\liminf_{n\to\infty} \mathrm{Sc}_F(\rho_0 \cdots \rho_n) = \infty \ .$$

Furthermore, this set F is the infinity set of ρ.

The following example shows that there can be more than one set whose score tends to infinity. Hence, we cannot replace the limit inferior by the maximal score function in Remark 4.5.

Example 4.6. Consider the play

$$\rho = 012\,210\,012\,32 \cdot \prod_{n\geq 1} (0^n 1) = 012\,210\,012\,32\,01\,001\,0001\,00001\cdots$$

with infinity set $\{0,1\}$. The set $\{0\}$ reaches a score of n during each infix $0^{n'}$ for $n' > n$, but it is also reset after each such infix by the occurrence of 1. Hence, we have $\mathrm{MaxSc}_{\{0\}}(\rho) = \infty$, but $\liminf_{n\to\infty} \mathrm{Sc}_{\{0\}}(\rho_0 \cdots \rho_n) = 0$. The score for the infinity set $\{0,1\}$ tends to infinity while being reset only finitely often. Hence, we have $\mathrm{MaxSc}_{\{0,1\}}(\rho) = \liminf_{n\to\infty} \mathrm{Sc}_{\{0,1\}}(\rho_0 \cdots \rho_n) = \infty$. Finally, the score for the set $\{0,1,2\}$ increases to three, is reset by the occurrence of the vertex 3 and then increases to one again, after which it is never increased or reset again. Therefore, we have $\mathrm{MaxSc}_{\{0,1,2\}}(\rho) = 3$ and $\liminf_{n\to\infty} \mathrm{Sc}_{\{0,1,2\}}(\rho_0 \cdots \rho_n) = 1$. ◇

As already mentioned in the introductory paragraph, McNaughton originally defined the scores of w by a maximization over decompositions of w and presented the inductive definition above as an algorithm to compute the scores. Furthermore, the accumulators were originally defined by decompositions as well [FZ12]. In the following, we show these definitions to be equivalent, a task that was (for the case of scores) left as an exercise to the reader in [McN00].

Lemma 4.7. *Let $F \subseteq V$ and $w \in V^*$.*

i. $\mathrm{Sc}_F(w)$ *is the maximal $k \in \mathbb{N}$ such that a suffix $x_1 \cdots x_k$ of w exists, where every x_j is a non-empty word satisfying* $\mathrm{Occ}(x_j) = F$.

ii. $\mathrm{Acc}_F(w)$ *is the occurrence set* $\mathrm{Occ}(x)$ *of the longest suffix* x *of* w *such that the score for* F *does not change and is not reset throughout* x*, i.e., we have* $\mathrm{Sc}_F(w) = \mathrm{Sc}_F(wy^{-1})$ *for every suffix* y *of* x*, and we have* $\mathrm{Occ}(x) \subseteq F$*.*

Proof. The claims are trivially true for $w = \varepsilon$. Thus, let $w = w_0 \cdots w_n$.

i.) Let $s = \mathrm{Sc}_F(w)$ and let k be maximal such that a suffix $x_1 \cdots x_k$ of w exists where every x_j is a non-empty word satisfying $\mathrm{Occ}(x_j) = F$. We show $s \leq k$ and $s \geq k$.

Let n_0 be maximal such that $w_{n_0} \notin F$ (and $n_0 = -1$ if no such letter exists) and for each j in the range $1 \leq j \leq s$ let $n_j > n_0$ be minimal such that $\mathrm{Sc}_F(w_0 \cdots w_{n_j}) = j$. The position n_0 is the last reset of the score for F (and -1 if there is no reset during w) and n_j denotes the position after n_0 where the score for F is increased to j. We have $n_0 < n_1 < \cdots < n_s$. Now define $x_j = w_{n_{j-1}+1} \cdots w_{n_j}$ for every j in the range $1 \leq j < s$ and $x_s = w_{n_{s-1}+1} \cdots w_n$. Since each vertex of F has to occur at least once to increase the score between $n_{j-1} + 1$ and n_j, we have $\mathrm{Occ}(x_j) = F$ for every j. Hence, we have shown $s \leq k$.

On the other hand, let $x_1 \cdots x_k$ be a suffix of w such that every x_j is a non-empty word satisfying $\mathrm{Occ}(x_j) = F$ and let $w(x_1 \cdots x_k)^{-1} = y$ be the prefix of w obtained by removing the x_j. Then, we have $\mathrm{Sc}_F(yx_1 \cdots x_j) \geq j$ for every j and hence $s \geq k$.

ii.) Let x be the longest suffix of w satisfying $\mathrm{Sc}_F(w) = \mathrm{Sc}_F(wy^{-1})$ for every suffix y of x, and $\mathrm{Occ}(x) \subseteq F$. If $x = \varepsilon$, then the score for F is increased or reset by visiting w_n and we have $\mathrm{Acc}_F(w) = \emptyset = \mathrm{Occ}(\varepsilon)$.

Otherwise, let $x = w_m \cdots w_n$ for some $m \leq n$. Since $\mathrm{Occ}(x) \subseteq F$ and since the score for F is increased or reset for the last time at position $m - 1$ (or not at all if $m = 0$), we have $\mathrm{Acc}_F(w_0 \cdots w_{m+j}) = \{w_m, \ldots, w_{m+j}\}$ for every j in the range $0 \leq j \leq n - m$. Thus, $\mathrm{Acc}_F(w) = \{w_m, \ldots, w_n\} = \mathrm{Occ}(x)$. □

A simple consequence of these characterizations is that sets with non-zero score and the accumulators for all sets are all pairwise comparable in the subset relation. For the sets with non-zero score, this was already proven by McNaughton. However, it turns out to be useful to consider also the accumulators.

Lemma 4.8 (cf. Theorem 4.2 of [McN00])**.** *Let* $w \in V^*$*. The sets* $F \subseteq V$ *with* $\mathrm{Sc}_F(w) \geq 1$ *together with the sets* $\mathrm{Acc}_F(w)$ *for some* $F \subseteq V$ *form a chain in the subset relation.*

Proof. It suffices to show that all such sets are pairwise comparable: let F and F' be two sets such that either $\mathrm{Sc}_F(w) \geq 1$ or $F = \mathrm{Acc}_H(w)$ for some $H \subseteq V$ and either $\mathrm{Sc}_{F'}(w) \geq 1$ or $F' = \mathrm{Acc}_{H'}(w)$ for some $H' \subseteq V$.

Lemma 4.7 implies the existence of two decompositions $w = w_0 w_1$ and $w = w_0' w_1'$ with $\mathrm{Occ}(w_1) = F$ and $\mathrm{Occ}(w_1') = F'$. Now, either w_1 is a suffix

of w_1' or vice versa. In the first case, we have $F \subseteq F'$ and in the second case $F' \subseteq F$. □

Lemma 4.8 implies that there can be at most $|V|$ sets that have a non-zero score at the same time.

4.2 Finite-time Muller Games

McNaughton used scores to define a finite-duration variant of Muller games. A play ρ is stopped as soon as the score for some set F reaches a given threshold score $\mathrm{Tr}(F)$. If $F \in \mathcal{F}_i$, then Player i is declared to be the winner of ρ. To obtain a zero-sum finite-duration game, infinite plays that never reach a threshold score for some set and also plays in which two sets reach their threshold score at the same time have to be ruled out. McNaughton mentions that every infinite play exceeds every threshold eventually and proves that no two scores can reach their threshold at the same time, provided the thresholds are non-trivial. Before we introduce finite-time Muller games we give a tight bound on the length of a play that avoids a certain score, thereby quantifying McNaughton's claim, and explain why there is always a unique winner. Then, we introduce finite-time Muller games and discuss some basic properties.

Lemma 4.9. *Let V be a finite non-empty set of vertices.*

i. For every $w \in V^+$ and $k > 0$, if $|w| \geq k^{|V|}$, then $\mathrm{MaxSc}_{2^V}(w) \geq k$.

ii. For every $k > 1$, there exists a $w \in V^+$ with $|w| = k^{|V|} - 1$ and $\mathrm{MaxSc}_{2^V}(w) < k$.

Proof. i.) We show by induction over the cardinality of the set V that every word $w \in V^+$ with $|w| \geq k^{|V|}$ contains an infix x that can be decomposed into parts $x_1, \ldots, x_k$ such that $x = x_1 \cdots x_k$ where every x_j is a non-empty word with $\mathrm{Occ}(x_j) = \mathrm{Occ}(x)$. This implies $\mathrm{MaxSc}_{2^V}(w) \geq k$.

The claim holds trivially for $|V| = 1$ by choosing x to be the prefix of w of length $k \leq |w|$ and $x_j = v$ for the single vertex $v \in V$. For the induction step, consider a set V with $n+1$ vertices and a word w with $|w| \geq k^{n+1}$. If w contains an infix y of length k^n which contains at most n distinct vertices, then we can apply the induction hypothesis to y and obtain a decomposition of an infix x of y with the desired properties. Since x is also an infix of w, we are done. Otherwise, every infix of w of length k^n contains every vertex of V at least once. Let x be the prefix of length k^{n+1} of w and let $x = x_1 \cdots x_k$ be the decomposition of x such that each x_j is of length k^n. Then, we have $\mathrm{Occ}(x_j) = \mathrm{Occ}(x) = V$ for every j. Therefore, the decomposition has the desired properties.

ii.) We fix some threshold $k > 1$ and assume without loss of generality $V = [n] = \{0, \ldots, n-1\}$. We inductively define words w_n over the alphabet $[n]$ satisfying $|w_n| = k^n - 1$ and $\mathrm{MaxSc}_{2^{[n]}}(w_n) < k$.

To this end, let $w_1 = 0^{k-1}$ and for $n > 1$ let

$$w_n = (w_{n-1} \cdot (n-1))^{k-1} \cdot w_{n-1} \ .$$

Note that an occurrence of the letter $(n-1)$ in w_n resets all non-zero scores for sets $F \subseteq [n-1] = \mathrm{Occ}(w_{n-1})$ that were reached during the infixes w_{n-1}. Furthermore, as $(n-1)$ appears only $k-1$ times, it can also not contribute to a score of k during w_n.

Let us verify our claims formally. We have $|w_1| = k^1 - 1$ and for $n > 1$

$$|w_n| = k \cdot |w_{n-1}| + (k-1) = k \cdot (k^{n-1} - 1) + (k-1) = k^n - 1 \ .$$

Finally, we show $\mathrm{MaxSc}_{2^{[n]}}(w_n) < k$ by induction over n. The induction start is immediate, as we have

$$\mathrm{MaxSc}_{2^{[1]}}(0^{k-1}) = \mathrm{Sc}_{\{0\}}(0^{k-1}) = k - 1 < k \ .$$

So, assume $n > 1$. Since the letter $(n-1)$ only appears $k-1$ times in w_n, every set containing it never reaches a score of k during w_n. Finally, consider the sets $F \subseteq [n-1]$. Since their scores are reset by an occurrence of the letter $(n-1)$, the score for such a set F can only increase during an infix w_{n-1}. However, the induction hypothesis bounds the score for F during such an infix by $k-1$. Hence, the scores for all sets are bounded by $k-1$. □

Lemma 4.9 implies that a finite-time Muller game with maximal threshold score k must end after at most $k^{|V|}$ steps and shows that this bound is tight for the case of uniform threshold scores. Figure 4.2 depicts a solitary arena $\mathcal{A}_n$ with vertex set $[n]$ in which Player 1 is able to produce the words w_n (for any k) as defined in the proof of Lemma 4.9(ii) and thereby avoids a score of k as long as possible.

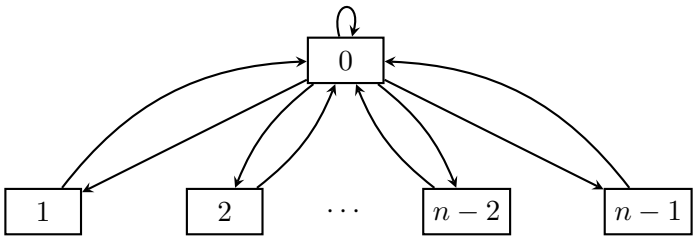

Figure 4.2: The arena $\mathcal{A}_n$ for the lower bounds on the play length

The arena $\mathcal{A}_n$ is very simple in terms of graph-theoretic complexity measures such as cycle rank [Egg63], treewidth [RS86], pathwidth [RS83],

cliquewidth [CO00], DAG-width [BDH+], Kelly-width [HK08], and entanglement [BG04]. Hence, the bound $k^{|V|}$ on the maximal play length cannot be improved by considering arenas of bounded complexity in any of these measures. However, the in- and outdegree of vertex 0 is n, hence as large as it can get in a graph with n vertices.

Hence, one might ask whether bounding the degree of the vertices leads to shorter plays, possibly in combination with bounding one of the complexity measures mentioned above. The arena $\mathcal{A}'_n$ depicted in Figure 4.3 shows that this is not the case. It is obtained by subdividing the self-loop in $\mathcal{A}_n$ of Figure 4.2 into a cycle of n vertices. Hence, the construction of the play prefix witnessing the lower bounds has to be adopted: fix some k. For every j we define a play prefix x_j in $\mathcal{A}'_n$ by $x_1 = (00')^{k-1}01\cdots(n-1)$ and for j in the range $1 < j \leq n$ by

$$x_j = (x_{j-1}01\cdots(j-1)(j-1)'(j-1)j\cdots(n-1))^{k-1}x_{j-1} \ .$$

A simple induction shows $|x_j| \geq k^j$ and one can prove $\mathrm{MaxSc}_{2^V}(x_j) < k$, where $V = \{0,\ldots,n-1,0',\ldots,n-1'\}$, using the same reasoning as in the proof of Lemma 4.9(ii). Thus, there is an arena $\mathcal{A}'_n$ with $2n$ vertices in which Player 1 can prevent a score of k for at least k^n steps by producing the play prefix x_n. To conclude, we mention that $\mathcal{A}'_n$ has not only maximal in- and outdegree two, but is still very simple in terms of the graph complexity measures listed above.

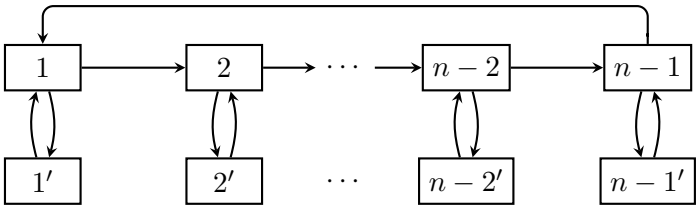

Figure 4.3: The arena $\mathcal{A}'_n$ with degree four for lower bounds on the play length

After we have ruled out draws due to infinite plays, we show that every play has a unique winner. To this end, we must exclude the case where two sets reach their threshold score at the same time. McNaughton proved that this cannot happen, provided the thresholds are non-trivial.

Lemma 4.10 ([McN00])**.** *Let $k, l \geq 2$, $F, F' \subseteq V$, $w \in V^*$ and $v \in V$ such that $\mathrm{Sc}_F(w) < k$ and $\mathrm{Sc}_{F'}(w) < l$. If $\mathrm{Sc}_F(wv) = k$ and $\mathrm{Sc}_{F'}(wv) = l$, then $F = F'$.*

For the sake of completeness, we give a proof of this result.

Proof. Towards a contradiction assume $F \neq F'$. By Lemma 4.8 we can assume without loss of generality $F' \subset F$, i.e., there exists some $q \in F \setminus F'$.

Then, $\mathrm{Sc}_F(wv) = k$ and $\mathrm{Sc}_{F'}(wv) = l$ imply the existence of decompositions $wv = x_0x_1 \cdots x_k$ and $wv = y_0y_1 \cdots y_l$ such that $\mathrm{Occ}(x_j) = F$ and $\mathrm{Occ}(y_j) = F'$ for every $j > 1$. As $q \notin F'$, $y_1 \cdots y_l$ is a proper suffix of x_k. The situation is sketched in Figure 4.4.

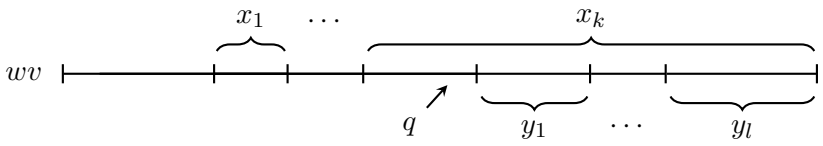

Figure 4.4: The decomposition of wv as in the proof of Lemma 4.10

Furthermore, as $\mathrm{Sc}_F(w) < k$ and $\mathrm{Sc}_F(wv) = k$, we have $v \notin \mathrm{Occ}(x_kv^{-1})$. If v would appear in x_kv^{-1}, then the decomposition $x_1 \cdots x_{k-1}(x_kv^{-1})$ would show $\mathrm{Sc}_F(w) \geq k$. However, we have $v \in F'$ and thus $v \in \mathrm{Occ}(y_{l-1})$, which is an infix of x_kv^{-1}. This yields the desired contradiction. □

The word 01 shows that the requirement of k and l being greater than one is necessary: by the occurrence of 1 after 0, the scores for $\{1\}$ and $\{0, 1\}$ are increased to one.

Now, we are able to define a finite-time Muller game to be a Muller game equipped with a threshold score function. To declare a unique winner by applying Lemma 4.10, we require the thresholds to be greater than one. Then, we can again restrict $\mathcal{F}_0$ and $\mathcal{F}_1$ to contain only loops of the arena, since only these can have a score greater than one (see Remark 4.2).

Definition 4.11. A finite-time Muller game $(\mathcal{A}, \mathcal{F}_0, \mathcal{F}_1, \mathrm{Tr})$ consists of an arena $\mathcal{A}$ with vertex set V, a set $\mathcal{F}_0 \subseteq 2^V$ of loops of $\mathcal{A}$, the set $\mathcal{F}_1$ of loops of $\mathcal{A}$ which are not in $\mathcal{F}_0$, and a threshold score function $\mathrm{Tr}\colon \mathcal{F}_0 \cup \mathcal{F}_1 \to \mathbb{N}\setminus\{0, 1\}$.

If the threshold function is constant, i.e., there exists a k such that $\mathrm{Tr}(F) = k$ for every $F \in \mathcal{F}_0 \cup \mathcal{F}_1$, then we write $(\mathcal{A}, \mathcal{F}_0, \mathcal{F}_1, k)$ instead of $(\mathcal{A}, \mathcal{F}_0, \mathcal{F}_1, \mathrm{Tr})$ and refer to the game as a finite-time Muller game with uniform threshold score k. We are mostly interested in these games.

By Lemma 4.9 we have that every infinite path must reach the threshold score for some set F after a bounded number of steps. Therefore, we define a play in the finite-time Muller game to be a finite path $\rho_0 \cdots \rho_n$ with $\mathrm{MaxSc}_F(\rho_0 \cdots \rho_n) = \mathrm{Tr}(F)$ for some loop F, but $\mathrm{MaxSc}_{F'}(\rho_0 \cdots \rho_{n-1}) < \mathrm{Tr}(F')$ for every loop F', i.e., n is the first position where a loop F reaches its threshold score. Due to Lemma 4.10, there is a unique loop F such that $\mathrm{Sc}_F(\rho_0 \cdots \rho_n) = \mathrm{Tr}(F)$. Player i wins the play $\rho_0 \cdots \rho_n$ if $F \in \mathcal{F}_i$. The notions of strategies and winning regions can all be redefined for finite games.

Example 4.12. Consider the Muller game $\mathcal{G} = (\mathcal{A}, \mathcal{F}_0, \mathcal{F}_1)$, where $\mathcal{A}$ is depicted in Figure 4.5, $\mathcal{F}_0 = \{\{0\}, \{2\}, \{0, 1, 2\}\}$ and $\mathcal{F}_1 = \{\{0, 1\}, \{1, 2\}\}$.

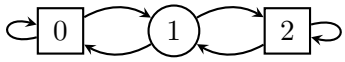

Figure 4.5: The arena $\mathcal{A}$ for Example 4.12

By alternatingly moving from 1 to 0 and to 2, Player 0 wins from every vertex, i.e., we have $W_0(\mathcal{G}) = \{0, 1, 2\}$.

Now, consider the finite-time Muller game $\mathcal{G}' = (\mathcal{A}, \mathcal{F}_0, \mathcal{F}_1, \mathrm{Tr})$ where $\mathrm{Tr}(F) = 3$ for $F \in \{\{0\}, \{2\}\}$ and $\mathrm{Tr}(F) = 2$ for every other loop. Player 1 has a winning strategy from vertex 1: assume Player 0 moves the token from 1 to 0 (the other case is symmetric). Then Player 1 uses the self-loop once before returning the token to 1. The resulting play 1001 is stopped at that point, since the loop $\{0, 1\}$ has reached its threshold score of two. The same approach guarantees him a win from one of the outer vertices: assume that token is at vertex 0. Then, he moves the token to 1. If Player 0 moves the token back to 0, then Player 1 just moves it back to 1 and wins the resulting play 0101. On the other hand, if Player 0 moves the token to 2, then, he wins again by using the self-loop once before moving the token back to 1. The case for a play starting at vertex 2 is again symmetric. Hence, we have $W_1(\mathcal{G}') = \{0, 1, 2\}$. ◇

Since there are only finite plays in a finite-time Muller game $\mathcal{G}$ due to Lemma 4.9, Zermelo's determinacy theorem for finite games [Zer13] yields the following remark.

Remark 4.13. Finite-time Muller games are determined.

In his work on playing Muller games in finite time, McNaughton considered the threshold score $|F|! + 1$ for every loop F and showed that a Muller game and the finite-time Muller game this threshold function have the same winning regions.

Theorem 4.14 ([McN00])**.** *Let $\mathcal{G} = (\mathcal{A}, \mathcal{F}_0, \mathcal{F}_1)$ be a Muller game and let $\mathcal{G}' = (\mathcal{A}, \mathcal{F}_0, \mathcal{F}_1, \mathrm{Tr})$ be the corresponding finite-time Muller game with $\mathrm{Tr}(F) = |F|! + 1$. Then, $W_i(\mathcal{G}) = W_i(\mathcal{G}')$.*

McNaughton's proof is based on properties of the LAR memory structure [GH82], which is sufficient to win Muller games. He showed that such a winning strategy for Player i bounds the losing player's scores by $|F|!$. If the strategy would allow to traverse a loop $F \in \mathcal{F}_{1-i}$ at least $|F|! + 1$ times, then a memory state would be repeated during this infix. Hence, the strategy also allows to traverse F infinitely often which yields a losing play that is consistent with the winning strategy, which is a contradiction. This shows $W_i(\mathcal{G}) \subseteq W_i(\mathcal{G}')$ for $i \in \{0, 1\}$ and thus $W_i(\mathcal{G}) = W_i(\mathcal{G}')$ due to determinacy. This observation can be generalized to arbitrary finite-state strategies: a

winning strategy of size n bounds the losing player's scores by $n \cdot |\mathcal{A}|$. McNaughton obtained better bounds for sets of smaller size by exploiting the special structure of the LAR memory.

The upper and lower bounds on the play length in Lemma 4.9 hold for a uniform threshold score k that is independent of the size of the loop. To obtain tight bounds on the duration of a play with McNaughton's threshold function we adapt the lemma to the threshold score $|F|! + 1$.

Lemma 4.15. *Let V be a finite set of vertices.*

i. For every $w \in V^+$, if $|w| \geq \prod_{j=1}^{|V|}(j!+1)$, then $\mathrm{MaxSc}_{\{F\}}(w) \geq |F|!+1$ for some $F \subseteq V$.

ii. There exists a $w \in V^+$ with $|w| = \frac{1}{2}\prod_{j=1}^{|V|}(j!+1)$ and $\mathrm{MaxSc}_{\{F\}}(w) < |F|!+1$ for every $F \subseteq V$.

Proof. i.) Analogously to the proof of Lemma 4.9(i) we show by induction over the cardinality of V that every word $w \in V^+$ with $|w| \geq \prod_{j=1}^{|V|}(j!+1)$ contains an infix x such that x can be decomposed into parts $x_1, \ldots, x_{|F|!+1}$ for some non-empty set $F \subseteq V$ such that $x = x_1 \cdots x_{|F|!+1}$ with $\mathrm{Occ}(x_j) = F$ for every j. This implies $\mathrm{MaxSc}_F(w) \geq |F|! + 1$.

The claim is true for $|V| = 1$ by choosing x to be the prefix of w of length two and $F = V$. For the induction step, we consider a set V with $n+1$ vertices and a word w with $|w| \geq \prod_{j=1}^{n+1}(j!+1)$. If w contains an infix y of length $\prod_{j=1}^{n}(j!+1)$ that does not contain some vertex $v \in V$, then the induction hypothesis yields the desired decomposition of some infix x of y (which is also an infix of w) for some set $F \subseteq V \setminus \{v\}$. Otherwise, every infix of length $\prod_{j=1}^{n}(j!+1)$ contains every vertex of V. Hence, we can decompose the prefix of length

$$((n+1)!+1) \cdot \prod_{j=1}^{n}(j!+1) = \prod_{j=1}^{n+1}(j!+1) \geq |w|$$

of w into $|V|!+1$ parts x_j of length $\prod_{j=1}^{n}(j!+1)$ all satisfying $\mathrm{Occ}(x_j) = V$.

ii.) Analogously to the proof of Lemma 4.9(ii) we assume without loss of generality $V = [n] = \{0, \ldots, n-1\}$ and inductively define words w_n over the alphabet $[n]$ satisfying $|w_n| = \frac{1}{2}\prod_{j=1}^{n}(j!+1)$ and $\mathrm{MaxSc}_{\{F\}}(w_n) < |F|!+1$ for every $F \subseteq [n]$.

To this end, let $w_1 = 0$ and for $n > 1$ let

$$w_n = (w_{n-1} \cdot (n-1))^{n!} \cdot w_{n-1} \ .$$

Note that $(n-1)$ is a letter. We have $|w_1| = 1 = \frac{1}{2}(1!+1)$ and for $n > 1$

$$\begin{aligned}
&|w_n| \\
=& n! \cdot (|w_{n-1}| + 1) + |w_{n-1}| \\
\geq& (n!+1) \cdot |w_{n-1}| \\
=& (n!+1) \cdot \frac{1}{2} \prod_{j=1}^{n-1} (j!+1) \\
=& \frac{1}{2} \prod_{j=1}^{n} (j!+1) \ .
\end{aligned}$$

Finally, we show $\mathrm{MaxSc}_{\{F\}}(w_n) < |F|!+1$ for every $F \subseteq [n]$ by induction over n. The induction start is immediate since we have $|w| = 1$. So, assume $n > 1$. First, consider a set F that does not contain $(n-1)$. Then, the score for F is reset by each occurrence of $(n-1)$ and can only increase during an infix w_{n-1}. Hence, the induction hypothesis is applicable and bounds the score for F by $|F|!$. Now consider a set F containing the letter $(n-1)$. If $F = [n]$, then the score for F is bounded by $|F|! = n!$, since w_n contains $(n-1) \in F$ only $n!$ times. Finally, consider a set F which contains n, but there is some $n' \in [n-1]$ such that $n' \notin F$. Since n' occurs in w_{n-1}, the score for F is reset between any two occurrences of $(n-1)$. Hence, the score for F is bounded by 1. □

Again, Player 1 can produce the words w_n in the arena $\mathcal{A}_n$ presented in Figure 4.2. Hence, the observations about bounding the play length in arenas with bounded graph complexity measures hold for McNaughton's finite-time Muller games as well. The same holds true for the alternative construction in the arena $\mathcal{A}'_n$ with small degree presented in Figure 4.3.

The threshold function in McNaughton's theorem grows factorially in the size of the loop and the maximal length of a play even grows superfactorially. In the following we investigate finite-time Muller games with a uniform threshold score k for some fixed k, where k may or may not depend on the size of $\mathcal{A}$ or on the complexity of $(\mathcal{F}_0, \mathcal{F}_1)$, but not on the size of a loop F itself. In such a game, the maximal length of a play is bounded by an exponential function. We investigate how small such a uniform threshold score can be while still yielding a finite-time Muller game which has the same winning regions as the original Muller game. The existence of a uniform threshold score that is smaller than $|V|! + 1$ would already be a strengthening of McNaughton's theorem and the strongest conceivable result is a uniform threshold score that is correct for every Muller game, independent of $\mathcal{A}$ and $(\mathcal{F}_0, \mathcal{F}_1)$. Our main theorem about finite-time Muller games is that there exists such a uniform threshold score. As the singleton set $\{v\}$ has a score of one as soon as a play starts in v, the threshold one is obviously too small (and is already ruled out in the definition of a finite-time Muller

game). Before we state our main theorem we give a Muller game that does not have the same winning regions as the corresponding finite-time Muller game with uniform threshold score two.

Theorem 4.16. *There is a Muller game $\mathcal{G} = (\mathcal{A}, \mathcal{F}_0, \mathcal{F}_1)$ and corresponding finite-time Muller game $\mathcal{G}' = (\mathcal{A}, \mathcal{F}_0, \mathcal{F}_1, 2)$ such that $W_i(\mathcal{G}) \neq W_i(\mathcal{G}')$.*

Proof. Consider the arena $\mathcal{A}$ in Figure 4.6 with $\mathcal{F}_1 = \{\{0, 1, 2\}, \{0, 2, 3\}\}$.

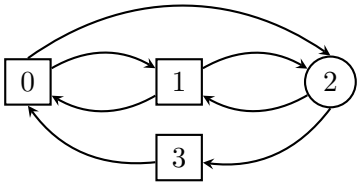

Figure 4.6: The arena $\mathcal{A}$ for Theorem 4.16

The following strategy σ is winning for Player 0 from every vertex in the Muller game $(\mathcal{A}, \mathcal{F}_0, \mathcal{F}_1)$: at vertex 2 alternate between moving to 1 and to 3. Every play ρ consistent with σ either ends up in the loop $\{0, 1\}$ or visits every vertex infinitely often. In both cases, ρ is won by Player 0.

On the other hand, Player 1 has a winning strategy from vertex 3 in the finite-time Muller game $(\mathcal{A}, \mathcal{F}_0, \mathcal{F}_1, 2)$: starting at 3, Player 1 moves to 0 and then to 2. Now, if Player 0 moves to 3, Player 1 answers by moving to 0 and then to 2. The resulting play 302302 is won by Player 1, as the set $\{0, 2, 3\} \in \mathcal{F}_1$ is the first to reach a score of two. On the other hand, if Player 0 moves from 2 to 1, then Player 1 answers by moving to 0, 1, and then to 2, which yields the play 3021012 that is also won by Player 1. □

The remainder of this chapter is devoted to showing our main theorem: the finite-time Muller game with threshold three has always the same winning regions as the original Muller game. In Theorem 4.16 we have shown that three is the smallest uniform threshold score which can possibly yield equivalent games in any case. Hence, our theorem is optimal in this sense.

Theorem 4.17. *Let $\mathcal{G} = (\mathcal{A}, \mathcal{F}_0, \mathcal{F}_1)$ be a Muller game and denote the corresponding finite-time Muller game with uniform threshold score three by $\mathcal{G}' = (\mathcal{A}, \mathcal{F}_0, \mathcal{F}_1, 3)$. Then, $W_i(\mathcal{G}) = W_i(\mathcal{G}')$.*

We proceed as follows: in Subsection 4.2.1, we introduce Zielonka's algorithm for solving Muller games. The internal structure of the winning regions of $\mathcal{G}$ computed by this algorithm is then used in Subsection 4.2.2 to construct a strategy that bounds the losing player's scores by two. By using this strategy, the winning player wins not only the Muller game, but also the finite-time Muller game with threshold three. This suffices to prove Theorem 4.17.

4.2.1 Digression: Zielonka's Algorithm

This subsection presents Zielonka's algorithm for Muller games [Zie98], a reinterpretation of an earlier algorithm due to McNaughton [McN93]. In the next subsection, we use the internal structure of the winning regions as computed by this algorithm to give a winning strategy for a Muller game that bounds the losing player's scores by two. The existence of such strategies is the crucial step in the proof of our main theorem about finite-time Muller games. We begin this subsection by introducing Zielonka trees, a data structure to encode winning conditions of a Muller game. It is the structure of these trees that guides Zielonka's algorithm.

Both the algorithm and the trees were originally defined for Muller games in colored arenas as discussed in Subsection 2.3.1. Recall that the vertices are colored by a function $\Omega\colon V \to [k]$ for some k and the winning condition is a partition $(\mathcal{F}_0, \mathcal{F}_1)$ of $2^{[k]}$. Player i wins a play ρ if and only if $\Omega(\mathrm{Inf}(\rho)) \in \mathcal{F}_i$. We prefer to use uncolored arenas, since dealing with colors would complicate our theorems and proofs unnecessarily: for example, Theorem 4.17 is obviously false for colored arenas: suppose the arena consists of a cycle of n vertices, all but one labeled by 0 and the remaining one by 1. Then, Player i with $\{0,1\} \in \mathcal{F}_i$ wins every play, but the set $\{0\}$, which could be in $\mathcal{F}_{1-i}$, reaches score $n-1$ infinitely often. In Subsection 4.2.3, we briefly discuss how to adapt our results to colored arenas.

According to our definition, a winning condition $(\mathcal{F}_0, \mathcal{F}_1)$ of a Muller game with vertex set V contains exactly the loops of the arena and therefore does not necessarily form a partition of 2^V. When we consider Muller games or finite-time Muller games, the sets of vertices that do not form a loop are irrelevant, since they cannot form an infinity set and do not reach scores greater than one. However, to be able to define the Zielonka tree of the winning condition, we require $\mathcal{F}_0 \cup \mathcal{F}_1$ to contain every subset of vertices of the arena, and not only the loops of the arena. Furthermore, to formulate Zielonka's algorithm as simple as possible we also allow the winning condition to be a partition of the power set of a superset of the arenas vertices. Hence, in the following two subsections, a Muller game $\mathcal{G} = (\mathcal{A}, \mathcal{F}_0, \mathcal{F}_1)$ consists of an arena $\mathcal{A}$ with vertex set V, a family of sets $\mathcal{F}_0 \subseteq 2^{V'}$, and $\mathcal{F}_1 = 2^{V'} \setminus \mathcal{F}_0$, where $V' \supseteq V$ is an arbitrary finite set. In this case we say that $(\mathcal{F}_0, \mathcal{F}_1)$ is an (exhaustive) winning condition over V'. The semantics of $\mathcal{G}$ remains unchanged: Player i wins a play ρ if and only if $\mathrm{Inf}(\rho) \in \mathcal{F}_i$. Since we assume all our winning conditions in this and the next subsection to be exhaustive ones over some superset V' of the arenas vertices, we drop the qualifier "exhaustive" and refer to them as winning conditions as well.

Consider a winning condition $(\mathcal{F}_0, \mathcal{F}_1)$ containing only the loops of an arena $\mathcal{A}$ with vertex set V and an exhaustive winning condition $(\mathcal{F}'_0, \mathcal{F}'_1)$ over $V' \supseteq V$ such that $\mathcal{F}_0 \subseteq \mathcal{F}'_0$ and $\mathcal{F}_1 \subseteq \mathcal{F}'_1$. Since the infinity set of a play in $\mathcal{A}$ is always a loop, a strategy is winning for Player i in $\mathcal{G} = (\mathcal{A}, \mathcal{F}_0, \mathcal{F}_1)$

if and only if it is winning for her in $\mathcal{G}' = (\mathcal{A}, \mathcal{F}_0', \mathcal{F}_1')$. Hence, we also have $W_i(\mathcal{G}) = W_i(\mathcal{G}')$. Thus, we can turn a Muller game with a winning condition that only contains the loops of the arena into an exhaustive winning condition over some superset V' of the arenas vertices by adding each missing set to either one of the families $\mathcal{F}_i$.

We begin by introducing Zielonka trees (originally called split trees by Zielonka [Zie98]). To do this, we need some notation: given a family of sets $\mathcal{F} \subseteq 2^{V'}$ and $X \subseteq V'$, we define

$$\mathcal{F} \restriction X = \{F \in \mathcal{F} \mid F \subseteq X\} \ .$$

We have $\mathcal{F} \restriction X \subseteq \mathcal{F}$ by definition. Given a winning condition $(\mathcal{F}_0, \mathcal{F}_1)$ over V' and $X \subseteq V'$, we define

$$(\mathcal{F}_0, \mathcal{F}_1) \restriction X = (\mathcal{F}_0 \restriction X, \mathcal{F}_1 \restriction X) \ .$$

Note that $(\mathcal{F}_0 \restriction X) \cup (\mathcal{F}_1 \restriction X) = 2^X$.

Definition 4.18 (Zielonka tree)**.** For a winning condition $(\mathcal{F}_0, \mathcal{F}_1)$ defined over a set V', its Zielonka tree $\mathcal{Z}_{\mathcal{F}_0,\mathcal{F}_1}$ is defined as follows: suppose that $V' \in \mathcal{F}_i$ and let $V_0', V_1', \ldots, V_{k-1}'$ be the $\subseteq$-maximal sets in $\mathcal{F}_{1-i}$. The tree $\mathcal{Z}_{\mathcal{F}_0,\mathcal{F}_1}$ consists of a root vertex labeled by (V', i) with k children which are defined by the trees $\mathcal{Z}_{(\mathcal{F}_0,\mathcal{F}_1)\restriction V_0'}, \ldots, \mathcal{Z}_{(\mathcal{F}_0,\mathcal{F}_1)\restriction V_{k-1}'}$.

For every Zielonka tree T with root label (V', i), we define $\mathrm{RtLbl}(T) = V'$ to be the set that labels the root and $\mathrm{RtPlr}(T) = i$ to be the index with $V' \in \mathcal{F}_i$. Furthermore, we define $\mathrm{BrnchFctr}(T)$ to be the number of children of the root and $\mathrm{Chld}(T, j)$ for $0 \leq j < \mathrm{BrnchFctr}(T)$ to be the j-th child of the root. Here, we assume that the children of every vertex are ordered by some fixed linear order.

Let us remark a simple consequence of the inductive definition and the fact that we have $(\mathcal{F}_0 \restriction X) \cup (\mathcal{F}_1 \restriction X) = 2^X$.

Remark 4.19. Let $\mathcal{Z}_{\mathcal{F}_0,\mathcal{F}_1}$ be a Zielonka tree and $T = \mathrm{Chld}(\mathcal{Z}_{\mathcal{F}_0,\mathcal{F}_1}, j)$ for some $j < \mathrm{BrnchFctr}(\mathcal{Z}_{\mathcal{F}_0,\mathcal{F}_1})$. Then, T is the Zielonka tree of the winning condition $(\mathcal{F}_0, \mathcal{F}_1) \restriction \mathrm{RtLbl}(T)$, which is defined over $\mathrm{RtLbl}(T)$.

Remember that the sets V_j' labeling the roots of the children are strict subsets of the root label V'. Hence, the previous remark implies that the height of $\mathcal{Z}_{\mathcal{F}_0,\mathcal{F}_1}$ is at most $|V'|$. Furthermore, the branching factor of a node labeled by some set X is bounded by $2^{|X|}$.

Remark 4.20. Every Zielonka tree is finite.

Let us illustrate these definition by an example.

Example 4.21. Consider the winning condition $(\mathcal{F}_0, \mathcal{F}_1)$ over $\{0,1,2,3\}$ where

$$\mathcal{F}_0 = \{\emptyset, \{0\}, \{1\}, \{2\}, \{0,3\}, \{0,1,2\}, \{1,2,3\}, \{0,1,2,3\}\}$$

and

$$\mathcal{F}_1 = \{\{3\}, \{0,1\}, \{0,2\}, \{1,2\}, \{1,3\}, \{2,3\}, \{0,1,3\}, \{0,2,3\}\} \ .$$

This winning condition is compatible with the Muller game from Example 4.12: all loops of the arena belong to the same $\mathcal{F}_i$ as in the example.

The resulting Zielonka tree $\mathcal{Z}_{\mathcal{F}_0,\mathcal{F}_1}$ is depicted in Figure 4.7. Note that a label may appear more than once in the tree. We have $\mathrm{RtLbl}(\mathcal{Z}_{\mathcal{F}_0,\mathcal{F}_1}) = \{0,1,2,3\}$, $\mathrm{RtPlr}(\mathcal{Z}_{\mathcal{F}_0,\mathcal{F}_1}) = 0$, $\mathrm{BrnchFctr}(\mathcal{Z}_{\mathcal{F}_0,\mathcal{F}_1}) = 3$, and $\mathrm{Chld}(\mathcal{Z}_{\mathcal{F}_0,\mathcal{F}_1}, 2)$ is the Zielonka tree of $(\mathcal{F}_0, \mathcal{F}_1) \restriction \{1,2\}$. Here, we assume the children of each vertex to be ordered from left to right. ◇

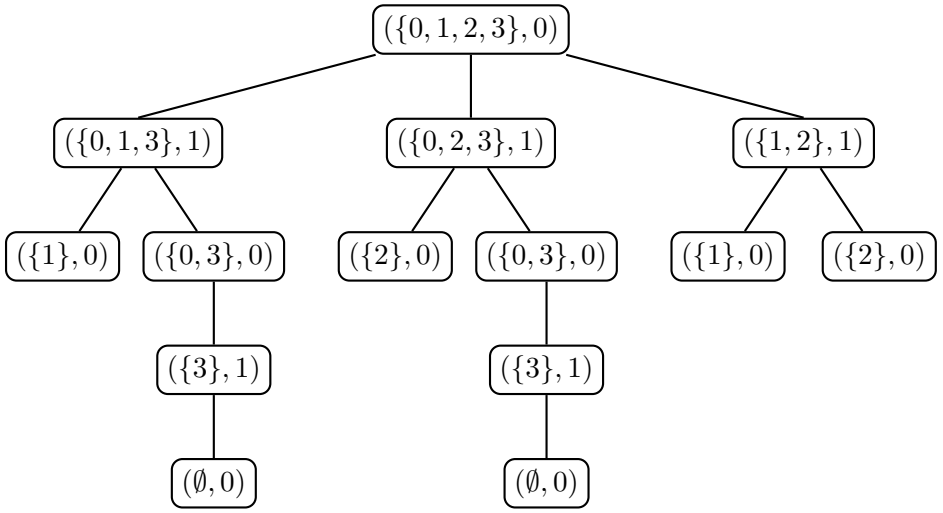

Figure 4.7: A Zielonka tree

By definition, every winning condition induces a Zielonka tree. On the other hand, every Zielonka tree T allows to reconstruct the winning condition $(\mathcal{F}_0, \mathcal{F}_1)$ from T as follows: a set $F \subseteq V'$ is in $\mathcal{F}_i$ if and only if there is a vertex labeled by (X, i) and with children whose roots are labeled by $(X_j, 1-i)$ such that $F \subseteq X$ and $F \not\subseteq X_j$ for every j. In the example above, we have $\{2,3\} \in \mathcal{F}_1$, since it is a subset of $\{0,2,3\}$, but not a subset of its children $\{2\}$ and $\{0,3\}$.

Using this property, one can illustrate the idea underlying Zielonka's algorithm. Consider the root label (V', i) of a Zielonka tree $\mathcal{Z}_{\mathcal{F}_0,\mathcal{F}_1}$ and the labels $(V'_j, 1-i)$ of its children. All sets F whose difference $F \setminus V'_j$ is non-empty for every j are in $\mathcal{F}_i$. Thus, Player i either wins by visiting the

complement of each set V'_j infinitely often or by winning in a game in a subarena with winning condition $(\mathcal{F}_0, \mathcal{F}_1) \restriction V'_j$ over V'_j for some j. Since the latter winning condition has a Zielonka tree of smaller height than $\mathcal{Z}_{\mathcal{F}_0,\mathcal{F}_1}$ (see Remark 4.19) the algorithm can proceed by induction over the height of the tree. Here it becomes apparent why we allow the winning condition to be defined over a superset of the arenas vertices. The set V'_j does not necessarily induce a subarena. Hence, to make a recursive call the algorithm determines an appropriate subset of V'_j which does induce a subarena. If this is a proper subset, then the winning condition is defined over a superset of the subarena's vertices.

The input for Zielonka's algorithm (see Algorithm 1) is an arena $\mathcal{A}$ with vertex set V and the Zielonka tree of a winning condition $(\mathcal{F}_0, \mathcal{F}_1)$ over V' for some finite set $V' \supseteq V$.

Algorithm 1 Zielonka$(\mathcal{A}, \mathcal{Z}_{\mathcal{F}_0,\mathcal{F}_1})$.

$i := \mathrm{RtPlr}(\mathcal{Z}_{\mathcal{F}_0,\mathcal{F}_1})$
$k := \mathrm{BrnchFctr}(\mathcal{Z}_{\mathcal{F}_0,\mathcal{F}_1})$
if The root of $\mathcal{Z}_{\mathcal{F}_0,\mathcal{F}_1}$ has no children **then**
  $W_i := V$; $W_{1-i} := \emptyset$
  return(W_0, W_1)
end if
$U_0 := \emptyset$; $j := 0$
repeat
  $j := j + 1$
  $A_j := \mathrm{Attr}^V_{1-i}(U_{j-1})$
  $X_j := V \setminus A_j$
  $T_j := \mathrm{Chld}(\mathcal{Z}_{\mathcal{F}_0,\mathcal{F}_1}, j \bmod k)$
  $Y_j := X_j \setminus \mathrm{Attr}^{X_j}_i(X_j \setminus \mathrm{RtLbl}(T_j))$
  $(W^j_0, W^j_1) := \mathrm{Zielonka}(\mathcal{A}[Y_j], T_j)$
  $U_j := A_j \cup W^j_{1-i}$
until $U_j = U_{j-1} = \cdots = U_{j-k}$
$W_i := V \setminus U_j$; $W_{1-i} := U_j$
return (W_0, W_1)

Let us explain the algorithm. For the sake of exposition, we assume $i = \mathrm{RtPlr}(\mathcal{Z}_{\mathcal{F}_0,\mathcal{F}_1}) = 1$ in the subsequent paragraphs. If this is not the case then the roles of the two players have to be be swapped. If the root of $\mathcal{Z}_{\mathcal{F}_0,\mathcal{F}_1}$ has no children, then $\mathcal{F}_0 = \emptyset$ and every play is winning for Player 1. Hence, the algorithm returns $W_1(\mathcal{G}) = V$. If the root has children, then Zielonka's algorithm computes the winning regions of the players by successively identifying parts of Player 0's winning region (the sets $U_0 \subseteq U_1 \subseteq U_2 \subseteq \cdots$) until they converge to the winning region. Figure 4.8 depicts the situation in the j-th iteration of the algorithm.

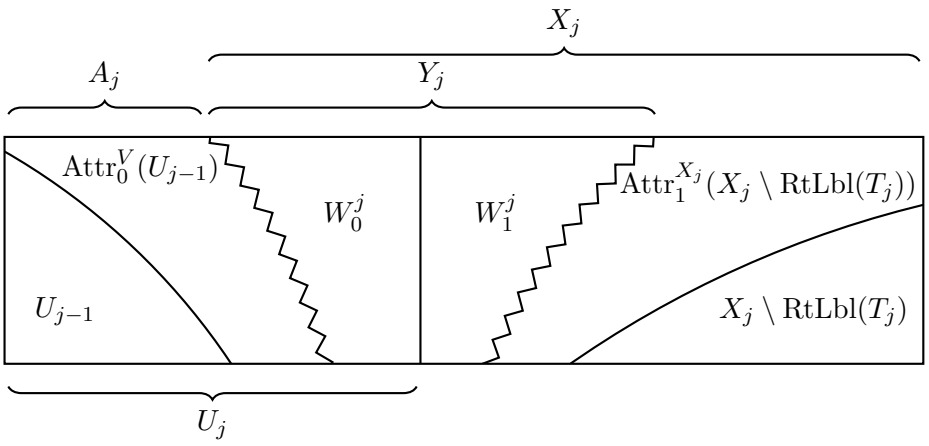

Figure 4.8: The sets computed by Zielonka's algorithm in its j-th iteration

The vertices in U_{j-1} are already determined to belong to $W_0(\mathcal{G})$. Thus, all vertices in the 0-attractor of U_{j-1} also belong to $W_0(\mathcal{G})$. As a complement of an attractor, the set $X_j = V \setminus A_j$ induces a subarena. After removing the vertices in A_j from the arena, the algorithm also removes the vertices in the 1-attractor of $X_j \setminus \mathrm{RtLbl}(T_j)$. Thus, we are in a situation as described above: the remaining vertices induce a subarena whose vertex set is a subset of $\mathrm{RtLbl}(T_j)$. Hence, the algorithm can recursively compute the winning regions W_i^j in this subarena with Zielonka tree T_j. By construction, the winning region W_0^j is also a subset of the winning region $W_0(\mathcal{G})$. Thus, the algorithm can move into the next iteration with $U_j = A_j \cup W_0^j$. It terminates only when the size of the set U_j does not increase for $k = \mathrm{BrnchFctr}(\mathcal{Z}_{\mathcal{F}_0,\mathcal{F}_1})$ consecutive iterations. Since the sets U_j are increasing, the algorithm terminates after at most $k \cdot |\mathcal{A}|$ many iterations. Furthermore, in the last k iterations, the sets U_j do not increase, hence we have $(A_j \setminus U_{j-1}) = \emptyset$ and $W_0^j = \emptyset$ in each of these iterations. The following theorem states that the sets W_0 and W_1 returned by the algorithm are indeed the winning regions.

Theorem 4.22 ([Zie98])**.** *Let $\mathcal{G} = (\mathcal{A}, \mathcal{F}_0, \mathcal{F}_1)$ be a Muller game. On input $(\mathcal{A}, \mathcal{Z}_{\mathcal{F}_0,\mathcal{F}_1})$, Algorithm 1 terminates with output $(W_0(\mathcal{G}), W_1(\mathcal{G}))$.*

The execution of Zielonka's algorithm gives us a structure for $W_0 = W_0(\mathcal{G})$ and $W_1 = W_1(\mathcal{G})$ that we use in Subsection 4.2.2 to construct winning strategies that bound the losing player's scores by two. The set W_0 is partitioned into the attractors given by the sets $A_j \setminus U_{j-1}$, and the recursively computed winning regions given by the sets W_0^j. On the other hand, the structure of W_1 is given by the final k iterations of the algorithm. In each of these iterations, the algorithm computes the attractor $\mathrm{Attr}_1^{X_j}(X_j \setminus \mathrm{RtLbl}(T_j))$, where $X_j = W_1$, and it recursively computes the

winning region $W_1^j \subseteq W_1$. The attractor and the winning region are a partition of the set W_1. The final k iterations of the algorithm give k partitions, one for each child of the root of the Zielonka tree.

This structure of the sets W_0 and W_1 restricts the possible moves available to the players. These restrictions are key to proving the algorithm to be correct. First, let us consider the set W_0 and the moves available to Player 1. All outgoing edges of a vertex $v \in W_0^j \cap V_1$ either lead to W_0^j or to A_j. They cannot lead to W_1^j, since the winning region W_0^j of the Muller game with arena $\mathcal{A}[Y_j]$ and winning condition encoded by T_j is a trap in the arena $\mathcal{A}[Y_j]$. Furthermore, such an edge cannot lead to $\mathrm{Attr}_1^{X_j}(X_j \setminus \mathrm{RtLbl}(T_j))$, since this would imply $v \in \mathrm{Attr}_1^{X_j}(X_j \setminus \mathrm{RtLbl}(T_j))$. Hence, when the play in is W_0^j, Player 1's only choices are to stay in the "smaller" Muller game or to move to A_j, which means Player 0 can force the token into U_{j-1}.

Now, let us consider W_1 and the moves of Player 0. All outgoing edges of a vertex $v \in W_1 \cap V_0$ lead back to W_1. If there is an edge (v, v') with $v' \in W_0$, then v' would be added to some set A_j, since Player 0 can force the token from v into some set U_j. In the last k iterations of the algorithm, the sets X_j are always equal to W_1 and the sets Y_j are the winning region of Player 1 in the Muller game with arena $\mathcal{A}[Y_j]$ and winning condition encoded by T_j. Here, Player 0's only choices are to stay in in the winning region of the "smaller" Muller game or to move to $\mathrm{Attr}_1^{X_j}(X_j \setminus \mathrm{RtLbl}(T_j))$.

Zielonka used these two properties to prove Theorem 4.22 by inductively defining winning strategies for the sets W_0 and W_1 returned by the algorithm, thereby proving that these sets are indeed the winning regions of the Muller game. Player 0 plays the attractor strategy to U_{j-1} on each set $A_j \setminus U_{j-1}$, and a recursively defined winning strategy on each set W_0^j. We have argued above that Player 1 can escape from a set W_0^j, but only to vertices in A_j. Hence, every play consistent with this strategy must eventually be confined to one of the sets W_0^j. Thus, the strategy is winning for Player 0 by induction hypothesis, since the finite play prefix is irrelevant when it comes to determining the winner by the infinity set.

On the other hand, Player 1 uses a cyclic counter c ranging over $0, \ldots, k-1$ to implement his strategy: suppose $c = j$ and let n be the index at which the algorithm terminated. In W_1^{n-j}, the strategy plays according to a recursively defined winning strategy. If Player 0 chooses to leave W_1^{n-j}, then she can only move to a vertex in $\mathrm{Attr}_1^{X_n}(X_n \setminus \mathrm{RtLbl}(T_{n-j}))$ as seen above. Hence, the strategy for Player 1 starts playing an attractor strategy to reach $X_n \setminus \mathrm{RtLbl}(T_{n-j}) \subseteq V \setminus \mathrm{RtLbl}(T_{n-j})$. Once this set has been reached, the counter c is incremented modulo k, and the strategy repeats the behavior described above with respect the new value of c. There are two possibilities for a play consistent with this strategy: if it stays from some point onwards in some W_1^{n-j}, then it is winning by the induction hypothesis. Otherwise, it visits infinitely many vertices in $V \setminus \mathrm{RtLbl}(\mathrm{Chld}(\mathcal{Z}_{\mathcal{F}_0,\mathcal{F}_1}, j))$ for every j in

the range $0 \leq j < \mathrm{BrnchFctr}(\mathcal{Z}_{\mathcal{F}_0,\mathcal{F}_1})$, which implies that the infinity set of the play is not a subset of any $\mathrm{RtLbl}(\mathrm{Chld}(\mathcal{Z}_{\mathcal{F}_0,\mathcal{F}_1}, j))$. Hence, it is in $\mathcal{F}_1$ and the play is indeed winning for Player 1.

We conclude this subsection by showing that these winning strategies do not bound the score of the losing player by a constant. In the next subsection we show how to refine them to achieve our goal.

Theorem 4.23. *There exists a family of Muller games $\mathcal{G}_n = (\mathcal{A}_n, \mathcal{F}_0^n, \mathcal{F}_1^n)$ such that $\mathrm{MaxSc}_{\mathcal{F}_0^n}(\rho) = n$ for some $\rho \in \mathrm{Beh}(W_1(\mathcal{G}_n), \tau_n)$, where τ_n is Zielonka's strategy for Player 1 in $\mathcal{G}_n$.*

Proof. Let $\mathcal{A}_n = (V^n, V_0^n, V_1^n, E^n)$ with $V^n = [n+1]$, $V_0^n = \emptyset$, $V_1^n = V^n$, $E^n = \{(m, m-1) \mid 1 \leq m \leq n\} \cup \{(0,n), (1,n)\}$ (see Figure 4.9(a)), and $\mathcal{F}_1^n = \{V_n\}$. It is easy to verify that we have $W_1(\mathcal{G}_n) = V^n$.

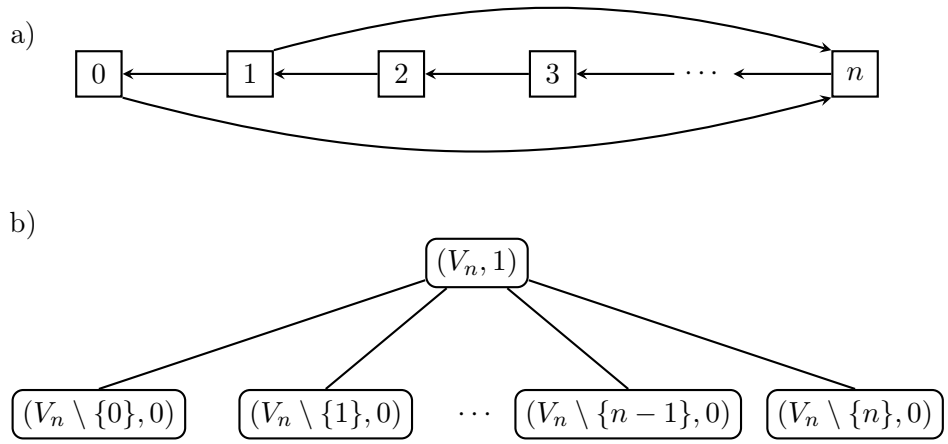

Figure 4.9: a) The arena $\mathcal{A}_n$ and b) the Zielonka tree $\mathcal{Z}_{(\mathcal{F}_0^n, \mathcal{F}_1^n)}$ for Theorem 4.23

The Zielonka tree for the winning condition $(\mathcal{F}_0^n, \mathcal{F}_1^n)$ is depicted in Figure 4.9(b). It has a root labeled by V_n and $n+1$ children which are leaves and are labeled by $V_n \setminus \{i\}$ for every $i \in V_n$. Assume the children are ordered from left to right: $(V_n \setminus \{0\}, 0) < \cdots < (V_n \setminus \{n\}, 0)$.

Zielonka's strategy for Player 1 in $\mathcal{G}_n$, which depends on the ordering of the children, can be described as follows. Initialize a counter $c := 0$ and repeat:

i. Use an attractor strategy to move to vertex c.

ii. Increment c modulo $n+1$.

iii. Go to i.

This strategy is winning from every vertex, since it visits every vertex infinitely often. Now assume a play consistent with this strategy has just

visited 0. Then, it visits all vertices $1, \dots, n$ in this order by cycling through the loop $n, \dots, 1$ exactly n times. Hence, the score for the set $\{1, \dots, n\} \in \mathcal{F}_0^n$ is infinitely often n. □

Note that the arena above is solitary for Player 1, i.e., Zielonka's strategy fails to bound the scores even without interference of the losing player. By contrast, Player 1 has a positional winning strategy for $\mathcal{G}_n$ that bounds the opponents scores by one. The reason the strategy described above fails to do this is that it ignores the fact that all other vertices are visited while moving to vertex 0. To be able to prove Theorem 4.17, we need to refine Zielonka's strategies to recognize such unnecessary iterations, and this turns out to be sufficient to bound the opponent's scores by two.

The unnecessary iterations through the loop $n, \dots, 1$ are caused by the children labeled by sets $V \setminus \{j\}$ with $j > 1$, which do not form a loop of the arena. By distributing the non-loops more cleverly, we obtain a Zielonka tree with one child such that Zielonka's strategy (induced by the new tree) bounds Player 0's scores by one. It is open whether Theorem 4.17 can be proven using Zielonka's strategies, if the Zielonka tree has a certain normal form.

4.2.2 Bounding the Scores of the Opponent

This subsection is dedicated to proving our main theorem about finite-time Muller games: a Muller game $\mathcal{G} = (\mathcal{A}, \mathcal{F}_0, \mathcal{F}_1)$ and the corresponding finite-time Muller game $\mathcal{G}' = (\mathcal{A}, \mathcal{F}_0, \mathcal{F}_1, 3)$ with uniform threshold score three have the same winning regions. Our proof strategy is as follows: using the structure of the winning regions of the Muller game $\mathcal{G}$ as computed by Zielonka's algorithm, we define uniform winning strategies for both players for $\mathcal{G}$ that bound the losing player's scores by two. These strategies are also uniform winning strategies for the finite-time Muller game $\mathcal{G}'$: suppose the strategy σ for Player i bounds Player $(1-i)$'s scores by two. Then, in every play that is consistent with σ some score eventually reaches its threshold score three. However, this cannot be a set of Player $1-i$, since his scores are bounded by two. Hence, these strategies witness $W_i(\mathcal{G}) \subseteq W_i(\mathcal{G}')$ for $i \in \{0, 1\}$. Due to determinacy, the sets $W_i(\mathcal{G})$ (and the sets $W_i(\mathcal{G}')$) partition the arena's set of vertices, hence we conclude $W_i(\mathcal{G}) = W_i(\mathcal{G}')$ for $i \in \{0, 1\}$. Thus, it remains to prove the existence of such strategies.

Lemma 4.24. *Let $\mathcal{G} = (\mathcal{A}, \mathcal{F}_0, \mathcal{F}_1)$ be a Muller game. Player i has a strategy σ such that $\mathrm{MaxSc}_{\mathcal{F}_{1-i}}(\rho) \leq 2$ for every play $\rho \in \mathrm{Beh}(W_i(\mathcal{G}), \sigma)$.*

Three comments are due before we begin the proof. First, in Theorem 4.23 we saw that Zielonka's strategies do not necessarily satisfy the property required by Lemma 4.24. This is not surprising, since they just have to be winning in a Muller game, but were never designed to keep the

scores small. But, by paying close attention to the scores during the inductive composition of strategies, we are able to devise strategies that bound the losing player's scores by two.

Second, the bound two on the losing player's scores is optimal. To see this, consider the Muller game $\mathcal{G}$ in Example 4.12 and assume the token is placed at vertex $1 \in W_0(\mathcal{G})$. No matter to which vertex Player 0 moves the token, she cannot prevent her opponent from reaching a score of two: if she moves it to 0, then Player 1 can use the self-loop once and then return the token to 1. The resulting play 1001 reaches score two for the set $\{0, 1\} \in \mathcal{F}_1$. The case where Player 0 moves the token to 2 is symmetric: Player 1 can enforce a score of two for the set $\{1, 2\} \in \mathcal{F}_1$. However, note that in both plays, Player 0 is the first to reach a score of two. Hence, we cannot use $\mathcal{G}$ to prove Theorem 4.16. In contrast, the theorem can be used to give another proof that the losing player's score cannot be bounded by one. If it could, then the finite-time Muller game with uniform threshold score two has the same winning regions as the original Muller game. However, Theorem 4.16 showed that this is not always the case. These two arguments show that if we are just interested in bounding the scores of the losing player, then Lemma 4.24 is optimal. We discuss possible strengthenings that go beyond bounding the scores at the end of this subsection.

Third, by formulating Lemma 4.24, we have transformed a combination of a reachability condition and a safety condition ("Player i is the first to reach a score of three") into a safety condition ("Player i prevents her opponent from reaching a score of three"). The fact that Player i is the first to reach a score of three follows then from Lemma 4.9(i). While the first condition speaks about the scores of both players, the safety condition only refers to the scores of one player. This simplifies our proof obligation considerably: we do not try to achieve a high score for the winning player but only bound the scores of the losing player.

We continue by discussing why Zielonka's strategies fail to bound the scores. This will be instructive when we define strategies that do satisfy the requirements of Lemma 4.24. As already noted before, Zielonka's strategies have a recursive structure, which means that a strategy σ for an arena with vertex set V often proceeds by playing a recursively defined strategy σ' for a subarena with vertex set $X \subset V$. For example, the two players could construct a play prefix $v_0 \cdots v_n$, where $v_n \in X$, and then σ could start executing σ' starting from vertex v_n. However, the vertex v_n may not be the first position at which the play entered the set X, and there could be a suffix $v_m v_{m+1} \cdots v_n$ of the play such that each vertex in the suffix is contained in X. The strategies produced by Zielonka's algorithm ignore this suffix, because it is not relevant when we only want to construct a winning strategy for a Muller game, since the infinity set – which determines the winner – is not influenced by a finite prefix.

Example 4.25. In the Muller game presented in Theorem 4.23, Zielonka's strategy for Player 0 is composed of the attractor strategies to the singleton sets $\{j\}$ for every j in the range $0 \leq j \leq n$, each of them being defined on the set of all vertices. The overlap of the "domain" of the strategies and the strict cycling through these attractor strategies – which ignores that fact that all vertices are visited when attracting from 0 to 1 – is the reason why these strategies allow high scores for the losing player. ◇

By contrast, when we want to construct a winning strategy that satisfies the properties given by Lemma 4.24, this suffix turns out to be vitally important. We now give some definitions that allow us to work with such suffixes. First, we extend the notion of a play. Previously, we had that a play begins at a starting vertex and its evolution is induced by the strategies for the players. Now we allow a play to begin with an initial play prefix over which the players have no control. Only after this prefix, the players construct a continuation using their strategies. This new definition is useful, because it allows strategies to base their decisions on the properties of the initial play prefix, even if they had no influence on forming it.

Definition 4.26. Let $\mathcal{A} = (V, V_0, V_1, E)$ be an arena. For a non-empty play prefix $w = w_0 \cdots w_m \in V^+$, and strategies $\sigma \in \Pi_i^{\mathcal{A}}$, $\tau \in \Pi_{1-i}^{\mathcal{A}}$, we define the infinite play $\rho(w, \sigma, \tau) = \rho_0\rho_1\rho_2 \cdots$ inductively by $\rho_n = w_n$ for $0 \leq n \leq m$ and for $n > m$ by

$$\rho_n = \begin{cases} \sigma(\rho_0 \cdots \rho_{n-1}) & \text{if } \rho_{n-1} \in V_i, \\ \tau(\rho_0 \cdots \rho_{n-1}) & \text{if } \rho_{n-1} \in V_{1-i}. \end{cases}$$

Note that this definition subsumes our original definition of $\rho(v, \sigma, \tau)$ where v is a vertex.

In fact, the play prefixes that are passed to our strategies are not totally arbitrary. As described previously, these prefixes arise out of decisions made before the strategy is recursively applied. Therefore, we have some control over the form that these prefixes take. We think of such a prefix as a burden that is handed down to a recursively defined strategy and we have to construct our strategies such that they can handle burdens and only pass burdens when calling other strategies. Hence, the following property – which is an equilibrium between these two requirements – plays a crucial role in our construction.

Definition 4.27. Let $\mathcal{F} \subseteq 2^V$. A play prefix w is an $\mathcal{F}$-burden if we have $\mathrm{MaxSc}_{\mathcal{F}}(w) \leq 2$ and if for every $F \in \mathcal{F}$ either

- $\mathrm{Sc}_F(w) = 0$, or
- $\mathrm{Sc}_F(w) = 1$ and $\mathrm{Acc}_F(w) = \emptyset$.

We do not require the burden to contain only vertices from the set V, because it is typically a play prefix in some superset of V only ending in V. A play prefix w satisfies the criteria of a burden if it has the following two properties. First, the requirement that $\mathrm{MaxSc}_{\mathcal{F}}(w) \leq 2$ means that the score for every set $F \in \mathcal{F}$ must be bounded by two at *every* position during the play prefix w. Second, the score for each set $F \in \mathcal{F}$ *at the end* of the prefix must either be zero or one. Additionally, if the score is one, then the accumulator of this set must be empty. In other words, while the scores are allowed to reach two during the play prefix, we insist that they satisfy a more restricted condition at the end.

Example 4.28. Consider $w = 012212210$. It is a $\{\{2\}, \{0,1\}\}$-burden, since the score for both sets is bounded by two throughout w and since we have $\mathrm{Sc}_{\{2\}}(w) = 0$, $\mathrm{Sc}_{\{0,1\}}(w) = 1$ and $\mathrm{Acc}_{\{0,1\}}(w) = \emptyset$. On the other hand, it is not a $\{\{1,2\}\}$-burden, since the score for $\{1,2\}$ reaches value three. ◇

Before we finally begin proving Lemma 4.24, we state two useful properties of burdens that are applied when we pass burdens between strategies.

Lemma 4.29. *Let $\mathcal{F} \subseteq 2^V$.*

i. Every suffix of an $\mathcal{F}$-burden is an $\mathcal{F}'$-burden for every $\mathcal{F}' \subseteq \mathcal{F}$.

ii. Let w be a play prefix and let v be a vertex. If $\mathrm{MaxSc}_{\mathcal{F}}(w) \leq 2$ and if $\mathrm{Lst}(w) \notin F$ for every $F \in \mathcal{F}$, then wv is an $\mathcal{F}$-burden.

Proof. i.) Let $w = w_0 \cdots w_n$ be an $\mathcal{F}$-burden and $w' = w_m \cdots w_n$ for some m in the range $0 \leq m \leq n$. The characterization of the scores by decompositions of suffixes implies

$$\mathrm{Sc}_F(w_m \cdots w_{m+j}) \leq \mathrm{Sc}_F(w_0 \cdots w_m \cdots w_{m+j}) \tag{4.1}$$

for every j in the range $0 \leq j \leq n - m$ and every $F \subseteq V$. Thus,

$$\mathrm{MaxSc}_{\mathcal{F}'}(w') \leq \mathrm{MaxSc}_{\mathcal{F}}(w') \leq \mathrm{MaxSc}_{\mathcal{F}}(w) \leq 2 \ .$$

Furthermore, by plugging $j = n - m$ into (4.1), we obtain $\mathrm{Sc}_F(w') \leq \mathrm{Sc}_F(w) \leq 1$ for every $F \in \mathcal{F}' \subseteq \mathcal{F}$. Now, assume $\mathrm{Sc}_F(w') = 1$. Then, we also have $\mathrm{Sc}_F(w) = 1$ due to (4.1). If the accumulator $\mathrm{Acc}_F(w')$ is non-empty, then the accumulator $\mathrm{Acc}_F(w)$ has to be non-empty as well, which contradicts the fact that w is an $\mathcal{F}$-burden.

ii.) It suffices to consider the scores of wv, since the scores throughout w are bounded by two. Let $F \in \mathcal{F}$. We have $\mathrm{Sc}_F(w) = 0$ and $\mathrm{Acc}_F(w) = \emptyset$. Hence, if $F = \{v\}$, then $\mathrm{Sc}_F(wv) = 1$ and $\mathrm{Acc}_F(wv) = \emptyset$. Otherwise, we have $\mathrm{Sc}_F(wv) = 0$. Thus, wv is indeed an $\mathcal{F}$-burden. □

Now, we have introduced all tools necessary to prove Lemma 4.24 by induction over the height of the Zielonka tree of a Muller game $\mathcal{G}$. The induction hypothesis is the following property:

> Player i has a strategy for $\mathcal{G}$ that bounds the scores for every set in $\mathcal{F}_{1-i} \restriction W_i(\mathcal{G})$ by two and never leaves her winning region, even if the play starts with an $\mathcal{F}_{1-i} \restriction W_i(\mathcal{G})$-burden through her winning region.

Note that we only bound the scores for subsets of the winning region. This suffices when we pass burdens, all other sets – which contain a vertex of the opponent's winning region – are taken care of by the fact that the strategy does not leave the winning region. Hence, these sets do not even reach a score of one. In the following, to be in line with the explanation of Zielonka's algorithm in the previous subsection, we always assume that the root label of a Zielonka tree is in $\mathcal{F}_1$. If this is not the case then the roles of the two players have to be swapped.

We begin with the base case of the induction, which occurs when $\mathcal{G}$ is a Muller game whose Zielonka tree $\mathcal{Z}_{\mathcal{F}_0,\mathcal{F}_1}$ is a leaf. Since we assume $\mathrm{RtLbl}(\mathcal{Z}_{\mathcal{F}_0,\mathcal{F}_1}) \in \mathcal{F}_1$, we have $\mathcal{F}_0 = \emptyset$. Thus, the induction base is given by the following lemma.

Lemma 4.30. *Let $\mathcal{G} = (\mathcal{A}, \mathcal{F}_0, \mathcal{F}_1)$ be a Muller game such that $\mathcal{F}_0 = \emptyset$. Then, every strategy $\tau^* \in \Pi_1^{\mathcal{A}}$ for Player 1 satisfies $\mathrm{MaxSc}_{\mathcal{F}_0}(\rho(w, \sigma, \tau^*)) = 0$ and $\mathrm{Occ}(\rho(w, \sigma, \tau^*)) \subseteq W_1(\mathcal{G})$ for every strategy $\sigma \in \Pi_0^{\mathcal{A}}$ and every $\mathcal{F}_0$-burden $w \in (W_1(\mathcal{G}))^+$.*

Proof. As $\mathcal{F}_0$ is empty, we have $W_1(\mathcal{G}) = V$, and every play ρ satisfies $\mathrm{MaxSc}_{\mathcal{F}_1}(\rho) = 0$ and $\mathrm{Occ}(\rho) \subseteq W_1(\mathcal{G})$. □

For the induction step consider a Muller game $\mathcal{G} = (\mathcal{A}, \mathcal{F}_0, \mathcal{F}_1)$ with arena $\mathcal{A} = (V, V_0, V_1, E)$ and Zielonka tree $\mathcal{Z}_{\mathcal{F}_0,\mathcal{F}_1}$ with $\mathrm{BrnchFctr}(\mathcal{Z}_{\mathcal{F}_0,\mathcal{F}_1}) = k > 0$. We have to give two proofs: one for the set $W_0(\mathcal{G})$, and the other for the set $W_1(\mathcal{G})$. We begin with the former set.

Recall that the algorithm computes $W_0(\mathcal{G})$ as increasing chain of sets

$$\emptyset = U_0 \subseteq U_1 \subseteq \cdots \subseteq U_n = W_0(\mathcal{G})$$

where $U_j = W_0^j \cup A_j$ for each j in the range $1 \leq j \leq n$. Here, W_0^j is the winning region of Player 0 in the Muller game

$$(\mathcal{A}[Y_j], \mathcal{F}_0 \restriction \mathrm{RtLbl}(T_j), \mathcal{F}_1 \restriction \mathrm{RtLbl}(T_j))$$

defined for the recursive call in the j-th iteration of Zielonka's algorithm. The Zielonka tree of this game is $T_j = \mathrm{Chld}(\mathcal{Z}_{\mathcal{F}_0,\mathcal{F}_1}, j \bmod k)$, i.e., a child of $\mathcal{Z}_{\mathcal{F}_0,\mathcal{F}_1}$. Hence, the induction hypothesis is applicable and yields a strategy

$$\sigma_j^{\mathrm{R}} \colon (Y_j)^*(Y_j \cap V_0) \to Y_j$$

for Player 0 with $\mathrm{MaxSc}_{\mathcal{F}_1 \restriction W_0^j}(\rho(w, \sigma_j^{\mathrm{R}}, \tau)) \leq 2$ and $\mathrm{Occ}(\rho(w, \sigma_j^{\mathrm{R}}, \tau)) \subseteq W_0^j$ for every $\tau \in \Pi_1^{\mathcal{A}[Y_j]}$ and every $\mathcal{F}_1 \restriction W_0^j$-burden $w \in (W_0^j)^+$. Here, we

applied the equality $(\mathcal{F}_1 \restriction \mathrm{RtLbl}(T_j)) \restriction W_0^j = \mathcal{F}_1 \restriction W_0^j$, which holds due to $\mathrm{RtLbl}(T_j) \supseteq W_0^j$. Furthermore, A_j is the 0-attractor of U_{j-1} and we denote by σ_j^{A} a positional attractor strategy for Player 0 which is defined on $A_j \setminus U_{j-1}$.

We can now construct our proposed winning strategy. It is similar to Zielonka's strategy, but is careful to pass the appropriate burden to the strategies σ_j^{R}. For every play prefix w and every vertex $v \in V_0 \cap W_0(\mathcal{G})$, we define

$$\sigma^*(wv) = \begin{cases} \sigma_j^{\mathrm{R}}(w'v) & \text{if } v \in W_0^j, \text{ where } w' \text{ is the longest suffix of } w \text{ with} \\ & \mathrm{Occ}(w') \subseteq W_0^j, \\ \sigma_j^{\mathrm{A}}(v) & \text{if } v \in A_j \setminus U_{j-1}. \end{cases}$$

If $v \in V_0 \setminus W_0(\mathcal{G})$, then we define $\sigma^*(wv)$ to be an arbitrary successor of v. This is just for completeness as our strategy never leaves the winning region. The strategy σ^* passes the complete suffix of wv that is contained in W_0^j to σ_j^{R}. This allows us to apply the induction hypothesis for σ_j^{R} in the following proof, which shows that σ^* has the property required by the induction hypothesis. Therefore, the next lemma proves the part of the induction step that deals with $W_0(\mathcal{G})$.

Lemma 4.31. *Let $\tau \in \Pi_1^{\mathcal{A}}$ and let $w \in (W_0(\mathcal{G}))^+$ be an $\mathcal{F}_1 \restriction W_0(\mathcal{G})$-burden. Then, $\mathrm{MaxSc}_{\mathcal{F}_1 \restriction W_0(\mathcal{G})}(\rho(w, \sigma^*, \tau)) \leq 2$ and $\mathrm{Occ}(\rho(w, \sigma^*, \tau)) \subseteq W_0(\mathcal{G})$.*

Proof. Consider such a play $\rho = \rho(w, \sigma^*, \tau)$ and a position $n \geq |w| - 1$. If $\rho_n \in W_0^j \subseteq U_j$ for some j, then ρ is from position n onwards consistent with the strategy σ_j^{R} (which means that Player 0 always moves to a successor in W_0^j) until Player 1 decides to leave W_0^j, if he does at all. We have seen in the previous subsection that he can only leave W_0^j by moving to a vertex in A_j. On the other hand, if $\rho_n \in (A_j \setminus U_{j-1}) \subseteq U_j$ for some j, then ρ is from position n onwards consistent with the attractor strategy σ_j^{A} and remains in $A_j \setminus U_{j-1}$ until a vertex in U_{j-1} is reached. At this position, the play is either in W_0^{j-1} or in A_{j-1}. Hence, we have shown $\mathrm{Occ}(\rho) \subseteq W_0(\mathcal{G})$ and it remains to bound the scores.

We denote the last vertex of w by v and w without its last vertex by w'. By assumption, v is in $W_0(\mathcal{G})$ and thereby in some W_0^j or $A_j \setminus U_{j-1}$. Hence, there is a k in the range $0 < k \leq n$ such that the play can be decomposed as

$$\rho = w' w_n a_n w_{n-1} a_{n-1} \cdots a_{k+1} w_k \ ,$$

where w' is the burden without its last vertex (which is in the first non-empty portion w_j or a_j), w_j is the portion of the play after w' that is contained in W_0^j, and a_j is the portion of the play after w' that is contained in $A_j \setminus U_{j-1}$. One or both of these infixes could be empty, but the portion w_k contains an infinite suffix of the play. We prove our claim by induction over this

decomposition. The base case follows from the fact that w is an $\mathcal{F}_1 \restriction W_0(\mathcal{G})$-burden, which means that we have $\mathrm{MaxSc}_{\mathcal{F}_1 \restriction W_0(\mathcal{G})}(w') \leq 2$.

We show the induction step in two parts. First, we have to prove that if we have

$$\mathrm{MaxSc}_{\mathcal{F}_1 \restriction W_0(\mathcal{G})}(w' w_n a_n \cdots w_{j+1} a_{j+1}) \leq 2 \ ,$$

then also

$$\mathrm{MaxSc}_{\mathcal{F}_1 \restriction W_0(\mathcal{G})}(w' w_n a_n \cdots w_{j+1} a_{j+1} w_j) \leq 2 \ .$$

Here, we assume that w_j is non-empty, as the claim trivially holds otherwise. Hence, let s be the first vertex of w_j and let $F \in \mathcal{F}_1 \restriction W_0(\mathcal{G})$. Due to the induction hypothesis, we only have to show that the score for F is bounded by two during the portion w_j.

If F contains at least one vertex in $W_0(\mathcal{G}) \setminus W_0^j$, then the score for F can increase at most once during the portion w_j, because the vertices in $W_0(\mathcal{G}) \setminus W_0^j$ do not occur in w_j. If it does not increase during w_j, we are done, since the score for F is bounded by two before w_j and does not increase during w_j, hence it is bounded by two during w_j. On the other hand, if the score for F does increase during w_j, then F contains at least one vertex $t \in W_0^j$. If $w_n a_n \cdots w_{j+1} a_{j+1}$ is empty, then we have $s = v$, i.e., the first vertex of w_j is the last vertex of the burden w. Hence, the score for F is at most one at the beginning of w_j and can increase at most once during w_j, hence it is bounded by two during w_j. Now suppose $w_n a_n \cdots w_{j+1} a_{j+1}$ is non-empty. Then, the last vertex of the burden $w = w'v$, is the first vertex of $w_n a_n \cdots w_{j+1} a_{j+1}$ and the vertex $t \in W_0^j$ does not occur in this infix. Therefore, the score for F can increase during $v^{-1} w_n a_n \cdots w_{j+1} a_{j+1}$ (note that we removed the first vertex) only once, and this only if t is already in the accumulator $\mathrm{Acc}_F(w'v)$. Thus, if the score for F increases during $v^{-1} w_n a_n \cdots w_{j+1} a_{j+1}$, then we have $\mathrm{Acc}_F(w'v) \neq \emptyset$, which implies $\mathrm{Sc}_F(w'v) = 0$. On the other hand, if $\mathrm{Sc}_F(w'v) = 1$, then $\mathrm{Acc}_F(w'v) = \emptyset$ and the score for F cannot increase during $v^{-1} w_n a_n \cdots w_{j+1} a_{j+1}$. Thus, we have $\mathrm{Sc}_F(w' w_n a_n \cdots w_{j+1} a_{j+1}) \leq 1$, and even if the score for F increases once during w_j, it cannot increase to more than two throughout w_j.

To conclude the first part of the induction step, we consider the sets $F \in \mathcal{F}_1 \restriction W_0^j$, which are exactly the sets $F \in \mathcal{F}_1 \restriction W_0(\mathcal{G})$ that do not contain a vertex in $W_0(\mathcal{G}) \setminus W_0^j$. In this case the claim follows from the induction hypothesis for the strategy σ_j^{R}: to invoke it, we have to show that $w' w_n a_n \cdots w_{j+1} a_{j+1} s$ is an $\mathcal{F}_1 \restriction W_0^j$-burden. Then, the strategy bounds the scores for the sets $F \in \mathcal{F}_1 \restriction W_0^j$ by assumption on σ_j^{R}, as the portion of the play that is consistent with σ_j^{R} starts with a burden, which is passed to the strategy. If $w_n a_n \cdots w_{j+1} a_{j+1}$ is non-empty, then its last vertex is not in W_0^j. Since the scores during $w' w_n a_n \cdots w_{j+1} a_{j+1}$ are bounded by two,

Lemma 4.29(ii) is applicable and shows that $w'w_na_n \cdots w_{j+1}a_{j+1}s$ is indeed an $\mathcal{F}_1 \restriction W_0^j$-burden. On the other hand, if $w_na_n \cdots w_{j+1}a_{j+1}$ is empty, then $s = v$, i.e., the first vertex of w_j is the last vertex of the burden $w = w'v$. As $w = w'w_na_n \cdots w_{j+1}a_{j+1}s$ is an $\mathcal{F}_1 \restriction W_0(\mathcal{G})$-burden by assumption, it is also an $\mathcal{F}_1 \restriction W_0^j$-burden by Lemma 4.29(i), since we have $\mathcal{F}_1 \restriction W_0^j \subseteq \mathcal{F}_1 \restriction W_0(\mathcal{G})$ due to $W_0^j \subseteq W_0(\mathcal{G})$. This finishes the first part of the induction step.

In the second part of the induction step, we have to prove that if

$$\mathrm{MaxSc}_{\mathcal{F}_1 \restriction W_0(\mathcal{G})}(ww_na_n \cdots a_{j+1}w_j) \leq 2 \ ,$$

then also

$$\mathrm{MaxSc}_{\mathcal{F}_1 \restriction W_0(\mathcal{G})}(w'w_na_n \cdots a_{j+1}w_ja_j) \leq 2 \ .$$

Again, we can assume a_j to be non-empty with first vertex s, since the claim trivially holds if this is not the case. Let $F \in \mathcal{F}_1 \restriction W_0(\mathcal{G})$. Again, due to the induction hypothesis, we only have to show that the score for F is bounded by two during the portion a_j.

If F contains a vertex in $W_0(\mathcal{G}) \setminus (A_j \setminus U_{j-1})$, then the score for F must remain below two for exactly the same reasons as in the first part of the induction step. Otherwise, if $F \subseteq A_j \setminus U_{j-1}$, then we claim that the score for F can rise to at most two during the portion a_j. As above, we have $\mathrm{Sc}_F(w'w_na_n \cdots a_{j+1}w_j) = 0$, if $w_na_n \cdots a_{j+1}w_j$ is non-empty. Since Player 0 plays an attractor strategy during a_j, every vertex in $A_j \setminus U_{j-1}$ occurs at most once during a_j. Thus, as F is a subset of $A_j \setminus U_{j-1}$, its score increases at most once during a_j and is thereby even bounded by one. If $w_na_n \cdots a_{j+1}w_j$ is empty, then we have $s = v$, i.e., the first vertex of a_j is the last vertex of the burden $w = w'v$. Thus, $\mathrm{Sc}_F(w'v) \leq 1$. Again, no vertex occurs twice in $v^{-1}a_j$, hence the score for F is increased at most once during $v^{-1}a_j$. This bounds the score for F by two during a_j, since it is at most one at the beginning of a_j. □

We now turn our attention to the set $W_1(\mathcal{G})$. Remember that $k > 0$ denotes the number of children of $\mathcal{Z}_{\mathcal{F}_0,\mathcal{F}_1}$. The structure of $W_1(\mathcal{G})$ is induced by the last k iterations of Zielonka's algorithm. For the sake of readability, we (mis)use the index j, which ranges over $0, \ldots, k-1$, to refer to the sets U_j, X_j, Y_j, W_i^j, and the tree T_j computed in the last k iterations. In each of these iterations, we have $U_j = U_{j-1}$ and therefore $X_j = W_1(\mathcal{G})$ and $Y_j = W_1^j$. Hence, for each child T_j of $\mathcal{Z}_{\mathcal{F}_0,\mathcal{F}_1}$, the winning region $W_1(\mathcal{G})$ can be decomposed as depicted in Figure 4.10: here, W_1^j is the winning region of Player 1 in the Muller game $(\mathcal{A}[Y_j], \mathcal{F}_0 \restriction \mathrm{RtLbl}(T_j), \mathcal{F}_1 \restriction \mathrm{RtLbl}(T_j))$ with vertex set $Y_j = W_1^j$ defined for the recursive call of Zielonka's algorithm. Since the Zielonka tree T_j of this game is a child of $\mathcal{Z}_{\mathcal{F}_0,\mathcal{F}_1}$, the induction hypothesis is applicable and yields a strategy

$$\tau_j^{\mathrm{R}} \colon (W_1^j)^*(W_1^j \cap V_1) \to W_1^j$$

for Player 1 with $\mathrm{MaxSc}_{\mathcal{F}_0 \restriction W_1^j}(\rho(w, \sigma, \tau_j^{\mathrm{R}})) \leq 2$ and $\mathrm{Occ}(\rho(w, \sigma, \tau_j^{\mathrm{R}})) \subseteq W_1^j$ for every $\sigma \in \Pi_0^{\mathcal{A}[Y_j]}$ and every $\mathcal{F}_0 \restriction W_1^j$-burden $w \in (W_1^j)^+$. Here, we applied the equality $(\mathcal{F}_1 \restriction \mathrm{RtLbl}(T_j)) \restriction W_1^j = \mathcal{F}_1 \restriction W_1^j$, which holds due to $\mathrm{RtLbl}(T_j) \supseteq W_1^j$. All other vertices in $W_1(\mathcal{G}) \setminus W_1^j$ belong to the 1-attractor of $W_1(\mathcal{G}) \setminus \mathrm{RtLbl}(T_j)$, i.e., Player 1 is able to visit a vertex that is not in the root label of T_j. We denote a positional attractor strategy for this set by τ_j^{A}.

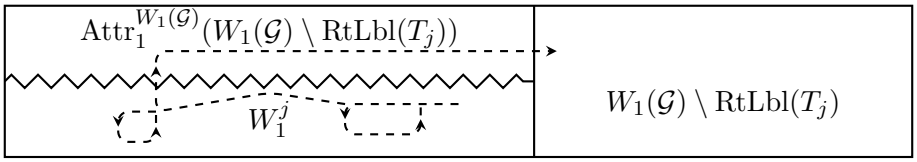

Figure 4.10: The structure of $W_1(\mathcal{G})$ with respect to T_j. The dashed line indicates a part of a play according to τ^* between two change points

Figure 4.10 shows the outcome when Player 1 plays τ_j^{R} and τ_j^{A}. The play remains in the set W_1^j until Player 0 chooses to leave W_1^j, if he does at all. At this position the play is forced to visit some vertex in $W_1(\mathcal{G}) \setminus \mathrm{RtLbl}(T_j)$. Once the play enters $W_1(\mathcal{G}) \setminus \mathrm{RtLbl}(T_j)$, a new index j' has to be selected, and $\tau_{j'}^{\mathrm{R}}$ and $\tau_{j'}^{\mathrm{A}}$ is played. Zielonka's strategy chooses j' to be $j + 1 \bmod k$, and Theorem 4.23 shows that this method does not bound the scores of the losing Player by two. Our goal is to provide a score-based method for choosing a new index that does bound the scores of the opponent by two.

Recall that Lemma 4.8 states that the sets with non-zero score and the accumulators form a chain with respect to the subset relation. Since this property is universal, it still holds if we restrict ourselves to sets in $\mathcal{F}_0$. The indicator function of a play selects the maximal element of this chain, when it is restricted to sets in $\mathcal{F}_0$. For every play prefix w, we define

$$\mathrm{Ind}(w) = \bigcup_{\substack{F \in \mathcal{F}_0 : \\ \mathrm{Sc}_F(w) > 0}} F \cup \bigcup_{F \in \mathcal{F}_0} \mathrm{Acc}_F(w) \ .$$

The following lemma gives an important property that is used in our index selection method: there is always some child whose label contains the indicator.

Lemma 4.32. *For every play prefix w, there is some j in the range $0 \leq j < k$ such that $\mathrm{Ind}(w) \subseteq \mathrm{RtLbl}(T_j)$.*

Proof. Lemma 4.8 implies that there is a $\subseteq$-maximal set I such that $\mathrm{Ind}(w) = I$, with either $\mathrm{Sc}_I(w) > 0$ and $I \in \mathcal{F}_0$ or $\mathrm{Acc}_F(w) = I$ for some $F \in \mathcal{F}_0$, which implies $I \subseteq F$. Hence, $\mathrm{Ind}(w) \subseteq F$ for some $F \in \mathcal{F}_0$. Since we assume $\mathrm{RtLbl}(\mathcal{Z}_{\mathcal{F}_0,\mathcal{F}_1}) \in \mathcal{F}_1$, by definition of $\mathcal{Z}_{\mathcal{F}_0,\mathcal{F}_1}$, there is some child of the root labeled by $\mathrm{RtLbl}(T_j)$ such that $F \subseteq \mathrm{RtLbl}(T_j)$. Hence, we have $\mathrm{Ind}(w) = I \subseteq F \subseteq \mathrm{RtLbl}(T_j)$. □

When a new child must be chosen, our strategy picks one whose label contains the value of the indicator function for the play up to that position. Lemma 4.32 implies that such a child always exists. Also, this condition has to be used when picking the child in the first step after the burden w: in this case the indicator of w is used.

Assume, the j-th child is chosen. We define our strategy so that a play either stays in the smaller winning region W_1^j ad infinitum (which allows us to apply the induction hypothesis) or it eventually leaves $\mathrm{RtLbl}(T_j)$, which resets the scores for all sets which contributed to the accumulator at the position where j is picked. However, by leaving $\mathrm{RtLbl}(T_j)$ other scores may be increased. These scores are taken care of by the fact that an attractor strategy visits every vertex at most once. Hence, these scores increase at most once before a new index j' i selected.

To formalize this, we define an auxiliary function

$$c\colon (W_1(\mathcal{G}))^* \to \{0, 1, \ldots, k-1\} \cup \{\bot\}$$

that uses the indicator of a play prefix to specify which child the strategy is currently considering. For each play w, if $c(w) = j$ then the strategy follows τ_j^{A} and τ_j^{R}. If $c(w) = \bot$ then the strategy moves arbitrarily. This is only the case when the last vertex of the play prefix is not contained in any of the labels of the children. This condition is equivalent to the indicator being empty. We define $c(\varepsilon) = \bot$, and for every play prefix w and every vertex v

$$c(wv) = \begin{cases} c(w) & \text{if } v \in \mathrm{RtLbl}(T_{c(w)}), \\ j & \text{if } v \notin \mathrm{RtLbl}(T_{c(w)}),\ \mathrm{Ind}(wv) \neq \emptyset, \text{ and} \\ & j \text{ is minimal with } \mathrm{Ind}(wv) \subseteq \mathrm{RtLbl}(T_j), \\ \bot & \text{if } v \notin \bigcup_{0 \leq j < k} \mathrm{RtLbl}(T_j). \end{cases}$$

The case distinction in the definition of c is exhaustive, since $\mathrm{Ind}(wv) = \emptyset$ is equivalent to $v \notin \bigcup_{0 \leq j < k} \mathrm{RtLbl}(T_j)$. Furthermore, the minimality in the second case is not essential, but makes the definition unambiguous. The following remark is useful when we prove that our strategy has the desired properties.

Remark 4.33. If $\mathrm{Ind}(wv) = j \neq \bot$, then $v \in \mathrm{RtLbl}(T_j)$.

Finally, we compose the strategies τ_j^{R} and τ_j^{A} to a strategy τ^* for Player 1 for $\mathcal{A}$ as described above. For every play prefix w and every vertex $v \in V_1 \cap W_1(\mathcal{G})$, define

$$\tau^*(wv) = \begin{cases} \tau_j^{\mathrm{R}}(w'v) & \text{if } c(wv) = j, v \in W_1^j, \text{ and } w' \text{ is the longest suffix} \\ & \text{of } w \text{ with } \mathrm{Occ}(w') \subseteq W_1^j, \\ \tau_j^{\mathrm{A}}(v) & \text{if } c(wv) = j, \text{ and } v \in \mathrm{RtLbl}(T_j) \setminus W_1^j, \\ x & \text{if } c(wv) = \bot, \text{ where } x \text{ is some vertex in } W_1(\mathcal{G}) \\ & \text{with } (v, x) \in E. \end{cases}$$

If $v \in V_1 \setminus W_1(\mathcal{G})$, then we define $\tau^*(wv)$ to be an arbitrary successor of v. This is just for completeness as our strategy never leaves the winning region. Furthermore, a vertex x as above always exists, as not every successor of a vertex in $V_1 \cap W_1(\mathcal{G})$ can be in $W_0(\mathcal{G})$. Finally, Remark 4.33 shows that the definition of τ^* is complete. Note that τ^* passes the complete suffix of wv that is contained in W_1^j to τ_j^{R}. This allows us to apply the induction hypothesis for τ_j^{R} in the part of the induction step that deals with the set $W_1(\mathcal{G})$. We now prove that τ^* has the required properties. Our proof uses change points[16], which are positions in a play where the function c changes its value.

Definition 4.34. Let ρ be a play. A position $n > 0$ of ρ is a change point in ρ if $c(\rho_0\rho_1 \cdots \rho_{n-1}) \neq c(\rho_0\rho_1 \cdots \rho_{n-1}\rho_n)$.

In the next Lemma, we prove that if Player 1 plays according to τ^* starting from a burden, then the play up to the next change point n is also a burden. Our intention is to use this as part of an inductive proof that every play bounds the scores for the opponent's sets by two.

Lemma 4.35. *Let ρ be a play such that $\rho_0 \cdots \rho_m$ is an $\mathcal{F}_0 \restriction W_1(\mathcal{G})$-burden and ρ is consistent with τ^* from (at least) m onwards. If n is the smallest change point in ρ satisfying $m < n$, then $\rho_0 \cdots \rho_n$ is an $\mathcal{F}_0 \restriction W_1(\mathcal{G})$-burden.*

Proof. We first provide a proof for the case $c(\rho_0 \cdots \rho_m) = \bot$. By definition this implies $\rho_{n'} \notin \mathrm{RtLbl}(T_j)$ for every n' in the range $m \leq n' < n$ and every j in the range $0 \leq j < k$. Therefore, we have $\mathrm{Sc}_F(\rho_0 \cdots \rho_{n'}) = 0$ and $\mathrm{Acc}_F(\rho_0 \cdots \rho_{n'}) = \emptyset$ for every $F \in \mathcal{F}_0$ and for every n' in the range $m \leq n' < n$, i.e., the scores are zero from position m up to, but not including, position n. Applying Lemma 4.29(ii) shows that $\rho_0 \cdots \rho_n$ is an $\mathcal{F}_0 \restriction W_1(\mathcal{G})$-burden, since we have $\rho_{n-1} \notin F$ for every $F \in \mathcal{F}_0 \restriction W_1(\mathcal{G})$ and since the scores are bounded by two during $\rho_0 \cdots \rho_{n-1}$.

Now, consider the case $c(\rho_0 \cdots \rho_m) = j \neq \bot$. We split the play ρ into four pieces, as depicted in Figure 4.11. The piece p_1 contains the portion of ρ up to and including the position ρ_m and the piece p_4 contains the portion of ρ after and including the change point ρ_n. The piece p_2 contains the portion of ρ between the positions ρ_m and ρ_n that is contained in the set W_1^j, and the piece p_3 contains the portion of ρ between the positions ρ_m and ρ_n that is contained in the set $\mathrm{Attr}_1^{W_1(\mathcal{G})}(W_1(\mathcal{G}) \setminus \mathrm{RtLbl}(T_j))$. Clearly, we have $\rho = p_1p_2p_3p_4$ and the first vertex of p_4 is not in $\mathrm{RtLbl}(T_j)$.

We now prove that $\mathrm{MaxSc}_{\mathcal{F}_0 \restriction W_1(\mathcal{G})}(p_1p_2p_3) \leq 2$. The scores at position ρ_n are considered later. For the portion p_1 the scores are bounded by two by assumption. Now, consider a set $F \in \mathcal{F}_0 \restriction W_1(\mathcal{G})$. During the portion p_2, τ_j^{R} is being played. Since p_1 is an $\mathcal{F}_0 \restriction W_1(\mathcal{G})$-burden, the induction hypothesis

[16] The definition of a change point here is not related to the one in the alternating color technique presented in Subsection 3.2.1.

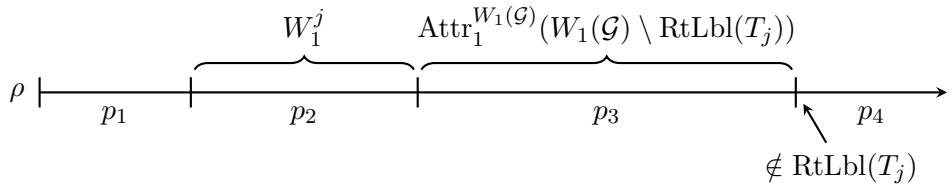

Figure 4.11: The decomposition of a play for Lemma 4.35

for τ_j^{R} is applicable, since τ^* passes the complete suffix of p_1 that is contained in W_1^j to τ_j^{R}. This proves the claim if we have $F \subseteq W_1^j$. On the other hand, if there is a vertex $s \in F \setminus W_1^j$, then s cannot be visited during the portion p_2. This implies that the score for F can increase at most once during p_2. Since p_1 is a burden, we have $\mathrm{Sc}_F(p_1) \leq 1$, which implies $\mathrm{MaxSc}_{\{F\}}(p_1 p_2) \leq 2$.

During the portion p_3 we know that the attractor strategy τ_j^{A} is being played. Since ρ_n is the smallest change point after m, the portion p_3 is in $\mathrm{Attr}_1^{W_1(\mathcal{G})}(W_1(\mathcal{G}) \setminus \mathrm{RtLbl}(T_j))$, but not in $W_1(\mathcal{G}) \setminus \mathrm{RtLbl}(T_j)$. Hence, each vertex can occur at most once during this portion. Consider a set $F \in \mathcal{F}_0 \restriction W_1(\mathcal{G})$. If F contains no vertex from $\mathrm{Attr}_1^{W_1(\mathcal{G})}(W_1(\mathcal{G}) \setminus \mathrm{RtLbl}(T_j))$, then the score for F is 0 during the portion p_3. Therefore, we only need to consider the case where F contains at least one vertex in $\mathrm{Attr}_1^{W_1(\mathcal{G})}(W_1(\mathcal{G}) \setminus \mathrm{RtLbl}(T_j))$, which implies that the score for F can increase at most once during p_3, since this vertex occurs at most once in p_3. So, we only need to show that the score for F is at most one at the end of $p_1 p_2$. The assumption that p_1 is a burden implies that $\mathrm{Sc}_F(p_1) \leq 1$. If p_2 is empty, then we are done, since the score for F is at most one before p_3 and increases at most once during p_3. If p_2 is non-empty, then we distinguish two cases: if the score for F is increased during p_2, then the vertex in $F \cap \mathrm{Attr}_1^{W_1(\mathcal{G})}(W_1(\mathcal{G}) \setminus \mathrm{RtLbl}(T_j))$ is in the accumulator of p_1, which implies $\mathrm{Sc}_F(p_1) = 0$. Furthermore, this is the only increase, since the vertex in the intersection does not occur again. Hence, the score for F can increase at most to two during $p_2 p_3$. On the other hand, if the score for F is not increased during p_2, it can reach at most two during $p_2 p_3$, since it is at most one at the end of p_1. Therefore, we have shown $\mathrm{MaxSc}_{\mathcal{F}_0 \restriction W_1(\mathcal{G})}(p_1 p_2 p_3) \leq 2$.

Thus, we have bounded the scores during $p_1 p_2 p_3$. To complete the proof, we have to show that $p_1 p_2 p_3 \rho_n = \rho_0 \cdots \rho_n$ is an $\mathcal{F}_0 \restriction W_1(\mathcal{G})$-burden, i.e., we have for every $F \in \mathcal{F}_0 \restriction W_1(\mathcal{G})$ either $\mathrm{Sc}_F(p_1 p_2 p_3 \rho_n) = 0$, or $\mathrm{Sc}_F(p_1 p_2 p_3 \rho_n) = 1$ and $\mathrm{Acc}_F(p_1 p_2 p_3 \rho_n) = \emptyset$. We split this proof into two cases. Recall that p_1 is a burden. First, we consider sets $F \in \mathcal{F}_0 \restriction W_1(\mathcal{G})$ such that $\mathrm{Sc}_F(p_1) = 1$ and $\mathrm{Acc}_F(p_1) = \emptyset$. By definition of c we have $F \subseteq \mathrm{Ind}(p_1)$, and therefore by definition of our strategy, we have $F \subseteq \mathrm{RtLbl}(T_j)$. Since $\rho_n \in W_1(\mathcal{G}) \setminus \mathrm{RtLbl}(T_j)$, we have $\rho_n \notin F$. This

implies $\mathrm{Sc}_F(p_1p_2p_3\rho_n) = 0$.

Now, we consider the case $\mathrm{Sc}_F(p_1) = 0$. If $\rho_n \in F$, then $\rho_n \notin \mathrm{Acc}_F(p_1)$, as we have $\mathrm{Acc}_F(p_1) \subseteq \mathrm{RtLbl}(T_j)$ and $\rho_n \notin \mathrm{RtLbl}(T_j)$. The score for F cannot increase during p_2p_3, since the vertex $\rho_n \notin \mathrm{RtLbl}(T_j)$ is not visited during p_2p_3 which is confined to $\mathrm{RtLbl}(T_j)$. Hence, $\mathrm{Sc}_F(p_1p_2p_3\rho_n)$ is at most one. If it is zero, we are done. So, assume the score for F is one. Then, it is increased by visiting ρ_n. Hence, the accumulator $\mathrm{Acc}_F(p_1p_2p_3\rho_n)$ is empty as required. On the other hand, if $\rho_n \notin F$ then $\mathrm{Sc}_F(p_1p_2p_3\rho_n) = 0$. □

We use Lemma 4.35 inductively to show that the strategy τ^* bounds the scores of Player 0 by two. For the base case of this inductive proof, we have to require the play prefix that is passed to the strategy to satisfy the burden property. The next lemma proves the part of the induction step that deals with $W_1(\mathcal{G})$.

Lemma 4.36. *Let $\sigma \in \Pi_0^{\mathcal{A}}$ and let $w \in (W_1(\mathcal{G}))^+$ be an $\mathcal{F}_0 \restriction W_1(\mathcal{G})$-burden. Then, $\mathrm{MaxSc}_{\mathcal{F}_0 \restriction W_1(\mathcal{G})}(\rho(w, \sigma, \tau^*)) \leq 2$ and $\mathrm{Occ}(\rho(w, \sigma, \tau^*)) \subseteq W_1(\mathcal{G})$.*

Proof. Consider such a play $\rho = \rho(w, \sigma, \tau^*)$. We begin by showing that ρ does not leave $W_1(\mathcal{G})$. We have seen in the previous subsection that Player 0 cannot leave $W_1(\mathcal{G})$ from one of his vertices. Furthermore, all moves of Player 1 which are consistent with τ^* lead back to $W_1(\mathcal{G})$: if τ^* uses a strategy τ_j^{R}, then it keeps the token in $W_1^j \subseteq W_1(\mathcal{G})$ until Player 0 moves it to the attractor $\mathrm{Attr}_1^{W_1(\mathcal{G})}(W_1(\mathcal{G}) \setminus \mathrm{RtLbl}(T_j))$, which is also a subset of $W_1(\mathcal{G})$. From there, Player 1 forces the token to $W_1(\mathcal{G}) \setminus \mathrm{RtLbl}(T_j)$. At this point, a new strategy $\tau_{j'}^{\mathrm{R}}$ is picked and the same argument applies, unless the function c returns $\bot$. But also in this situation, the token is moved to a successor in $W_1(\mathcal{G})$. Hence, we have shown $\mathrm{Occ}(\rho) \subseteq W_1(\mathcal{G})$ and it remains to bound the scores.

Since w is an $\mathcal{F}_0 \restriction W_1(\mathcal{G})$-burden, we can use Lemma 4.35 inductively to show that, if $n \geq |w|$ is a change point in ρ, then $\rho_0\rho_1 \cdots \rho_n$ is a burden. If ρ contains infinitely many change points, then the proof is complete. This is because if the play up to every change point is a burden and there is an infinite number of change points, then $\mathrm{MaxSc}_{\mathcal{F}_0 \restriction W_1(\mathcal{G})}(\rho) \leq 2$.

On the other hand, if there is only a finite number of change points after position $|w|-1$, then let n be the final change point after $|w|-1$ in ρ (if there is no such change point at all, then let $n = |w| - 1$). Since $\rho_0 \cdots \rho_n$ is an $\mathcal{F}_0 \restriction W_1(\mathcal{G})$-burden, we have $\mathrm{MaxSc}_{\mathcal{F}_0 \restriction W_1(\mathcal{G})}(\rho_0 \cdots \rho_n) \leq 2$ in both cases. If $c(\rho_0 \cdots \rho_n) = j$ for some j in the range $0 \leq j < k$, then we have $\rho_m \in W_1^j$ for every $m \geq n$. Assume this is not the case: we have $\rho_m \in \mathrm{RtLbl}(T_j)$ due to Remark 4.33 and if there is some $m \geq n$ such that $\rho_m \in \mathrm{RtLbl}(T_j) \setminus W_1^j$, then it is in the attractor of $W_1(\mathcal{G}) \setminus \mathrm{RtLbl}(T_j)$. Hence, there would be another change point after position n.

Thus, τ^* follows σ_j^{R} and is confined to W_1^j from the position n onwards. Due to Lemma 4.29(i), $\rho_0 \cdots \rho_n$ is also an $\mathcal{F}_0 \restriction W_1^j$-burden and we can apply

the induction hypothesis to bound the scores for the sets $\mathcal{F}_0 \restriction W_1^j$ after position n. Furthermore, the scores for the sets $F \in (\mathcal{F}_0 \restriction W_1(\mathcal{G}))\backslash(\mathcal{F}_0 \restriction W_1^j)$ are bounded by one at position n. Since each such set F contains a vertex in $W_1(\mathcal{G})\backslash W_1^j$ (which is not visited after position n), the score can only increase once after n, and this only if this vertex is in the accumulator, which implies that the score for the set is zero at position n. Hence, the scores for these sets are bounded by one after position n. Finally, the scores up to position n are bounded by the burden property. Therefore, we have $\mathrm{MaxSc}_{\mathcal{F}_0 \restriction W_1(\mathcal{G})}(\rho) \leq 2$.

On the other hand, if $c(\rho_0 \cdots \rho_n) = \bot$, then also $c(\rho_0 \cdots \rho_m) = \bot$ for every $m > n$. This implies $\rho_m \notin \mathrm{RtLbl}(T_j)$ for every j in the range $0 \leq j < k$. Since every $F \in \mathcal{F}_0$ is a subset of some $\mathrm{RtLbl}(T_j)$, we have $\mathrm{Sc}_F(\rho_0 \cdots \rho_m) = 0$ for every $m > n$ and every $F \in \mathcal{F}_0$. Therefore, $\mathrm{MaxSc}_{\mathcal{F}_0 \restriction W_1(\mathcal{G})}(\rho) \leq 2$. □

Finally, we can prove Lemma 4.24, which also completes the proof of Theorem 4.17.

Proof. We prove the following by induction over the height of $\mathcal{Z}_{\mathcal{F}_0,\mathcal{F}_1}$:

> Player i has a strategy σ with $\mathrm{MaxSc}_{\mathcal{F}_{1-i} \restriction W_i(\mathcal{G})}(\rho(w,\sigma,\tau)) \leq 2$ and $\mathrm{Occ}(\rho(w,\sigma,\tau)) \subseteq W_i(\mathcal{G})$ for every strategy $\tau \in \Pi^{\mathcal{A}}_{1-i}$ and every $\mathcal{F}_{1-i} \restriction W_i(\mathcal{G})$-burden $w \in (W_i(\mathcal{G}))^+$.

This suffices to prove our claim: let $\rho \in \mathrm{Beh}(W_i(\mathcal{G}), \sigma)$. Then, there is a strategy τ for Player $1-i$ such that $\rho = \rho(\rho_0, \sigma, \tau)$. Furthermore, the play prefix ρ_0 is an $\mathcal{F}_{1-i} \restriction W_i(\mathcal{G})$-burden. Thus, we have $\mathrm{MaxSc}_{\mathcal{F}_{1-i} \restriction W_i(\mathcal{G})}(\rho) \leq 2$. It remains to consider the scores for the sets in $\mathcal{F}_{1-i} \setminus (\mathcal{F}_{1-i} \restriction W_i(\mathcal{G}))$: each such set contains a vertex $v \in W_{1-i}(\mathcal{G})$ which cannot occur in a play that is consistent with σ since it does not visit v. Therefore the scores for the sets in $\mathcal{F}_{1-i} \setminus (\mathcal{F}_{1-i} \restriction W_i(\mathcal{G}))$ never reach a score of one. Altogether, we have $\mathrm{MaxSc}_{\mathcal{F}_{1-i}}(\rho) \leq 2$.

The actual inductive proof is now very simple: for the induction start, apply Lemma 4.30 and in the induction step, use the strategies obtained from the induction hypothesis to define σ^* and τ^* as above. For σ^*, Lemma 4.31 guarantees $\mathrm{MaxSc}_{\mathcal{F}_1 \restriction W_0(\mathcal{G})}(\rho(w,\sigma^*,\tau)) \leq 2$ for every strategy $\tau \in \Pi^{\mathcal{A}}_1$ and every $\mathcal{F}_1 \restriction W_0(\mathcal{G})$-burden w with $v \in W_0(\mathcal{G})$. The reasoning for $W_1(\mathcal{G})$ is analogous and applies Lemma 4.36. □

We have just shown that the winning player (from a given vertex) in a Muller game can always bound the losing player's scores by two. Example 4.12 shows that this bound is optimal in the sense that there are Muller games in which the losing player can enforce a score of two. let us conclude by mentioning two possibilities to strengthen Lemma 4.24: in Example 4.12, just after increasing his score for the set $\{0,1\}$ to two by moving the token from vertex 0 to vertex 1, Player 0 can move the token to 2 and thereby reset the score for $\{0,1\}$. It is open, whether this is always possible:

> does the winning player have a winning strategy that bounds the losing player's scores by two and additionally resets every score after it has reached two?

The second strengthening is related to the first one: in the example, Player 1 is able to reach a score of two, but only with empty accumulator. It is open, whether this is the best he can achieve:

> does the winning player have a winning strategy that bounds the losing player's scores by two and keeps the accumulator empty while the score is two?

Note that the second strengthening implies the first one: if the losing player reaches a score of two for some set F, then the next vertex of the play has to be in $V \setminus F$ and thereby a reset. If it were in F, then it would yield a non-empty accumulator while the score is two. However, the other implication does not hold: it might take the winning player some moves through the set F that has reached score two before she is able to reset the score. However, this means she has visited some vertex of F two times after the score for F is increased to one and could (and should) have reset the score after the first visit.

Let us explain why the proof presented above does not yield a strategy that has one or both of the properties described above. Consider the construction of σ^*: starting after the burden w, the play proceeds through the hierarchical traps $U_n \supseteq U_{n-1} \supseteq \cdots \supseteq U_1$ in this order and eventually stays in some U_k ad infinitum. Say the burden has the form u_1u_2 where u_1 is a finite path through U_1 and u_2 is a finite path through U_2 (note the reversal of the order). A play starting with burden u_1u_2 that is consistent with σ^* might proceed from U_2 to U_1 which is never left, i.e., the complete play has the form $u_1u_2u_2'u_1'$ where u_2' is a finite path through U_2 and u_1' is an infinite path through U_1. Let $F \subseteq U_1 \cup U_2$ be a set such that $F \cap U_1 \neq \emptyset$ and $F \cap U_2 \neq \emptyset$. Then, the score for F might be one after the burden u_1u_2 and might be increased to two during u_2'. Since U_2 is never left, the score for F is not necessarily reset and the accumulator could be non-empty as well.

4.2.3 Playing Muller Games in Colored or Infinite Arenas in Finite Time

In this section we discuss two possible extensions of Theorem 4.17, which is formulated for Muller games in uncolored, finite arenas. We begin by dropping the first restriction and consider an arena whose vertices are labeled by $\Omega\colon V \to [k]$ for some k. In this case, the winning condition $(\mathcal{F}_0, \mathcal{F}_1)$ is a partition of $2^{[k]}$. Then, we can define scoring functions Sc_F for every $F \subseteq [k]$, which are defined on the sequence of colors visited by a play. However, for trivial reasons, the losing player's scores can no longer be bounded by any constant.

Example 4.37. Consider the Muller game $(\mathcal{A}_n, \mathcal{F}_0, \mathcal{F}_1)$ in the colored arena $\mathcal{A}_n$ depicted in Figure 4.12, where the number below a vertex denotes its color, and where $\mathcal{F}_0 = \{\{1\}\}$ and $\mathcal{F}_1 = 2^{\{0,1\}} \setminus \mathcal{F}_0$. In this Muller game with n vertices, the losing player reaches a score of $n-1$ for his set $\{0\}$ when starting at vertex v_0. ◇

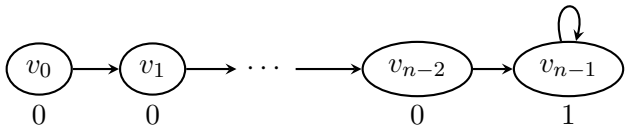

Figure 4.12: The arena $\mathcal{A}_n$ for Example 4.37

The existence of strategies that bound the losing player's score in an uncolored arena very much relied on the fact that an attractor strategy visits a vertex v at most once. Hence, the score for any set F containing v can increase at most once while an attractor strategy is played. This fails for colored arenas: here, on an attractor of size n, the score for a set F of colors can increase up to $\lfloor n/|F| \rfloor$ times. The proof of Lemma 4.24 could be adapted, but this requires to keep track of the size of the arenas along the induction. For this reason, we prefer uncolored arenas, which simplifies the proof and also yields nicer results in the form of the constant bound two instead of a bound that depends on the size of the arena and the size of the set F. Since visiting a set $C \subseteq [k]$ repeatedly requires visits to sets $F_j \subseteq V$ such that $\Omega(F_j) = C$, our results on uncolored arenas imply a bound on the scores the losing player can achieve. However, these bounds are nowhere near the linear lower bounds of Example 4.37, which we conjecture to be tight.

Now we drop the restriction to finite arenas and consider infinite ones. There are several ways to define Muller games in such arenas. Note that there could be plays whose infinity set is empty. Furthermore, if we define $\mathcal{F}_0$ and $\mathcal{F}_1$ to be a partition of the set of vertices, then the scores of every infinite simple path[17] are bounded by one and reset immediately after they are increased to one. However, as there also could be arbitrarily long paths which end in a self-loop (which means that such a play could be winning for either player), there is no way to stop a play after a finite number of steps that is only based on the scores of the prefix produced so far.

To guarantee the existence of a non-empty infinity set for each play, it is standard to consider (finitely) colored infinite arenas, which also guarantees high scores after an exponential number of steps. In the following, we give three examples of Muller games in infinite arenas and highlight some properties of scores in such arenas. All arenas are simple in the sense that they are isomorphic to configuration graphs of one-counter automata, i.e.,

[17] A path is simple, if it visits each vertex at most once.

pushdown automata with a single stack symbol (besides the stack bottom symbol, which may neither be deleted nor pushed onto the stack). Hence, every vertex represents a configuration (q, γ) consisting of a state q and a stack content γ. Since the automaton has only a single stack symbol, we identify stack contents by natural numbers in our examples. In this setting, the color of a vertex is usually its state q. Thus, the winning condition $(\mathcal{F}_0, \mathcal{F}_1)$ is a partition of 2^Q. We refrain from giving formal definitions (see, e.g., [Wal01] for the definition of a pushdown arena) and thus also from proving these arenas to be indeed configuration graphs of one-counter machines.

Firstly, we show that there are Muller games in which the winning player has to allow arbitrary high scores for the losing player. Note that we cannot use the game of Example 4.37 to show this, if we require the color of a configuration to be equal to its state.

Example 4.38. Consider the arena $\mathcal{A}_4$ depicted in Figure 4.13 and assume Player 0 wins a play if and only if the state q_4 is visited infinitely often. Thus, she has to reach the vertex $(q_4, 0)$ to win. But by doing this when starting at vertex $(q_0, 0)$, she has to visit five vertices in the first row, thereby allowing Player 1 a score of five. This example can obviously be generalized to show that for every n, there is a Muller game, whose arena $\mathcal{A}_n$ is isomorphic to the configuration graph of a one-counter machine, in which the winning player has to allow the losing player a score of $n + 1$. ◇

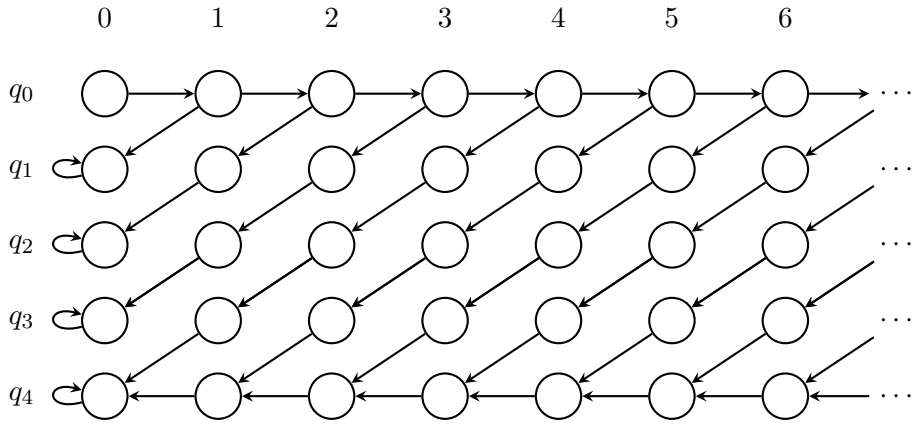

Figure 4.13: The arena $\mathcal{A}_4$ for Example 4.38 (the labels on the left denote the state of a configuration, the labels on top its stack height)

Our second example shows that there is an arena in which a player can make sure to be the first player to reach a certain threshold score, but loses eventually. Hence, the winner of the infinite Muller game cannot be determined by any fixed threshold score.

Example 4.39. Consider the arena $\mathcal{A}$ depicted in Figure 4.14 and assume Player 0 wins a play if and only if either the state q_0 or the state q_2 is visited infinitely often. Thus, every play is winning for her. Now, consider a threshold score $k \geq 3$. By moving from the starting vertex $(q_0, 0)$ to the right $k-2$ times and then down, Player 1 enforces the state sequence $(q_0)^{k-1}(q_1)^k(q_2)^\omega$, since Player 0 only has non-trivial choices at her vertices. Hence, Player 1 can ensure to be the first player to reach a score of k, although he loses every play. ◇

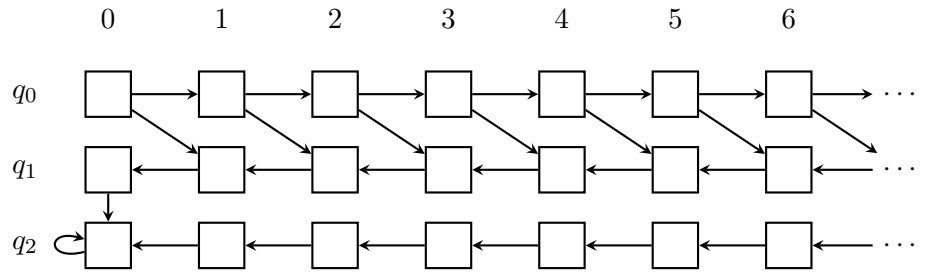

Figure 4.14: The arena $\mathcal{A}$ for Example 4.39 (the labels on the left denote the state of a configuration, the labels on top its stack height)

In the previous example, Player 1 is the first to achieve a score of k, but only after Player 0 has reached a score of $k-1$. The lower bounds on play prefixes that avoid a score of k presented in Lemma 4.9 are used in our last example to widen this gap: in this game, the gap between the losing player's scores reached so far and the winning player's scores reached so far is exponential (in the number of states of the underlying one-counter machine) for some prefix. Hence, the winner of the infinite Muller game cannot be determined by any fixed gap between the scores reached so far.

Example 4.40. Consider the arena $\mathcal{A}_3$ depicted in Figure 4.15 and assume we have $\mathcal{F}_1 = \{\{q_4\}\}$. Thus, every play is winning for Player 0, since it either ends up in the self-loop at vertex $(q_5, 0)$ or has an infinity set which is a subset of $\{q_0, q_1, q_2, q_3\}$. Now, consider a threshold score $k \geq 1$. By using the lower bound w_4 (for this k) from Lemma 4.9 as guideline, Player 1 is able to move the token from the starting vertex $(q_0, 0)$ to the vertex $(q_0, k^4 - 1)$ without allowing Player 0 to reach a score of k. Then, he reaches a score of k^4 by moving the token to (q_4, k^4) and then all the way to $(q_4, 0)$. Hence, Player 1 can ensure to reach a score of k^4 while Player 0 has not yet reached a score of k, although he loses every play. This construction can be easily generalized to an arena $\mathcal{A}_n$ in which Player 1 can enforce a score of k^n while Player 0 has not yet reached a score of k. ◇

In the latter two examples, Player 1 reaches a high score for a set which cannot be the infinity set of a play. This defect can be rectified by adding

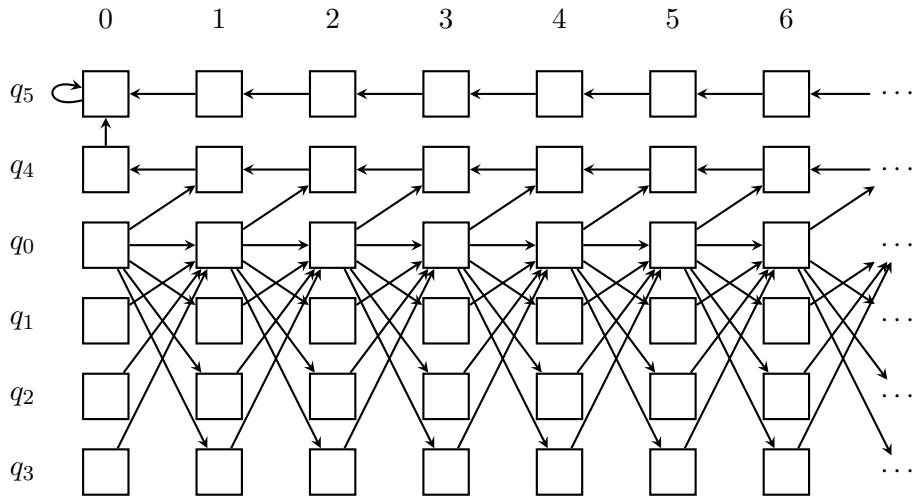

Figure 4.15: The arena $\mathcal{A}_3$ for Example 4.40 (the labels on the left denote the state of a configuration, the labels on top its stack height)

inverse edges in this row (and letting Player 0 move at these vertices). However, Player 0 has no reason to use these additional edges, which makes them superfluous. One way to find an analogue of Theorem 4.17 might be to find a formal definition of "superfluous edges", remove them, and then consider only the scores for sets $F \in 2^Q$ that can be the infinity set of a play. Note that this policy disregards the set $\{q_1\}$ in Example 4.39 and the set $\{q_4\}$ in Example 4.40.

Finally, let us mention that another possibility to devise a score-based finite-duration variant of an infinite game played in an infinite arena whose investigation seems promising: Walukiewicz [Wal01] shows that the winner of a parity game $\mathcal{G}$ in a pushdown arena can be determined by solving a parity game $\mathcal{G}'$ in a finite arena. In Section 5.2, we present a score-based finite-time duration variant of a parity game. In ongoing research we investigate whether this can be transfered to a finite-duration variant of a parity game in a pushdown arena by reverse-engineering Walukiewicz's transformation.

4.3 Summary of Results

We improved McNaughton's results on finite-time Muller games by showing that his threshold score $|F|! + 1$ can be reduced to three. We have complemented this with tight upper and lower bounds on the maximal play length in such a finite-duration game. Table 4.1 compares the threshold score (if applicable) and the maximal play length of three finite-duration variants of Muller games: the second row represents the reduction to a parity game via latest appearance records. A play in the Muller game can be stopped

if the induced play in the parity game visits a memory state for the second time. If the maximal priority occurring in the cycle of this play prefix is even, then Player 0 is declared to win the finite play of the Muller game, otherwise Player 1 wins. Due to positional determinacy of parity games, this finite-duration version also has the same winning regions as the original Muller game. The construction and a proof of the lower bound due to Chaturvedi [Cha11] can be found in the appendix.

Variant	Threshold score	Maximal play length	
		upper bound	lower bound
McNaughton	|F|!+1	$\prod_{j=1}^{n}(j!+1)$	$\frac{1}{2}\prod_{j=1}^{n}(j!+1)$
LAR-Reduction	n/a	$n \cdot n! + 1$	$n \cdot n!$
here	3	3^n	$3^n - 1$

Table 4.1: Comparison of finite-duration variants of a Muller game with n vertices

Our results imply that the winning regions of a Muller game can be determined by solving a finite game in a tree of height 3^n. The main technical contribution of this section consists of uniform winning strategies for Muller games that bound the opponent's scores by two. In the next chapter, we investigate further applications of the existence of such "strong" winning strategies for Muller games.

Finally, we have illustrated that a score-based finite-duration variant of a Muller game in an infinite arena (even in very simple ones induced by pushdown automata) has to be more complex than just relying on a certain threshold score that has to be reached.

Chapter 5

Reductions Down the Borel Hierarchy

Game reductions as introduced in Subsection 2.3.3 are subject to limitations arising from the topological complexity of the sets of winning plays as evidenced by their classification in the Borel hierarchy. For example, reachability and safety conditions are in the first level of the hierarchy, Büchi and co-Büchi conditions in the second one, and parity and Muller conditions in the third level. A game reduction $\mathcal{G} \leq_{\mathcal{M}} \mathcal{G}'$ between two games $\mathcal{G}$ and $\mathcal{G}'$ can only exist if the set of winning plays in $\mathcal{G}'$ is on the same or a higher level of the hierarchy than the set of winning plays in $\mathcal{G}$. Therefore, parity and Muller games cannot be reduced to safety games.

Nevertheless, there are several reductions down the Borel hierarchy in the literature that allow to determine the winning regions (and in some cases also a winning strategy for one player). For example, Bernet et al. show how to determine a permissive strategy, a non-deterministic winning strategy that subsumes the behavior of every positional winning strategy, for a parity game by solving a safety game [BJW02]. Similarly, reductions from co-Büchi games to safety games are used in so-called Safraless algorithms for LTL games [KV05, KPV06, FJR11] and in bounded synthesis [SF07].

In this chapter, we show that similar results also hold for Muller games, i.e., both player's winning regions in a Muller game and a winning strategy for one player can be computed by solving a safety game. The correctness of this construction follows from the existence of winning strategies that bound the losing player's scores. This also yields a new memory structure as well as a notion of permissive strategies for Muller games. As a brief introduction, we rephrase the construction of Bernet et al. for parity games in terms of scoring functions for parity games, to which our construction for Muller games is compared to. After the main contribution of this chapter, the reduction from Muller games to safety games, we present a general framework that allows to determine winning regions and one winning strategy for a game by

solving a safety game. This framework encompasses all results mentioned above.

5.1 Digression: The Borel Hierarchy

We begin this chapter by giving a brief introduction to the Borel hierarchy of ω-languages and classify the games we consider in this work in the hierarchy. For a detailed exposition, we refer to [Kec95]. As a consequence of the results presented here, we obtain limitations on reductions between different types of games. These serve as background to the constructions presented in the remainder of this chapter.

Let V be a finite set. A set $L \subseteq V^\omega$ is open, if it is of the form $K \cdot V^\omega$ for some $K \subseteq V^*$. The Borel hierarchy on V^ω consists of levels $\mathbf{\Sigma}_n$ and $\mathbf{\Pi}_n$ for every positive integer[18] n, defined inductively as follows:

- $\mathbf{\Sigma}_1 = \{L \subseteq V^\omega \mid L \text{ open } \}$,
- $\mathbf{\Pi}_n = \{L \subseteq V^\omega \mid V^\omega \setminus L \in \mathbf{\Sigma}_n\}$, and
- $\mathbf{\Sigma}_{n+1} = \{L \subseteq V^\omega \mid L = \bigcup_{i \in \mathbb{N}} L_i \text{ with } L_i \in \mathbf{\Pi}_n\}$.

For every $n \geq 1$, we have $\mathbf{\Sigma}_n \cup \mathbf{\Pi}_n \subseteq \mathbf{\Sigma}_{n+1} \cap \mathbf{\Pi}_{n+1}$. We say that a set L is on a smaller level than a set L', if we have $L \in \mathbf{\Sigma}_n \cup \mathbf{\Pi}_n$ and $L' \in (\mathbf{\Sigma}_{n'} \cup \mathbf{\Pi}_{n'}) \setminus (\mathbf{\Sigma}_{n'-1} \cup \mathbf{\Pi}_{n'-1})$ for some $n' > n$.

Remark 5.1. Let $\mathcal{G} = (\mathcal{A}, \mathrm{Win})$ be a game.

i. If $\mathcal{G}$ is a reachability game, then $\mathrm{Win} \in \mathbf{\Sigma}_1$.

ii. If $\mathcal{G}$ is a safety game, then $\mathrm{Win} \in \mathbf{\Pi}_1$.

iii. If $\mathcal{G}$ is a Büchi game, then $\mathrm{Win} \in \mathbf{\Pi}_2$.

iv. If $\mathcal{G}$ is a co-Büchi game, then $\mathrm{Win} \in \mathbf{\Sigma}_2$.

v. If $\mathcal{G}$ is a parity game, then $\mathrm{Win} \in \mathbf{\Sigma}_3 \cap \mathbf{\Pi}_3$.

vi. If $\mathcal{G}$ is a Muller game, then $\mathrm{Win} \in \mathbf{\Sigma}_3 \cap \mathbf{\Pi}_3$.

Furthermore, for each type of game appearing in the list above, there is a game which is in this class, but not in a smaller class or in the co-class (this only applies to the first four types), e.g., there is a parity game $(\mathcal{A}, \mathrm{Win})$ (which is also a Muller game) such that $\mathrm{Win} \notin \mathbf{\Sigma}_2 \cup \mathbf{\Pi}_2$.

A function $f\colon U^\omega \to V^\omega$ is continuous, if $f^{-1}(L)$ is open for every open set $L \subseteq V^\omega$. A set $L \subseteq U^\omega$ is Wadge-reducible to a set $L' \subseteq V^\omega$, written $L \leq L'$, if there exists a continuous function $f\colon U^\omega \to V^\omega$ such that $f^{-1}(L') = L$.

[18] The hierarchy also has levels $\mathbf{\Sigma}_\alpha$ and $\mathbf{\Pi}_\alpha$ for countably infinite ordinals α. Since we are only interested in the first three levels, we omit these from the definition.

Lemma 5.2 ([Wad72, Wad83])**.** *Let $L \subseteq U^\omega$ and let $L' \subseteq V^\omega$ such that $L \leq L'$.*

i. If $L' \in \mathbf{\Sigma}_n$, then $L \in \mathbf{\Sigma}_n$.

ii. If $L' \in \mathbf{\Pi}_n$, then $L \in \mathbf{\Pi}_n$.

Since the mapping from a play to its extended play used in game reductions is continuous, Lemma 5.2 shows that games cannot be reduced "down the Borel hierarchy". Using the framework presented in Section 5.4 allows to overcome this obstacle.

Lemma 5.3. *Let $\mathcal{G} = (\mathcal{A}, \mathrm{Win})$ and $\mathcal{G}' = (\mathcal{A}', \mathrm{Win}')$ be games with $\mathcal{G} \leq_{\mathcal{M}} \mathcal{G}'$ for some memory structure $\mathcal{M}$. Then, we have $\mathrm{Win} \leq \mathrm{Win}'$.*

Proof. Let $\mathcal{M} = (M, \mathrm{init}, \mathrm{upd})$ and consider the function $f\colon V^\omega \to (V \times M)^\omega$ defined by $f(\rho) = (\rho_0, m_0)(\rho_1, m_1)(\rho_2, m_2)\cdots$, where $m_0 = \mathrm{init}(\rho_0)$ and $m_{n+1} = \mathrm{upd}(m_n, \rho_{n+1})$. Hence, we have $\rho \in \mathrm{Win}$ if and only if $f(\rho) \in \mathrm{Win}'$ by definition of a game reduction. Thus, $f^{-1}(\mathrm{Win}') = \mathrm{Win}$. So, to prove $\mathrm{Win} \leq \mathrm{Win}'$, it remains to show that f is continuous, i.e., $f^{-1}(L')$ is open for every open L'.

Let $L' = K' \cdot (V \times M)^\omega$ be open. We define

$$K = \{v_0 \cdots v_n \mid (v_0, m_0) \cdots (v_n, m_n) \in K' \text{ for some } m_0, \ldots, m_n \in M\}$$

to be the projection of the words in K' to their first components. Now, let $w \in f^{-1}(L')$, i.e., $f(w) \in K' \cdot (V \times M)^\omega$. Since the j-th letter of $f(w)$ only depends on the first j letters of w, this is equivalent to $w \in K \cdot V^\omega$. Thus, $f^{-1}(L') = K \cdot V^\omega$ is indeed open. □

Lemma 5.2 shows that a game $(\mathcal{A}, \mathrm{Win})$ cannot be reduced to another game $(\mathcal{A}', \mathrm{Win}')$, if Win' is on a strictly smaller level of the Borel hierarchy than Win. The same holds true, if we have $\mathrm{Win} \in \mathbf{\Sigma}_n \setminus \mathbf{\Pi}_n$ and $\mathrm{Win}' \in \mathbf{\Pi}_n$ or vice versa. Especially, reachability, Büchi, co-Büchi, parity, and Muller games are in general not reducible to safety games.

5.2 Reducing Parity Games to Safety Games

Bernet et al. showed how to reduce a parity game to a safety game. Since we have seen in the previous section, that parity games can in general not be reduced to safety games using the reductions presented in Subsection 2.3.3, their reduction cannot be complete in the sense that it yields winning strategies for both players. Indeed, they obtain both winning regions, but only a winning strategy for one player. However, this is a special kind of strategy: their main motivation for the construction is to compute a permissive strategy, which is a non-deterministic winning strategy that subsumes the

behaviors of every positional winning strategy. We introduce such strategies in the following subsection. In the current section, we present a reformulation of the proof of Bernet et al. that is based on scoring functions for parity games. Then, in Section 5.3 we show how to lift our construction for parity games to Muller games. This allows us to reduce a Muller game to a safety game and to transfer the notion of permissive strategies from positionally determined games to games which are not necessarily positionally determined. For Muller games, we also obtain both winning regions and a winning strategy for one player.

The main idea underlying the construction for parity games is the following observation: for a priority c, let n_c denote the number of vertices labeled by c. A play that is consistent with a positional winning strategy for Player 0 does not visit $n_c + 1$ vertices of an odd priority c without visiting a larger priority in between. Hence, a vertex is in the winning region of Player 0, if she can, starting at this vertex, prevent her opponent from seeing some odd priority c exactly $n_c + 1$ times without seeing a larger priority in between. On the other hand, if Player 1 is able to enforce $n_c + 1$ visits without larger priority in between from a vertex, then this vertex is in his winning region. This is a safety condition that can be expressed by scoring functions: the function Sc_c measures how often the priority c has occurred since a larger priority has occurred. We construct an arena which keeps track of Player 1's scores up to n_c and declares him the winner if he is able to exceed this threshold. Thus, a winning strategy for Player 0 for the safety game can be turned into a strategy for her for the parity game that prevents Player 1 from reaching a score of n_c+1 for every odd priority c. Such a strategy is winning for Player 0. On the other hand, if Player 1 is able to reach a score of n_c+1 in the safety game, then the play ends. Thus, we are not able to derive a winning strategy for Player 1. To obtain a winning strategy for him, we have to construct the safety game which keeps track of Player 0's scores. In this game, it is Player 1's objective to prevent his opponent from reaching a score of $n_c + 1$ for some even priority. Alternatively, we could track the scores of both players at the same time, but would need two safety conditions in this arena to obtain winning strategies for both players: one in which Player 0 avoids high scores for Player 1 and vice versa. In the previous section, we have seen that it is impossible to reduce a parity game to a safety game in order to obtain winning strategies for both players. The construction just explained forms an alternative proof of the following theorem of Bernet et al. We discuss the differences to the original proof at the end of this section.

Theorem 5.4 ([BJW02])**.** *Let $\mathcal{G}$ be a parity game with vertex set V and priority function $\Omega\colon V \to [k]$. One can effectively construct a safety game $\mathcal{G}_S$ with vertex set V^S and a mapping $f\colon V \to V^S$ with the following properties:*

i. For every $v \in V$: $v \in W_i(\mathcal{G})$ if and only if $f(v) \in W_i(\mathcal{G}_S)$.

ii. Player 0 has a finite-state winning strategy from $W_0(\mathcal{G})$ with memory $M \subseteq W_0(\mathcal{G}_S)$.

iii. $|V^S| \leq |V|^{\lceil k/2 \rceil + 1}$.

We proceed as follows: we introduce scoring functions for parity games and construct a safety game as required by Theorem 5.4. Then, in the next subsection we show how to compute permissive strategies using this game. These constructions are then lifted to Muller games in Section 5.3 and Subsection 5.3.1.

Fix a parity game $\mathcal{G} = (\mathcal{A}, \Omega)$ with $\mathcal{A} = (V, V_0, V_1, E)$ and $\Omega \colon V \to [k]$ for some $k \in \mathbb{N}$. For each $c \in [k]$, we define

$$n_c = |\{v \in V \mid \Omega(v) = c\}|$$

to be the number of vertices with priority c. Next, we define scoring functions for parity games.

Definition 5.5. Let $c \in [k]$. We define the scoring function $\mathrm{Sc}_c \colon V^* \to \mathbb{N}$ inductively as follows: $\mathrm{Sc}_c(\varepsilon) = 0$ and for $w \in V^*$ and $v \in V$, we define

$$\mathrm{Sc}_c(wv) = \begin{cases} \mathrm{Sc}_c(w) + 1 & \text{if } \Omega(v) = c, \\ \mathrm{Sc}_c(w) & \text{if } \Omega(v) < c, \\ 0 & \text{if } \Omega(v) > c. \end{cases}$$

Furthermore, for $c \in [k]$, we define $\mathrm{MaxSc}_c \colon V^* \cup V^\omega \to \mathbb{N} \cup \{\infty\}$ by

$$\mathrm{MaxSc}_c(w) = \sup_{w' \sqsubseteq w} \mathrm{Sc}_c(w') \ .$$

This function returns the maximal score that is reached for c during w, and ∞, if arbitrarily large scores are reached. We illustrate these definitions in the following example.

Example 5.6. Consider the play prefix $w = 12210211$ with $\Omega(j) = j$. We have $\mathrm{Sc}_0(w) = 0$, $\mathrm{Sc}_1(w) = 2$, and $\mathrm{Sc}_2(w) = 3$. The score for 0 is one after the prefix 012210, but is reset to zero with the occurrence of 2. ◇

Consider a play ρ and its sequence of priorities. From some point onwards the score for the maximal priority that occurs infinitely often is no longer reset, but increased infinitely often. The score for every smaller priority is reset to zero infinitely often, while the score for every larger priority is increased only finitely often. Hence, the maximal priority that occurs infinitely often can be characterized by the limit inferior of the scoring functions.

Remark 5.7. In every play ρ, there is a unique priority c such that

$$\liminf_{n \to \infty} \mathrm{Sc}_c(\rho_0 \cdots \rho_n) = \infty \ .$$

Furthermore, this priority is equal to $\max(\Omega(\mathrm{Inf}(\rho)))$.

A simple consequence of this remark is that a play ρ is winning for Player i, if $\mathrm{MaxSc}_c(\rho) < \infty$ for every priority $c \in [k]$ with $\mathrm{Par}(c) = 1 - i$, since the limit inferior for a score is always smaller than the maximal score.

Lemma 5.8. *Let σ be a positional strategy σ for Player i. Then, σ is winning from a set W of vertices if and only if $\mathrm{MaxSc}_c(\rho) \leq n_c$ for every $\rho \in \mathrm{Beh}(W, \sigma)$ and for every priority $c \in [k]$ with $\mathrm{Par}(c) = 1 - i$.*

Proof. Let σ be winning for Player i from W and assume $\mathrm{Sc}_c(\rho_0 \cdots \rho_n) = n_c + 1$ for some $\rho \in \mathrm{Beh}(W, \sigma)$ and some $c \in [k]$ such that $\mathrm{Par}(c) = 1 - i$. For every j in the range $1 \leq j \leq n_c + 1$ let n_j denote the position of $\rho_0 \cdots \rho_n$ where the score for c is increased to j for the last time. Since the score for c is never reset between the positions n_1 and n_{n_c+1}, every other priority that occurs between these positions are strictly smaller than c. Furthermore, the pigeon-hole principle guarantees indices j, j' such that $\rho_{n_j} = \rho_{n_{j'}}$, since there are only n_c vertices with priority c. The play

$$\rho^* = \rho_0 \cdots \rho_{n_j - 1} \left(\rho_{n_j} \cdots \rho_{n_{j'} - 1} \right)^\omega$$

starts in W and is consistent with σ, since the strategy is positional. However, this play is winning for Player $1 - i$, since the largest priority that is seen infinitely often is c, which has parity $1 - i$. This is a contradiction to the fact that σ is a winning strategy for Player i from W.

The other direction of the statement follows directly from Remark 5.7, since the limit inferior of the scores for a priority c is always smaller than the maximal score for a priority. □

Hence, positional determinacy of parity games implies the existence of strategies that bound the losing players scores by n_c.

Corollary 5.9. *Player i has a uniform positional winning strategy σ such that $\mathrm{MaxSc}_c(\rho) \leq n_c$ for every $\rho \in \mathrm{Beh}(W_i(\mathcal{G}), \sigma)$ and for every priority $c \in [k]$ with $\mathrm{Par}(c) = 1 - i$*

This corollary is the basis of the reduction from a parity game to a safety game: due to determinacy, a vertex v is in the winning region of Player 0 if and only if she is able, starting in v, to prevent her opponent from reaching a score of $n_c + 1$.

As already mentioned, the safety game's arena keeps track of Player 1's scores. Hence, we can ignore the scores of Player 0 and identify play prefixes having the same scores for Player 1 by an equivalence relation. For the construction of the finite-state winning strategy it is also useful to consider a preoder on play prefixes: a play prefix w is better for Player 0 than a play prefix w', if for all all odd priorities the scores of w are smaller than the scores of w'. For technical reasons, we also require the last vertices to be equal in both relations.

Definition 5.10. Let $w, w' \in V^+$.

i. $w \leq_1 w'$, if $\mathrm{Lst}(w) = \mathrm{Lst}(w')$ and if $\mathrm{Sc}_c(w) \leq \mathrm{Sc}_c(w')$ for every odd $c \in [k]$.

ii. $w =_1 w'$, if $\mathrm{Lst}(w) = \mathrm{Lst}(w')$ and if for all odd $c \in [k]$ the equalities $\mathrm{Sc}_c(w) = \mathrm{Sc}_c(w')$ hold.

It is easy to verify that $\leq_1$ and $=_1$ are reflexive and transitive and that $=_1$ is even symmetric. Hence, $\leq_1$ is a preorder and $=_1$ is an equivalence relation. However, $\leq_1$ is not a partial order since it fails to be antisymmetric: let v and v' be two distinct vertices of priority one. Then, we have $v \leq_1 v'$ and $v' \leq_1 v$, but $v \neq v'$. Next, we show that $\leq_1$ and $=_i$ are preserved under concatenation, i.e., $=_1$ is a right-congruence. To simplify the proof, we observe that $w =_1 w'$ is equivalent to $w \leq_1 w'$ and $w' \leq_1 w$.

Lemma 5.11. *Let $w, w' \in V^+$.*

i. If $w \leq_1 w'$, then $wu \leq_1 w'u$ for every $u \in V^*$.

ii. If $w =_1 w'$, then $wu =_1 w'u$ for every $u \in V^*$.

Proof. i.) It suffices to show $wv \leq_1 w'v$ for a vertex $v \in V$. The last vertices of wv and $w'v$ are obviously equal, so consider an odd priority $c \in [k]$. We have $\mathrm{Sc}_c(w) \leq \mathrm{Sc}_c(w')$ and proceed by case distinction over the definition of Sc_c. If $\Omega(v) = c$, then

$$\mathrm{Sc}_c(wv) = \mathrm{Sc}_c(w) + 1 \leq \mathrm{Sc}_c(w') + 1 = \mathrm{Sc}_c(w'v) \ .$$

If $\Omega(v) < c$, then

$$\mathrm{Sc}_c(wv) = \mathrm{Sc}_c(w) \leq \mathrm{Sc}_c(w') = \mathrm{Sc}_c(w'v) \ .$$

Finally, if $\Omega(v) > c$, then $\mathrm{Sc}_c(wv) = 0 = \mathrm{Sc}_c(w'v)$.

ii.) Since $=_1$ can be expressed by $\leq_1$, the claim follows. □

Now, we can define the arena of the safety game to be the $=_1$-quotient of the unraveling of $\mathcal{A}$ up to the positions where Player 1 reaches a score of $n_c + 1$ for some odd priority. First, let

$$\begin{aligned}\mathrm{Plays}_{<n_c+1} = \{w \mid & w \text{ play prefix in } \mathcal{G} \text{ with} \\ & \mathrm{MaxSc}_c(w) < n_c + 1 \text{ for every odd } c \in [k]\}\end{aligned}$$

be the set of play prefixes during which Player 1 has not reached a score of $n_c + 1$ for some odd priority c, and let

$$\begin{aligned}\mathrm{Plays}_{=n_c+1} = \{wv \mid & wv \text{ play prefix in } \mathcal{G} \text{ with} \\ & \mathrm{MaxSc}_c(w) \leq n_c \text{ for every odd } c \in [k] \text{ and} \\ & \mathrm{Sc}_c(wv) = n_c + 1 \text{ for some odd } c \in [k]\}\end{aligned}$$

be the set of play prefixes in which Player 1 has just reached a score of n_c+1 for some odd priority c. Finally, let

$$\text{Plays}_{\leq n_c+1} = \text{Plays}_{<n_c+1} \cup \text{Plays}_{=n_c+1} \ .$$

We define $\mathcal{G}_S = ((V^S, V_0^S, V_1^S, E^S), F)$ where

- $V^S = \{[w]_{=_1} \mid w \in \text{Plays}_{\leq n_c+1}\}$,
- $V_0^S = \{[w]_{=_1} \mid [w]_{=_1} \in V^S \text{ and } \text{Lst}(w) \in V_0\}$,
- $V_1^S = \{[w]_{=_1} \mid [w]_{=_1} \in V^S \text{ and } \text{Lst}(w) \in V_1\}$,
- $([w]_{=_1}, [wv]_{=_1}) \in E^S$ for all $w \in \text{Plays}_{<n_c+1}$ and all v such that $(\text{Lst}(w), v) \in E$ [19], and
- $F = \{[w]_{=_1} \mid w \in \text{Plays}_{<n_c+1}\}$.

The sets V_0^S and V_1^S are well-defined, since $w =_1 w'$ implies $\text{Lst}(w) = \text{Lst}(w')$ and we have $V^S = V_0^S \cup V_1^S$ due to $V = V_0 \cup V_1$. Also, the edge relation is well-defined, since $=_1$ is a right-congruence. Finally, every equivalence class in F is also an equivalence class in V^S, since $\text{Plays}_{<n_c+1} \subseteq \text{Plays}_{\leq n_c+1}$ and since $w \in \text{Plays}_{<n_c+1}$ and $w' \in \text{Plays}_{=n_c+1}$ cannot be $=_1$-equivalent. Thus, we have $F \subseteq V^S$. For the sake of readability, we drop the subscripts from now on and denote the $=_1$-equivalence class of w by $[w]$. Furthermore, all our arguments below are independent of representatives, hence we refrain from mentioning this again.

We split the proof of Theorem 5.4 into several lemmata. We begin with the first statement: Lemma 5.12 shows the implication from left to right and Lemma 5.13 the implication from right to left. Due to determinacy of both games, if suffices to consider $i = 0$.

To show that $v \in W_0(\mathcal{G})$ implies $[v] \in W_0(\mathcal{G}_S)$ we translate a strategy for $\mathcal{G}$ that is winning from v into a strategy for $\mathcal{G}_S$ that is winning from $[v]$. There is a straightforward translation of a strategy for an arena to a strategy for its unraveling. However, in our case, the construction is complicated by two factors: $\mathcal{G}_S$ is not played in the complete unraveling, but a play is stopped as soon as a score of n_c+1 is reached for Player 1, and this incomplete unraveling is quotiented with respect to $=_1$. Hence, a play prefix in the (quotiented) unraveling does not determine a unique play prefix in the original arena. In the following, we show how to overcome these two obstacles.

Lemma 5.12. *Let $v \in V$. If $v \in W_0(\mathcal{G})$, then $[v] \in W_0(\mathcal{G}_S)$.*

[19] Hence, every vertex in $\text{Plays}_{=n_c+1}$ is terminal, contrary to our requirements on an arena. However, every play visiting these vertices is losing for Player 0 no matter how it is continued. To simplify the following proofs, we refrain from defining outgoing edges for these vertices.

Proof. Player 0 has a strategy σ for $\mathcal{G}$ such that $\mathrm{MaxSc}_c(\rho) \leq n_c$ for every play $\rho \in \mathrm{Beh}(W_0(\mathcal{G})$ and for every odd priority $c \in [k]$. Since $([w], [w']) \in E^S$ implies $(\mathrm{Lst}(w), \mathrm{Lst}(w')) \in E$, every play prefix $[w_0] \cdots [w_n]$ in $\mathcal{G}_S$ can be mapped to a play $p([w_0] \cdots [w_n]) = \mathrm{Lst}(w_0) \cdots \mathrm{Lst}(w_n)$ in $\mathcal{G}$. Furthermore, if $w_0 =_1 v$ for some $v \in V$, then $\mathrm{Lst}(w_0) = v$, i.e., if $[w_0] \cdots [w_n]$ starts in $[v]$, then $p([w_0] \cdots [w_n])$ starts in v.

We use this to define a strategy σ' for $\mathcal{G}_S$ by

$$\sigma'([w_0] \cdots [w_n]) = [w_n \cdot \sigma(p([w_0] \cdots [w_n]))]$$

for every play prefix $[w_0] \cdots [w_n]$ of $\mathcal{G}_S$ with $[w_n] \in V_0^S \cap F$. This is a legal move as $\sigma(p([w_0] \cdots [w_n]))$ is a successor of the last vertex of $p([w_0] \cdots [w_n])$ which is equal to the last vertex of w_n. Combining this with the assumption $[w_n] \in F$ yields the necessary edge between $[w_n]$ and the vertex $[w_n \cdot \sigma(p([w_0] \cdots [w_n]))]$. Finally, the restriction to play prefixes ending in F is sufficient, since all other play prefixes are already losing for Player 0 and have no outgoing edges.

It remains to show that σ' is winning from $W' = \{[v] \mid v \in W_0(\mathcal{G})\}$. We begin by showing inductively that if $[w_0] \cdots [w_n]$ starts in W' and is consistent with σ', then $p([w_0] \cdots [w_n])$ starts in $W_0(\mathcal{G})$ and is consistent with σ and that we have $p([w_0] \cdots [w_n]) \in [w_n]$. This suffices to prove that σ' is indeed a winning strategy from W', since the scores encoded by vertices reachable via σ' are equal to the scores of play prefixes that are consistent with σ and therefore at most n_c. Hence, a play that is consistent with σ' never leaves F.

Consider a play prefix $[w_0]$ of length one. Since we assume $[w_0] \in W'$, we have $w_0 =_1 v$ for some $v \in W_0(\mathcal{G})$. By definition of $=_1$, we have $\mathrm{Lst}(w_0) = \mathrm{Lst}(v) = v$ and therefore $p([w_0]) = \mathrm{Lst}(w_0) = v \in [v] = [w_0]$. Furthermore, the play prefix $p(w_0) = \mathrm{Lst}(w_0)$ is consistent with every strategy.

Now, consider a play prefix $[w_0] \cdots [w_n][w_{n+1}]$ that is consistent with σ' and remember that we have

$$p([w_0] \cdots [w_n][w_{n+1}]) = \mathrm{Lst}(w_0) \cdots \mathrm{Lst}(w_n)\mathrm{Lst}(w_{n+1})$$

by definition. We begin by showing $\mathrm{Lst}(w_0) \cdots \mathrm{Lst}(w_n)\mathrm{Lst}(w_{n+1}) \in [w_{n+1}]$. Applying Lemma 5.11 to the induction hypothesis

$$p([w_0] \cdots [w_n]) = \mathrm{Lst}(w_0) \cdots \mathrm{Lst}(w_n) =_1 w_n$$

yields

$$\mathrm{Lst}(w_0) \cdots \mathrm{Lst}(w_n)\mathrm{Lst}(w_{n+1}) =_1 w_n \cdot \mathrm{Lst}(w_{n+1}) \ .$$

Since there is an edge from $[w_n]$ to $[w_{n+1}]$, we have $w_n \cdot \mathrm{Lst}(w_{n+1}) =_1 w_{n+1}$ by definition of E^S. Hence, we have shown $p([w_0] \cdots [w_n][w_{n+1}]) \in [w_{n+1}]$.

It remains to show that $\mathrm{Lst}(w_0) \cdots \mathrm{Lst}(w_n)\mathrm{Lst}(w_{n+1})$ is consistent with σ. The induction hypothesis yields that $\mathrm{Lst}(w_0) \cdots \mathrm{Lst}(w_n)$ is consistent with σ, i.e., we only have to consider the last move, provided it is Player 0's turn at $\mathrm{Lst}(w_n)$. So, suppose we have $\mathrm{Lst}(w_n) \in V_0$, which implies $[w_n] \in V_0^S$ and thus

$$[w_{n+1}] = [w_n \cdot \sigma(p([w_0] \cdots [w_n]))] \tag{5.1}$$

by definition of σ'. Hence, we have to show $\sigma(p([w_0] \cdots [w_n]) = \mathrm{Lst}(w_{n+1})$: this follows directly from (5.1), since the $=_1$-equivalent play prefixes w_{n+1} and $w_n \cdot \sigma(p([w_0] \cdots [w_n]))$ share their last vertex. □

Now, consider the implication from right to left of Theorem 5.4(i).

Lemma 5.13. *Let $v \in V$. If $[v] \in W_0(\mathcal{G}_S)$, then $v \in W_0(\mathcal{G})$.*

For the proof, we refer to Lemma 5.18, which states a more general result than we need it here. However, let us mention an alternative proof. We can use $W_0(\mathcal{G}_S)$ as memory structure for $\mathcal{G}$ as follows: imagine the token is placed at a vertex v such that $[v] \in W_0(\mathcal{G}_S)$. Then, there is a $\leq_1$-maximal play prefix w such that $[w] \in W_0(\mathcal{G}_S)$ and $v \leq_1 w$.

Now, suppose it is Player 0's turn at v, which implies that it is also her turn at $[w]$. Since $[w]$ is in the winning region of Player 0, it has some successor $[wv'] \in W_0(\mathcal{G})$ for some vertex v'. By construction, this vertex is a successor of v in the parity game $\mathcal{G}$. In this situation, Player 0 moves to v' and updates her memory to some $[w'] \in W_0(\mathcal{G}_S)$ with $wv \leq_1 w'$ such that w' is $\leq_1$-maximal with this property. Such a vertex exists, since $[wv'] \in W_0(\mathcal{G})$.

On the other hand, if it is Player 1's turn at v, then also at $[w]$ as specified above. Since $[w]$ is in the winning region of Player 0 in a safety game, which is a trap for Player 1, each successor of $[w]$ is also in her winning region. The successors of $[w]$ have the form $[wv']$ for every successor v' of v. Hence, no matter to which vertex Player 1 moves in $\mathcal{G}$, the memory can be updated to some $[w']$ such that w' is $\leq_1$-maximal with $[w'] \in W_0(\mathcal{G}_S)$ and $wv' \leq_1 w'$.

This procedure can be iterated ad infinitum. Since the scores of a play prefix are always bounded by its memory state – which is an element of Player 0's winning region – we have implemented a finite state winning strategy from every vertex v such that $[v] \in W_0(\mathcal{G}_S)$. The memory states used by the strategy are $\leq_1$-maximal elements, hence they form a $\leq_1$-antichain in $W_0(\mathcal{G}_S)$.

This construction is explained in full detail – for the case of Muller games – in the proof of Lemma 5.24. By replacing the order relations $\leq_{\mathcal{F}_1}$ and $=_{\mathcal{F}_1}$ for Muller games by their counterparts $\leq_1$ and $=_1$ for parity games, the proof applies to parity games as well. For this reason, we do not give it here. But this construction proves the second statement of Theorem 5.4.

Corollary 5.14. *Player 0 has a finite-state winning strategy from $W_0(\mathcal{G})$ whose memory states form a $\leq_1$-antichain in $W_0(\mathcal{G}_S)$.*

Proof. We have $\{[v] \mid v \in W_0(\mathcal{G})\} \subseteq W_0(\mathcal{G}_S)$ due to Lemma 5.12. Hence, the construction described above yields a finite-state winning strategy for Player 0 from $W_0(\mathcal{G})$ whose memory states form a $\leq_1$-antichain in $W_0(\mathcal{G}_S)$ as required. □

It remains to bound the size of the safety game.

Lemma 5.15. *We have*

$$|V^S| \leq |V| \cdot \prod_{\substack{c \in [k] \\ \mathrm{Par}(c)=1}} (n_c + 1) \leq |V|^{\lceil k/2 \rceil + 1} \ .$$

Proof. In every safety game, we can merge the vertices in $V \setminus F$ to a single vertex without changing $W_0(\mathcal{G})$. Since $[v] \in F$ for every vertex v of $\mathcal{G}$, we also retain the equivalence $v \in W_i(\mathcal{G}) \Leftrightarrow [v] \in W_i(\mathcal{G}_S)$.

Hence, it remains to bound the index of $=_i$ on $\mathrm{Plays}_{<n_c+1}$. Two play prefixes in $\mathrm{Plays}_{<n_c+1}$ are equivalent, if they have the same last vertex and the same score for every odd priority c, of which there are at most $\lceil k/2 \rceil$. Furthermore, since we have $0 \leq \mathrm{Sc}_c(w) \leq n_c < |V|$ for every play prefix $w \in \mathrm{Plays}_{<n_c+1}$, we obtain our result. In the last inequality, we assume $n_c < |V|$, since the parity game is trivial, if we have $n_c = |V|$ for some c. □

5.2.1 Permissive Strategies for Parity Games

Bernet et. al. used the reduction from parity to safety games to compute permissive strategies for parity games: in this framework, a strategy is allowed to be non-deterministic in the sense that it does not prescribe one successor, but a non-empty set of successors. A strategy is permissive, if it subsumes the behavior of every positional (non-deterministic) winning strategy. Let us formalize these notions following [BJW02]. Then, we show how to derive a permissive strategy from the safety game constructed above.

Fix an arena $\mathcal{A} = (V, V_0, V_1, E)$. A multi-strategy for Player i is a mapping $\sigma \colon V^* V_i \to 2^V \setminus \{\emptyset\}$ such that $v' \in \sigma(wv)$ implies $(v, v') \in E$. A play ρ is consistent with σ if $\rho_{n+1} \in \sigma(\rho_0 \cdots \rho_n)$ for every n such that $\rho_n \in V_i$. Again, a play prefix $\rho_0 \cdots \rho_m$ is consistent with σ, if $\rho_{n+1} \in \sigma(\rho_0 \cdots \rho_n)$ for every $n < m$ such that $\rho_n \in V_i$. We still denote the plays starting in a vertex v that are consistent with a multi-strategy σ by $\mathrm{Beh}_{\mathcal{A}}(v, \sigma)$ and we define $\mathrm{Beh}_{\mathcal{A}}(W, \sigma) = \bigcup_{v \in W} \mathrm{Beh}(v, \sigma)$ for every subset $W \subseteq V$. Again, we drop the subscript whenever this is possible without introducing ambiguity. A multi-strategy σ is winning for Player 0 from a set of vertices W in a game $(\mathcal{A}, \mathrm{Win})$ if $\mathrm{Beh}_{\mathcal{A}}(W, \sigma) \subseteq \mathrm{Win}$, and a multi-strategy τ is winning for Player 1 from W, if $\mathrm{Beh}_{\mathcal{A}}(W, \sigma) \subseteq V^\omega \setminus \mathrm{Win}$. It is clear that the winning regions of a game do not change when we allow multi-strategies instead of standard strategies, since every strategy can be seen as a multi-strategy.

To define finite-state multi-strategies we have to adapt the notion of a next-move function. Consider a memory structure $\mathcal{M} = (M, \text{init}, \text{upd})$ a (non-deterministic) next-move function $\text{nxt}\colon V_i \times M \to 2^V \setminus \{\emptyset\}$ such that $v' \in \text{nxt}(v, m) \in E$ implies (v, v'). It implements a multi-strategy σ via $\sigma(wv) = \text{nxt}(v, \text{upd}^*(wv))$. A multi-strategy σ is called positional, if it can be implemented with a single memory state. Again, this is equivalent to $\sigma(wv) = \sigma(v)$ for every play prefix wv.

Multi-strategies are compared by the amount of non-determinism they allow: let σ and σ' be multi-strategies for Player i in $(\mathcal{A}, \text{Win})$ that are winning from exactly the vertices in W and W', respectively. We have $\sigma \sqsubseteq_p \sigma'$, if $W \subseteq W'$ and $\text{Beh}(W, \sigma) \subseteq \text{Beh}(W, \sigma')$, i.e., σ' is at least winning from all vertices from which σ is winning and from those vertices, σ' allows all plays that σ allows. Note that the definition only refers to plays starting in vertices from which σ is winning. The behavior of both strategies from the other vertices is irrelevant.

Definition 5.16. Let $\mathcal{G}$ be a game. A multi-strategy σ for Player i is permissive, if we have $\sigma' \sqsubseteq_p \sigma$ for every positional multi-strategy σ'.

A permissive strategy for Player i for a parity game is winning from every vertex in $W_i(\mathcal{G})$ since it is $\sqsubseteq_p$-greater than a uniform positional winning strategy, which means that it is at least winning from every vertex in the winning region of Player i. On the other hand, it cannot be winning from any vertex in $W_{1-i}(\mathcal{G})$. Furthermore, it is essential to require to only subsume positional strategies in a parity game: the following example shows that there is not necessarily a winning multi-strategy that subsumes the behaviors of all strategies.

Example 5.17. Consider the parity game $(\mathcal{A}, \Omega)$ where the arena $\mathcal{A}$ with vertex set $\{0, 1\}$ is depicted in Figure 5.1 and the priority of a vertex is given by its name. Player 0 has a winning strategy by moving the token to 0 (if it is not already there) and then staying in this vertex ad infinitum.

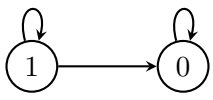

Figure 5.1: The arena $\mathcal{A}$ for Example 5.17

On the other hand, consider the strategy σ_n (for every $n \in \mathbb{N}$) which uses the self-loop at vertex 1 n times before it moves the token to vertex 0. A multi-strategy σ with $\text{Beh}(1, \sigma_n) \subseteq \text{Beh}(1, \sigma)$ for every n satisfies $1^\omega \in \text{Beh}(1, \sigma)$. However, this play is not winning for Player 0, which means that σ is not winning from 1. Thus, $\sigma_n \not\sqsubseteq_p \sigma$, since the strategies σ_n are winning from both vertices. ◇

There is a tight connection between safety games and permissive strategies. On the one hand, Bernet et al. showed that every safety game has a positional $\sqsubseteq_p$-maximal winning strategy. On the other hand, they showed that if a parity game $\mathcal{G}$ has a $\sqsubseteq_p$-maximal winning strategy, then $\mathcal{G}$ is a safety game, i.e., there is a subset F of the vertices such that a play satisfies the parity condition if and only if the play only visits vertices in F. Note that in the game $\mathcal{G}$ in Example 5.17 – which has no $\sqsubseteq_p$-maximal winning strategy – there is indeed no such set F.

We conclude this section by showing how to construct a finite-state permissive strategy for a parity game $\mathcal{G}$ from the safety game $\mathcal{G}_S$ defined in the previous section. This construction again relies on the fact that a positional winning strategy only allows plays that visit odd priorities c at most n_c times without visiting a larger priority in between. Hence, the prefix of each play that is consistent with such a strategy is contained in the winning region of Player 0 in the safety game. A strategy that allows exactly the plays whose prefixes lie in $W_0(\mathcal{G}_S)$ is permissive. This theorem also implies Lemma 5.13, since every winning multi-strategy can be restricted to be a winning (standard) strategy.

Theorem 5.18. *Let $\mathcal{G}$ be a parity game and let $\mathcal{G}_S$ be the safety game as above. Then, Player 0 has a finite-state permissive strategy for $\mathcal{G}$ with memory states $W_0(\mathcal{G}_S)$.*

Proof. We define $\mathcal{M} = (M, \text{init}, \text{upd})$ where $M = W_0(\mathcal{G}_S) \cup \{\bot\}$ [20], where

$$\text{init}(v) = \begin{cases} [v] & \text{if } [v] \in W_0(\mathcal{G}_S), \\ \bot & \text{otherwise,} \end{cases}$$

and

$$\text{upd}([w], v) = \begin{cases} [wv] & \text{if } [wv] \in W_0(\mathcal{G}_S), \\ \bot & \text{otherwise.} \end{cases}$$

Hence, we have $\text{upd}^*(w) = [w] \in W_0(\mathcal{G}_S)$ as long as every prefix x of w satisfies $[x] \in W_0(\mathcal{G}_S)$, and $\text{upd}^*(w) = \bot$ otherwise. We define a next-move function by $\text{nxt}(v, \bot) = \{v'\}$ for some successor v' of v and

$$\text{nxt}(v, [w]) = \begin{cases} \{v' \mid [wv'] \in W_0(\mathcal{G}_S)\} & \text{if } \text{Lst}(w) = v, \\ \{v''\} & \text{otherwise, where } v'' \text{ is some} \\ & \text{successor of } v. \end{cases}$$

[20] We use the memory state $\bot$ to simplify our proof. It is not reachable via plays that are consistent with the strategy implemented by $\mathcal{M}$ and can therefore eliminated and its incoming transitions can be redefined arbitrarily.

We need to show that the function always returns a non-empty set of successors of v. The first and third case are clear by definition, so consider the second one: let $[w] \in W_0(\mathcal{G}_S)$ and $v = \mathrm{Lst}(w)$. Then, there is at least one successor of $[w]$ in $W_0(\mathcal{G}_S)$ and all of these successors are of the form $[wv']$ for some successor v' of v.

We continue by showing that the strategy σ implemented by $\mathcal{M}$ and nxt is winning from every vertex in the winning region of Player 0 in the parity game. So, let $\rho \in \mathrm{Beh}(W_0(\mathcal{G}), \sigma)$. We claim that we have $\mathrm{upd}^*(\rho_0 \cdots \rho_n) = [\rho_0 \cdots \rho_n] \in W_0(\mathcal{G}_S)$ for every n. This implies that the scores of Player 1 are bounded by n_c. Thus, ρ is winning for Player 0.

For $n = 0$, we have $\rho_n \in W_0(\mathcal{G})$ which implies $[v] \in W_0(\mathcal{G}_S)$ due to Lemma 5.12. Hence, $\mathcal{M}$ initializes the memory to $\mathrm{init}(\rho_0) = [\rho_0]$, which yields the induction start.

For $n > 0$, we distinguish two cases. If $\rho_{n-1} \in V_0$, then we have

$$\begin{aligned} &\rho_n \in \sigma(\rho_0 \cdots \rho_{n-1}) \\ =&\mathrm{nxt}(\rho_{n-1}, \mathrm{upd}^*(\rho_0 \cdots \rho_{n-1})) \\ =&\mathrm{nxt}(\rho_{n-1}, [\rho_0 \cdots \rho_{n-1}]) \end{aligned}$$

where the last equality is the application of the induction hypothesis. The hypothesis also yields $\rho_0 \cdots \rho_{n-1} \in W_0(\mathcal{G})$ and therefore

$$\rho_n \in \{v' \mid [\rho_0 \cdots \rho_{n-1} v'] \in W_0(\mathcal{G}_S)\}$$

by definition of the next-move function. Hence, we have $[\rho_0 \cdots \rho_{n-1}\rho_n] \in W_0(\mathcal{G}_S)$, which implies $\mathrm{upd}^*(\rho_0 \cdots \rho_n) = [\rho_0 \cdots \rho_{n-1}\rho_n]$.

On the other hand, if $\rho_{n-1} \in V_1$, then $[\rho_0 \cdots \rho_{n-1}] \in V_1^S$ by definition of $\mathcal{G}_S$. Furthermore, since we have $(\rho_{n-1}, \rho_n) \in E$, $[\rho_0 \cdots \rho_{n-1}\rho_n]$ is a successor of $[\rho_0 \cdots \rho_{n-1}]$ in $\mathcal{G}_S$. Finally, since $W_0(\mathcal{G}_S)$ as winning region of Player 0 in a safety game is a trap for Player 1, we conclude $[\rho_0 \cdots \rho_{n-1}\rho_n] \in W_0(\mathcal{G})$, which implies $\mathrm{upd}^*(\rho_0 \cdots \rho_n) = [\rho_0 \cdots \rho_{n-1}\rho_n]$.

It remains to show that we have $\mathrm{Beh}(v, \sigma') \subseteq \mathrm{Beh}(v, \sigma)$, where σ' is a positional strategy that is winning from v. Towards a contradiction, assume there is a play $\rho \in \mathrm{Beh}(v, \sigma) \setminus \mathrm{Beh}(v, \sigma')$. Then, there is a smallest position $\rho_n \in V_0$ such that

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

We have shown above that we have $\mathrm{upd}^*(\rho_0 \cdots \rho_n) = [\rho_0 \cdots \rho_n]$, as $\rho_0 \cdots \rho_n$ is consistent with σ, i.e., the definition of nxt yields $[\rho_0 \cdots \rho_{n+1}] \in W_1(\mathcal{G}_S)$. Hence, Player 1 is able to enforce a visit to $V^S \setminus F$ in the safety games $\mathcal{G}_S$ using an attractor strategy τ. This strategy can translated into a strategy τ' for him for $\mathcal{G}$ that enforces a score of $n_c + 1$ for some odd priority c starting from the play prefix $\rho_0 \cdots \rho_{n+1}$. Consider the continuation ρ' of $\rho_0 \cdots \rho_{n+1}$ that is consistent with σ' and τ'. In this play, Player 1 reaches a score of $n_c + 1$ for

some odd priority. Thus, Lemma 5.8 yields the desired contradiction, since ρ' is consistent with the positional strategy σ' that is winning from ρ and therefore bounds Player 1's scores by n_c. □

Note that we did not only show that the plays of every positional strategy are subsumed, but actually that the strategy allows every play in which Player 1, starting from any prefix, cannot enforce a score of $n_c + 1$ for some odd c. It is just the case that a positional winning strategy bounds the scores of the losing player by these values. We come back to this when we generalize the notion of permissiveness to Muller games.

Let us briefly discuss the original proof of Theorem 5.4. The equivalence classes used as memory states in the construction of the safety game above can be seen as elements in

$$V \times [n_1 + 1] \times [n_3 + 1] \times \cdots \times [n_{k'} + 1] \ ,$$

where k' is the largest odd integer such that $k' < k$. Such a vector encodes the last vertex of the play prefix and the score for every odd priority. It is the same memory structure that Bernet et al. use in their proof of Theorem 5.4 and to construct permissive strategies for parity games. However, they use a different update function: just as in our definition, the occurrence of priority c resets the scores for all smaller priorities and increases the score for c. If the score for an odd priority c has already value n_c and is increased again, then the score for the smallest odd $c' > c$ whose score has not yet reached value $n_{c'}$ is increased. If no such c' exists, then the play is stopped. In contrast, according to our definition, a play is stopped if the score of n_c is reached. Hence, the plays according to our definition are stopped earlier.

However, the update function of Bernet et al. has a great advantage: it is monotonic with respect to the (inverse) lexicographic order on

$$[n_1 + 1] \times [n_3 + 1] \times \cdots \times [n_{k'} + 1] \ ,$$

i.e., vectors are compared from right to left. This reflects the fact that larger priorities are more important, since the maximal priority that is seen infinitely often determines the winner of a play. The totality of the lexicographic order allows to find small representations of permissive strategies and is also the basis of Jurdziński's small progress measure algorithm for parity games [Jur00], which is closely related to this memory structure. We, on the other hand, compare vectors componentwise, an order which lacks totality. In the next section, we lift the construction presented above from parity games to Muller games. The order we use for Muller games is also not a total one, since it is unclear how to linearly order the sets in $\mathcal{F}_1$ according to their importance (especially those of equal cardinality). Hence, the situation there is much closer to the one presented in this section than to the one of Bernet et al. This is the reason we presented the alternative construction.

5.3 Reducing Muller Games to Safety Games

In this section, we show how to determine the winning regions of a Muller game and a winning strategy for one player by solving a safety game. Obviously, this can be done by reducing a Muller game to a parity game and then applying the reduction of Bernet et al. presented in the previous section. A priori, each reduction involves an exponential blowup, although it might be the case that the games obtained by composing these reductions are simple on some sense, which could prevent a doubly-exponential blowup. But instead of taking the detour via parity games, we give a direct reduction from Muller games to safety games. Again, note that this is not a classical reduction, since this is ruled by the fact that Muller conditions are on a higher level of the Borel hierarchy than safety conditions. This is witnessed by the fact that we only obtain a winning strategy for one player.

The proof idea is completely similar to the score-based reduction from parity games to safety games: from her winning region, Player i can prevent her opponent from ever reaching a score of three. Hence, if Player 1 is able, starting at a vertex v, to enforce a score of three, then v is not in the winning region of Player 0, and therefore in the winning region of Player 1. Hence, we again construct an arena which tracks the scores of Player 1 and define a safety condition which declares Player 0 to win a play if and only if Player 1 never reaches a score of three during the play. By analyzing the intrinsic structure of the vertices of the safety game, we are able to construct a new, antichain-based memory structure for Muller games. Furthermore, in the next subsection, we prove that this construction also allows to transfer the notion of permissiveness to Muller games.

The reduction presented in the following is implemented in the tool `GAVS+`[21] [CKLB11].

Theorem 5.19. *Let $\mathcal{G}$ be a Muller game with vertex set V. One can effectively construct a safety game $\mathcal{G}_S$ with vertex set V^S and a mapping $f\colon V \to V^S$ with the following properties:*

i. For every $v \in V$: $v \in W_i(\mathcal{G})$ if and only if $f(v) \in W_i(\mathcal{G}_S)$.

ii. Player 0 has a uniform finite-state winning strategy from $W_0(\mathcal{G})$ with memory $M \subseteq W_0(\mathcal{G}_S)$.

iii. $|V^S| \leq (n!)^3$ where $n = |V|$.

Compare this theorem to its analogue for parity games, Theorem 5.4. The first two statements are similar: the reduction yields the winning regions for both players and a finite-state winning strategy for Player 0. The third statement shows that our construction yields a safety game that is smaller

[21] See `http://www6.in.tum.de/~chengch/gavs/` for details and to download the tool.

than the one obtained by reducing a Muller game first to a parity game and then apply Theorem 5.4: we have

$$(n!)^3 = \left(2^{\log_2(n!)}\right)^3 = \left(2^{\sum_{j=1}^{n} \log_2(j)}\right)^3 \leq \left(2^{n \log_2(n)}\right)^3 = 2^{3n \log_2(n)}$$

while the two-step reduction via parity games yields a safety game of doubly-exponential size. However, it might be possible that this can be improved by exploiting the intrinsic structure of the memory used to reduce the Muller game to a parity game in the first step.

Also, just as in the analogue for parity games, we can obtain a winning strategy for Player 1 by swapping the roles of the players and construct a safety game which keeps track of the scores of Player 0 or by constructing the arena which keeps track of both player's scores. However, that would require to define two safety games in this arena: one in which Player 0 has to avoid a score of three for Player 1 and vice versa. Also, this arena is larger than the ones which only track the scores of one player (but still smaller than $(|V|!)^3$). We have seen in Section 5.1, that it is impossible to reduce a Muller game to a safety game to obtain winning strategies for both players.

The proof of Theorem 5.19 proceeds as the one for parity games: we define an equal-score relation $\leq_{\mathcal{F}_1}$ for play prefixes in Muller games and define the safety game's arena as the $=_{\mathcal{F}_1}$-quotient of the unraveling of $\mathcal{A}$ restricted to those plays in which Player 1's scores are bounded by two. Then, we show how to translate winning strategies between these two games to prove the first two statements of the theorem. We finish by bounding the index of $=_{\mathcal{F}_1}$ to prove our claim about the size of the safety game.

We begin by defining the safety game $\mathcal{G}_S$. Let $\mathcal{G} = (\mathcal{A}, \mathcal{F}_0, \mathcal{F}_1)$ with $\mathcal{A} = (V, V_0, V_1, E)$. Since we are only interested in the scores of Player 1, we can identify play prefixes having the same score and accumulator for every set in $\mathcal{F}_1$. We need to keep track of the accumulators as well, since they are used to compute the scores. For technical reason we also require the last vertices to be equal. We complement this equivalence relation by a score-based preorder on play prefixes. Intuitively, a play prefix w' is better for Player 0 than another play prefix w if for each $F \in \mathcal{F}_1$ the score for F of w is smaller than the score for F of w'. If the score is equal for some set $F \in \mathcal{F}_1$, then we require the accumulator of w' to be a superset of the accumulator of w. Again, we also require the last vertices of the play prefixes to be equal.

Definition 5.20. Let $w, w' \in V^+$.

i. w is $\mathcal{F}_1$-smaller than w', denoted by $w \leq_{\mathcal{F}_1} w'$, if $\mathrm{Lst}(w) = \mathrm{Lst}(w')$ and for all $F \in \mathcal{F}_1$:

   - $\mathrm{Sc}_F(w) < \mathrm{Sc}_F(w')$, or
   - $\mathrm{Sc}_F(w) = \mathrm{Sc}_F(w')$ and $\mathrm{Acc}_F(w) \subseteq \mathrm{Acc}_F(w')$.

ii. w and w' are $\mathcal{F}_1$-equivalent, denoted by $w =_{\mathcal{F}_1} w'$, if $\mathrm{Lst}(w) = \mathrm{Lst}(w')$ and for all $F \in \mathcal{F}_1$ the equalities $\mathrm{Sc}_F(w) = \mathrm{Sc}_F(w')$ and $\mathrm{Acc}_F(w) = \mathrm{Acc}_F(w')$ are satisfied.

Note that $=_{\mathcal{F}_1}$ can be expressed by $\leq_{\mathcal{F}_1}$: we have $w =_{\mathcal{F}_1} w'$ if and only if $w \leq_{\mathcal{F}_1} w'$ and $w' \leq_{\mathcal{F}_1} w$. It is easy to verify that $\leq_{\mathcal{F}_1}$ and $=_{\mathcal{F}_1}$ are reflexive and transitive and that $=_{\mathcal{F}_1}$ is even symmetric. Hence, $\leq_{\mathcal{F}_1}$ is a preorder and $=_{\mathcal{F}_1}$ is an equivalence relation. We denote the $\mathcal{F}_1$ equivalence class of w by $[w]_{=_{\mathcal{F}_1}}$. Note that $\leq_{\mathcal{F}_1}$ is not a partial order since it fails to be antisymmetric for any $\mathcal{F}_1$: we have $abab \leq_{\mathcal{F}_1} baab$ and $baab \leq_{\mathcal{F}_1} abab$ for every $\mathcal{F}_1 \subseteq 2^{\{a,b\}}$, but obviously not $abab = baab$. Both $\leq_{\mathcal{F}_1}$ and $=_{\mathcal{F}_1}$ are preserved under concatenation, i.e., $=_{\mathcal{F}_1}$ is a right-congruence.

Lemma 5.21. *Let $w, w' \in V^+$.*

i. If $w \leq_{\mathcal{F}_1} w'$, then $wu \leq_{\mathcal{F}_1} w'u$ for all $u \in V^*$.

ii. If $w =_{\mathcal{F}_1} w'$, then $wu =_{\mathcal{F}_1} w'u$ for all $u \in V^*$.

Proof. i.) It suffices to show $w \leq_{\mathcal{F}_1} w'$ implies $wv \leq_{\mathcal{F}_1} w'v$ for every $v \in V$. The last vertices of wv and wv' are obviously equal. So, let $F \in \mathcal{F}_1$. We have $\mathrm{Sc}_F(w) \leq \mathrm{Sc}_F(w')$ and $\mathrm{Acc}_F(w) \subseteq \mathrm{Acc}_F(w')$ in case $\mathrm{Sc}_F(w) = \mathrm{Sc}_F(w')$; and we need to show $\mathrm{Sc}_F(wv) \leq \mathrm{Sc}_F(w'v)$ and $\mathrm{Acc}_F(wv) \subseteq \mathrm{Acc}_F(w'v)$ in case that $\mathrm{Sc}_F(wv) = \mathrm{Sc}_F(w'v)$.

If $v \notin F$, then the score for F is reset by the occurrence of v and we have $\mathrm{Sc}_F(wv) = \mathrm{Sc}_F(w'v) = 0$ and $\mathrm{Acc}_F(wv) = \mathrm{Acc}_F(w'v) = \emptyset$.

Now, suppose we have $v \in F$. First, consider the case $\mathrm{Sc}_F(w) < \mathrm{Sc}_F(w')$: either the score for F does not increase by visiting v after w and we have

$$\mathrm{Sc}_F(wv) = \mathrm{Sc}_F(w) < \mathrm{Sc}_F(w') \leq \mathrm{Sc}_F(w'v)$$

or the score increases in wv and we have

$$\mathrm{Sc}_F(wv) = \mathrm{Sc}_F(w) + 1 \leq \mathrm{Sc}_F(w') \leq \mathrm{Sc}_F(w'v)$$

and $\mathrm{Acc}_F(wv) = \emptyset$, due to the score increase. Hence, $\mathrm{Acc}_F(wv) \subseteq \mathrm{Acc}_F(w'v)$ holds in case $\mathrm{Sc}_F(wv) = \mathrm{Sc}_F(w'v)$.

Now, consider the case $\mathrm{Sc}_F(w) = \mathrm{Sc}_F(w')$, which implies $\mathrm{Acc}_F(w) \subseteq \mathrm{Acc}_F(w')$. If $\mathrm{Acc}_F(w) = F \setminus \{v\}$, then also $\mathrm{Acc}_F(w') = F \setminus \{v\}$, as the accumulator for F can never be F. In this situation, we have

$$\mathrm{Sc}_F(wv) = \mathrm{Sc}_F(w) + 1 = \mathrm{Sc}_F(w') + 1 = \mathrm{Sc}_F(w'v)$$

and $\mathrm{Acc}_F(wv) = \mathrm{Acc}_F(w'v) = \emptyset$. Otherwise, if $\mathrm{Acc}_F(w) \neq F \setminus \{v\}$, then

$$\mathrm{Sc}_F(wv) = \mathrm{Sc}_F(w) = \mathrm{Sc}_F(w') \leq \mathrm{Sc}_F(w'v) \ .$$

If $\mathrm{Sc}_F(w') < \mathrm{Sc}_F(w'v)$, then we are done. So, consider the case $\mathrm{Sc}_F(w') = \mathrm{Sc}_F(w'v)$: we have

$$\mathrm{Acc}_F(wv) = \mathrm{Acc}_F(w) \cup \{v\} \subseteq \mathrm{Acc}_F(w') \cup \{v\} = \mathrm{Acc}_F(w'v) \ ,$$

due to $\mathrm{Acc}_F(w) \subseteq \mathrm{Acc}_F(w')$ and the fact that the score at $w'v$ does not increase, which implies that the accumulator for wv' is obtained by adding v' to the accumulator of w'.

ii.) The play prefixes w and w' are $\mathcal{F}_1$-equivalent if and only if both $w \leq_{\mathcal{F}_1} w'$ and $w' \leq_{\mathcal{F}_1} w$ hold. Hence, applying i.) yields $wu \leq_{\mathcal{F}_1} w'u$ and $w'u \leq_{\mathcal{F}_1} wu$ and hence $wu =_{\mathcal{F}_1} w'u$. □

The arena of the safety game we are about to define is the $=_{\mathcal{F}_1}$-quotient of the unraveling of $\mathcal{A}$ up to the positions where Player 1 reaches a score of three for the first time (if he does at all). Thus, we define

$$\mathrm{Plays}_{<3} = \{w \mid w \text{ play prefix in } \mathcal{G} \text{ and } \mathrm{MaxSc}_{\mathcal{F}_1}(w) < 3\}$$

to be the set of play prefixes in $\mathcal{G}$ in which the scores of Player 1 are at most 2 and we define

$$\mathrm{Plays}_{=3} = \{wv \mid wv \text{ play prefix in } \mathcal{G}, \mathrm{MaxSc}_{\mathcal{F}_1}(w) \leq 2, \text{ and } \mathrm{MaxSc}_{\mathcal{F}_1}(wv) = 3\}$$

to be the set of play prefixes in which Player 1 just reached a score of three. Furthermore, let

$$\mathrm{Plays}_{\leq 3} = \mathrm{Plays}_{<3} \cup \mathrm{Plays}_{=3} \ .$$

We define $\mathcal{G}_S = ((V^S, V_0^S, V_1^S, E^S), F)$ where

- $V^S = \{[w]_{=_{\mathcal{F}_1}} \mid w \in \mathrm{Plays}_{\leq 3}\}$,
- $V_0^S = \{[w]_{=_{\mathcal{F}_1}} \mid [w]_{=_{\mathcal{F}_1}} \in V^S \text{ and } \mathrm{Lst}(w) \in V_0\}$,
- $V_1^S = \{[w]_{=_{\mathcal{F}_1}} \mid [w]_{=_{\mathcal{F}_1}} \in V^S \text{ and } \mathrm{Lst}(w) \in V_1\}$,
- $([w]_{=_{\mathcal{F}_1}}, [wv]_{=_{\mathcal{F}_1}}) \in E^S$ for every $w \in \mathrm{Plays}_{<3}$ and every v such that $(\mathrm{Lst}(w), v) \in E$ [22], and
- $F = \{[w]_{=_{\mathcal{F}_1}} \mid w \in \mathrm{Plays}_{<3}\}$.

[22] Hence, every vertex in $\mathrm{Plays}_{=3}$ is terminal, contrary to our requirements on an arena. However, every play visiting these vertices is losing for Player 0 no matter how it is continued. To simplify the following proofs, we refrain from defining outgoing edges for these vertices.

The definitions of V_0^S and V_1^S are independent of representatives, as $w =_{\mathcal{F}_1} w'$ implies $\mathrm{Lst}(w) = \mathrm{Lst}(w')$, and we have $V^S = V_0^S \cup V_1^S$ due to $V = V_0 \cup V_1$. The edge relation E^S is well-defined, since $=_{\mathcal{F}_1}$ is a right-congruence. Finally, we have $F \subseteq V$, since we have $\mathrm{Plays}_{<3} \subseteq \mathrm{Plays}_{\leq 3}$ and since $w \in \mathrm{Plays}_{<3}$ and $w' \in \mathrm{Plays}_{=3}$ cannot be $\mathcal{F}_1$-equivalent. For the sake of readability, we drop the subscripts from now on and denote the $=_{\mathcal{F}_1}$-equivalence class of w by $[w]$. Furthermore, all definitions and statements below are independent of representatives; hence, we refrain from mentioning independence of representatives from now on.

Remark 5.22. If $([w]_{=_{\mathcal{F}_1}}, [w']_{=_{\mathcal{F}_1}}) \in E^S$, then $(\mathrm{Lst}(w), \mathrm{Lst}(w')) \in E$.

We illustrate the definition of $\mathcal{G}_S$ in the following example, which runs through the whole section.

Example 5.23. The safety game $\mathcal{G}_S$ for the Muller game $\mathcal{G}$ from Example 4.12 is depicted in Figure 5.2. Its vertices are the $=_{\mathcal{F}_1}$ equivalence classes of the play prefixes of $\mathcal{G}$ in which Player 1 never reached a score of three (in the first five columns) or in which he just reached a score of three (in the last column). The latter equivalence classes are the only vertices which are not in F. Let us illustrate the definition of the edge relation: there is an edge from [1001] to [12] since there is an edge from 1 to 2 in the original game $\mathcal{G}$, and since $10012 =_{\mathcal{F}_1} 12$. Furthermore, there is a self-loop at vertex [2], since there is one at vertex 2 in $\mathcal{G}$ and since we have $2 =_{\mathcal{F}_1} 22$.

One can verify easily that the vertices $[v]$ for $v \in V$ are in the winning region of Player 0. This corresponds to the fact that Player 0's winning region in the Muller game contains every vertex of the arena. ◇

The proof of Theorem 5.19 is split into several lemmata. Due to determinacy of both Muller and safety games, it suffices to consider $i = 0$ to prove Theorem 5.19(i). The direction from left to right is shown in Lemma 5.24, the direction from right to left in Lemma 5.28.

To show that $v \in W_0(\mathcal{G})$ implies $[v] \in W_0(\mathcal{G}_S)$ we translate a strategy for $\mathcal{G}$ that is winning from v into a strategy for $\mathcal{G}_S$ that is winning from $[v]$. The construction is the same as the one in the proof of izs analogue, Lemma 5.12, we just have to replace $=_1$ by $=_{\mathcal{F}_1}$. For this reason, we skip the proof.

Lemma 5.24. *For every $v \in V$: if $v \in W_0(\mathcal{G})$ then $[v] \in W_0(\mathcal{G}_S)$.*

For the other direction of Theorem 5.19(i), we show that a subset of $W_0(\mathcal{G}_S)$ can be turned into a memory structure for Player 0 in the Muller game that implements a winning strategy. We intend to use the $=_{\mathcal{F}_1}$-equivalence class $[w]$ of w as memory state to keep track of Player 1's scores in $\mathcal{G}$. Suppose we have $[w] \in W_0(\mathcal{G}_S)$: if it is Player 0's turn at the last vertex of w in the Muller game, then she has to determine a successor to move the token to. By construction of $\mathcal{G}_S$, it is also Player 0's turn at $[w]$. Since we

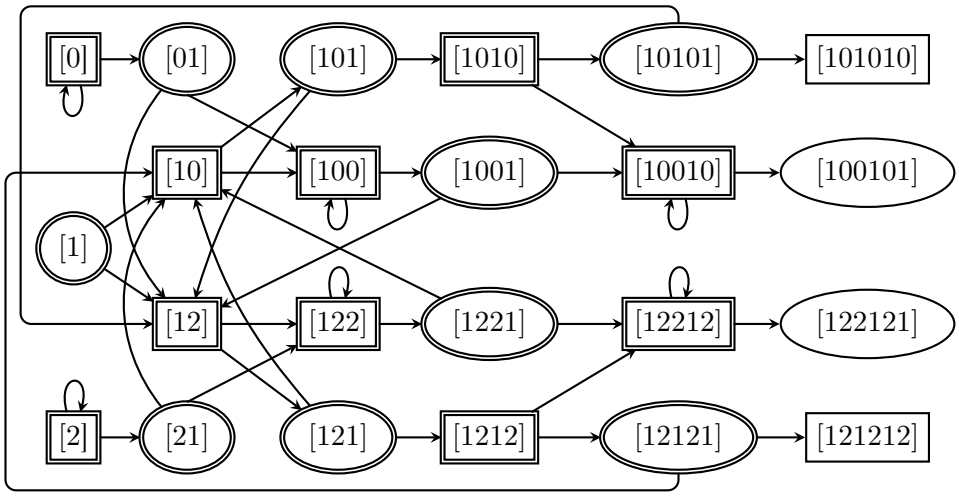

Figure 5.2: The safety game $\mathcal{G}_S$ for $\mathcal{G}$ from Example 4.12 (vertices drawn with double lines are in F)

have $[w] \in W_0(\mathcal{G}_S)$, there is a successor $[wv] \in W_0(\mathcal{G}_S)$ of $[w]$, where v is a vertex in V such that $(\mathrm{Lst}(w), v) \in E$ due to Remark 5.22. Our proposed strategy for the Muller game prescribes to move to v and updates the memory to $[wv] \in W_0(\mathcal{G})$. On the other hand, assume it is Player 1's turn at the last vertex of w and he moves the token to a successor v. Then, it is also his turn at $[w]$ in $\mathcal{G}_S$ and the successor $[wv]$ of $[w]$ is in $W_0(\mathcal{G}_S)$ as well, since $W_0(\mathcal{G}_S)$ is a trap for him. Hence, we can update the memory to $[wv]$ while maintaining the invariant that the memory state – which is the equivalence class of the current play prefix – is always an element of $W_0(\mathcal{G}_S)$. This strategy is winning for Player 0 for the Muller game $\mathcal{G}$ from every vertex v such that $[v] \in W_0(\mathcal{G}_S)$, since it bounds the scores by the scores encoded by the equivalence classes in $W_0(\mathcal{G})$, which are in turn bounded by two.

Example 5.25. Consider the winning region $W_0(\mathcal{G}_S)$ in the safety game $\mathcal{G}_S$ of Example 5.23 as depicted in Figure 5.3 and recall that this is the safety game that corresponds to the Muller game $\mathcal{G}$ from Example 4.12.

We show how to use this winning region to implement a winning strategy for $\mathcal{G}$: assume the token is placed at vertex 0 in $\mathcal{G}$. Then, we initialize the memory to [0]. If Player 1 uses the self-loop and moves the token to 0, the memory state is left unchanged. This reflects the fact that he does not change any of his scores or accumulators by using the self-loop in this situation. On the other hand, if he moves the token to vertex 1, then the memory is updated to [01] and its Player 0's turn. Now, she has two choices, which correspond to the successors [100] (move to 0 in $\mathcal{G}$) and [12] (move to 2 in $\mathcal{G}$). Assume she moves to 0. Then, the memory is updated to [010] = [100]

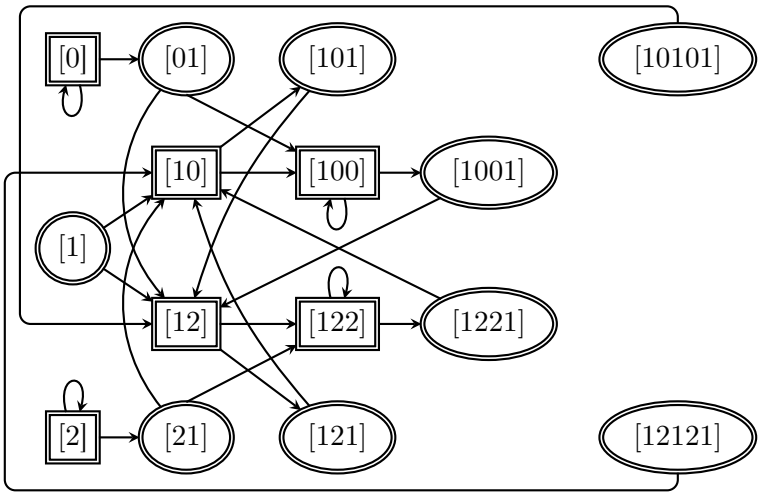

Figure 5.3: The winning region $W_0(\mathcal{G}_S)$ of the safety game $\mathcal{G}_S$ from Example 5.23

accordingly, and left unchanged as long as Player 1 uses the self-loop. If he moves back to 1, the memory is updated to [1001]. Now, Player 0 has only one choice: move to 2, induced by the fact that the only successor of [1001] in $W_0(\mathcal{G}_S)$ is [12]. The successor [10010], which corresponds to moving another time back to 0, is in the winning region of Player 1, since he can enforce to leave F from there. Hence, Player 0 moves to 2 in the Muller game and updates her memory to [10012] = [12]. By iterating this, she is able to prevent her opponent from ever reaching a score of three. Hence, this is a winning strategy from 0. In the same way, she can win from any vertex of $\mathcal{G}$, since [1] and [2] are in $W_0(\mathcal{G}_S)$ as well. ◇

But we can do better than using all equivalence classes in the winning region of Player 0 to keep track of the scores. In the previous example, in memory state [01], Player 0 has two choices, which both prevent her opponent from reaching a score of three. This freedom is the subject of the next section, but here we are interested in obtaining a small finite-state strategy. Hence, we eliminate such choices by restricting the winning region of Player 0 to the set of vertices that are reachable by some uniform winning strategy for $\mathcal{G}_S$.

Example 5.26. Once more, consider the safety game $\mathcal{G}_S$ from Example 5.23 as depicted in Figure 5.2 that corresponds to the Muller game $\mathcal{G}$ from Example 4.12. Figure 5.4 depicts the winning region of Player 0 restricted to the vertices reachable from $\{[0], [1], [2]\}$ via a uniform winning strategy. Using the construction presented in Example 5.25, Player 0 can use this restriction as memory structure to implement a winning strategy for the Muller game.

This saves two memory states. ◇

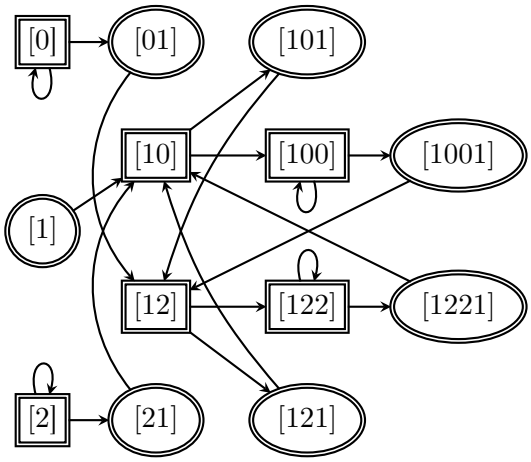

Figure 5.4: The winning region $W_0(\mathcal{G}_S)$ of the safety game $\mathcal{G}_S$ from Example 5.23 restricted to a winning strategy

But we can do even better than that: instead of keeping track of the exact scores, we over-approximate the scores, i.e., instead of considering all vertices in the restricted winning region, we only consider the maximal ones with respect to $\leq_{\mathcal{F}_1}$. Here, we compare equivalence classes by comparing their representatives, which is well-defined, since they are all $\mathcal{F}_1$-equivalent.

Example 5.27. Once more, consider the vertices in Figure 5.4. The $\mathcal{F}_1$-maximal ones are [100], [122], [1001], and [1221]. We show how to use these memory states to implement a winning strategy for Player 0 for the Muller game $\mathcal{G}$ from Example 4.12. Assume the token is placed at vertex 0. Then, the memory is initialized to [100], since we have $0 \leq_{\mathcal{F}_1} 100$. If Player 1 uses the self-loop twice, then the memory is left unchanged, since we have $00 \leq_{\mathcal{F}_1} 100$ and $000 \leq_{\mathcal{F}_1} 100$. If he then moves the token to vertex 1, the memory is updated to [1001], since we have $0001 \leq_{\mathcal{F}_1} 1001$ and it is Player 0's turn. Now, there is only one successor v available such that $0001v$ can be covered by some maximal element, namely $v = 2$. Hence, she moves the token to 2 and the memory is updated to [122] due to $00012 \leq_{\mathcal{F}_1} 122$. By iterating this, she is able to prevent Player 1 from ever reaching a score of three. Furthermore, by initializing the memory to 1001 (or 1221) and 122, she is able to win $\mathcal{G}$ from the vertices 1 and 2, respectively.

Hence, we have constructed a winning strategy of size four. However, it is easy to see that two states suffice to win this game. ◇

We prove the direction from right to left of Theorem 5.19(i) by showing that the idea explained in the previous example can be implemented in every

game: the maximal elements reachable by some winning strategy for Player 0 for the safety game can be turned into a memory structure that implements a uniform winning strategy for the Muller game.

Lemma 5.28. *For every $v \in V$: if $[v] \in W_0(\mathcal{G}_S)$, then $v \in W_0(\mathcal{G})$.*

Proof. Let σ' be a uniform positional winning strategy for Player 0 for $\mathcal{G}_S$ and let $R \subseteq V^S$ be the set of vertices that are reachable from $W_0(\mathcal{G}_S) \cap \{[v] \mid v \in V\}$ by plays consistent with σ'. Every $[w] \in R \cap V_0^S$ has exactly one successor in R (which is of the form $[wv]$ for some $v \in V$ with $(\mathrm{Lst}(w), v) \in E$) and dually, every successor of $[w] \in R \cap V_1^S$ (which are exactly the equivalence classes $[wv]$ with $(\mathrm{Lst}(w), v) \in E$) is in R.

Now, we lift $\leq_{\mathcal{F}_1}$ to equivalence classes by defining $[w] \leq_{\mathcal{F}_1} [w']$ if and only if $w \leq_{\mathcal{F}_1} w'$. Let $R_{\max}$ be the $\leq_{\mathcal{F}_1}$-maximal elements of R. Applying the facts about successors of vertices in R, we obtain the following remark.

Remark 5.29. Let $R_{\max}$ be defined as above.

i. For every $[w] \in R_{\max} \cap V_0^S$, there is a $v \in V$ with $(\mathrm{Lst}(w), v) \in E$ and there is a $[w'] \in R_{\max}$ such that $[wv] \leq_{\mathcal{F}_1} [w']$.

ii. For every $[w] \in R_{\max} \cap V_1^S$ and each of its successors $[wv]$, there is a $[w'] \in R_{\max}$ such that $[wv] \leq_{\mathcal{F}_1} [w']$.

Thus, instead of updating the memory from $[w]$ to $[wv]$ (and thereby keeping track of the exact scores) when processing a vertex v, we can directly update it to a maximal element that is $\mathcal{F}_1$-larger than $[wv]$ (and thereby over-approximate the exact scores).

Formally, we define $\mathcal{M} = (M, \mathrm{init}, \mathrm{upd})$ by $M = R_{\max} \cup \{\bot\}$ [23], where

$$\mathrm{init}(v) = \begin{cases} [w] & \text{if } [v] \in W_0(\mathcal{G}_S) \text{ and there exists some } [w] \in R_{\max} \\ & \text{such that } [v] \leq_{\mathcal{F}_1} [w], \\ \bot & \text{otherwise,} \end{cases}$$

and

$$\mathrm{upd}([w], v) = \begin{cases} [w'] & \text{if there is some } [w'] \in R_{\max} \text{ such that } [wv] \leq_{\mathcal{F}_1} [w'], \\ \bot & \text{otherwise.} \end{cases}$$

We claim that we have $[w] \leq_{\mathcal{F}_1} \mathrm{upd}^*(w)$ for every $w \in V^+$ with $\mathrm{upd}^*(w) \neq \bot$. It is true for a play prefix of length one, since we have $[v] \leq_{\mathcal{F}_1} \mathrm{init}(v) = \mathrm{upd}^*(v)$ for every v such that $\mathrm{init}(v) \neq \bot$. For the induction step, we have

[23] We use the memory state $\bot$ to simplify our proof. It is not reachable via plays that are consistent with the strategy implemented by $\mathcal{M}$ and can therefore eliminated and its incoming transitions can be redefined arbitrarily.

$[w] \leq \mathrm{upd}^*(w)$ by induction hypothesis and have to show $[wv] \leq \mathrm{upd}^*(wv)$ for every v such that $\mathrm{upd}^*(wv) \neq \bot$. Let $\mathrm{upd}^*(w) = [x]$. Then, we have

$$\mathrm{upd}^*(wv) = \mathrm{upd}(\mathrm{upd}^*(w), v) = [w']$$

for some w' such that $[xv] \leq_{\mathcal{F}_1} [w']$. Thus,

$$[wv] \leq_{\mathcal{F}_1} [xv] \leq_{\mathcal{F}_1} [w'] = \mathrm{upd}^*(wv) \ ,$$

where the first inequality is due to Lemma 5.21, which is easily lifted to $\mathcal{F}_1$ equivalence classes.

Thus, if $\mathrm{upd}^*(w) = [x]$, then $\mathrm{Lst}(w) = \mathrm{Lst}(x)$. Using Remark 5.29, we define a next-move function by

$$\mathrm{nxt}(v, [w]) = \begin{cases} v' & \text{if } \mathrm{Lst}(w) = v,\ (v, v') \in E, \text{ and} \\ & \text{there exists } [w'] \in R_{\max} \text{ such that } [wv'] \leq_{\mathcal{F}_1} [w'], \\ v'' & \text{otherwise, where } v'' \text{ is some vertex with } (v, v'') \in E, \end{cases}$$

and $\mathrm{nxt}(v, \bot) = v''$ for some v'' with $(v, v'') \in E$. The second case in the case distinction above is just to match the formal definition of a next-move function; it is never be invoked since $\mathrm{upd}^*(w)$ is either equal to $\bot$ or we have $\mathrm{Lst}(w) = \mathrm{Lst}(x)$, where $\mathrm{upd}^*(w) = [x]$.

It remains to show that the strategy σ implemented by $\mathcal{M}$ and nxt is a winning strategy for Player 0 from $W = \{v \in V \mid [v] \in W_0(\mathcal{G}_S)\}$. An inductive application of Remark 5.29 shows that every play w that starts in W and is consistent with σ satisfies $\mathrm{upd}^*(w) \neq \bot$: for the induction start, let $v \in W$, i.e., $[v] \in W_0(\mathcal{G}_S)$. Hence, there exists an $[w] \in R_{\max}$ such that $[v] \leq_{\mathcal{F}_1} [w]$ which shows that we have $\mathrm{upd}^*(v) = \mathrm{init}(v) \neq \bot$.

For the induction step, let wv be consistent with σ. By induction hypothesis, we can assume $\mathrm{upd}^*(w) \neq \bot$. Hence, let $\mathrm{upd}^*(w) = [x]$. We consider two cases: if $\mathrm{Lst}(w) \in V_0$, then we have $v = \sigma(w) = \mathrm{nxt}(\mathrm{Lst}(w), \mathrm{upd}^*(w))$. By definition of nxt, there exists some $[w'] \in R_{\max}$ such that $[xv] \leq_{\mathcal{F}_1} [w']$. This equivalence class witnesses $\mathrm{upd}([x], v) \neq \bot$. This finishes the first case, since we have $\mathrm{upd}^*(wv) = \mathrm{upd}([x], v)$. Now, we consider the case $\mathrm{Lst}(w) \in V_1$. Then, there is an edge between $[x]$ and $[xv]$ and we have $[xv] \in R$. Hence, there is also some equivalence class $[w'] \in R_{\max}$ such that $[xv] \leq_{\mathcal{F}_1} [w']$. The class $[w']$ witnesses that we have $\mathrm{upd}^*(wv) = \mathrm{upd}([x], v) \neq \bot$.

Let us conclude the proof: since the memory state $\bot$ is never reached by a play that starts in W and is consistent with σ, the scores of Player 1 are bounded by two, as we have shown above $[w] \leq_{\mathcal{F}_1} \mathrm{upd}^*(w) \in R_{\max} \subseteq F$ for every play prefix w of such a play, which implies $w \in \mathrm{Plays}_{<3}$. Hence, σ is indeed a winning strategy for Player 0 from W. □

Using Lemma 5.24 and the construction in the proof of Lemma 5.28 proves Theorem 5.19(ii).

Corollary 5.30. *Player 0 has a finite-state winning strategy from $W_0(\mathcal{G})$ whose memory states form an $\leq_{\mathcal{F}_1}$-antichain in $W_0(\mathcal{G}_S)$.*

Proof. We have $\{[v] \mid v \in W_0(\mathcal{G})\} \subseteq W_0(\mathcal{G}_S)$ due to Lemma 5.24. Hence, the construction in the proof of Lemma 5.28 yields a finite-state winning strategy for Player 0 from $W_0(\mathcal{G})$ whose memory states form an $\leq_{\mathcal{F}_1}$-antichain in $W_0(\mathcal{G}_S)$. □

To finish the proof of Theorem 5.19, we determine the size of $\mathcal{G}_S$ to prove the third statement. To this end, we use a variation of latest appearance records (LAR) [GH82, McN93]. Note that we do not need a hit position for our purposes. A word $\ell \in V^+$ is an LAR if every vertex $v \in V$ appears at most once in ℓ. Next, we map each $w \in V^+$ to a unique LAR, denoted by $\mathrm{LAR}(w)$, as follows: $\mathrm{LAR}(v) = v$ for every $v \in V$ and for $w \in V^+$ and $v \in V$ we define

$$\mathrm{LAR}(wv) = \begin{cases} \mathrm{LAR}(w)v & \text{if } v \notin \mathrm{Occ}(w), \\ p_1 p_2 v & \mathrm{LAR}(w) = p_1 v p_2. \end{cases}$$

A simple induction shows that $\mathrm{LAR}(w)$ is indeed an LAR, which also ensures that the decomposition of w in the second case of the inductive definition is unique. We continue by showing that $\mathrm{LAR}(w)$ determines all but $|\mathrm{LAR}(w)|$ many of w's scores and accumulators.

Lemma 5.31. *Let $w \in V^+$ and $\mathrm{LAR}(w) = v_k v_{k-1} \cdots v_1$.*

i. $w = x_k v_k x_{k-1} v_{k-1} \cdots x_2 v_2 x_1 v_1$ *for some $x_j \in V^*$ with $\mathrm{Occ}(x_j) \subseteq \{v_1, \ldots, v_j\}$ for every j.*

ii. $\mathrm{Sc}_F(w) > 0$ *if and only if $F = \{v_1, \ldots, v_j\}$ for some j.*

iii. If $\mathrm{Sc}_F(w) = 0$, then $\mathrm{Acc}_F(w) = \{v_1, \ldots, v_j\}$ for the maximal j such that $\{v_1, \ldots, v_j\} \subseteq F$ and $\mathrm{Acc}_F(w) = \emptyset$ if no such j exists.

iv. Let $\mathrm{Sc}_F(w) > 0$ and $F = \{v_1, \ldots, v_j\}$. Then, $\mathrm{Acc}_F(w) \in \{\emptyset\} \cup \{\{v_1, \ldots, v_{j'}\} \mid j' < j\}$.

Proof. i.) By induction over $|w|$. If $|w| = 1$, then the claim follows immediately from $w = \mathrm{LAR}(w)$. Now, let $|wv| > 1$. If $v \notin \mathrm{Occ}(w)$, then $\mathrm{LAR}(wv) = \mathrm{LAR}(w)v$ and the claim follows by induction hypothesis.

Now, consider the case $v \in \mathrm{Occ}(w)$. So, there is an index h such that $\mathrm{LAR}(w) = p_1 v p_2$ with $p_1 = v_k \cdots v_{h+1}$, $p_2 = v_{h-1} \cdots v_1$, and hence $v_h = v$. By induction hypothesis, there exists a decomposition

$$w = x_k v_k x_{k-1} v_{k-1} \cdots v_2 x_1 v_1$$

for some $x_j \in V^*$ such that $\mathrm{Occ}(x_j) \subseteq \{v_1, \ldots, v_j\}$ for every j. Furthermore, we have $\mathrm{LAR}(wv) = p_1 p_2 v = v'_k \cdots v'_1$ where $v'_1 = v_h$, $v'_j = v_{j-1}$ for every j in the range $1 < j \leq h$, and $v'_j = v_j$ for every j in the range $h < j \leq k$.

Now, define $x'_1 = \varepsilon$, $x'_j = x_{j-1}$ for every j in the range $1 < j < h$, $x'_h = x_h v_h x_{h-1}$, and $x'_j = x_j$ for every j in the range $h < j \leq k$. It is easy to verify, that the decomposition

$$wv = x'_k v'_k x'_{k-1} v'_{k-1} \cdots v'_2 x'_1 v'_1$$

has the desired properties.

ii.) We have $\mathrm{Sc}_F(w) > 0$ if and only if there exists a suffix x of w with $\mathrm{Occ}(x) = F$. Due to the decomposition characterization, having a suffix x with $\mathrm{Occ}(x) = F$ is equivalent to $F = \{v_1, \ldots, v_j\}$ for some i.

iii.) By Lemma 4.7 we have $\mathrm{Acc}_F(w) = \mathrm{Occ}(x)$ where x is the longest suffix of w such that the score for F does not change throughout x and $\mathrm{Occ}(x) \subseteq F$. Consider the decomposition characterization of w as above. We have $\{v_1, \ldots, v_i\} \subseteq \mathrm{Acc}_F(w)$, since $x_i v_i \cdots v_1 v_1$ is a suffix of w satisfying $\mathrm{Occ}(x) \subseteq F$. Furthermore, since $v_{i+1} \notin F$ by the maximality of i, this is the longest such suffix and we have indeed $\mathrm{Acc}_F(w) = \{v_1, \ldots, v_i\}$.

iv.) The latest increase of $\mathrm{Sc}_F(w)$ occurs after (or at) the last visit of v_i, since we have $\mathrm{Occ}(v_i x_{i-1} \cdots x_1 v_1) = F$. Hence, $\mathrm{Acc}_F(w)$ is the occurrence set of a suffix of $x_{i-1} \cdots x_1 v_1$ and the decomposition characterization yields the result. □

This characterization allows us to bound the size of $\mathcal{G}_S$ and to prove Theorem 5.19(iii).

Lemma 5.32. *We have* $|V^S| \leq \left(\sum_{k=1}^{n} \binom{n}{k} \cdot k! \cdot 2^k \cdot k!\right) + 1 \leq (n!)^3$, *where* $n = |V|$.

Proof. In every safety game, we can merge the vertices in $V \setminus F$ to a single vertex without changing $W_0(\mathcal{G})$. Since $[v] \in F$ for every vertex v of $\mathcal{G}$, we also retain the equivalence $v \in W_i(\mathcal{G}) \Leftrightarrow [v] \in W_i(\mathcal{G}_S)$.

Hence, it remains to bound the index of $\mathrm{Plays}_{<3}/_{=_{\mathcal{F}_1}}$. Lemma 5.31 shows that a play prefix $w \in V^+$ has $|\mathrm{LAR}(w)|$ many sets with non-zero score. Furthermore, the accumulator of the sets with score 0 is determined by $\mathrm{LAR}(w)$. Now, consider a play $w \in \mathrm{Plays}_{<3}$ and a set $F \in \mathcal{F}_1$ with non-zero score. We have $\mathrm{Sc}_F(w) \in \{1, 2\}$ and there are exactly $|F|$ possible values for $\mathrm{Acc}_F(w)$ due to Lemma 5.31(iv). Finally, $\mathrm{LAR}(w) = \mathrm{LAR}(w')$ implies $\mathrm{Lst}(w) = \mathrm{Lst}(w')$. Hence, the index of $\mathrm{Plays}_{<3}/_{=_{\mathcal{F}_1}}$ is bounded by the number of LARs, which is $\sum_{k=1}^{n} \binom{n}{k} \cdot k!$, times the number of possible score and accumulator combinations for each LAR ℓ of length k, which is bounded by $2^k \cdot k!$. □

If we only want to determine which player has a winning strategy from a given vertex v (and such a strategy in case it is Player 0), it suffices to construct the part of $\mathcal{G}_S$ that is reachable from $[v]$. This can even be improved by starting with an $\mathcal{F}_1$-burden wv ending in v: we have shown in Subsection 4.2.2 that Player 0 can prevent Player 1 from reaching a score of

three starting in v if and only if she can prevent him from reaching a score of three when starting the play with the burden wv.

Also, while a player in general cannot prevent her opponent from reaching a score of two, there are arenas in which she can do so. By first constructing the subgame $\mathcal{G}'_S$ up to threshold 2, which is smaller than the one for threshold three, we can possibly determine a subset of Player 0's winning region faster and obtain a (potentially) smaller finite-state winning strategy for this subset: we have $W_0(\mathcal{G}'_S) \subseteq W_0(\mathcal{G}_S)$. However, the converse $W_1(\mathcal{G}'_S) \subseteq W_1(\mathcal{G}_S)$ is in general false as we have seen in Theorem 4.16.

5.3.1 Permissive Strategies for Muller Games

In the previous section, we have introduced permissive strategies for parity games: according to this definition, a multi-strategy σ is permissive, if it subsumes the behavior of every positional multi-strategy σ' (restricted to those plays that start in a vertex from which σ' is winning). So, what is the right notion of permissiveness for Muller games, which are not positionally determined, i.e., which behaviors should a permissive strategy for a Muller game subsume? One possibility is to rely on finite-state determinacy: Muller games are determined with strategies of size $n!$, where n is the size of the game's arena. Already Bernet et al. mentioned that their results could be lifted to M-permissive strategies, i.e., strategies which subsume the behavior of every finite-state strategy of size at most M. One could define the same notion for Muller games and rely on the fact that a finite-state winning strategy of size M bounds the scores of the losing player by some bound which depends only on M and the size of the arena.

However, we prefer to pursue an alternative direct approach: our score-based construction (and also the original construction of Bernet et al.) for parity games yields permissive strategies that do not only subsume the behaviors of all positional strategies, but allow every play in which Player 0 can prevent Player 1 from reaching a score of $n_c + 1$ for some odd priority c. We think that this is the right way to generalize permissiveness to Muller games: a multi-strategy for a Muller game is permissive, if it is winning an if it allows every play ρ such that from every prefix of ρ, Player 0 is able to prevent her opponent from reaching a score of three. Such a strategy is winning from every vertex of her winning region. In the following, we construct such a strategy by turning the winning region of Player 0 in the safety game $\mathcal{G}_S$ into a memory structure. By construction, the winning region contains exactly the play prefixes from which Player 0 can prevent her opponent from reaching a score of three.

Obviously, we can replace the threshold three by any other value k and obtain a multi-strategy that keeps the scores below k. To this end, we just have to construct the safety game that keeps track of Player 1's scores up to k. However, when using a score that is strictly smaller than three, then such

a strategy cannot be winning from every vertex in the winning region, e.g., in an arena in which Player 0 cannot prevent her opponent from reaching a score of two (cf. Theorem 4.16).

We begin by giving a formal definition of permissiveness in Muller games. Recall that $\rho(w, \sigma, \tau)$ is the play obtained if the players use their strategies σ and τ to prolong the play prefix w on whose formation the players had no influence (see Definition 4.26). We say that Player 0 can prevent a score of three for Player 1 starting at w, if she has a strategy σ such that $\mathrm{MaxSc}_{\mathcal{F}_1}(\rho(w, \sigma, \tau)) \leq 2$ for every strategy τ of Player 1. Note that Player 0 cannot prevent a score of three for Player 1 starting from a play prefix w with $\mathrm{MaxSc}_{\mathcal{F}_1}(w) \geq 3$. Hence, if she is able to prevent him from reaching a score of three from every prefix of a play ρ, then we have $\mathrm{MaxSc}_{\mathcal{F}_1}(\rho) \leq 2$.

Remark 5.33. The permissive strategies constructed by Bernet et al. and us allow every play ρ such that Player 0 can prevent a score of $n_c + 1$ for every odd priority c starting at every prefix of ρ.

As we have explained in the introductory remarks, this is the generalization we pursue for Muller games.

Definition 5.34. A multi-strategy σ for Player 0 for a Muller game $\mathcal{G}$ is permissive, if it is winning from every vertex in $W_0(\mathcal{G})$ and if $\mathrm{Beh}(W_0(\mathcal{G}), \sigma)$ contains every play ρ that satisfies the following condition: starting at every prefix of ρ, Player 0 is able to prevent a score of three for Player 1.

Using the safety game $\mathcal{G}_S$ defined in the previous section, we are able to show that Player 0 always has a finite-state permissive strategy and how to compute one. The construction is analogous to the one of Theorem 5.18, but the correctness proof differs slightly due to our new definition of permissiveness.

Theorem 5.35. *Let $\mathcal{G}$ be a Muller game and $\mathcal{G}_S$ the corresponding safety game as above. Then, Player 0 has a finite-state permissive strategy for $\mathcal{G}$ with memory states $W_0(\mathcal{G}_S)$.*

The proof is very similar to the one for Theorem 5.18, hence, we only discuss the differences in greater detail.

Proof. The definition of the memory structure and the next-move function is exactly the same as in the proof of Theorem 5.18: we define $\mathcal{M} = (M, \mathrm{init}, \mathrm{upd})$ where $M = W_0(\mathcal{G}_S) \cup \{\bot\}$ [24],

$$\mathrm{init}(v) = \begin{cases} [v] & \text{if } [v] \in W_0(\mathcal{G}_S), \\ \bot & \text{otherwise,} \end{cases}$$

[24] Again, we use the memory state $\bot$ to simplify our proof. It is not reachable via plays that are consistent with the strategy implemented by $\mathcal{M}$ and can therefore eliminated and its incoming transitions can be redefined arbitrarily.

and

$$\mathrm{upd}([w], v) = \begin{cases} [wv] & \text{if } [wv] \in W_0(\mathcal{G}_S), \\ \bot & \text{otherwise.} \end{cases}$$

As we have done in the proof of Theorem 5.18 one can show that nxt always returns a non-empty set of successors. Similarly, one can show that the strategy σ implemented by $\mathcal{M}$ and nxt is winning from every vertex in the winning region of Player 0 in the Muller game.

It remains to show that σ is permissive. Let ρ be a play such that starting at every prefix of ρ, Player 0 is able to prevent a score of three for Player 1. It suffices to show that we have $[\rho_0 \cdots \rho_n] \in W_0(\mathcal{G}_S)$ for every prefix $\rho_0 \cdots \rho_n$, since every such play is consistent with σ by construction.

Towards a contradiction, assume that we have $[\rho_0 \cdots \rho_n] \notin W_0(\mathcal{G}_S)$, i.e., Player 1 has a strategy to move the token from $[\rho_0 \cdots \rho_n]$ into $V \setminus F$. This strategy can be translated into a strategy for him that enforces a score of three starting at $\rho_0 \cdots \rho_n$, which contradicts our assumption on ρ. Hence, the strategy σ is indeed permissive. □

5.4 Safety Reductions

The reductions presented in the previous two sections are very similar: a closed[25], ω-regular subset $L \subseteq \mathrm{Win}$ of the winning plays for Player 0 in a game $\mathcal{G}$ is identified such that Player 0 has a strategy that only allows plays which are in L when starting in her winning region. Such a strategy is necessarily winning. In the reductions above, the sets L are the plays with bounded scores for Player 1.

A closed, ω-regular language is the set of winning plays for Player 0 in a suitable safety game $\mathcal{G}_S$. In this situation, Player 0 can prove a vertex to be in her winning region by showing that a suitable vertex is in her winning region of $\mathcal{G}_S$. Furthermore, a uniform winning strategy for Player 0 for $\mathcal{G}$ can be constructed by turning $W_0(\mathcal{G})$ into a memory structure. Dually, if $\mathcal{G}$ is determined, then Player 1 can prove a vertex to be in his winning region by showing that a suitable vertex is in his winning region of the safety game. However, we do not obtain a winning strategy for Player 1.

In the following, we present a general notion of such a safety reduction which not only compasses the constructions for parity and Muller games presented above, but is also implicitly used in so-called Safraless synthesis algorithms. We show that this new reduction is also applicable to almost all types of winning conditions found in the literature. While this does not yield optimal strategies in terms of memory size (indeed, the safety reductions only

[25] A language is closed, if its complement is open (cf. Subsection 5.1).

yield finite-state strategies, even for positionally determined games), these reductions show that the winning regions and one winning strategy in all these types of games can be obtained by just solving safety games.

We begin by formalizing the new notion of reduction.

Definition 5.36. A game $\mathcal{G} = (\mathcal{A}, \mathrm{Win})$ with vertex set V is safety reducible, if there is a regular language $L \subseteq V^*$ of finite words such that:

- For every $\rho \in V^\omega$: if $\mathrm{Pref}(\rho) \subseteq L$, then $\rho \in \mathrm{Win}$.
- If $v \in W_0(\mathcal{G})$, then Player 0 has a strategy σ with $\mathrm{Pref}(\mathrm{Beh}(v, \sigma)) \subseteq L$.

A strategy σ satisfying $\mathrm{Pref}(\mathrm{Beh}(v, \sigma)) \subseteq L$ is winning for Player 0 from v. We continue by showing that many games appearing in the literature on infinite games are safety reducible.

Example 5.37. For a finite set X, let $X^{\leq k} = \{w \in X^* \mid |w| \leq k\}$.

- In a reachability game $\mathcal{G}$, Player 0 has an attractor winning strategy such that every consistent play visits F after at most $k = |V \setminus F|$ steps. Hence, $\mathcal{G}$ is safety reducible with $L = \mathrm{Pref}((V \setminus F)^{\leq k} \cdot F \cdot V^\omega)$.
- In a Büchi game $\mathcal{G}$, Player 0 has a positional winning strategy such that every consistent play visits a vertex in F at least every $k = |V \setminus F|$ steps. Hence, $\mathcal{G}$ is safety reducible with $L = \mathrm{Pref}(((V \setminus F)^{\leq k} \cdot F)^\omega)$.
- In a co-Büchi game $\mathcal{G}$, Player 0 has a positional winning strategy such that every consistent play stays in F after visiting each vertex in $V \setminus F$ at most once. Hence, $\mathcal{G}$ is safety reducible with

$$L = \mathrm{Pref}(\{w \cdot F^\omega \mid \text{ each } v \in V \setminus F \text{ appears at most once in } w\}) \ .$$

- In a request-response game $\mathcal{G}$ [WHT03], Player 0 has a finite-state winning strategy such that in every consistent play every request is answered within $k = |V| \cdot r \cdot 2^{r+1}$ steps [HTW08], where r is the number of request-response pairs. Thus, $\mathcal{G}$ is safety reducible to the language of play prefixes in which every request is answered within k steps. The same approach is applicable to Poset games [Zim09], an extension of request-response games with partially ordered responses.
- Corollary 5.9 shows that a parity game is safety reducible to the language of play prefixes that never allow a score of $n_c + 1$ for some odd priority c.
- The results of Piterman and Pnueli on small progress-measures for Rabin and Streett games [PP06] can also be rephrased in terms of safety reductions.

- Lemma 4.24 shows that a Muller game is safety reducible to the language of play prefixes in which the scores of Player 1 are bounded by two. ◇

Next, we define the cartesian product of an arena and an automaton reading play prefixes. This is essentially the same as the product of an arena and a memory structure. However, since we need to identify runs of the automaton in the product, we prefer to keep the automaton explicit. Let $\mathcal{A} = (V, V_0, V_1, E)$ be an arena and let $\mathfrak{A} = (Q, V, q_0, \delta, F)$ be a deterministic finite automaton recognizing a language over V. We define the arena

$$\mathcal{A} \times \mathfrak{A} = (V \times Q, V_0 \times Q, V_1 \times Q, E \circ \delta)$$

where $((v, q), (v', q')) \in E \circ \delta$ if and only if $(v, v') \in E$ and $\delta(q, v') = q'$. Every play prefix $v_1 \cdots v_n$ in $\mathcal{A}$ is be mapped to an extended play $e(v_1 \cdots v_n) = (v_1, q_1) \cdots (v_n, q_n)$ with $q_1 = \delta(q_0, v_1)$ and $q_{j+1} = \delta(q_j, v_{j+1})$, i.e., in its second component, the extended play in $\mathcal{A} \times \mathfrak{A}$ simulates the run of $\mathfrak{A}$ on the original play in $\mathcal{A}$. Dually, a play prefix $(v_1, q_1) \cdots (v_n, q_n)$ in $\mathcal{A} \times \mathcal{M}$ is mapped to its projected play by $p((v_1, q_1) \cdots (v_n, q_n)) = v_1 \cdots v_n$.

Theorem 5.38. *Let $\mathcal{G}$ be a game with vertex set V that is safety reducible with language $L(\mathfrak{A})$ for some DFA $\mathfrak{A} = (Q, V, q_0, \delta, F)$. Define the safety game $\mathcal{G}_S = (\mathcal{A} \times \mathfrak{A}, V \times F)$. Then:*

i. $v \in W_0(\mathcal{G})$ if and only if $(v, \delta(q_0, v)) \in W_0(\mathcal{G}_S)$.

ii. Player 0 has a finite-state winning strategy from $W_0(\mathcal{G})$ with memory states Q.

The proof is very similar to the proofs for the reductions from parity and Muller games to safety games presented above.

Proof. i.) Let $v \in W_0(\mathcal{G})$ and let σ be a winning strategy for Player 0 from v that satisfies $\mathrm{Pref}(\mathrm{Beh}(v, \sigma)) \subseteq L(\mathfrak{A})$. We define a strategy for $\mathcal{G}_S$ by $\sigma'((v_1, q_1) \cdots (v_n, q_n)) = (v', \delta(q_n, v'))$ where $v' = \sigma(v_1 \cdots v_n)$. We show that this strategy is winning for Player 0 from $(v, \delta(q_0, v))$. A simple induction shows that $(v_1, q_1) \cdots (v_n, q_n)$ being consistent with σ' implies $p((v_1, q_1) \cdots (v_n, q_n))$ being consistent with σ. So, suppose σ' is not winning in the safety game, i.e., there exists a play prefix w' in $\mathcal{G}_S$ starting in $(v, \delta(q_0, v))$ that is consistent with σ such that its last vertex (v_n, q_n) is in $V \times (Q \setminus F)$. Since the second component simulates the run of $\mathfrak{A}$ on $p(w')$, the projected play $p(w')$, which is consistent with σ, is not accepted by $\mathfrak{A}$. This yields the desired contradiction to our assumption that σ allows only play prefixes that are in $L(\mathfrak{A})$.

For the other direction, we construct a finite-state winning strategy with memory states Q that is winning for Player 0 from $W_0(\mathcal{G})$. Fix a uniform positional winning strategy σ' for Player 0 for $\mathcal{G}_S$ that is winning from $W_0(\mathcal{G}_S)$.

We define $\mathcal{M} = (Q, \text{init}, \delta)$ with $\text{init}(v) = \delta(q_0, v)$ and a next-move function by $\text{nxt}(v, q) = v'$, if $\sigma'(v, q) = (v', q')$ for some q'.

Let $(v, \delta(q_0, v)) \in W_0(\mathcal{G}_S)$. We show that the strategy σ implemented by $\mathcal{M}$ and nxt is winning for Player 0 from v. A simple induction shows that w starting in v and being consistent with σ implies $e(w)$ being consistent with σ'. Hence, we have $\text{Pref}(\text{Beh}(v, \sigma)) \subseteq L(\mathfrak{A})$, since the memory simulates the run of $\mathfrak{A}$ on w and does not leave F. Thus, σ is winning, as every play whose prefixes are all in $L(\mathfrak{A})$ is winning for Player 0.

ii.) We have $\{(v, \delta(q_0, v)) \mid v \in W_0(\mathcal{G})\} \subseteq W_0(\mathcal{G}_S)$ due to the first part of the proof for i.). Hence, the construction in the second part of the proof for i.) yields a finite-state winning strategy for Player 0 from $W_0(\mathcal{G})$ with memory states Q. □

Let us mention that a converse of Theorem 5.38 holds as well: if Player 0 has a uniform finite-state strategy σ implemented by a memory structure $\mathcal{M}$ for a game $\mathcal{G} = (\mathcal{A}, \text{Win})$, then $\mathcal{A} \times \mathcal{M}$ can be turned into a finite automaton $\mathfrak{A}$ with

$$L(\mathfrak{A}) = \text{Pref}(\text{Beh}(W_0(\mathcal{G}), \sigma)) \ .$$

Then, $\mathcal{G}$ is trivially safety reducible to $L(\mathfrak{A})$. Hence, every game in which Player 0 has a finite-state winning strategy (which can always be assumed to be uniform) can be reduced to a safety game. However, for the construction of the safety game as just described, we need a finite-state winning strategy and there is no need to determine another one by a safety reduction. The interesting case is when we have a generic construction of a language L that allows a safety reduction without having to solve the game first, as we have it in all examples mentioned above.

If $\mathcal{G}$ is determined, then Theorem 5.38(i) is equivalent to $v \in W_i(\mathcal{G})$ if and only if $(v, \delta(q_0, v)) \in W_i(\mathcal{G}_S)$. Hence, the winning regions (and a winning strategy for Player 0) for the games discussed in Example 5.37 can be determined by solving safety games. Let us conclude by mentioning that the safety reducibility of parity games was used implicitly to construct an algorithm for parity games [Jur00] and to compute permissive strategies for parity games [BJW02] as discussed in Section 5.2. Similarly, the safety reducibility of co-Büchi games is used implicitly in work on bounded synthesis [SF07] and LTL realizability [KV05, KPV06, FJR11].

Furthermore, there is a tight connection between permissive strategies, progress measure algorithms, and safety reductions for parity games: the progress measure algorithm due to Jurdziński [Jur00] and the reduction from parity games to safety game due to Bernet et al. [BJW02] to compute permissive strategies are essentially the same. Whether the safety reducibility of Muller games can be turned into a progress measure algorithm is subject to ongoing research.

What we can do is to generalize the notion of permissiveness to the games discussed in Example 5.23: if a game is safety reducible to L, then we can construct a multi-strategy that allows every play ρ in which Player 1 cannot leave L starting from any prefix of ρ. Thereby, we obtain what one could call L-permissive strategies.

5.5 Summary of Results

We presented a score-based variant of Bernet et al.'s reduction from parity to safety games to compute permissive strategies. While it is conceptually very close to the original proof, it has the advantage to be generalizable to Muller games as well: using the existence of strategies that bound the losing player's scores by two in a Muller game, we presented a reduction from Muller to safety games that yields both winning regions and a winning strategy for one player. The safety game here is only cubically larger than the parity game obtained by an LAR-reduction. This blowup is made up for by the fact that safety games can be solved in linear time while the question whether parity games can be solved in polynomial time is open at the time of writing. Also, our construction yields a novel type of antichain-based memory structure for Muller games and the first definition of permissive strategies for Muller games.

Furthermore, we have introduced a new type of reduction from arbitrary games to safety games which generalizes both reductions mentioned above, but can be (and has already been) applied to many other winning conditions appearing in the literature. As above, solving the safety game yields both winning regions (if the original game is determined, otherwise only one) and a winning strategy for one player.

Chapter 6

Conclusion

In this thesis, we have investigated the existence and the complexity of computing optimal strategies for infinite games.

We have shown that the complexity of solving games with winning conditions in linear temporal logics does not increase when adding operators parameterized by variable bounds, i.e., determining whether a player wins a PLTL game with respect to some, infinitely many, or all variable valuations is **2Exptime**-complete, as is solving LTL games. When we consider optimization problems for unipolar PLTL games, there is an exponential gap: we have presented an algorithm to compute optimal variable valuations in triply-exponential time, but the best lower bound is the doubly-exponential one of solving LTL games. Furthermore, we have complemented these results with doubly-exponential upper and lower bounds on the values of optimal variable valuations in PLTL games. Finally, we have shown that the set of variable valuations or Player i in a unipolar game is semilinear. The same holds true for the projection of this set in an arbitrary PLTL game to variables of one polarity.

The exact complexity of the optimization problems remains open and is subject of ongoing research. The lower bounds on values of optimal variable valuations show that our algorithms with triply-exponential running time cannot be improved to run in doubly-exponential time.

For Muller games, we have proved the existence of winning strategies which are optimal with respect to McNaughton's scoring functions: our strategies bound the losing player's scores by two (which is the best she can achieve), an improvement over the previous bound $|F|!$ for the score for a set F. We have used these strategies to show how to determine the winning regions of a Muller game and a winning strategy for one player by solving a safety game. This reduction also yields a score-based notion of permissive strategies and a new memory structure for Muller games. While the original definition of permissive strategies for parity games is not based on scores, we have shown that it can (and probably should) be formulated

using scoring functions for parity games. Finally, we have presented a general framework of so-called safety reductions, which allow to determine the winning regions and one winning strategy for an infinite game by solving a safety game. We have listed several implicit applications of these reductions in many earlier works found in the literature and have shown applications of safety reductions to other common winning conditions.

6.1 Further Research and Open Questions

The most interesting open problem about PLTL games is to settle the exact complexity of the unipolar optimization problems. Another challenging problem concerns the memory requirements of winning strategies realizing optimal variable valuations: these strategies are finite-state, but it is open whether being optimal requires more memory than just being winning: is there a tradeoff between the size and the quality of a winning strategy? Note that this question is very general and can be posed for many other winning conditions with an induced quality measure as well. We come back to this question below. Other questions worthwhile pursuing include the addition of (parameterized) past modalities and extensions to infinite arenas.

Finally, we propose to investigate the following variant of PLTL games: according to our definition, the emptiness problem for PLTL games asks whether there *exists* a strategy σ and a variable valuation α such that *every* play that is consistent with σ is a model of the winning condition with respect to α, i.e., the order of quantifiers is $\exists\sigma\exists\alpha\forall\rho$. If we change the order to $\exists\sigma\forall\rho\exists\alpha$, we ask whether there is a strategy such that the winning condition is satisfied on every consistent play, but with a variable valuation that may depend on the play. Thus, instead of guaranteeing uniform bounds for all plays consistent with a strategy, Player 0 only has to guarantee some bound on each play. This non-uniform variant of PLTL games is reminiscent of finitary objectives [CHH09].

For Muller games let us recall the two strengthenings of Lemma 4.24 discussed at the end of Subsection 4.2.2. Recall that the Lemma stated the existence of strategies that bound the losing player's scores by two, but does not yield results about the accumulator in situations where the score is two. Proving these strengthenings would show that the upper and lower bound of two a player can achieve against an optimal strategy is not only tight when we consider scores, but also when we consider scores and accumulators. Furthermore, this would also reduce the size of the safety game constructed in Section 5.3.

Another question regards the implementability of our strategies: we have shown them to be implemented by a memory structure whose states are LARs decorated by a small number of scores and accumulators. The size of this memory structure is bounded by $(n!)^3$, where n is the number of vertices

of the Muller game. Hence, this memory is only polynomially larger than the LAR-memory or the one induced by Zielonka trees. It would be interesting to investigate whether our strategies can be implemented with LARs or the memory induced by the Zielonka tree only. These questions can even be sharpened to the following conjecture:

Conjecture 6.1. *In every Muller game, Player i has a finite-state strategy of minimal size that bounds the losing player's scores by two (and by one, if this is possible at all).*

There is one strong argument for this conjecture to be true: allowing the losing player to reach a high score requires memory to keep track of the scores. If the memory is not updated while the score for some set F increases, then the strategy also allows an infinite play with infinity set F. Proving the conjecture would show that there is no tradeoff between the size and the quality of a strategy for a Muller game and could be a first step to proving similar results for PLTL games and others.

There are two examples of Muller games in this thesis (in Example 4.12 and in the proof of Theorem 4.16) in which the losing player is able to enforce a score of two against any strategy of the winning player. Intuition tells us that this happens when the winning player has to visit a sequence of vertices (1 and 0 in Example 4.12) to reach a certain vertex (vertex 0), and then has to visit each of these vertices again (as in the example: Player 1 can force her to visit 1 and 0 on the way out of the set $\{0, 1\}$). This leads to a score of two for the losing player. However, such a situation requires the arena (and the winning condition) to have a certain structure. It would be interesting to find a non-trivial characterization of this structure.

Also mentioned in Section 4.2.3 is the problem of extending our results on strategies that bound scores to infinite arenas, thereby showing that infinite games in infinite arenas can be played in finite time. We have highlighted some problems with the definition of scoring functions in infinite arenas. They show the need for domain-specific approaches that rely on intrinsic properties of arenas obtained by finite representations such as pushdown arenas. This is also subject to ongoing research.

Recall that we constructed a memory structure for Muller games using the maximal elements in the winning region in the associated safety game that are reachable via a winning strategy. Hence, the memory size is influenced by the choice of the strategy for the safety game, but the exact relation between these remains open: it is not even obvious that trying to minimize the number of reachable states yields the smallest number of maximal elements, since the have to be maximal with respect to the set of reachable states in the winning region, not with respect to all states of the winning region.

Finally, for parity games, there is a tight connection between permissive strategies and progress measure algorithms for solving them. In ongoing

research we investigate whether our notion of permissiveness yields a progress measure algorithm for Muller games. Here, the main obstacle to such an algorithm is the lack of a total ordering for equivalence classes of the equal-score relation $=_{\mathcal{F}_1}$ that is compatible with a suitable lifting operator.

Appendix A

Playing Muller Games in Finite Time via Parity Games

In Section 4.3, we claimed an $n \cdot n!$ lower bound on the length of a play in the finite-duration Muller game induced by the reduction to parity games. Here, we consider the reduction via latest appearance records (LAR). The number of LARs yields a trivial upper bound of $n \cdot n! + 1$ on the play length. In the following, we present a construction due to Chaturvedi [Cha11] which shows this bound to be tight.

We begin by fixing some necessary notation. An LAR over a finite set V is a word $\ell \in V^+$ in which every $v \in V$ appears exactly once. Thus, we have $|\ell| = |V|$. Furthermore, an extended LAR (ℓ, h) over V consists of an LAR ℓ over V and a hit-position h in the range $0 \leq h \leq |\ell| - 1$. There are exactly $|V| \cdot |V|!$ many extended LARs.

Given a vertex $v \in V$ and an extended LAR (ℓ, h) over V with $\ell = p_2 v p_1$, we define $\mathrm{upd}((\ell, h), v) = (p_2 p_1 v, |p_1|)$. A Muller game with vertex set V can be reduced to a parity game via the memory structure $\mathcal{M} = (M, \mathrm{init}, \mathrm{upd})$, where M is the set of extended LARs over V, init is any function mapping a vertex v to an extended LAR ending in v, and upd is defined as above.

Consider a Muller game $\mathcal{G} = (\mathcal{A}, \mathcal{F}_0, \mathcal{F}_1)$ with vertex set V and the parity game $\mathcal{G}' = (\mathcal{A} \times \mathcal{M}, \Omega)$ obtained in the reduction described above (the exact definition of Ω is irrelevant for our purposes, hence we do not specify it). We define a finite-duration game in $\mathcal{A}$ with the following rules: starting at a vertex v of $\mathcal{A}$, the players move the token through $\mathcal{A}$ building a play prefix $\rho_0 \rho_1 \cdots \rho_n$ until the extended play

$$(\rho_0, \mathrm{upd}^*(\rho_0))(\rho_1, \mathrm{upd}^*(\rho_0 \rho_1)) \cdots (\rho_n, \mathrm{upd}^*(\rho_0 \rho_1 \cdots \rho_n))$$

in $\mathcal{A} \times \mathcal{M}$ visits a vertex in $V \times M$ for the second time. If the maximal priority occurring in the cycle constructed in this way is even, then Player 0 wins the (finite) play $\rho_0 \rho_1 \cdots \rho_n$ of the Muller game; if it is odd, then Player 1 wins. Since $\mathcal{G}$ is determined with finite-state strategies implemented by $\mathcal{M}$,

Player i has a winning strategy for the Muller game from v if and only if she has a winning strategy for the finite-duration game from v.

Let $|V| = n$. We have $v = \mathrm{Lst}(\ell)$ for every vertex $(v, (\ell, h))$ in the parity game reachable by an extended play of $\mathcal{G}$. Thus, $\mathcal{G}'$ has at most $n \cdot n!$ reachable vertices and a play in the finite-duration game ends after at most $n \cdot n! + 1$ steps. In the following, we present a construction due to Chaturvedi that proves this bound to be tight.

To this end, we show that the set of extended LARs over a finite set V with $|V| = n$ can be enumerated as $e_0, e_1, \ldots, e_{n \cdot n! - 1}$ such that we have $\mathrm{init}(v_0) = e_0$ and $e_{j+1} = \mathrm{upd}(e_j, v_{j+1})$ for vertices v_j with $0 \leq j \leq n \cdot n! - 1$. Then, the sequence $v_0, v_1, \ldots, v_{n \cdot n! - 1}$ induces an extended play of length $n \cdot n!$ in $\mathcal{A} \times \mathcal{M}$ that visits each vertex exactly once. This shows that a play in the finite-duration game in a complete arena with n vertices can last $n \cdot n!$ steps. In the following, we construct a directed graph G and a bijective labeling of its edges by extended LARs satisfying the following property: if an incoming edge at a vertex is labeled by e and an outgoing edge at the same vertex by e', then we have $\mathrm{upd}(e, v) = e'$ for some $v \in V$. Thus, proving that G has a directed eulerian cycle yields the desired enumeration, since we can assume without loss of generality that the cycle starts in an extended LAR e such that $\mathrm{init}(v) = e$ for some vertex v.

The vertices of the graph G are the LARs over V and we add an edge from ℓ to ℓ', if we have $\mathrm{upd}((\ell, 0), v) = (\ell', h)$ for some v (note that the update-function is independent of the hit position, i.e., we have $\mathrm{upd}((\ell, 0), v) = \mathrm{upd}((\ell, j), v)$ for every j). If we label this edge by (ℓ', h), then the requirement on the edge labelings formulated above is satisfied. Furthermore, every vertex in G has in- and outdegree $|V|$ and the graph is connected, hence it contains a directed eulerian cycle. Finally, each extended LAR appears exactly once as edge label in G, which proves our claim.

Bibliography

[AETP99] Rajeev Alur, Kousha Etessami, Salvatore La Torre, and Doron Peled. Parametric temporal logic for "model measuring". In Jirí Wiedermann, Peter van Emde Boas, and Mogens Nielsen, editors, *ICALP*, volume 1644 of *LNCS*, pages 159–168. Springer, 1999. Conference version of [AETP01].

[AETP01] Rajeev Alur, Kousha Etessami, Salvatore La Torre, and Doron Peled. Parametric temporal logic for "model measuring". *ACM Trans. Comput. Log.*, 2(3):388–407, 2001. Journal version of [AETP99].

[AH90] Rajeev Alur and Thomas A. Henzinger. Real-time logics: Complexity and expressiveness. In *LICS*, pages 390–401. IEEE Computer Society, 1990. Conference version of [AH93].

[AH93] Rajeev Alur and Thomas A. Henzinger. Real-time logics: Complexity and expressiveness. *Inf. Comput.*, 104(1):35–77, 1993. Journal version of [AH90].

[AL11] Parosh Aziz Abdulla and K. Rustan M. Leino, editors. *Tools and Algorithms for the Construction and Analysis of Systems - 17th International Conference, TACAS 2011, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2011, Saarbrücken, Germany, March 26-April 3, 2011. Proceedings*, volume 6605 of *LNCS*. Springer, 2011.

[AT01] Rajeev Alur and Salvatore La Torre. Deterministic generators and games for LTL fragments. In *LICS*, pages 291–302, 2001. Conference version of [AT04].

[AT04] Rajeev Alur and Salvatore La Torre. Deterministic generators and games for LTL fragments. *ACM Trans. Comput. Log.*, 5(1):1–25, 2004. Journal version of [AT01].

[ATM03] Rajeev Alur, Salvatore La Torre, and P. Madhusudan. Playing games with boxes and diamonds. In Roberto M. Amadio and

Denis Lugiez, editors, *CONCUR*, volume 2761 of *LNCS*, pages 127–141. Springer, 2003.

[BBF+] Aaron Bohy, Véronique Bruyère, Emmanuel Filiot, Naiyong Jin, and Jean-François Raskin. Acacia+, a tool for LTL systhesis. To appear.

[BCHJ09] Roderick Bloem, Krishnendu Chatterjee, Thomas A. Henzinger, and Barbara Jobstmann. Better quality in synthesis through quantitative objectives. In Bouajjani and Maler [BM09], pages 140–156.

[BDH+] Dietmar Berwanger, Anuj Dawar, Paul Hunter, Stephan Kreutzer, and Jan Obdržálek. DAG-width and parity games. *J. Comb. Theory, Ser. B*. To appear. Journal version of [BDHK06, Obd06].

[BDHK06] Dietmar Berwanger, Anuj Dawar, Paul Hunter, and Stephan Kreutzer. DAG-width and parity games. In Bruno Durand and Wolfgang Thomas, editors, *STACS*, volume 3884 of *LNCS*, pages 524–536. Springer, 2006. A conference version of [BDH+].

[BG04] Dietmar Berwanger and Erich Grädel. Entanglement – a measure for the complexity of directed graphs with applications to logic and games. In Franz Baader and Andrei Voronkov, editors, *LPAR*, volume 3452 of *LNCS*, pages 209–223. Springer, 2004.

[BJW02] Julien Bernet, David Janin, and Igor Walukiewicz. Permissive strategies: from parity games to safety games. *ITA*, 36(3):261–275, 2002.

[BK08] Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. The MIT Press, 2008.

[BL69] J. Richard Büchi and Lawrence H. Landweber. Solving sequential conditions by finite-state strategies. *Transactions of the American Mathematical Society*, 138:pp. 295–311, 1969.

[BM09] Ahmed Bouajjani and Oded Maler, editors. *Computer Aided Verification, 21st International Conference, CAV 2009, Grenoble, France, June 26 - July 2, 2009. Proceedings*, volume 5643 of *LNCS*. Springer, 2009.

[CH+11] Pavol Černý, Krishnendu Chatterjee, Thomas A. Henzinger, Arjun Radhakrishna, and Rohit Singh. Quantitative synthesis for concurrent programs. In Ganesh Gopalakrishnan and Shaz Qadeer, editors, *CAV*, volume 6806 of *LNCS*, pages 243–259. Springer, 2011.

[CH06] Krishnendu Chatterjee and Thomas A. Henzinger. Finitary winning in omega-regular games. In Holger Hermanns and Jens Palsberg, editors, *TACAS*, volume 3920 of *LNCS*, pages 257–271. Springer, 2006. A conference version of [CHH09].

[Cha11] Namit Chaturvedi, 2011. Personal communication.

[CHH09] Krishnendu Chatterjee, Thomas A. Henzinger, and Florian Horn. Finitary winning in omega-regular games. *ACM Trans. Comput. Log.*, 11(1), 2009. Journal version of [CH06, Hor07].

[CHJS11] Krishnendu Chatterjee, Thomas A. Henzinger, Barbara Jobstmann, and Rohit Singh. Quasy: Quantitative synthesis tool. In Abdulla and Leino [AL11], pages 267–271.

[Chu57] Alonzo Church. Applications of recursive arithmetic to the problem of circuit synthesis. In *Summaries of the Summer Institute of Symbolic Logic*, volume 1, pages 3–50. Cornell University, 1957.

[Chu63] Alonzo Church. Logic, arithmetic, and automata. In *Proc. Int. Congr. Math. 1962*, pages 23–35. Inst. Mittag-Lefler, Djursholm, Sweden, 1963.

[CKLB11] Chih-Hong Cheng, Alois Knoll, Michael Luttenberger, and Christian Buckl. Gavs+: An open platform for the research of algorithmic game solving. In Abdulla and Leino [AL11], pages 258–261.

[CO00] Bruno Courcelle and Stephan Olariu. Upper bounds to the clique width of graphs. *Discrete Applied Mathematics*, 101(1-3):77–114, 2000.

[Dic13] Leonard E. Dickson. Finiteness of the odd perfect and primitive abundant numbers with n distinct prime factors. *Amer. Journal Math.*, 35(4):413–422, 1913.

[DJW97] Stefan Dziembowski, Marcin Jurdziński, and Igor Walukiewicz. How much memory is needed to win infinite games? In *LICS*, pages 99–110, 1997.

[Egg63] Lawrence C. Eggan. Transition graphs and the star-height of regular events. *Michigan Math. J.*, 10(4):385–397, 1963.

[Ehl11] Rüdiger Ehlers. Unbeast: Symbolic bounded synthesis. In Abdulla and Leino [AL11], pages 272–275.

[EJ91] E. Allen Emerson and Charanjit S. Jutla. Tree automata, mu-calculus and determinacy (extended abstract). In *FOCS*, pages 368–377. IEEE, 1991.

[EJS93] E. Allen Emerson, Charanjit S. Jutla, and A. Prasad Sistla. On model-checking for fragments of μ-calculus. In Costas Courcoubetis, editor, *CAV*, volume 697 of *LNCS*, pages 385–396. Springer, 1993. Conference version of [EJS01].

[EJS01] E. Allen Emerson, Charanjit S. Jutla, and A. Prasad Sistla. On model checking for the μ-calculus and its fragments. *Theor. Comput. Sci.*, 258(1-2):491–522, 2001. Journal version of [EJS93].

[EL85] E. Allen Emerson and Chin-Laung Lei. Modalities for model checking: Branching time strikes back. In *POPL*, pages 84–96, 1985.

[FJR09] Emmanuel Filiot, Naiyong Jin, and Jean-François Raskin. An antichain algorithm for LTL realizability. In Bouajjani and Maler [BM09], pages 263–277. A conference version of [FJR11].

[FJR10] Emmanuel Filiot, Naiyong Jin, and Jean-François Raskin. Compositional algorithms for LTL synthesis. In Ahmed Bouajjani and Wei-Ngan Chin, editors, *ATVA*, volume 6252 of *LNCS*, pages 112–127. Springer, 2010. A conference version of [FJR11].

[FJR11] Emmanuel Filiot, Naiyong Jin, and Jean-François Raskin. Antichains and compositional algorithms for LTL synthesis. *Formal Methods in System Design*, 39(3):261–296, 2011. Journal version of [FJR09, FJR10].

[FZ10] John Fearnley and Martin Zimmermann. Playing Muller games in a hurry. In Angelo Montanari, Margherita Napoli, and Mimmo Parente, editors, *GandALF*, volume 25 of *EPTCS*, pages 146–161, 2010. Conference version of [FZ12].

[FZ12] John Fearnley and Martin Zimmermann. Playing Muller games in a hurry. *Int. J. Found. Comput. Sci.*, 2012. To appear. Journal version of [FZ10].

[GH82] Yuri Gurevich and Leo Harrington. Trees, automata, and games. In STOC [STO82], pages 60–65.

[GS64] Seymour Ginsburg and Edwin H. Spanier. Bounded Algol-like languages. *Transactions of the American Mathematical Society*, 113(2):333–368, 1964.

[GTN10] Barbara Di Giampaolo, Salvatore La Torre, and Margherita Napoli. Parametric metric interval temporal logic. In Adrian Horia Dediu, Henning Fernau, and Carlos Martín-Vide, editors, *LATA*, volume 6031 of *LNCS*, pages 249–260. Springer, 2010.

[HD05] Paul Hunter and Anuj Dawar. Complexity bounds for regular games. In Joanna Jedrzejowicz and Andrzej Szepietowski, editors, *MFCS*, volume 3618 of *LNCS*, pages 495–506. Springer, 2005.

[HK07] Paul Hunter and Stephan Kreutzer. Digraph measures: Kelly decompositions, games, and orderings. In Nikhil Bansal, Kirk Pruhs, and Clifford Stein, editors, *SODA*, pages 637–644. SIAM, 2007. Conference version of [HK08].

[HK08] Paul Hunter and Stephan Kreutzer. Digraph measures: Kelly decompositions, games, and orderings. *Theor. Comput. Sci.*, 399(3):206–219, 2008. Journal version of [HK07].

[Hor07] Florian Horn. Faster algorithms for finitary games. In Orna Grumberg and Michael Huth, editors, *TACAS*, volume 4424 of *LNCS*, pages 472–484. Springer, 2007. A conference version of [CHH09].

[Hor08] Florian Horn. Explicit Muller games are PTIME. In Ramesh Hariharan, Madhavan Mukund, and V. Vinay, editors, *FSTTCS*, volume 2 of *LIPIcs*, pages 235–243. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2008.

[HTW08] Florian Horn, Wolfgang Thomas, and Nico Wallmeier. Optimal strategy synthesis in request-response games. In Sung Deok Cha, Jin-Young Choi, Moonzoo Kim, Insup Lee, and Mahesh Viswanathan, editors, *ATVA*, volume 5311 of *LNCS*, pages 361–373. Springer, 2008.

[JB06] Barbara Jobstmann and Roderick Bloem. Optimizations for LTL synthesis. In *FMCAD*, pages 117–124. IEEE Computer Society, 2006.

[JPZ06] Marcin Jurdziński, Mike Paterson, and Uri Zwick. A deterministic subexponential algorithm for solving parity games. In SODA [SOD06], pages 117–123. Conference version of [JPZ08].

[JPZ08] Marcin Jurdziński, Mike Paterson, and Uri Zwick. A deterministic subexponential algorithm for solving parity games. *SIAM J. Comput.*, 38(4):1519–1532, 2008. Journal version of [JPZ06].

[Jur98] Marcin Jurdziński. Deciding the winner in parity games is in $\mathbf{UP} \cap \mathbf{Co\text{-}UP}$. *Inf. Process. Lett.*, 68(3):119–124, 1998.

[Jur00] Marcin Jurdziński. Small progress measures for solving parity games. In Horst Reichel and Sophie Tison, editors, *STACS*, volume 1770 of *LNCS*, pages 290–301. Springer, 2000.

[Kam68] Hans W. Kamp. *Tense Logic and the Theory of Linear Order*. PhD thesis, Computer Science Department, University of California at Los Angeles, USA, 1968.

[Kec95] Alexander Kechris. *Classical Descriptive Set Theory*, volume 156 of *Graduate Texts in Mathematics*. Springer, 1995.

[Koy90] Ron Koymans. Specifying real-time properties with metric temporal logic. *Real-Time Systems*, 2:255–299, 1990.

[Koz83] Dexter Kozen. Results on the propositional mu-calculus. *Theor. Comput. Sci.*, 27:333–354, 1983.

[KPV06] Orna Kupferman, Nir Piterman, and Moshe Y. Vardi. Safraless compositional synthesis. In Thomas Ball and Robert B. Jones, editors, *CAV*, volume 4144 of *LNCS*, pages 31–44. Springer, 2006.

[KPV07] Orna Kupferman, Nir Piterman, and Moshe Y. Vardi. From liveness to promptness. In Werner Damm and Holger Hermanns, editors, *CAV*, volume 4590 of *LNCS*, pages 406–419. Springer, 2007. Conference version of [KPV09].

[KPV09] Orna Kupferman, Nir Piterman, and Moshe Y. Vardi. From liveness to promptness. *Formal Methods in System Design*, 34(2):83–103, 2009. Journal version of [KPV07].

[KR10] Orna Kupferman and Adin Rosenberg. The blowup in translating LTL to deterministic automata. In Ron van der Meyden and Jan-Georg Smaus, editors, *MoChArt*, volume 6572 of *LNCS*, pages 85–94. Springer, 2010.

[KV05] Orna Kupferman and Moshe Y. Vardi. Safraless decision procedures. In *FOCS*, pages 531–542. IEEE Computer Society, 2005.

[LIC06] *21th IEEE Symposium on Logic in Computer Science (LICS 2006), 12-15 August 2006, Seattle, WA, USA, Proceedings*. IEEE Computer Society, 2006.

[McN93] Robert McNaughton. Infinite games played on finite graphs. *Ann. Pure Appl. Logic*, 65(2):149–184, 1993.

[McN00] Robert McNaughton. Playing infinite games in finite time. In Arto Salomaa, Derick Wood, and Sheng Yu, editors, *A Half-Century of Automata Theory*, pages 73–91. World Scientific, 2000.

[Mor10] Andreas Morgenstern. *Symbolic Controller Synthesis for LTL Specifications*. PhD thesis, Department of Computer Science, University of Kaiserslautern, Kaiserslautern, Germany, 2010.

[Mos91] Andrzej Mostowski. Games with forbidden positions. Technical Report 78, University of Gdańsk, 1991.

[MS95] David E. Muller and Paul E. Schupp. Simulating alternating tree automata by nondeterministic automata: New results and new proofs of the theorems of Rabin, McNaughton and Safra. *Theor. Comput. Sci.*, 141(1&2):69–107, 1995.

[MS08] Andreas Morgenstern and Klaus Schneider. From LTL to symbolically represented deterministic automata. In Francesco Logozzo, Doron Peled, and Lenore D. Zuck, editors, *VMCAI*, volume 4905 of *LNCS*, pages 279–293. Springer, 2008.

[NRY96] Anil Nerode, Jeffrey B. Remmel, and Alexander Yakhnis. McNaughton games and extracting strategies for concurrent programs. *Ann. Pure Appl. Logic*, 78(1-3):203–242, 1996.

[NRZ11] Daniel Neider, Roman Rabinovich, and Martin Zimmermann. Solving Muller games via safety games. Technical Report AIB-2011-14, RWTH Aachen University, July 2011.

[Obd06] Jan Obdržálek. DAG-width: connectivity measure for directed graphs. In SODA [SOD06], pages 814–821. A conference version of [BDH$^+$].

[Pap94] Christos H. Papadimitriou. *Computational complexity.* Addison-Wesley, 1994.

[Pit06] Nir Piterman. From nondeterministic Büchi and Streett automata to deterministic parity automata. In LICS [LIC06], pages 255–264. Conference version of [Pit07].

[Pit07] Nir Piterman. From nondeterministic Büchi and Streett automata to deterministic parity automata. *Logical Methods in Computer Science*, 3(3), 2007. Journal version of [Pit06].

[PP06] Nir Piterman and Amir Pnueli. Faster solutions of Rabin and Streett games. In LICS [LIC06], pages 275–284.

[PR89a] Amir Pnueli and Roni Rosner. On the synthesis of a reactive module. In *POPL*, pages 179–190, 1989.

[PR89b] Amir Pnueli and Roni Rosner. On the synthesis of an asynchronous reactive module. In Giorgio Ausiello, Mariangiola Dezani-Ciancaglini, and Simona Ronchi Della Rocca, editors, *ICALP*, volume 372 of *LNCS*, pages 652–671. Springer, 1989.

[Pre29] Mojżesz Presburger. Über die Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zahlen, in welchem die Addition als einzige Operation hervortritt. In *Comptes Rendus du I congrés de Mathématiciens des Pays Slaves. Warsaw*, pages 92–101, 1929.

[Ros91] Roni Rosner. *Modular Synthesis of Reactive Systems*. PhD thesis, Weizmann Institute of Science, February 1991.

[RS83] Neil Robertson and Paul D. Seymour. Graph minors. I. Excluding a forest. *J. Comb. Theory, Ser. B*, 35(1):39–61, 1983.

[RS86] Neil Robertson and Paul D. Seymour. Graph minors. II. Algorithmic aspects of tree-width. *J. Algorithms*, 7(3):309–322, 1986.

[Saf88] Shmuel Safra. On the complexity of omega-automata. In *FOCS*, pages 319–327. IEEE Computer Society, 1988.

[SC82] A. Prasad Sistla and Edmund M. Clarke. The complexity of propositional linear temporal logics. In STOC [STO82], pages 159–168. Conference version of [SC85].

[SC85] A. Prasad Sistla and Edmund M. Clarke. The complexity of propositional linear temporal logics. *J. ACM*, 32(3):733–749, 1985. Journal version of [SC82].

[Sch07] Sven Schewe. Solving parity games in big steps. In Vikraman Arvind and Sanjiva Prasad, editors, *FSTTCS*, volume 4855 of *LNCS*, pages 449–460. Springer, 2007.

[SF07] Sven Schewe and Bernd Finkbeiner. Bounded synthesis. In Kedar S. Namjoshi, Tomohiro Yoneda, Teruo Higashino, and Yoshio Okamura, editors, *ATVA*, volume 4762 of *LNCS*, pages 474–488. Springer, 2007.

[SOD06] *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2006, Miami, Florida, USA, January 22-26, 2006*. ACM Press, 2006.

[SS63] John C. Shepherdson and Howard E. Sturgis. Computability of recursive functions. *J. ACM*, 10(2):217–255, 1963.

[STO82] *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing, 5-7 May 1982, San Francisco, California, USA*. ACM, 1982.

[VW94] Moshe Y. Vardi and Pierre Wolper. Reasoning about infinite computations. *Inf. Comput.*, 115(1):1–37, 1994. Journal version of [WVS83].

[Wad72] William W. Wadge. Degrees of complexity of subsets of the Baire space. *Notices A.M.S.*, pages A714–A715, 1972.

[Wad83] William W. Wadge. *Reducibility and determinateness on the Baire space*. PhD thesis, University of California, Berkeley, 1983.

[Wal96] Igor Walukiewicz. Pushdown processes: Games and model checking. In Rajeev Alur and Thomas A. Henzinger, editors, *CAV*, volume 1102 of *LNCS*, pages 62–74. Springer, 1996. Conference version of [Wal01].

[Wal01] Igor Walukiewicz. Pushdown processes: Games and model-checking. *Inf. Comput.*, 164(2):234–263, 2001. Journal version of [Wal96].

[WHT03] Nico Wallmeier, Patrick Hütten, and Wolfgang Thomas. Symbolic synthesis of finite-state controllers for request-response specifications. In Oscar H. Ibarra and Zhe Dang, editors, *CIAA*, volume 2759 of *LNCS*, pages 11–22. Springer, 2003.

[WVS83] Pierre Wolper, Moshe Y. Vardi, and A. Prasad Sistla. Reasoning about infinite computation paths (extended abstract). In *FOCS*, pages 185–194. IEEE Computer Society, 1983. Conference version of [VW94].

[Zer13] Ernst Zermelo. Über eine Anwendung der Mengenlehre auf die Theorie des Schachspiels. In *Proceedings of the Fifth Congress of Mathematicians, Vol. 2*, pages 501–504. Cambridge Press, 1913.

[Zie98] Wiesław Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theor. Comput. Sci.*, 200(1-2):135–183, 1998.

[Zim09] Martin Zimmermann. Time-optimal winning strategies for poset games. In Sebastian Maneth, editor, *CIAA*, volume 5642 of *LNCS*, pages 217–226. Springer, 2009.

[Zim11] Martin Zimmermann. Optimal bounds in parametric LTL games. In Giovanna D'Agostino and Salvatore La Torre, editors, *GandALF*, volume 54 of *EPTCS*, pages 146–161, 2011.

Index

Symbol Index

$\overline{\mathcal{A}}$	dual arena for $\mathcal{A}$, 40
$\mathcal{A}_b$	arena for blinking semantics, 50
$\mathcal{A}[X]$	subarena induced by X, 13
$\mathcal{A} \times \mathfrak{A}$	cartesian product of $\mathcal{A}$ and $\mathfrak{A}$, 156
$\mathcal{A} \times \mathcal{M}$	product arena, 19
A_j	set computed by Zielonka's algorithm, 100
Acc_F	accumulator function, 86
Acpt	set of accepting runs, 11
alt_p	the formula $\mathbf{GF}p \wedge \mathbf{GF}\neg p$, 46
$\mathrm{Attr}_i^X(F)$	attractor, 21
β	variable valuation, 29
$\mathrm{Beh}(v, \sigma)$	set of plays starting in v consistent with σ, 17
$\mathrm{Beh}(W, \sigma)$	set of plays starting in W consistent with σ, 17
$\mathrm{BrnchFctr}(T)$	number of children of T's root, 98
$\mathrm{Chld}(T, j)$	j-th child of T, 98
$\mathrm{cl}(\varphi)$	set of subformulae of φ, 28
$\mathrm{cl}_p(\varphi)$	set of parameterized subformulae of φ, 66
Co-NP	complement class of **NP**, 10
Co-UP	complement class of **UP**, 10
ε	the empty word, 9
$\mathbf{F}$	eventually operator, 27
$\mathbf{F}_{\leq x}$	parameterized eventually operator, 27
$\mathcal{F} \restriction X$	restriction of $\mathcal{F}$ to subsets of X, 98
$(\mathcal{F}_0, \mathcal{F}_1)$	winning condition in a Muller game, 15
$(\mathcal{F}_0, \mathcal{F}_1) \restriction X$	restriction of $(\mathcal{F}_0, \mathcal{F}_1)$ to subsets of X, 98
$\mathcal{G}$	infinite game, 13
$\overline{\mathcal{G}}$	dual game for $\mathcal{G}$, 40

$\mathcal{G} \leq_{\mathcal{M}} \mathcal{G}'$	game reduction, 19
$\mathbf{G}$	always operator, 27
$\mathbf{G}_{\leq y}$	parameterized always operator, 27
$\mathrm{Inf}(w)$	infinity set of w, 10
init	initialization function, 18
ℓ	labeling function, 37
$L(\mathfrak{A})$	language recognized by $\mathfrak{A}$, 11
$L \leq L'$	Wadge reduction, 126
$L \cdot L'$	concatenation of L and L', 9
$\mathrm{Lst}(w)$	last letter of w, 9
$\mathcal{M}$	memory structure, 18
M	set of memory states, 18
MaxSc_c	maximum score function, 129
$\mathrm{MaxSc}_{\mathcal{F}}$	maximum score function, 86
$\mathbb{N}$	the set of non-negative integers, 9
n_c	number of vertices labeled by c, 128
NP	non-deterministic polynomial time, 10
nxt	next-move function, 18
Ω	priority function in a parity automaton, 12
Ω	priority function in a parity game, 14
$\mathrm{Occ}(w)$	occurrence set of w, 9
P	set of atomic propositions, 27
$\Pi_i^{\mathcal{A}}$	set of strategies for Player i in $\mathcal{A}$, 16
$\mathbf{\Pi}_n$	class of the Borel hierarchy, 126
φ_α	LTL formula equivalent to φ w.r.t. α, 35
φ_X	relativization of φ, 46
$\mathfrak{P}_{\varphi,\alpha}$	det. parity automaton recognizing models of φ w.r.t. α, 62

$\mathrm{Plays}_{<n_c+1}$	plays in which Player 1's scores are bounded by n_c, 131
$\mathrm{Plays}_{=n_c+1}$	plays in which Player 1's scores just reached value n_c+1, 131
$\mathrm{Plays}_{\leq n_c+1}$	plays in which Player 1's scores are bounded by n_c or just reached value n_c+1, 132
$\mathrm{Plays}_{<3}$	plays in which Player 1's scores are bounded by two, 143
$\mathrm{Plays}_{=3}$	plays in which Player 1's scores just reached value three, 143
$\mathrm{Plays}_{\leq 3}$	plays in which Player 1's scores are bounded by two or just reached value three, 143
$\mathrm{Pref}(w)$	the set of prefixes of w, 9
$\mathrm{Par}(n)$	parity of n, 9
PSPACE	polynomial space, 10
ρ	play, 13
$\rho(v, \sigma, \tau)$	play starting in vertex v consistent with σ and τ, 16
$\rho(w, \sigma, \tau)$	play starting with prefix w consistent with σ and τ, 106
$\mathbf{R}$	release operator, 27
$\mathrm{RtLbl}(T)$	root label of T, 98
$\mathrm{RtPlr}(T)$	root player of T, 98
σ	strategy (typically for Player 0 or Player i), 16
Σ	alphabet, 9
Σ^*	the set of finite words over Σ, 9
Σ^+	the set of non-empty finite words over Σ, 9
Σ^ω	the set of ω-words over Σ, 9
$\mathbf{\Sigma}_n$	class of the Borel hierarchy, 126
Sc_c	scoring function, 129
Sc_F	scoring function, 86
τ	strategy (typically for Player 1 or Player $1-i$), 16
T_j	tree computed by Zielonka's algorithm, 100
Tr	threshold score function, 92

Curriculum Vitae

Persönliche Daten

Name	Zimmermann
Vorname	Martin
Geburtstag	21. September 1982
Geburtsort	Köln
Staatsangehörigkeit	deutsch

Qualifikationen

2002	Abitur
09/2003 – 01/2009	Studium der Informatik mit Nebenfach Betriebswirtschaftslehre an der RWTH Aachen University
09/2007 – 06/2008	Fulbright-Stipendiat an der DePaul University, Chicago, USA
01/2009	Diplom in Informatik
02/2009 – 01/2012	Promotionsstudium in Informatik an der RWTH Aachen