Languages and Strategies: A Study of Regular Infinite Games

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

von

Diplom-Informatiker
Jörg Olschewski
aus Aachen.

Berichter: Universitätsprofessor Dr. Dr. h.c. Wolfgang Thomas
Chargé de Recherche Dr. habil. Nicolas Markey

Tag der mündlichen Prüfung: 25. Januar 2013

Zusammenfassung

Unendliche Spiele mit zwei Spielern stellen ein Rahmenwerk für das Studium des Problems der Controllersynthese in reaktiven Systemen dar. Das fundamentale Büchi-Landweber Theorem war eine erste Lösung dieses Problems für reguläre Gewinnbedingungen. Wir erweitern dieses Resultat in der vorliegenden Arbeit, indem wir verschiedene Möglichkeiten zur Bewertung die Komplexität von Strategien in unendlichen Spielen untersuchen.

Im ersten Teil verbessern wir die 50 Jahre alte Ramsey-basierte Komplementierung von Büchi-Automaten, sowohl auf der praktischen als auch auf der theoretischen Seite. Wir stellen Heuristiken vor, die den Ramsey-basierten Ansatz optimieren. Diese Heuristiken wurden auch in einem Java-Programm von uns implementiert. Durch Experimente verifizieren wir, dass dieser neue verbesserte Ramsey-basierte Algorithmus tatsächlich mit den modernen Komplementierungsmethoden konkurrieren kann. Wir stellen außerdem eine neue theoretische Komplementierungsmethode vor, die auf totalen Quasiordnungen (weak-orders) basiert. Die weak-order-basierte Methode hat eine $2^{\mathcal{O}(n \log n)}$ obere Schranke für die Größe des Komplements und ist eng mit dem verbesserten Ramsey-basierten Ansatz verknüpft.

Im zweiten Teil betten wir das Konzept der Spiele in den Bereich der formalen Sprachen ein und erhalten ein qualitatives Maß für die Komplexität von Gewinnstrategien und -bedingungen. Wir verfeinern das Büchi-Landweber Theorem auf Teilklassen der regulären Sprachen, insbesondere auf Hierarchien unterhalb der sternfreien Sprachen. Wir unterscheiden zwischen starken Spielen, die auf unendlichen Vorkommen von Mustern in einem Wort basieren, und schwachen Spielen, die nur auf endlichen Mustern beruhen. Wir zeigen dass zum Lösen schwacher Spiele Gewinnstrategien benötigt werden, die ein Level über dem Level der Gewinnbedingung liegen. Für starke Spiele auf Level i reichen Gewinnstrategien auf Level $i + 2$ aus.

Im dritten Teil stellen wir ein weiteres Maß für die Komplexität von Strategien vor, dieses Mal über Paritätsspielen auf Graphen. Die *Permissivität* ist ein quantitatives Maß für nichtdeterministische Strategien. Sie wird bestimmt, indem einer Strategie eine Strafe zugewiesen wird, nämlich der Mittelwert vom Gewicht der geblockten Kanten (im Limes). Wir reduzieren das Problem, den Wert eines solchen Mean-Penalty-Paritätspiels zu bestimmen, auf das Wert-Problem in einem entsprechenden Mean-Payoff-Paritätsspiel und wir zeigen, dass beide Probleme in $\mathrm{NP} \cap \mathrm{coNP}$ liegen. Wir entwickeln einen deterministischen Algorithmus der die Werte berechnet und der schneller ist als alle bekannten Algorithmen. Ein ähnlicher Algorithmus für Mean-Penalty-Paritätspiele wird ebenfalls vorgestellt.

Abstract

The theory of two-player infinite games provides a framework for studying the controller synthesis problem in reactive system. This problem was solved for regular winning conditions for the first time by the fundamental Büchi-Landweber Theorem. The present work extends this result by investigating possibilities to measure the complexity of strategies in infinite games.

In the first part of this work, we improve a 50 years old algorithm, the Ramsey-based Büchi automata complementation method, both on the practical as well as on the theoretical side. On the practical side, we present some heuristics for improving the Ramsey-based approach, which we also implemented in a Java program. We show that this algorithm in fact can compete with the more modern ones. On the theoretical side, we introduce a novel complementation construction, based on weak-orders, to which the improved Ramsey-based approach is tightly connected, and we prove a $2^{\mathcal{O}(n \log n)}$ upper bound on the size of the complement automaton.

In the second part, we embed the concept of games into the domain of formal languages. By doing this, we are able to give a qualitative measure of the complexity of a winning strategy, as well as of the complexity of the corresponding winning condition. In this way, we extend and refine the fundamental Büchi-Landweber Theorem to subclasses of the class of regular languages, in particular we consider hierarchies below the starfree languages. We distinguish between weak games and strong games. Strong games rely on infinite occurrence of patterns in a word, while weak games only rely on finite occurrences of patterns. We show that for solving weak games, winning strategies lie one level above winning conditions inside the hierarchy. For strong games on level i, we show that winning strategies on level $i + 2$ suffice.

In the third part, we introduce another measure for the complexity of strategies, but this time on graph-games with parity conditions. This measure is of quantitative nature and it determines the *permissiveness* of a given nondeterministic strategy. The permissiveness is measured by assigning to each strategy a penalty, namely the average sum (in the limit) of the weight of edges that are to be disallowed. We reduce the problem of determining the value of such a mean-penalty parity game to the value problem of a corresponding mean-payoff parity game, and we show that both problems are in NP$\cap$coNP. We revisit the study of mean-payoff parity games and obtain a deterministic algorithm, which computes the value and which runs faster than all previously known algorithms. A similar algorithm for mean-penalty parity games is developed as well.

Contents

Chapter 1

Introduction

In today's computer science, the design and analysis of reactive systems that have a nonterminating behavior [57] is a central issue of growing importance. Control systems, database systems, operating systems, integrated circuits all share these properties of being reactive and designed not to terminate. The theoretical challenges of these application domains are best met by game-theoretic models and terminology. The most basic case, which has led to a beautiful theory, is concerned with two-player games of complete information and with "regular winning conditions" [75, 33]. In this setting of two antagonistic players, often called the environment and the controller, the aim of the controller is to fulfill a certain specification (or winning condition) while the aim of the environment is to prevent this. In this thesis, we address three topics of this field.

Let us sketch the fundamental notions and landmark results of the background theory. Church's Problem [22] is the synthesis problem for regular winning conditions. It was stated by Church in 1957, however, not in a game-theoretic context. Church considers a requirement, given in some suitable logic, and he asks for finding a representation of "a circuit that satisfies the given requirement". Today, by "circuit" we understand the common notion of a finite automaton, so Church's Problem, rephrased for today, is whether for a given specification there exists a finite automaton with output which satisfies the specification, and how we can construct it. This problem was first put into a game-theoretic framework by McNaughton [46] in 1965. Later, Büchi and Landweber [14] solved the problem for finite-state games with regular winning conditions. They discovered that these games are determined and that the winning player has a winning strategy executable by a finite automaton. This fundamental result is called the Büchi-Landweber Theorem. Rabin [58] obtained the same result by a simpler approach, using tree automata.

The foundations of this algorithmic game theory are provided by the theory of regular ω-languages, which serve as winning conditions in infinite

games. Their study began with the seminal work of Büchi [12], in which he introduced a mathematical model for ω-regular languages. This model is what is nowadays called the model of "Büchi automata". The key result in Büchi's paper is that Büchi automata are closed under complementation. He used this result to show that Büchi automata recognize exactly the class of languages that are definable by monadic second-order logic formulas. The analogous bridge between languages of finite words and logics had been established by Büchi and Elgot [13] and Trakhtenbrot [76]. The model of Büchi automata is a nondeterministic one, and the class of languages recognized by deterministic Büchi automata is strictly less expressive than the class recognized by nondeterministic ones. However, by the fundamental determinization result due to McNaughton [47] Büchi automata are determinizable, and the result is a deterministic Muller automaton. Another determinization procedure was given by Safra [61]; it constructs a deterministic Rabin automaton with an optimal upper bound on the size of the resulting automaton.

The first two parts of this thesis are concerned with these foundations. In the first part, we address the fundamental problem which was at the heart of Büchi's paper: the complementation of Büchi automata. To prove his result, Büchi used a combinatorial result by Ramsey [60] to decompose the possible behaviors of a Büchi automaton. For this reason, his approach is called the *Ramsey-based approach*. Later, others developed different, more efficient complementation approaches, called the *determinization-based approach* due to Safra [61], the *rank-based approach* by Klarlund [42], and the *slice-based approach* by Kähler and Wilke [40]. These possess a $2^{\mathcal{O}(n \log n)}$ upper bound whereas the original Ramsey-based approach only has a $2^{\mathcal{O}(n^2)}$ upper bound [67] on the size of the complement automaton. The theory around these approaches has improved over time due to numerous research efforts, and the gap between upper bounds [31, 56, 62], and lower bounds [48, 81] for the size of the resulting automata and computation time is thereby closing.

The original Ramsey-based approach remained unattended while for the other approaches much effort was invested for their improvement. In experimental studies, conducted by Tsai, Fogarty, Vardi, and Tsay [77], the Ramsey-based was exposed as being inferior to the other approaches. The question that we want to raise is whether one can develop the Ramsey-based complementation approach as well, and make it able to compete with the others. Our main result is that indeed this is possible, in fact both on a practical and on a theoretical level. On the practical side, we improve the original Ramsey-based approach by three main ideas. We implement these heuristics in a Java program, which is publicly available [50], and we

show that our implementation can compete in experiments with the other three approaches studied in [77]. On the theoretical level, we develop a novel, *weak-order-based* approach which is tightly connected to our improved heuristics, and we establish a $2^{\mathcal{O}(n \log n)}$ upper bound for this approach.

The second part of the thesis addresses a basic question raised in the paper by Büchi and Landweber [14], concerning the solution of two-player games, namely "how simple winning strategies do exist" for games in a given class of games. There are many facets for the simplicity or complexity of strategies, for example the memory size needed for storing a strategy, or the timings for reactions to requests made by one of the players. Here we focus on "logical simplicity". The basic idea is that strategies *essentially are* languages, as one can describe a strategy by a tuple of languages, one language for each possible decision a strategy can make. There is a strong background for this in uniformization problems, where one seeks a simple deterministic function that *covers* a given relation. In the same spirit, the question of Büchi and Landweber is concerned with a deterministic function that *solves* a given game. Their solution shows that monadic second-order definable games have strategies definable again in monadic second-order logic. This intuitively beautiful theorem allures us to find similar results by replacing "monadic second-order" by other logics. There have been refinements of this theorem by Rabinovich and Thomas [59], who showed analogous results for first-order logics with various predicates, and by Fridman [30], who broadened the theorem to classes defined by distinct types of pushdown automata.

In the second part of this thesis, we continue these studies for fragments of first-order logic. We refer to the classical structure theory of regular languages, in particular to classes below the starfree languages. We consider piecewise testable languages and two hierarchies classifying the starfree languages, namely the dot-depth hierarchy by Cohen and Brzozowski [23] and the Straubing-Thérien hierarchy [54]. We obtain sharper results on definability of strategies in games defined in the above classes. We show that the quantifier rank in a first-order formula that defines the game has to strictly increase when generating a formula that solves this game. In a language-theoretic context, the increase of the quantifier rank corresponds to a jump from one level of the hierarchy to a level above. We show that in order to solve strong games of dot-depth i, winning strategies of dot-depth $i + 2$ are sufficient, but strategies of dot-depth i are not. We obtain analogous results for the Straubing-Thérien hierarchy. For piecewise testable games, however, it turns out that piecewise testable strategies can be found that solve these games.

The last part of this thesis is concerned with another aspect of measuring the complexity of strategies. We soften the notion of strategies for games on graphs by allowing strategies to choose multiple successor nodes instead of only one successor. There the intuition is that more nondeterminism gives better strategies. This perspective proves useful in any approach of refinement in system design. The more nondeterminism a strategy has, the less restrictive it is on other possibly competing goals. The philosophy here is, that we can put together several nondeterministic strategies to obtain a combined strategy, which still fulfills all the winning conditions. This view on nondeterministic strategies is common for the existing definitions of permissiveness, but there are differences in the precise definitions. Bernet, Janin, and Walukiewicz [3] and Pinchinat and Riedweg [55] use a qualitative notion of permissiveness by comparing each two strategies and specifying which one is more permissive than the other. With these notions, a most permissive strategy might not exist. Therefore, we build upon a different, already existing notion of permissiveness, which has been defined for (finitary) reachability games by Bouyer, Duflot, Markey, and Renault [5]. There the edges from one node to another are equipped with weights, and strategies are rated by the sum of the weights of the edges that they disallow. This sum is called the *penalty* of a strategy. We extend this definition to the infinitary parity objective, denoting games with these objectives "mean-penalty parity games". The *value problem* for this kind of games is a decision problem, which asks whether there is a strategy that does not exceed a given penalty. As it turns out, mean-penalty parity games are closely related to the model of mean-payoff parity games, which was introduced by Chatterjee, Henzinger, and Jurdziński [18]. Revisiting the topic of mean-payoff parity games, we prove that Player 2 has a memoryless winning strategy, and we give some simple proofs of the already known results that these games are determined and that the value problem is in $\mathrm{NP} \cap \mathrm{coNP}$. Furthermore, we develop a new deterministic algorithm, which computes the value of a mean-payoff parity game and which runs faster than all previously known algorithms. We provide two reductions of our model of mean-penalty parity games to mean-payoff parity games. First, a simple reduction, which is of simple nature, but which constructs a game graph of exponential size, and second a more sophisticated reduction, which constructs a game graph in polynomial time. Then we use the results about mean-payoff parity games to solve mean-penalty parity games. We show that the value problem for these games is in $\mathrm{NP} \cap \mathrm{coNP}$ as well, and we present a deterministic algorithm for computing the value of these games, which has similar properties as the algorithm for mean-payoff parity games.

Outline

The further structure of this thesis is as follows.

In Chapter 2, we introduce the main concepts and terminology that we use throughout the entire thesis.

In Chapter 3, we study the complementation of Büchi automata, a model for automata on infinite words. Complementation of this automata model is necessary for switching the roles of the two players in infinite games. If the winning condition of a game is given as a Büchi language, i.e., as a finitely representable Büchi automaton, then by complementing the automaton, we can make every play that was winning for Player 1 to be winning for Player 2 and vice versa. Note that this chapter solely concentrates on the concept of language. The results were obtained in collaboration with Stefan Breuers and Christof Löding and are based on [10].

In the next chapter, Chapter 4, we connect the concept of language with the concept of game. We regard games to be played on the basis of words and winning conditions and winning strategies to be languages. We measure the complexity of winning strategies by utilizing some already established measures of the complexitiy of languages, for example the dot-depth hierarchy and the Straubing-Thérien hierarchy. We show that to solve games inside this hierarchy, you have to go up a few levels in the hierarchy to obtain appropriate winning strategies. The results were obtained in collaboration with Namit Chaturvedi and Wolfgang Thomas and are based on [20, 21].

In Chapter 5, we concentrate on the complexity of games on graphs, as this is the model of games most frequently used in the literature. We measure the complexity of a strategy in such a game by the amount or sum of the weight of edges that are "forbidden" by the strategy. Note that this chapter is based solely on the concept of game. To solve our problems here, we reduce to problems on an already known game type, namely mean-payoff parity games, which mix two different objectives in a graph game. The results were obtained in collaboration with Patricia Bouyer-Decitre, Nicolas Markey, and Michael Ummels and are based on [7, 8, 9].

Finally, in Chapter 6, we give a summary and a short perspective for future research.

Acknowledgments

I am grateful to many people who contributed directly or indirectly to this thesis. In particular, I want to thank the following people, whose assistance was very helpful:

My supervisor Wolfgang Thomas for his constant support and encouragement, for his advice and for his endeavor for revealing to me at least a glimpse at the art of writing. Nicolas Markey for being prepared to act as examiner for this thesis. Michael Ummels for being my teacher and mentor, both in writing and in typesetting, and for numerous fruitful discussions. Christof Löding for a lot of helpful discussions, in which I learned many new things and many old things afresh, due to his patience. Nicolas Markey and Patricia Bouyer-Decitre for giving me the great opportunity to work with them in Cachan and for their great support. Namit Chaturvedi for his compelling force in research, for his motivation and persistence, and for questioning my beliefs and my doubts. Ming-Hsien Tsai and Yih-Kuen Tsay for helpful discussions and for providing example material for Chapter 3. An anonymous reviewer for pointing out the polynomial reduction from mean-penalty parity games to mean-payoff parity games in Chapter 5.

Furthermore, I want to thank all my other colleagues in Aachen, Paris and elsewhere, who made my time both enjoyable and instructive, in particular Alex Spelten for the common need for coffee, Benedikt Bollig for sharing thoughts and ambitions, Bernd Puchala, Claus Thrane for many amusing evenings, Daniel Neider, Diana Fischer, Faried Abu Zaid, Francicleber Ferreira, Frank Radmacher, Graham Steel, Ingo Felscher, Joe-Kai Tsay, Jules Villard for lending his bed, Karianto Wong, Luc Segoufin for lending his car, Łukasz Kaiser, Marcus Gelderie, Martin Lang, Martin Zimmermann for his endeavor for joint research, Michael Holtmann, Michaela Slaats, Nathanaël Fijalkow, Nils Jansen for sharing illuminating thoughts and ideas at conference banquets, Ocan Sankur, Rolf Eschmann, Roman Rabinovich for many pleasant discussions, Romain Brenguier, Rohit Chadha, Silke Cormann for showing me every day that there is a life beyond the ivory tower, Simon Lessenich, Stefan Repke, Steve Kremer, Wied Pakusa, and Wladimir Fridman for enlightening discussions about the absurdity of Sudoku.

The last "thank you" – and also the most cordial – is for my family: for my parents, for my two brothers, and for Barbara and Edwin: their continuous and selfless support and their love were far more important than they could ever imagine.

Chapter 2

Technical Preliminaries

In this chapter, we introduce the basic notations and definitions that we use throughout this entire work. The definitions that we require exclusively for a certain chapter are not introduced here, but instead in the beginning of the respective chapter.

More often than not, we use standard notations [37] regarding languages and automata. For the reader who is familiar with most of the preliminaries addressed in this chapter and who wants to skip them, we advise to at least have a look at the definitions of $*$-languages and ω-languages and the definition of the transition function δ for nondeterministic automata. These differ a little bit from what one would expect from common knowledge of the literature.

2.1 Basic Notations

We use common set-theoretical notation. By $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$ and $\mathbb{R}$ we denote the set of natural numbers, integer numbers, rational numbers and reals, respectively. The natural numbers start at 0, so $\mathbb{N} = \{0, 1, 2, \ldots\}$.

2.2 Semigroups and Monoids

A *semigroup* is a pair $(S, \cdot)$, where S is a set and $\cdot$ is a binary operation on S that is associative, i.e., for two elements $s_1, s_2 \in S$ also the element $s_1 \cdot s_2$ is in S, and the identity $(s_1 \cdot s_2) \cdot s_3 = s_1 \cdot (s_2 \cdot s_3)$ holds for all $s_1, s_2, s_3 \in S$. We simply write $s_1 \cdot s_2 \cdot s_3$ or short $s_1 s_2 s_3$ instead of $(s_1 \cdot s_2) \cdot s_3$.

A *monoid* is a triple $(M, \cdot, 1)$, where $(M, \cdot)$ is a semigroup and $1 \in M$ is a neutral element, i.e., it holds $1 \cdot s = s \cdot 1 = s$ for all $s \in M$. A *monoid homomorphism* from a monoid $(M_1, \bullet, \mathrm{id})$ to a monoid $(M_2, \cdot, 1)$ is a mapping $h\colon M_1 \to M_2$ such that $h(s \bullet t) = h(s) \cdot h(t)$ for all $s, t \in M_1$, and $h(\mathrm{id}) = 1$. A triple $(N, \bullet, \mathrm{id})$ is a *submonoid* of a monoid $(M, \cdot, 1)$ if $(N, \bullet, \mathrm{id})$ is a monoid, N is a subset of M, and $\mathrm{id} = 1$.

2.3 Languages

An *alphabet* $\Sigma = \{a, b, c, \ldots\}$ is a finite nonempty set whose elements are usually called *symbols* or *letters*. A *finite word* (or $*$-word) over an alphabet Σ is a (possibly empty) sequence of letters $w = a_1 \cdot \ldots \cdot a_n$; its length is denoted by $|w| = n$. The *empty word* has length 0 and is denoted by ε. The set of all finite words over Σ is denoted by Σ^* and the set of all finite nonempty words by Σ^+. They are called the *free monoid* and the *free semigroup*, respectively. A *$*$-language* over the alphabet Σ is a subset $K \subseteq \Sigma^*$. An *infinite word* (or ω-word) over an alphabet Σ is an infinite sequence of letters $\alpha = a_0 \cdot a_1 \cdot \ldots$. For an ω-word α and $j < k \in \mathbb{N}$ we denote by $\alpha[j, k)$ or $\alpha[j, k-1]$ the finite infix $a_k \cdots a_{k-1}$, which starts at position j and ends at position $k-1$. By $\alpha[j, \infty)$ we denote the infinite suffix $a_j a_{j+1} \ldots$. The set of all infinite words over Σ is denoted by Σ^ω. An *ω-language* over the alphabet Σ is a subset $L \subseteq \Sigma^\omega$. We will also simply speak of a *language* (without the prefix "$*$-" or "ω-"), when the context determines its meaning. For a $*$-language $K \subseteq \Sigma^*$ and an ω-language $L \subseteq \Sigma^\omega$ we write $\overline{K}$ for $\Sigma^* \setminus K$ and $\overline{L}$ for $\Sigma^\omega \setminus L$. The *concatenation* of two $*$-languages K_1 and K_2 is defined as $K_1 \cdot K_2 \coloneqq \{u \cdot v \mid u \in K_1, v \in K_2\}$, and the *concatenation* of a $*$-language K and an ω-language L is $K \cdot L \coloneqq \{w \cdot \alpha \mid w \in K, \alpha \in L\}$. For a $*$-language K, the ω-language K^ω is defined as $K^\omega \coloneqq \{\alpha = w_0 \cdot w_1 \cdot \ldots \mid \forall i \in \mathbb{N} \colon w_i \in K\}$.

For a class $\mathcal{C}$ of languages, by $\mathrm{BC}(\mathcal{C})$ we denote the class of all finite *Boolean combinations* of languages in $\mathcal{C}$, i.e., $\mathrm{BC}(\mathcal{C})$ is the smallest class such that $\mathcal{C} \subseteq \mathrm{BC}(\mathcal{C})$ and for $A, B \in \mathrm{BC}(\mathcal{C})$ the languages $A \cup B$, $A \cap B$, and $\overline{A}$ are in $\mathrm{BC}(\mathcal{C})$. Analogously, the set of all *positive Boolean combinations* (also just called *positive combinations*) for a class $\mathcal{C}$ is denoted by $\mathrm{pos}(\mathcal{C})$, i.e., $\mathrm{pos}(\mathcal{C})$ is the smallest class such that $\mathcal{C} \subseteq \mathrm{pos}(\mathcal{C})$ and for $A, B \in \mathrm{pos}(\mathcal{C})$ the languages $A \cup B$ and $A \cap B$ are in $\mathrm{pos}(\mathcal{C})$.

For a $*$-language K over Σ, we define its *extension* and its *limit*

$$\mathrm{ext}(K) \coloneqq \{\alpha \in \Sigma^\omega \mid \exists i \colon \alpha[0, i) \in K\} \text{ and}$$
$$\lim(K) \coloneqq \{\alpha \in \Sigma^\omega \mid \forall k \; \exists i \geq k \colon \alpha[0, i) \in K\}$$

as the set of all ω-words that have a prefix in K and the set of all ω-words that have infinitely many prefixes in K, respectively. Note that $\mathrm{ext}(K)$ can also be written as $K \cdot \Sigma^\omega$ and that in the literature usually $\lim(K)$ is written as $\vec{K}$. Both operators are lifted to language classes by setting $\mathrm{ext}(\mathcal{C}) \coloneqq \{L \subseteq \Sigma^\omega \mid \exists K \in \mathcal{C} \colon L = \mathrm{ext}(K)\}$ and $\lim(\mathcal{C}) \coloneqq \{L \subseteq \Sigma^\omega \mid \exists K \in \mathcal{C} \colon L = \lim(K)\}$ for a class $\mathcal{C}$ of $*$-languages over Σ.

2.4 Automata

In this work, a finite automaton is considered to be a language accepting machine. Some automata models have the same syntactic structure, and distinct names of automata just refer to distinct semantics. For this reason, in the following we first define the syntactic structure of an "automaton" and second we give meanings to this structure by employing two distinct semantics, one for (nondeterministic) finite automata and one for Büchi automata. Third, we define the notions of a "Staiger-Wagner automaton" and of a "Muller automaton".

Formally, an *automaton* is a 5-tuple $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$, where

- Q is a finite set of *states*,
- Σ is an alphabet,
- $q_0 \in Q$ is the *initial state*,
- $\delta \colon Q \times \Sigma \to 2^Q$ is the *transition function*, and
- $F \subseteq Q$ is a set of *accepting states*.

The transition function δ is called *deterministic* if $|\delta(q, a)| \leq 1$ for all $q \in Q$ and all $a \in \Sigma$. We then also say that the automaton is *deterministic*. Furthermore, a deterministic automaton is called *complete* if every state has outgoing transitions for all letters, i.e., if $|\delta(q, a)| = 1$ for all $q \in Q$ and all $a \in \Sigma$. An automaton is called *normalized* if there is no transition entering the initial state, i.e., for all states q and letters a it holds $\delta(q, a) \cap \{q_0\} = \emptyset$.

For automata we consider two different semantics: the usual NFA semantics, in which the automaton accepts a language of finite words, and the Büchi semantics, in which the automaton accepts a language of infinite words. This is made precise in the following definitions.

A *path* in $\mathcal{A}$ from p to q of a word $u = a_1 \cdots a_n$ is a finite sequence of states $p_0, \ldots, p_n$ such that $p_0 = p$, $p_n = q$, and $p_i \in \delta(p_{i-1}, a_i)$ for all $1 \leq i \leq n$. Define $\delta^*(p, u) \coloneqq \{q \in Q \mid \text{there is a path of } u \text{ from } p \text{ to } q\}$. If $\mathcal{A}$ is deterministic, then this set is a singleton $\delta^*(p, u) = \{q\}$ and we sloppily write $\delta^*(p, u) = q$. An *infinite run* of $\mathcal{A}$ on a word $\alpha \in \Sigma^\omega$ is an infinite sequence of states $\rho = p_0, p_1, \ldots$ such that $p_0 = q_0$, and $p_{i+1} \in \delta(p_i, \alpha(i))$ for all $i \geq 0$.

An *NFA* ("nondeterministic finite automaton") is an automaton $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$. A finite word w is *NFA-accepted* by $\mathcal{A}$ if there is a state $q_\mathrm{f} \in$

F and a path in $\mathcal{A}$ from q_0 to q_f of w. A *DFA* ("deterministic finite automaton") is a deterministic automaton $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$. A finite word w is *DFA-accepted* by $\mathcal{A}$ if it is NFA-accepted by $\mathcal{A}$. A (nondeterministic) *Büchi automaton* is an automaton $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$. An infinite word α is *Büchi-accepted* by $\mathcal{A}$ if there is an infinite run ρ of $\mathcal{A}$ on α with $\rho(i) \in F$ for infinitely many $i \geq 0$.

An automaton $\mathcal{A}$ *recognizes* the languages

$$\mathrm{L}_*(\mathcal{A}) \coloneqq \{w \in \Sigma^* \mid w \text{ is NFA-accepted by } \mathcal{A}\} \text{ and}$$
$$\mathrm{L}_\omega(\mathcal{A}) \coloneqq \{\alpha \in \Sigma^\omega \mid \alpha \text{ is Büchi-accepted by } \mathcal{A}\}.$$

Usually it is obvious from the context which kind of semantics we mean. In this case, we omit the explicit reference of the semantics and just say that a word is *accepted* by an automaton $\mathcal{A}$, and speak of the language $\mathrm{L}(\mathcal{A})$ as the set of words that are accepted by $\mathcal{A}$. If a language is recognized by some automaton $\mathcal{A}$, then we say that the language $\mathrm{L}(\mathcal{A})$ is *regular*. The class of all regular $*$-languages over Σ is denoted by REG.

Büchi showed that the class of regular ω-languages can be characterized in the following way.

Characterization 1 *An ω-language L is regular iff there is an $n \in \mathbb{N}$, such that*

$$L = \bigcup_{i=1}^{n} U_i \cdot V_i^\omega, \text{ where all } U_i \text{ and } V_i \text{ are regular } *\text{-languages.}$$

A (deterministic) *SW-automaton* ("Staiger-Wagner automaton" [68]) or a *weak Muller automaton* is a 5-tuple $\mathcal{A} = (Q, \Sigma, q_0, \delta, \mathcal{F})$ where

- Q, Σ, and q_0 are defined as for automata,
- $\delta\colon Q \times \Sigma \to 2^Q$ is a deterministic *transition function*, and
- $\mathcal{F} \subseteq 2^Q$ is a set of *accepting state sets*.

An infinite run for $\mathcal{A}$ is defined as for automata. An infinite word α is *SW-accepted* by $\mathcal{A}$ if there is an infinite run ρ of $\mathcal{A}$ on α such that the set

$$\mathrm{Occ}(\rho) \coloneqq \{q \in Q \mid \exists i\colon \rho(i) = q\}$$

of states occurring in ρ is in $\mathcal{F}$. A weak Muller automaton $\mathcal{A}$ *recognizes* an ω-language L if $L = \{\alpha \in \Sigma^\omega \mid \alpha \text{ is SW-accepted by } \mathcal{A}\}$.

Characterization 2 *An ω-language L is recognized by an SW-automaton iff*

$$L \in \mathrm{BC}(\mathrm{ext}(\mathrm{REG})).$$

A (deterministic) *Muller automaton* $\mathcal{A}$ is syntactically the same object as an SW-automaton. An infinite word α is *Muller-accepted* by $\mathcal{A}$ if there is an infinite run ρ of $\mathcal{A}$ on α such that the set

$$\mathrm{Inf}(\rho) \coloneqq \{q \in Q \mid \forall k\ \exists i \geq k \colon \rho(i) = q\}$$

of states occurring infinitely often in ρ is in $\mathcal{F}$. A Muller automaton $\mathcal{A}$ *recognizes* an ω-language L if $L = \{\alpha \in \Sigma^\omega \mid \alpha \text{ is Muller-accepted by } \mathcal{A}\}$. A central result characterizing the ω-regular languages is the following (see [52, 74]).

Characterization 3 *An ω-language L is regular iff L is recognized by a Muller automaton iff*

$$L \in \mathrm{BC}(\lim(\mathrm{REG})).$$

2.5 Transducers

As a model of automata with output we use Moore machines (see [38]), which transform words over an alphabet Σ into words over an alphabet Γ via an output function $\lambda \colon Q \to \Gamma$ over the state set Q.

Formally, a *Moore machine* is a 6-tuple $\mathcal{A} = (Q, \Sigma, \Gamma, q_0, \delta, \lambda)$, where

- Q, q_0, and δ are defined as for automata,
- Σ and Γ are the *input alphabet* and the *output alphabet*, and
- $\lambda \colon Q \to \Gamma$ is a *labeling function*.

Moore machines are complete deterministic automata, i.e., we require $|\delta(q, a)| = 1$ for all $q \in Q$ and all $a \in \Sigma$. For Moore machines, δ^* is defined as for automata. We make use of Moore machines in Chapter 4, where we use them to express strategies for games.

2.6 Games on Graphs

The kind of games we consider in this thesis are two-player turn-based zero-sum games with perfect information and infinite duration played on finite graphs. These games are considered in the literature (see [33]) usually over a graph with two different types of vertices: one type belonging to Player 1, the other to Player 2.

A *game graph* or *arena* is a tuple $G = (Q_1, Q_2, E)$, where $Q \coloneqq Q_1 \cup Q_2$ is a finite set of *vertices* or *states*, $Q_1 \cap Q_2 = \emptyset$, and $E \subseteq Q \times Q$ is an *edge* or *transition relation*. The states in Q_1 are called *Player 1 states*, and the states in Q_2 are called *Player 2 states*.

For $q \in Q$, we write qE for the set $\{q' \in Q \mid (q, q') \in E\}$ of all successors of q. We require that $qE \neq \emptyset$ for all states $q \in Q$. A subset $S \subseteq Q$ is a *subarena* of G if $qE \cap S \neq \emptyset$ for all states $q \in S$. If $S \subseteq Q$ is a subarena of G, then we can restrict G to states in S, in which case we obtain the game graph $G \restriction S \coloneqq (Q_1 \cap S, Q_2 \cap S, E \cap (S \times S))$.

A *play* of G is an infinite sequence $\rho = \rho(0)\rho(1)\cdots \in Q^\omega$ of states such that $(\rho(i), \rho(i+1)) \in E$ for all $i \in \mathbb{N}$. We denote by $\mathrm{Out}^G(q)$ the set of all plays ρ with $\rho(0) = q$. By $\mathrm{Occ}(\rho)$ and $\mathrm{Inf}(\rho)$ we denote the set of states occurring in ρ and the set of states occurring infinitely often in ρ, respectively.

A *play prefix* or a *history* $\gamma = \gamma(0)\gamma(1)\cdots\gamma(n) \in Q^+$ is a finite, nonempty prefix of a play. For a play or a history ρ and $j < k \in \mathbb{N}$, its infix that starts at position j and ends at position $k-1$ is denoted by $\rho[j,k) \coloneqq \rho[j,k-1] \coloneqq \rho(j)\cdots\rho(k-1)$, its infinite suffix $\rho(j)\rho(j+1)\cdots$ is denoted by $\rho[j,\infty)$.

Strategies

A *(deterministic) strategy* for Player i in G is a function $\sigma\colon Q^*Q_i \to Q$ such that $\sigma(\gamma q) \in qE$ for all $\gamma \in Q^*$ and $q \in Q_i$. A strategy σ is *memoryless* if $\sigma(\gamma q) = \sigma(q)$ for all $\gamma \in Q^*$ and $q \in Q_i$. More generally, a strategy σ is *finite-memory* if the equivalence relation $\sim \subseteq Q^* \times Q^*$, defined by $\gamma_1 \sim \gamma_2$ if and only if $\sigma(\gamma_1 \cdot \gamma) = \sigma(\gamma_2 \cdot \gamma)$ for all $\gamma \in Q^*Q_i$, has finite index.

We say that a play ρ of G is *consistent* with a strategy σ for Player i if $\rho(k+1) = \sigma(\rho[0,k])$ for all $k \in \mathbb{N}$ with $\rho(k) \in Q_i$, and denote by $\mathrm{Out}^G(\sigma, q_0)$ the set of all plays ρ of G that are consistent with σ and start in $\rho(0) = q_0$. Given a strategy σ of Player 1, a strategy τ of Player 2, and a state $q_0 \in Q$, there exists a unique play $\rho \in \mathrm{Out}^G(\sigma, q_0) \cap \mathrm{Out}^G(\tau, q_0)$, which we denote by $\rho^G(\sigma, \tau, q_0)$.

Traps and attractors

Intuitively, a set $T \subseteq Q$ of states is a *trap* for one of the two players if the other player can enforce that the play stays in this set. Formally, a trap for Player 2 (or simply a 2-trap) is a subarena $T \subseteq Q$ such that $qE \subseteq T$ for all states $q \in T \cap Q_2$, and $qE \cap T \neq \emptyset$ for all $q \in T \cap Q_1$. A trap for Player 1 (or 1-trap) is defined analogously. Note that if T is a trap for Player i in $G \restriction S$ and S is a trap for Player 1 in G, then T is also a trap for Player i in G.

If $T \subseteq Q$ is not a trap for Player 1, then Player 1 has a strategy to reach a position in $Q \setminus T$. In general, given a subset $S \subseteq Q$, we denote by $\mathrm{Attr}_1^G(S)$ the set of states from where Player 1 can force a visit to S. This set can be characterized as the limit of the sequence $(A_i)_{i \in \mathbb{N}}$ defined by $A^0 = S$ and

$$A^{i+1} = A^i \cup \{q \in Q_1 \mid qE \cap A^i \neq \emptyset\} \cup \{q \in Q_2 \mid qE \subseteq A^i\}\,.$$

From every state in $\mathrm{Attr}_1^G(S)$, Player 1 has a memoryless strategy σ that guarantees a visit to S in at most $|Q|$ steps: the strategy chooses for each state $q \in (A^i \setminus A^{i-1}) \cap Q_1$ a state $p \in qE \cap A^{i-1}$ (which decreases the distance to S by 1). We call the set $\mathrm{Attr}_1^G(S) = \bigcup_{i \in \mathbb{N}} A_i$ the 1-*attractor of* S and σ an *attractor strategy for* S. The 2-*attractor* of a set S, denoted by $\mathrm{Attr}_2^G(S)$, and attractor strategies for Player 2 are defined symmetrically. Notice that for any set S, the set $Q \setminus \mathrm{Attr}_1^G(S)$ is a 1-trap, and if S is a subarena (2-trap), then $\mathrm{Attr}_1^G(S)$ is also a subarena (2-trap). Analogously, $Q \setminus \mathrm{Attr}_2^G(S)$ is a 2-trap, and if S is a subarena (1-trap), then $\mathrm{Attr}_2^G(S)$ is also a subarena (1-trap).

Winning Conditions

Formally, a *parity game* is a tuple $\mathcal{G} = (G, \chi)$, where G is a game graph, and $\chi \colon Q \to \mathbb{N}$ is a priority function assigning a *priority* to every state. A play $\rho = \rho(0)\rho(1)\cdots$ is *parity-winning* if the minimal priority occurring infinitely often in ρ is even, i.e., if $\min\{\chi(q) \mid q \in \mathrm{Inf}(\rho)\} \equiv 0 \pmod 2$. If $\chi(Q) \subseteq \{0, 1\}$, then we say that $\mathcal{G}$ is a *Büchi game*. Hence, in a Büchi game Player 1 needs to visit the set $\chi^{-1}(0)$ infinitely often.

We say that a strategy σ for Player 1 is a *winning strategy from* q_0 if each play $\rho \in \mathrm{Out}^G(\sigma, q_0)$ is parity-winning. Analogously, a strategy τ for Player 2 is a *winning strategy from* q_0 if each play $\rho \in \mathrm{Out}^G(\tau, q_0)$ is not parity-winning.

We say that a game is determined if for every state $q_0 \in Q$ either Player 1 or Player 2 has a winning strategy from q_0. Parity games, and therefore

also Büchi games, are determined and the winning player has a winning strategy that is memoryless. This is a well-known result by Emerson and Jutla [27], and Mostowski [49].

Formally, a *Muller game* is a tuple $\mathcal{G} = (G, \mathcal{F})$, where G is a game graph, and $\mathcal{F} \subseteq 2^Q$ is a set of subsets of Q. A play $\rho = \rho(0)\rho(1)\cdots$ is *Muller-winning* if the set of states occurring infinitely often in ρ is a member of $\mathcal{F}$, i.e., if $\mathrm{Inf}(\rho) \in \mathcal{F}$.

An *SW-game* ("Staiger-Wagner game" or *weak Muller game*) is a tuple $\mathcal{G} = (G, \mathcal{F})$, where G is a game graph, and $\mathcal{F} \subseteq 2^Q$ is a set of subsets of Q. A play $\rho = \rho(0)\rho(1)\cdots$ is *SW-winning* if the set of states occurring in ρ is a member of $\mathcal{F}$, i.e., if $\mathrm{Occ}(\rho) \in \mathcal{F}$.

All notions that we have defined for game graphs carry over to parity games and Muller games. In particular, a play of $\mathcal{G}$ is just a play of G and a strategy for Player i in $\mathcal{G}$ is nothing but a strategy for Player i in G. Hence, we write $\mathrm{Out}^{\mathcal{G}}(\sigma, q)$ for $\mathrm{Out}^{G}(\sigma, q)$, and so on.

Game Reductions

Games with Staiger-Wagner winning conditions or Muller winning conditions can be solved by converting the winning condition into a parity condition while expanding the game graph by an additional "memory component" (see [75, 72]).

For SW-games we replace a state q of the game graph by a pair (q, R) where R is the set of those states visited in the current play prefix so far. The set R is called the AR ("appearance record"). Formally, $\mathrm{AR}(\gamma)$ is defined by $\mathrm{AR}(q) := \{q\}$ for any state $q \in Q$, and $\mathrm{AR}(\gamma q) := \mathrm{AR}(\gamma) \cup \{q\}$ for any play prefix γ and state q.

Any SW-game $\mathcal{G} = (G, \mathcal{F})$ with $G = (Q_1, Q_2, E)$ can be reduced to a parity game $\mathcal{G}' = (G', \chi')$ where $G' = (Q_1 \times 2^Q, Q_2 \times 2^Q, E')$ is the expanded game graph with $((p, R), (q, S)) \in E'$ iff $R \cup \{q\} = S$, and where χ' is defined by

$$\chi'((p, R)) := \begin{cases} 2 \cdot |R|, & \text{if } R \in \mathcal{F}; \\ 2 \cdot |R| - 1, & \text{otherwise.} \end{cases}$$

For Muller games we make use of the LAR ("latest appearance record") (see [35, 75]). This memory component refines the above information by listing the visited states in order of most recent visits and by providing a pointer to that place in the sequence where the current state was located in the preceding step. For a state space $Q = \{q_1, \ldots, q_n\}$, the LAR is a pair consisting of a permutation *perm* of Q and an index $h \in \{1, \ldots, n\}$. It can

be defined inductively by $\mathrm{LAR}(q) \coloneqq ((q_1, \ldots, q_n), 1)$ for any state $q \in Q$, and $\mathrm{LAR}(\gamma q)$ is obtained from $\mathrm{LAR}(\gamma)$ by moving state q from position i to position 1 and setting $h \coloneqq i$. By LARs denote the set of all possible LARs over Q.

Any Muller game $\mathcal{G} = (G, \mathcal{F})$ with $G = (Q_1, Q_2, E)$ can be reduced to a parity game $\mathcal{G}' = (G', \chi')$ where $G' = (Q_1 \times \mathrm{LARs}, Q_2 \times \mathrm{LARs}, E')$ is the expanded game graph with $((p, R), (q, S)) \in E'$ iff the LAR $S = (\mathit{perm}', h')$ is obtained from the LAR $R = (\mathit{perm}, h)$ by moving state q from position i to position 1 and setting $h' \coloneqq i$, and where χ' is defined for an LAR $R = ((q_1, \ldots, q_n), h)$ by

$$\chi'((p, R)) \coloneqq \begin{cases} 2 \cdot |R|, & \text{if } \{q_1, \ldots, q_h\} \in \mathcal{F}; \\ 2 \cdot |R| - 1, & \text{otherwise.} \end{cases}$$

Both game reductions are correct in the sense that any play prefix (and thus any play) in the original game is translated into a play prefix (or a play) in the expanded game with the property that the original play is won by Player 1 iff the expanded play is won by Player 1. The memory component of the game reduction thus corresponds to a finite state winning strategy for the winning player.

2.7 First-order Logic

We use first-order logic over several structures throughout this thesis. For an overview, see [70, 25]. In this section, we define first-order logic over the specific signatures $\{(P_a)_{a \in \Sigma}, <\}$ and $\{(P_a)_{a \in \Sigma}, <, \mathrm{suc}, \mathrm{pre}, \min, \max\}$ with the intended interpretations over words.

Let Var be a set, called the set of *first-order variables*. A first-order variable is usually written as a lowercase letter, like x, y or z.

A *term* is inductively defined by the following rules. Every first-order variable $x \in \mathrm{Var}$ is a term; min and max are terms; if t is a term, then $\mathrm{suc}(t)$ and $\mathrm{pre}(t)$ are also terms. An *atomic formula* is defined by the following rules. $t_1 = t_2$ is an atomic formula for terms t_1 and t_2; $P_a(t)$ is an atomic formula for every $a \in \Sigma$ and every term t; $t_1 < t_2$ is an atomic formula for terms t_1 and t_2.

The set of all *first-order formulas* is defined inductively as follows. Every atomic formula is a formula; if φ and ψ are formulas, then $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $(\varphi \to \psi)$ and $\neg\psi$ are formulas; if ψ is a formula and $x \in \mathrm{Var}$ is a variable, then $\exists x \psi$ and $\forall x \psi$ are formulas.

Let $\mathrm{Var}(\psi)$ be the set of all variables occurring in the formula ψ. The set $\mathrm{Free}(\psi)$ of *free variables* of ψ is defined inductively by the following rules.

- If ψ is an atomic formula, then $\mathrm{Free}(\psi) = \mathrm{Var}(\psi)$.
- $\mathrm{Free}(\neg\psi) = \mathrm{Free}(\psi)$.
- $\mathrm{Free}(\varphi \wedge \psi) = \mathrm{Free}(\varphi \vee \psi) = \mathrm{Free}(\varphi \to \psi) = \mathrm{Free}(\varphi) \cup \mathrm{Free}(\psi)$.
- $\mathrm{Free}(\exists x\psi) = \mathrm{Free}(\forall x\psi) = \mathrm{Free}(\psi) \setminus \{x\}$.

A *sentence* is a formula ψ without any free variable, i.e., $\mathrm{Free}(\psi) = \emptyset$.

A formula ψ is in *prenex normal form* if has the form

$$\psi = Q_1 x_1 Q_2 x_2 \dots Q_n x_n \varphi$$

with $Q_i \in \{\exists, \forall\}$ and $x_i \in \mathrm{Var}$ and where φ is a quantifier-free formula. A Σ_n*-sentence* is a first-order sentence in prenex normal form with n alternating quantifier blocks starting with an existential block.

The corresponding first-order logic with signature $\{(P_a)_{a\in\Sigma}, <\}$, i.e., the symbols min, max, pre, and suc do not occur in formulas of this logic, is denoted by FO[<], whereas the first-order logic over the complete signature $\{(P_a)_{a\in\Sigma}, <, \mathrm{suc}, \mathrm{pre}, \mathrm{min}, \mathrm{max}\}$ is denoted by FO[$<, \mathrm{suc}, \mathrm{pre}, \mathrm{min}, \mathrm{max}$].

A finite word $w = a_1 \dots a_n$ over Σ with length $|w| = n > 0$ can be identified with the structure

$$\underline{w} = (\mathrm{dom}(w), <, (P_a)_{a\in\Sigma})$$

where $\mathrm{dom}(w) = \{1, \dots, n\}$, the relation $<$ is interpreted as the standard ordering on $\mathrm{dom}(w)$, $P_a = \{i \in \mathrm{dom}(w) \mid a_i = a\}$, the elements min and max are the first and last element of $\mathrm{dom}(w)$, and the successor and predecessor functions suc and pre respectively, such that $\mathrm{suc}(\mathrm{max}) = \mathrm{max}$ and $\mathrm{pre}(\mathrm{min}) = \mathrm{min}$. We write $\underline{w} \models \psi$ for any sentence ψ if ψ is satisfied in $\underline{w}$ under this canonical interpretation.

Analogously, an infinite word $\alpha = a_0 a_1 \dots$ over Σ is identified with the structure

$$\underline{\alpha} = (\mathbb{N}, <, (P_a)_{a\in\Sigma}),$$

and we write $\underline{\alpha} \models \psi$ for any sentence ψ if ψ is satisfied in $\underline{\alpha}$.

We say that a $*$-language $K \subseteq \Sigma^*$ *is defined* by a sentence ψ if for every word $w \in \Sigma^+$ it holds

$$w \in K \iff \underline{w} \models \psi.$$

We say that an ω-language $L \subseteq \Sigma^\omega$ *is defined* by a sentence ψ if for every word $\alpha \in \Sigma^\omega$ it holds

$$\alpha \in L \iff \underline{\alpha} \models \psi.$$

For a formula ψ and a variable x where x does not occur in ψ, we introduce the *relativisation* $\psi[x]$ of ψ with respect to the segment $[0, x]$ (see [25]). Formally this means that we introduce x as a new free variable and rewrite every subformula of ψ of the form $\exists y \varphi(y)$ as $\exists y(y \leq x \wedge \varphi(y))$, and every subformula of the form $\forall y \varphi(y)$ as $\forall y(y \leq x \to \varphi(y))$, and every occurrence of max as x.

Chapter 3

Büchi Automata Complementation via Ramsey's Theorem

In 1962, J. Richard Büchi showed [12] that the monadic second-order theory of the natural numbers with the successor relation is decidable. In his proof, he introduced a model for ω-regular languages, a model which we nowadays call a "Büchi automaton", and he also proved that this model is closed under the complementation operation. Büchi proved this result en passant, but hitherto it is considered to be one of the most intriguing topics in automata theory, and still every year several papers are published in the area of Büchi complementation.

For proving his complementation result, Büchi utilized a combinatorial result by Ramsey [60] to obtain a periodic decomposition of the behaviors of a Büchi automaton when reading an infinite word. For this reason, Büchi's complementation approach is also called the *Ramsey-based* approach. Over the past decades, a few other complementation methods were developed, which are namely the *determinization-based*, the *rank-based*, and the *slice-based* approach.

The number of states of the complement automaton is a measure by which all these complementation methods are judged. Let n be the number of states of the original automaton. Then for the Ramsey-based approach, an upper bound of $2^{\mathcal{O}(n^2)}$ states for the complement automaton was established [67] and Michel [48] proved a non-trivial lower bound of $n!$ for the complementation of Büchi automata in general. So there still remained a huge gap between both bounds. Then Safra [61] presented a method to transform a nondeterministic Büchi automaton into a deterministic Muller automaton and later on Piterman [56] optimized this construction. This lead to a *determinization-based* complementation construction with an upper bound of $2^{\mathcal{O}(n \log n)}$, which tightened the gap between the lower and the upper bound. [42] presented a very elegant and simple complementation construction based on progress measures, which possessed the same upper

bound as the determinization-based approach. These progress measures are today mostly referred to as ranking functions and therefore Klarlund's approach is also called the *rank-based* approach. The rank-based approach was later improved by Friedgut, Kupferman, and Vardi [31] and then even further by Schewe [62], leaving only a polynomial gap between the new upper bound and the improved lower bound that was obtained by Yan [81]. Contemporaneous with the optimizations of the rank-based approach in the past years, a completely new complementation method was established by Kähler and Wilke [40], which is called the *slice-based* approach. Table 3.1 gives a rough overview over the four complementation approaches.

Approach		
Name	discovered by	State blow-up
Ramsey-based	Büchi (1962)	$2^{\mathcal{O}(n^2)}$
Determinization-based	Safra (1988)	$2^{\mathcal{O}(n \log n)}$
Rank-based	Klarlund (1991)	$2^{\mathcal{O}(n \log n)}$
Slice-based	Kähler-Wilke (2008)	$2^{\mathcal{O}(n \log n)}$

Table 3.1: The four different complementation approaches.

Given that all these approaches have, on the theoretical side, a rather unpleasant upper bound on the size of the complement automaton of $2^{\mathcal{O}(n \log n)}$ or $2^{\mathcal{O}(n^2)}$, it stands to reason to compare them as well on the practical side. There has been an increased interest in experimental evaluations of the different complementation methods in the recent years. In experimental studies [77], it turned out that the Ramsey-based approach was inferior to the other three, more modern approaches, not only on the theoretical side, but also in practice. In Table 3.2 we present a quick overview of the results of Tsai, Fogarty, Vardi, and Tsay [77]. It shows the results of four different implementations, named `Ramsey`, `Safra-Piterman`, `Rank`, and `Slice`. Each implementation was fed the same set of 11 000 input automata, and for each of these task, the same timeout of 10 minutes and the same memory space of 1 GiB was given. More details on the setting can be found below in Section 3.3.4. The most noticeable value in this comparison is the amount of tasks that were not finished by `Ramsey`. In fact, none of the tasks finished in the predefined time and memory bounds. When we started researching possible improvements of the Ramsey-based approach, we did not know the reason for this bad result. As it turned out later [78], it is due to a flaw in the implementation of `Ramsey`. At the time we learned about this flaw,

we already had our own, improved implementation of the Ramsey-based approach. Despite of the named deficiencies of the Ramsey-based approach for the complementation of Büchi automata, it is still used in two neighboring fields, universality checking and inclusion checking [29, 2, 1]. For these two fields specific optimization techniques were developed and the resulting methods are claimed to be way more efficient than their complementation counterparts.

Algorithm	Unfinished Tasks
`Safra-Piterman`	4
`Slice`	216
`Rank`	3 383
`Ramsey`	11 000

Table 3.2: Four different implementations, studied in Tsai, Fogarty, Vardi, and Tsay [77].

This chapter is devoted to improving, based on recent research [10], the Ramsey-based approach, both on the theoretical side as well as on the practical side. We first present Ramsey's Theorem, then in Section 3.2 we remind the reader of how the complementation has been described originally by Büchi [12] and Sistla, Vardi, and Wolper [67]. In Section 3.3 we propose some heuristics to improve the Ramsey-based approach, in order to make it practically applicable. Furthermore we give an overview over the results of our experimental studies of our implementation. Then in Section 3.4 we explain how to obtain a better upper bound on the size of the resulting automata by the use of weak orders, and finally we connect our heuristically improved algorithm with the weak-order-based algorithm.

3.1 Ramsey's Theorem

For the sake of completeness, we state Ramsey's Theorem [60] here. It will be used in the following section to prove the so-called Sequential Lemma. For $r \in \mathbb{N}$ and an arbitrary set Γ, an *r-combination* of Γ is a subset $P \subseteq \Gamma$ with $|P| = r$.

Theorem 1 (Ramsey) *Let Γ be an infinite set and let $\mu, r \in \mathbb{N}$. Let Γ_r be the set of all r-combinations of Γ. Let $f\colon \Gamma_r \to \{1, \ldots, \mu\}$ be a finite coloring of Γ_r. Then there is an infinite subset $\Delta \subseteq \Gamma$ such that f restricted to the set $\Delta_r := \{P \subseteq \Delta \mid |P| = r\}$ of r-combinations of Δ is constant.*

3.2 The Ramsey-based Approach Revisited

In this section, we recap the original Ramsey-based complementation approach. This is necessary to prepare for the improved approach which is described in the next section. Detailed presentations of this original method can be found in the literature [67, 71]. We assume that $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ is an arbitrary automaton, which we want to complement regarding the Büchi acceptance condition. So we seek an automaton $\mathcal{A}'$ with $\mathrm{L}_\omega(\mathcal{A}') = \Sigma^\omega \setminus \mathrm{L}_\omega(\mathcal{A})$.

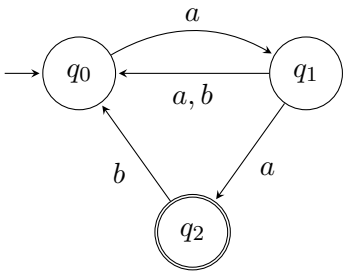


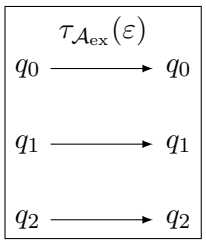
Figure 3.1: An automaton $\mathcal{A}_{\mathrm{ex}}$.

Example *Throughout this chapter, the automaton $\mathcal{A}_{\mathrm{ex}}$ depicted in Figure 3.1 serves as a running example. We will apply the developed methods to this automaton, and in the end we will obtain a complement automaton.*

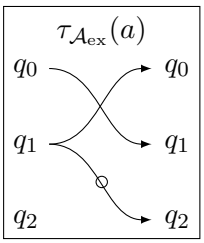
The idea of the Ramsey-based approach is for the complement automaton to guess a periodic decomposition of the input word into infinitely many finite segments. The decomposition is periodic in the sense that the original automaton $\mathcal{A}$ ultimately exhibits the exact same behavior on these segments. The different behaviors an automaton can possess are captured by so-called "transition profiles". A transition profile of a word u over an automaton $\mathcal{A}$ describes the possible run continuations for each state of the automaton $\mathcal{A}$ when reading the word u. Furthermore a transition profile distinguishes between run continuations that exhibit a final state when reading u and those that do not.

We introduce the following notations. If there is a (possible empty) path from p to q of u in $\mathcal{A}$, then we denote this fact by $p \xrightarrow{u} q$. If furthermore there is a nonempty path from p to q of u in $\mathcal{A}$ that visits a final state, then we additionally denote this by $p \xrightarrow[F]{u} q$ (in particular, this is the case if p is final or if q is final).

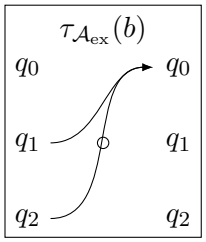
Definition 2 *Let $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ be an automaton. A pair $t = \langle \to_t, \not\to_t \rangle$ with $\to_t \subseteq Q \times Q$ and $\not\to_t \subseteq \to_t$ is called a* transition profile *over $\mathcal{A}$. The* transition profile $\tau_{\mathcal{A}}(u)$ of the word $u \in \Sigma^*$ *over $\mathcal{A}$ is the pair $\langle \to, \not\to \rangle$ with $p \to q$ iff $p \xrightarrow{u} q$ and $p \not\to q$ iff $p \xrightarrow[F]{u} q$.*



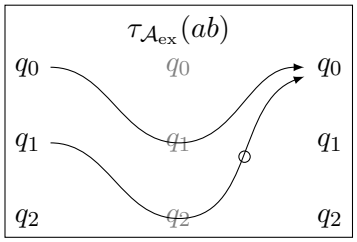
(a) The transition profile $\tau_{\mathcal{A}_{\text{ex}}}(\varepsilon)$.



(b) The transition profile $\tau_{\mathcal{A}_{\text{ex}}}(a)$.



(c) The transition profile $\tau_{\mathcal{A}_{\text{ex}}}(b)$.



(d) The concatenation $\tau_{\mathcal{A}_{\text{ex}}}(ab)$.

Figure 3.2: Some transition profiles of $\mathcal{A}_{\text{ex}}$.

There are several possible ways to visualize a transition profile. We present one of them in Figure 3.2. There, for each state there is a row with two occurrences of this state, one in each column. Arrows are drawn from the states in the left column to the states in the right column in the following way. There is an arrow from p to q if $p \to q$, i.e., if the word u permits a transition from p to q. An arrow from p to q is marked with a circle if $p \not\to q$, i.e., if the transition can occur via a final state. A transition profile $\tau_{\mathcal{A}}(u)$ contains information about the behavior of $\mathcal{A}$ on u, relevant to a Büchi automaton.

Let $\text{TP}_{\mathcal{A}}$ be the set of all transition profiles over $\mathcal{A}$. For transition profiles $s_1, s_2 \in \text{TP}_{\mathcal{A}}$, the concatenation transition profile $t = s_1 \cdot s_2$ is defined by

- $p \to_t q$ iff $\exists r \in Q$ such that $p \to_{s_1} r \wedge r \to_{s_2} q$, and
- $p \not\to_t q$ iff $\exists r \in Q$ such that $(p \not\to_{s_1} r \wedge r \to_{s_2} q) \vee (p \to_{s_1} r \wedge r \not\to_{s_2} q)$.

One can easily verify that this operation is associative. Thus, $(\mathrm{TP}_{\mathcal{A}}, \cdot)$, the set of all transition profiles over $\mathcal{A}$ in conjunction with the concatenation operation, forms a semigroup. In addition, $\tau_{\mathcal{A}}(\varepsilon)$ acts as a neutral element, and so $(\mathrm{TP}_{\mathcal{A}}, \cdot, \tau(\varepsilon))$ forms a monoid. It is called the *transition monoid* of $\mathcal{A}$. Furthermore, by induction on the length of the words one can show that $\tau_{\mathcal{A}}(u \cdot v) = \tau_{\mathcal{A}}(u) \cdot \tau_{\mathcal{A}}(v)$. This means that $\tau_{\mathcal{A}}$ is a monoid homomorphism, mapping each element of Σ^* to a transition profile in $\mathrm{TP}_{\mathcal{A}}$.

An automaton $\mathcal{A}$ with n states has exactly n^2 distinct pairs of states. For each pair (p, q), a transition profile can be in one of three states: either have $p \not\to q$, or $p \to q$ but $p \not\Rightarrow q$, or $p \Rightarrow q$. Thus, there are exactly 3^{n^2} distinct transition profiles in the transition monoid. However, not for all of these transition profiles there is a word u which induces this profile by $\tau_{\mathcal{A}}(u)$. This is why we introduce the notion of a reachable transtion profile. A transition profile t over $\mathcal{A}$ is *reachable* if there is a word $u \in \Sigma^*$ with $\tau_{\mathcal{A}}(u) = t$. By $\mathrm{RTP}_{\mathcal{A}}$ we denote the set of all reachable transition profiles. Note that $(\mathrm{RTP}_{\mathcal{A}}, \cdot, \tau_{\mathcal{A}}(\varepsilon))$ is a submonoid of $(\mathrm{TP}_{\mathcal{A}}, \cdot, \tau_{\mathcal{A}}(\varepsilon))$.

Note that the transition profile $\tau_{\mathcal{A}}(\varepsilon)$ plays a special role among all reachable transition profiles. Since there exists at least one accepting state (otherwise complementation is trivial), this accepting state induces a $p \Rightarrow q$ edge in each transition profile $\tau_{\mathcal{A}}(u)$ with $u \in \Sigma^+$. This means that $\tau_{\mathcal{A}}(\varepsilon)$ is not reachable from any other transition profile in $\mathrm{RTP}_{\mathcal{A}}$. For this reason, in the literature $\tau_{\mathcal{A}}(\varepsilon)$ is often omitted, and one speaks of the *transition semigroup* instead of the *transition monoid*. We, however, find it quite convenient to have a special element of the semigroup as the "initial" element of the semigroup, and it will become useful later on when we consider a deterministic automaton constructed from the transition monoid.

Definition 3 *For a transition profile t over $\mathcal{A}$, we define the language*

$$\mathrm{L}_*(t) \coloneqq \tau_{\mathcal{A}}^{-1}(t) = \{u \in \Sigma^* \mid \tau_{\mathcal{A}}(u) = t\}.$$

For a fixed automaton $\mathcal{A}$, the nonempty languages $\mathrm{L}_*(t)$ form a partition of the set Σ^* of all words into finitely many distinct classes, for there are only finitely many transition profiles over $\mathcal{A}$.

3.2.1 The Transition Monoid Automaton

The transition monoid of $\mathcal{A}$ can be represented by a deterministic finite automaton that we call the *transition monoid automaton*. Each state of this automaton corresponds to a reachable transition profile, the initial state is $\tau_{\mathcal{A}}(\varepsilon)$, and the a-successor of a state t is obtained as $t \cdot \tau_{\mathcal{A}}(a)$. We

can construct the transition monoid automaton by performing a breadth-first search: we start with the initial state $\tau_{\mathcal{A}}(\varepsilon)$ and for each state t and letter $a \in \Sigma$ we spawn a new state $t \cdot \tau_{\mathcal{A}}(a)$, until the automaton is complete.

Definition 4 *Given an automaton $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$, we define $\mathrm{TMA}_{\mathcal{A}} = (\tilde{Q}, \Sigma, \tilde{q_0}, \tilde{\delta}, \emptyset)$ with $\tilde{Q} = \mathrm{RTP}_{\mathcal{A}}$, and $\tilde{q_0} = \tau_{\mathcal{A}}(\varepsilon)$, and $\tilde{\delta}(\tilde{q_0}, a) = \tau_{\mathcal{A}}(a)$ and $\tilde{\delta}(t, a) = t \cdot \tau_{\mathcal{A}}(a)$ for every $a \in \Sigma$ and $t \in \mathrm{RTP}_{\mathcal{A}}$.*

Note that $\mathrm{TMA}_{\mathcal{A}}$ is normalized, since there is no incoming transition into $\tau_{\mathcal{A}}(\varepsilon)$. Furthermore, $\mathrm{TMA}_{\mathcal{A}}$ does not have any final states. By induction on the length of words, one can show that $\tilde{\delta}^*(\tilde{q_0}, w) = \tau_{\mathcal{A}}(w)$ and from that it follows that the language recognized by $\mathrm{TMA}_{\mathcal{A}}$ with final state set $F = \{t\}$ is exactly $\mathrm{L}_*(t)$. Later on we will instantiate $\mathrm{TMA}_{\mathcal{A}}$ with different sets of final states, depending on the language we want to accept.

Corollary 5 *For each $t \in \mathrm{TP}_{\mathcal{A}}$, the set $\mathrm{L}_*(t)$ is a regular language.*

Example *Figure 3.3 shows the transition modoid automaton for $\mathcal{A}_{\mathrm{ex}}$. Note that it has 19 states and exactly one sink, namely $\tau_{\mathcal{A}_{\mathrm{ex}}}(bb)$.*

3.2.2 The Sequential Lemma

Büchi's proof is based on two key observations. The first observation is formalized in the following lemma. It states that each infinite word can be decomposed into finite segments such that the induced sequence of transition profiles is of the form st^ω. One can even restrict the transition profiles to those which satisfy the equations $t \cdot t = t$ (so t is idempotent) and $s \cdot t = s$.

Lemma 6 (Sequential Lemma) *Let $\mathcal{A}$ be an automaton. For every $\alpha \in \Sigma^\omega$ there is a decomposition of α as $\alpha = uv_1v_2\cdots$ with $u, v_i \in \Sigma^+$ and reachable transition profiles $s, t \in \mathrm{RTP}_{\mathcal{A}}$ such that*

1. *$\tau_{\mathcal{A}}(u) = s$,*
2. *$\tau_{\mathcal{A}}(v_i) = t$ for every $i \geq 1$,*
3. *$t \cdot t = t$, and*
4. *$s \cdot t = s$.*

Proof This proof uses Ramsey's Theorem (Theorem 1). Let $\alpha \in \Sigma^\omega$ be an infinite word. Then each proper factor $\alpha[i, j)$ of α (with $i < j$) is a word from Σ^+. Construct the complete infinite graph with node set $\mathbb{N}$

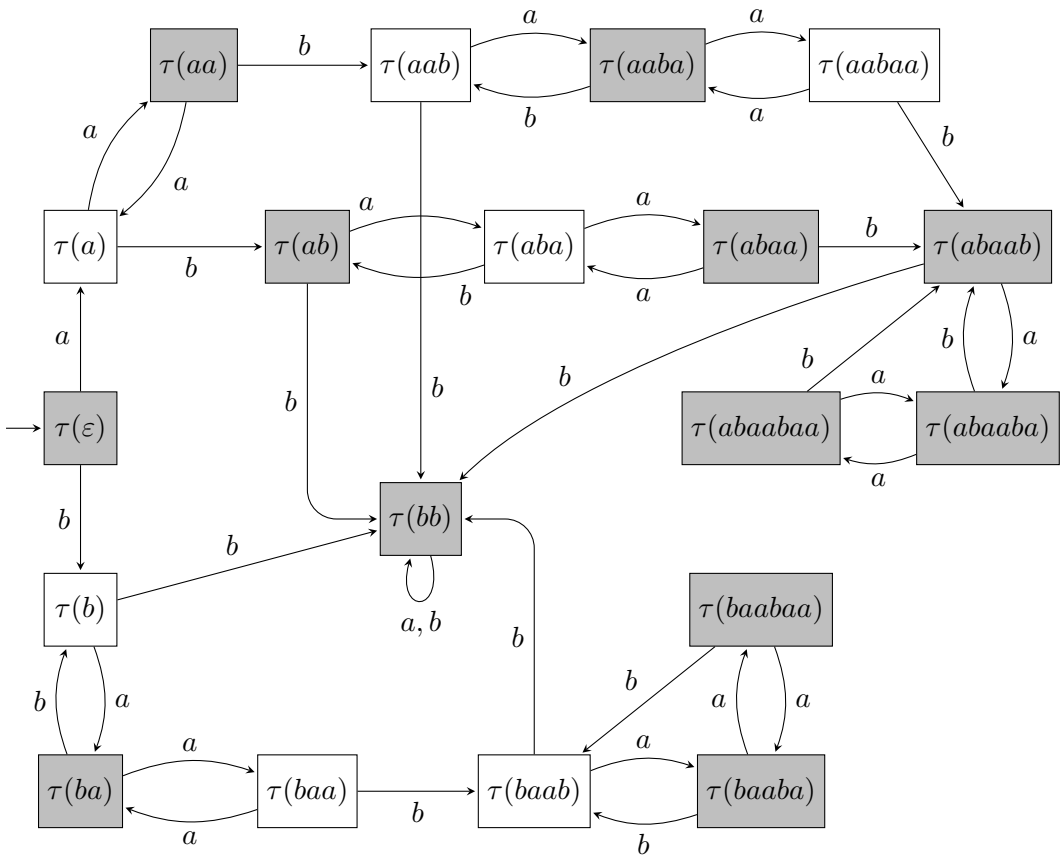


Figure 3.3: The transition monoid automaton of $\mathcal{A}_{\text{ex}}$ with 19 states. Idempotent elements are colored gray.

and coloring function $f(\{i,j\}) = \tau_{\mathcal{A}}(\alpha[i,j))$ for $i < j$. Since there are only finitely many transition profiles, the coloring is finite and we apply Ramsey's Theorem (with $\Gamma = \mathbb{N}$, $r = 2$, and $\mu = |\mathrm{RTP}_{\mathcal{A}}|$). Then there is an infinite subset $\Delta = \{k_0, k_1, k_2, \ldots\} \subseteq \mathbb{N}$ (w.l.o.g. $k_0 < k_1 < k_2 < \ldots$) such that for all $k_i, k_j \in \Delta$ with $k_i < k_j$ the values $\tau_{\mathcal{A}}(\alpha[k_i, k_j))$ are identical. Call this value t.

It holds $\tau_{\mathcal{A}}(\alpha[k_0, k_1)) = \tau_{\mathcal{A}}(\alpha[k_1, k_2)) = \tau_{\mathcal{A}}(\alpha[k_0, k_2))$. So $t \cdot t = \tau_{\mathcal{A}}(\alpha[k_0, k_1) \cdot \alpha[k_1, k_2)) = t$. Furthermore define $s' = \tau_{\mathcal{A}}(\alpha[0, k_0))$ and $s = s' \cdot t$. Then $s \cdot t = s' \cdot t \cdot t = s' \cdot t = s$. With $u = \alpha[0, k_1)$ and $v_i = \alpha[k_i, k_{i+1})$ the claim follows. □

In order to work with transition profiles fulfilling the conditions of the Sequential Lemma, we define

$$\text{s-t-Pairs} \coloneqq \{\langle s,t\rangle \in {\mathrm{RTP}_{\mathcal{A}}}^2 \mid s \cdot t = s,\ t \cdot t = t\} \text{ and}$$
$$\mathrm{L}_\omega(\langle s,t\rangle) \coloneqq \mathrm{L}_*(s) \cdot \mathrm{L}_*(t)^\omega \text{ for } \langle s,t\rangle \in \text{s-t-Pairs}.$$

The Sequential Lemma states that the languages $\mathrm{L}_\omega(\langle s,t\rangle)$ cover the set of all infinite words:

$$\Sigma^\omega = \bigcup \{\mathrm{L}_\omega(\langle s,t\rangle) \mid \langle s,t\rangle \in \text{s-t-Pairs}\}.$$

Idempotent TPs
$\tau_{\mathcal{A}_{\mathrm{ex}}}(aa)$
$\tau_{\mathcal{A}_{\mathrm{ex}}}(ab)$
$\tau_{\mathcal{A}_{\mathrm{ex}}}(ba)$
$\tau_{\mathcal{A}_{\mathrm{ex}}}(bb)$
$\tau_{\mathcal{A}_{\mathrm{ex}}}(aaba)$
$\tau_{\mathcal{A}_{\mathrm{ex}}}(abaa)$
$\tau_{\mathcal{A}_{\mathrm{ex}}}(abaab)$
$\tau_{\mathcal{A}_{\mathrm{ex}}}(baaba)$
$\tau_{\mathcal{A}_{\mathrm{ex}}}(abaaba)$
$\tau_{\mathcal{A}_{\mathrm{ex}}}(baabaa)$
$\tau_{\mathcal{A}_{\mathrm{ex}}}(abaabaa)$

Table 3.3: The 11 idempotent transition profiles of $\mathcal{A}_{\mathrm{ex}}$, excluding $\tau_{\mathcal{A}_{\mathrm{ex}}}(\varepsilon)$.

Accepting s-t-Pairs
$\langle \tau_{\mathcal{A}_{\text{ex}}}(abaab), \tau_{\mathcal{A}_{\text{ex}}}(abaab) \rangle$
$\langle \tau_{\mathcal{A}_{\text{ex}}}(abaaba), \tau_{\mathcal{A}_{\text{ex}}}(baaba) \rangle$
$\langle \tau_{\mathcal{A}_{\text{ex}}}(abaaba), \tau_{\mathcal{A}_{\text{ex}}}(abaaba) \rangle$
$\langle \tau_{\mathcal{A}_{\text{ex}}}(abaabaa), \tau_{\mathcal{A}_{\text{ex}}}(baabaa) \rangle$
$\langle \tau_{\mathcal{A}_{\text{ex}}}(abaabaa), \tau_{\mathcal{A}_{\text{ex}}}(abaabaa) \rangle$

(a) The 5 accepting s-t-Pairs of $\mathcal{A}_{\text{ex}}$.

Rejecting s-t-Pairs
$\langle \tau_{\mathcal{A}_{\text{ex}}}(a), \tau_{\mathcal{A}_{\text{ex}}}(aa) \rangle$
$\langle \tau_{\mathcal{A}_{\text{ex}}}(aa), \tau_{\mathcal{A}_{\text{ex}}}(aa) \rangle$
$\langle \tau_{\mathcal{A}_{\text{ex}}}(ba), \tau_{\mathcal{A}_{\text{ex}}}(aa) \rangle$
$\langle \tau_{\mathcal{A}_{\text{ex}}}(bb), \tau_{\mathcal{A}_{\text{ex}}}(aa) \rangle$
$\langle \tau_{\mathcal{A}_{\text{ex}}}(aba), \tau_{\mathcal{A}_{\text{ex}}}(aa) \rangle$
$\langle \tau_{\mathcal{A}_{\text{ex}}}(baa), \tau_{\mathcal{A}_{\text{ex}}}(aa) \rangle$
$\langle \tau_{\mathcal{A}_{\text{ex}}}(aaba), \tau_{\mathcal{A}_{\text{ex}}}(aa) \rangle$
$\langle \tau_{\mathcal{A}_{\text{ex}}}(abaa), \tau_{\mathcal{A}_{\text{ex}}}(aa) \rangle$
$\langle \tau_{\mathcal{A}_{\text{ex}}}(aabaa), \tau_{\mathcal{A}_{\text{ex}}}(aa) \rangle$
$\langle \tau_{\mathcal{A}_{\text{ex}}}(baaba), \tau_{\mathcal{A}_{\text{ex}}}(aa) \rangle$
$\langle \tau_{\mathcal{A}_{\text{ex}}}(abaaba), \tau_{\mathcal{A}_{\text{ex}}}(aa) \rangle$
$\langle \tau_{\mathcal{A}_{\text{ex}}}(baabaa), \tau_{\mathcal{A}_{\text{ex}}}(aa) \rangle$
$\langle \tau_{\mathcal{A}_{\text{ex}}}(abaabaa), \tau_{\mathcal{A}_{\text{ex}}}(aa) \rangle$
$\langle \tau_{\mathcal{A}_{\text{ex}}}(b), \tau_{\mathcal{A}_{\text{ex}}}(ab) \rangle$
$\langle \tau_{\mathcal{A}_{\text{ex}}}(ab), \tau_{\mathcal{A}_{\text{ex}}}(ab) \rangle$
$\langle \tau_{\mathcal{A}_{\text{ex}}}(bb), \tau_{\mathcal{A}_{\text{ex}}}(ab) \rangle$
$\langle \tau_{\mathcal{A}_{\text{ex}}}(aab), \tau_{\mathcal{A}_{\text{ex}}}(ab) \rangle$
$\langle \tau_{\mathcal{A}_{\text{ex}}}(baab), \tau_{\mathcal{A}_{\text{ex}}}(ab) \rangle$
$\langle \tau_{\mathcal{A}_{\text{ex}}}(abaab), \tau_{\mathcal{A}_{\text{ex}}}(ab) \rangle$
$\langle \tau_{\mathcal{A}_{\text{ex}}}(ba), \tau_{\mathcal{A}_{\text{ex}}}(ba) \rangle$
$\langle \tau_{\mathcal{A}_{\text{ex}}}(bb), \tau_{\mathcal{A}_{\text{ex}}}(ba) \rangle$
$\langle \tau_{\mathcal{A}_{\text{ex}}}(aba), \tau_{\mathcal{A}_{\text{ex}}}(ba) \rangle$
$\langle \tau_{\mathcal{A}_{\text{ex}}}(aaba), \tau_{\mathcal{A}_{\text{ex}}}(ba) \rangle$
$\langle \tau_{\mathcal{A}_{\text{ex}}}(baaba), \tau_{\mathcal{A}_{\text{ex}}}(ba) \rangle$
$\langle \tau_{\mathcal{A}_{\text{ex}}}(abaaba), \tau_{\mathcal{A}_{\text{ex}}}(ba) \rangle$

Rejecting s-t-Pairs
$\langle \tau_{\mathcal{A}_{\text{ex}}}(bb), \tau_{\mathcal{A}_{\text{ex}}}(bb) \rangle$
$\langle \tau_{\mathcal{A}_{\text{ex}}}(ba), \tau_{\mathcal{A}_{\text{ex}}}(aaba) \rangle$
$\langle \tau_{\mathcal{A}_{\text{ex}}}(bb), \tau_{\mathcal{A}_{\text{ex}}}(aaba) \rangle$
$\langle \tau_{\mathcal{A}_{\text{ex}}}(aba), \tau_{\mathcal{A}_{\text{ex}}}(aaba) \rangle$
$\langle \tau_{\mathcal{A}_{\text{ex}}}(aaba), \tau_{\mathcal{A}_{\text{ex}}}(aaba) \rangle$
$\langle \tau_{\mathcal{A}_{\text{ex}}}(baaba), \tau_{\mathcal{A}_{\text{ex}}}(aaba) \rangle$
$\langle \tau_{\mathcal{A}_{\text{ex}}}(abaaba), \tau_{\mathcal{A}_{\text{ex}}}(aaba) \rangle$
$\langle \tau_{\mathcal{A}_{\text{ex}}}(bb), \tau_{\mathcal{A}_{\text{ex}}}(abaa) \rangle$
$\langle \tau_{\mathcal{A}_{\text{ex}}}(baa), \tau_{\mathcal{A}_{\text{ex}}}(abaa) \rangle$
$\langle \tau_{\mathcal{A}_{\text{ex}}}(abaa), \tau_{\mathcal{A}_{\text{ex}}}(abaa) \rangle$
$\langle \tau_{\mathcal{A}_{\text{ex}}}(aabaa), \tau_{\mathcal{A}_{\text{ex}}}(abaa) \rangle$
$\langle \tau_{\mathcal{A}_{\text{ex}}}(baabaa), \tau_{\mathcal{A}_{\text{ex}}}(abaa) \rangle$
$\langle \tau_{\mathcal{A}_{\text{ex}}}(abaabaa), \tau_{\mathcal{A}_{\text{ex}}}(abaa) \rangle$
$\langle \tau_{\mathcal{A}_{\text{ex}}}(bb), \tau_{\mathcal{A}_{\text{ex}}}(abaab) \rangle$
$\langle \tau_{\mathcal{A}_{\text{ex}}}(baab), \tau_{\mathcal{A}_{\text{ex}}}(abaab) \rangle$
$\langle \tau_{\mathcal{A}_{\text{ex}}}(bb), \tau_{\mathcal{A}_{\text{ex}}}(baaba) \rangle$
$\langle \tau_{\mathcal{A}_{\text{ex}}}(baaba), \tau_{\mathcal{A}_{\text{ex}}}(baaba) \rangle$
$\langle \tau_{\mathcal{A}_{\text{ex}}}(bb), \tau_{\mathcal{A}_{\text{ex}}}(abaaba) \rangle$
$\langle \tau_{\mathcal{A}_{\text{ex}}}(baaba), \tau_{\mathcal{A}_{\text{ex}}}(abaaba) \rangle$
$\langle \tau_{\mathcal{A}_{\text{ex}}}(bb), \tau_{\mathcal{A}_{\text{ex}}}(baabaa) \rangle$
$\langle \tau_{\mathcal{A}_{\text{ex}}}(baabaa), \tau_{\mathcal{A}_{\text{ex}}}(baabaa) \rangle$
$\langle \tau_{\mathcal{A}_{\text{ex}}}(bb), \tau_{\mathcal{A}_{\text{ex}}}(abaabaa) \rangle$
$\langle \tau_{\mathcal{A}_{\text{ex}}}(baabaa), \tau_{\mathcal{A}_{\text{ex}}}(abaabaa) \rangle$

(b) The 48 rejecting s-t-Pairs of $\mathcal{A}_{\text{ex}}$.

Table 3.4: The set of all s-t-Pairs of $\mathcal{A}_{\text{ex}}$.

3.2.3 A Separation Lemma

The second key observation for Büchi's proof concerns the language containment of the basic building blocks $\mathrm{L}_\omega(\langle s,t\rangle)$ in the ω-languages $\mathrm{L}_\omega(\mathcal{A})$ and $\Sigma^\omega \setminus \mathrm{L}_\omega(\mathcal{A})$.

Given an arbitrary decomposition of an infinite word α into finite segments, the corresponding sequence of transition profiles contains all the information on possible runs of $\mathcal{A}$ on α relevant for acceptance of α. This means that if two infinite words are both in $\mathrm{L}_\omega(\langle s,t\rangle)$, either both are accepted by $\mathcal{A}$ or both are rejected. This is shown in the following lemma.

Lemma 7 *Let s and t be transition profiles over $\mathcal{A}$. Then either $\mathrm{L}_\omega(\langle s,t\rangle) \cap \mathrm{L}_\omega(\mathcal{A}) = \emptyset$, or $\mathrm{L}_\omega(\langle s,t\rangle) \subseteq \mathrm{L}_\omega(\mathcal{A})$.*

Proof If $\mathrm{L}_*(s) \cdot \mathrm{L}_*(t)^\omega \cap \mathrm{L}_\omega(\mathcal{A})$ is not empty, then there is a word α which lies in both sets. Then α can be decomposed as $\alpha = uv_1v_2\cdots$ with $u \in \mathrm{L}_*(s)$ and all v_i being in $\mathrm{L}_*(t)$. Because the word α is in $\mathrm{L}_\omega(\mathcal{A})$, there must be an accepting run ρ of $\mathcal{A}$ on α. Consider the form of this run ρ to be $q_0 \xrightarrow{u} q_1 \xrightarrow{v_1} q_2 \xrightarrow{v_2} q_3 \cdots$. This run visits infinitely often an accepting state. So for infinitely many i it holds $q_i \xrightarrow[F]{v_i} q_{i+1}$.

Now let β be any word in $\mathrm{L}_*(s) \cdot \mathrm{L}_*(t)^\omega$. Then β can also be decomposed as $\beta = u'v_1'v_2'\cdots$ with $u' \in \mathrm{L}_*(s)$ and all v_i' being in $\mathrm{L}_*(t)$. We have $\tau_\mathcal{A}(u) = \tau_\mathcal{A}(u')$ and $\tau_\mathcal{A}(v_i) = \tau_\mathcal{A}(v_i')$ for all $i \geq 1$. Then there is a run ρ' of $\mathcal{A}$ on β of the form $q_0 \xrightarrow{u'} q_1 \xrightarrow{v_1'} q_2 \xrightarrow{v_2'} q_3 \cdots$. It holds $q_i \xrightarrow[F]{v_i'} q_{i+1}$ for the very same i as above. So ρ' is an accepting run and $\beta \in \mathrm{L}_\omega(\mathcal{A})$. □

This separation lemma allows us to divide accepting and rejecting s-t-pairs into disjoint sets:

$$\mathrm{Accept}_{\mathrm{s,t}} \coloneqq \{\langle s,t\rangle \in \text{s-t-Pairs} \mid \mathrm{L}_\omega(\langle s,t\rangle) \subseteq \mathrm{L}_\omega(\mathcal{A})\} \text{ and}$$
$$\mathrm{Reject}_{\mathrm{s,t}} \coloneqq \{\langle s,t\rangle\rangle \in \text{s-t-Pairs} \mid \mathrm{L}_\omega(\langle s,t\rangle) \cap \mathrm{L}_\omega(\mathcal{A}) = \emptyset\}.$$

Example *The sets $\mathrm{Accept}_{\mathrm{s,t}}$ and $\mathrm{Reject}_{\mathrm{s,t}}$ for the automaton $\mathcal{A}_{\mathrm{ex}}$ are shown in Table 3.4a and Table 3.4b, respectively.*

From Lemmas 6 and 7 we can conclude the following characterizations of $\mathrm{L}_\omega(\mathcal{A})$ and its complement.

$$\mathrm{L}_\omega(\mathcal{A}) = \bigcup \{\mathrm{L}_\omega(\langle s,t\rangle) \mid \langle s,t\rangle \in \mathrm{Accept}_{\mathrm{s,t}}\},$$
$$\Sigma^\omega \setminus \mathrm{L}_\omega(\mathcal{A}) = \bigcup \{\mathrm{L}_\omega(\langle s,t\rangle) \mid \langle s,t\rangle \in \mathrm{Reject}_{\mathrm{s,t}}\}.$$

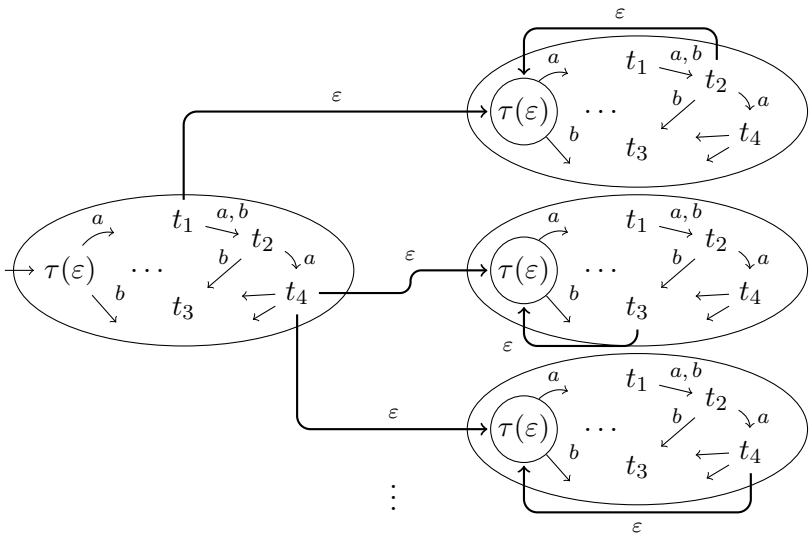


Figure 3.4: A layout of a complement automaton with $\text{Reject}_{\text{s,t}} = \{\langle t_1, t_2\rangle, \langle t_4, t_3\rangle, \langle t_4, t_4\rangle, \ldots\}$.

Naturally, we are interested in the latter of both equations. The right side of both equations are finite unions, as there are only finitely many transition profiles over $\mathcal{A}$. By Corollary 5 each $\text{L}_\omega(\langle s, t\rangle)$ is a regular ω-language. As a consequence, $\Sigma^\omega \setminus L$ is a finite union of regular ω-languages, so there must be a Büchi automaton recognizing this complement. This conclusion is called Büchi's Theorem.

Theorem 8 (Büchi) *For each ω-regular language L, the complement $\Sigma^\omega \setminus L$ is also ω-regular.*

3.2.4 Assembling the Complement Automaton

We are now able to assemble a complement automaton for any given Büchi automaton $\mathcal{A}$. For this, we make use of the transition monoid automaton, more precisely we take several copies of $\text{TMA}_\mathcal{A}$ with different sets of final states.

As we have seen in Section 3.2.1, for every transition profile t there is an automaton recognizing $\text{L}_*(t)$. In particular, for each rejecting s-t-pair (s, t) we have automata for the languages $\text{L}_*(s)$ and $\text{L}_*(t)$. We modify the second automaton (recognizing $\text{L}_*(t)$) to recognize $\text{L}_*(t)^\omega$ by adding an ε-transition from its final state to its initial state and then making the

final state non-final and making the initial state final. This is possible only because $\mathrm{TMA}_{\mathcal{A}}$ is normalized. After that, we connect the final state of the first automaton (recognizing $\mathrm{L}_*(s)$) with the initial state of the automaton recognizing $\mathrm{L}_*(t)^\omega$, again using an ε-transition. We obtain an automaton recognizing $\mathrm{L}_\omega(\langle s,t\rangle) = \mathrm{L}_*(s) \cdot \mathrm{L}_*(t)^\omega$.

An automaton recognizing the entire complement can now easily be obtained by introducing a new initial state and connecting this state with all the former initial states of all automata for $\mathrm{L}_\omega(\langle s,t\rangle)$. But there is a simpler and more efficient method described in Sistla, Vardi, and Wolper [67]: it is possible to reuse the same automaton for each of the different languages $\mathrm{L}_*(s)$ by connecting the different final states of the first automaton with the corresponding initial states of the second automaton. This construction is depicted in Figure 3.4.

We refer to the first part of the automaton, which reads a word from $\mathrm{L}_*(s)$, as the *initial part*, and to the second part of the automaton, which reads a word from $\mathrm{L}_*(t)^\omega$, as the *looping part*. The ε-transitions used in these illustrations can now be eliminated by standard techniques. Obviously, one can eliminate each ε-transition from the initial part to the looping part without adding a new state. For eliminating the ε-transitions within the looping part we don't need any additional states either, as each looping part consists of a normalized automaton.

The complement automaton consists of at most 3^{n^2} Büchi automata in the looping part, each with at most 3^{n^2} states, plus an additional initial automaton component of 3^{n^2} states. This results in a complement automaton with at most $3^{n^2} + 9^{n^2}$ states.

Example *In our running example, the complement automaton has 11 different idempotent transition profiles to consider for the looping part (see Table 3.3). This results in a complement automaton with exactly* $19 + 11 \cdot 19 = 228$ *states. The list of rejecting pairs in Table 3.4b defines which states of the initial part are connected to which automata of the looping part.*

3.3 Improved Ramsey-based Complementation

The complementation method described in the last section is the one that was established by Büchi [12] and improved by Sistla, Vardi, and Wolper [67]. Its upper bound on the state space is $2^{\mathcal{O}(n^2)}$, and we will not improve this upper bound in this section. Rather, we introduce three significant ideas to reduce the state space of the complement automaton in practice.

We call these ideas to be "significant", because only by the use of these three improvements, we will be able to establish a connection to the complement automaton of Section 3.4, which in turn possesses a $2^{\mathcal{O}(n \log n)}$ upper bound.

3.3.1 Subset Construction

When taking a look at the initial part of the complement automaton, we notice that in this part the infinite behavior of the input word is not yet significant for the automaton; it is only the finite behavior that matters here. Nevertheless, the transition profiles in this part distinguish between final states and nonfinal states, which is needless. What is required there is to distinguish between states that are reachable and states that are not reachable from the initial state. Therefore, for the initial part, we drop the generation of transition profiles and replace them with a simple subset construction.

Let $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ be an automaton. The set $\mathrm{RSC}_\mathcal{A} \coloneqq \{P \subseteq Q \mid \exists u \in \Sigma^* \colon \delta^*(\{q_0\}, u) = P\}$ describes the set of states that are reachable by a standard subset construction. For a set of states $P \subseteq Q$ define $\mathrm{L}_*(P) \coloneqq \{u \in \Sigma^* \mid \delta^*(\{q_0\}, u) = P\}$ to be the set of all words with which from q_0 one can reach all states in P and not any other states. For a transition profile t over $\mathcal{A}$, define $t(P) \coloneqq \{q \in Q \mid \exists p \in P \colon p \to_t q\}$ to be the set of all states which are reachable from P by reading a word u with $\tau_\mathcal{A}(u) = t$. Then we have $t(s(P)) = (s \cdot t)(P)$ for all $s, t \in \mathrm{TP}_\mathcal{A}$ and $P \subseteq Q$. With these definition, we reformulate the Sequential Lemma for the initial part of the complement automaton, which corresponds to the s in the decomposition st^ω.

Lemma 9 *Let $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ be an automaton. For every $\alpha \in \Sigma^\omega$ there is a decomposition of α as $\alpha = uv_1v_2\cdots$ with $u, v_i \in \Sigma^+$, a set $P \subseteq Q$ and a reachable transition profile $t \in \mathrm{RTP}_\mathcal{A}$ such that*

1. *$u \in \mathrm{L}_*(P)$,*
2. *$\tau_\mathcal{A}(v_i) = t$ for every $i \geq 1$,*
3. *$t \cdot t = t$, and*
4. *$t(P) = P$.*

Proof We apply Lemma 6 and obtain a decomposition $\alpha = uv_1v_2\cdots$ and transition profiles s and t with properties 2. and 3., as well as $\tau_\mathcal{A}(u) = s$ and $s \cdot t = s$. We let $P \coloneqq s(\{q_0\})$. Then all words in $\mathrm{L}_*(s)$ lead from q_0

to P and nowhere else, in particular the word $u \in \mathrm{L}_*(s)$. So $u \in \mathrm{L}_*(P)$. Furthermore $t(P) = t(s(\{q_0\})) = (s \cdot t)(\{q_0\}) = s(\{q_0\}) = P$. □

With $\mathrm{RSC}_\mathcal{A}$ describing the reachable states of the subset construction and $\mathrm{RTP}_\mathcal{A}$ describing the reachable transition profiles, we are now able to move the construction from s-t-pairs to P-t-pairs by defining

$$\text{P-t-Pairs} \coloneqq \{\langle P, t\rangle \in \mathrm{RSC}_\mathcal{A} \times \mathrm{RTP}_\mathcal{A} \mid t(P) = P,\ t \cdot t = t\} \text{ and}$$
$$\mathrm{L}_\omega(\langle P, t\rangle) \coloneqq \mathrm{L}_*(P) \cdot \mathrm{L}_*(t)^\omega \text{ for } \langle P, t\rangle \in \text{P-t-Pairs}.$$

Then Lemma 9 shows that the P-t-pairs again cover the set of all infinite words:

$$\Sigma^\omega = \bigcup \{\mathrm{L}_\omega(\langle P, t\rangle) \mid \langle P, t\rangle \in \text{P-t-Pairs}\}.$$

Again each P-t-pair represents an ω-language which is completely inside $\mathrm{L}_\omega(\mathcal{A})$ or completely outside of it.

Lemma 10 *Let $P \subseteq Q$ and t be a transition profile over $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$. Then either $\mathrm{L}_\omega(\langle P, t\rangle) \cap \mathrm{L}_\omega(\mathcal{A}) = \emptyset$, or $\mathrm{L}_\omega(\langle P, t\rangle) \subseteq \mathrm{L}_\omega(\mathcal{A})$.*

The proof is the same as for Lemma 7 but $\mathrm{L}_*(s)$ replaced by $\mathrm{L}_*(P)$.

Proof If $\mathrm{L}_*(P) \cdot \mathrm{L}_*(t)^\omega \cap \mathrm{L}_\omega(\mathcal{A})$ is not empty, then there is a word α which lies in both sets. Then α can be decomposed as $\alpha = uv_1v_2\cdots$ with $u \in \mathrm{L}_*(P)$ and all v_i being in $\mathrm{L}_*(t)$. Because the word α is in L, there must be an accepting run of $\mathcal{A}$ on α. Consider the form of this run to be $q_0 \xrightarrow{u} q_1 \xrightarrow{v_1} q_2 \xrightarrow{v_2} q_3 \cdots$. Note that $q_1 \in P$. This run visits infinitely often an accepting state. So for infinitely many i it holds $q_i \xrightarrow[F]{v_i} q_{i+1}$.

Now let β be any word in $\mathrm{L}_*(P) \cdot \mathrm{L}_*(t)^\omega$. Then β can also be decomposed as $\beta = u'v_1'v_2'\cdots$ with $u' \in \mathrm{L}_*(P)$ and all v_i' being in $\mathrm{L}_*(t)$. We have $u' \in \mathrm{L}_*(P)$ and $\tau_\mathcal{A}(v_i) = \tau_\mathcal{A}(v_i')$ for all $i \geq 1$. Then there is a run ρ' of $\mathcal{A}$ on β of the form $q_0 \xrightarrow{u'} q_1 \xrightarrow{v_1'} q_2 \xrightarrow{v_2'} q_3 \cdots$. It holds $q_i \xrightarrow[F]{v_i'} q_{i+1}$ for the very same i as above. So ρ' is an accepting run and $\beta \in \mathrm{L}_\omega(\mathcal{A})$. □

We adapt the definition of the set of accepting and rejecting pairs to

$$\mathrm{Accept}_{\mathrm{P,t}} \coloneqq \{\langle P, t\rangle \in \text{P-t-Pairs} \mid \mathrm{L}_\omega(\langle P, t\rangle) \subseteq \mathrm{L}_\omega(\mathcal{A})\} \text{ and}$$
$$\mathrm{Reject}_{\mathrm{P,t}} \coloneqq \{\langle P, t\rangle \in \text{P-t-Pairs} \mid \mathrm{L}_\omega(\langle P, t\rangle) \cap \mathrm{L}_\omega(\mathcal{A}) = \emptyset\}.$$

We assemble the new complement automaton by retaining all the automata in the looping part of the old complement automaton and by replacing the initial part by a simple subset automaton, which keeps track of the set of reachable states.

Definition 11 *Given an automaton* $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$*, the subset automaton* $\mathrm{PA}_\mathcal{A} = (\mathrm{RSC}_\mathcal{A}, \Sigma, \{q_0\}, \delta^*, F')$ *is defined by* $\mathrm{RSC}_\mathcal{A} \coloneqq \{P \subseteq Q \mid \exists u \in \Sigma^* \colon \delta^*(\{q_0\}, u) = P\}$ *and* $F' \coloneqq \{P \in \mathrm{RSC}_\mathcal{A} \mid P \cap F \neq \emptyset\}$.

In the new complement automaton, a set $P \in \mathrm{RSC}_\mathcal{A}$ is connected to the copy of $\mathrm{TMA}_\mathcal{A}$ for the transition profile t if $\langle P, t\rangle \in \mathrm{Reject}_{\mathrm{P,t}}$.

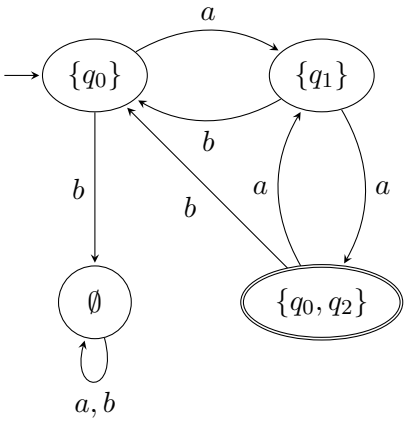


Figure 3.5: The subset construction $\mathrm{PA}_{\mathcal{A}_{\mathrm{ex}}}$ of $\mathcal{A}_{\mathrm{ex}}$.

Example *Let us remark on the size of the new complement automaton, after this first simple reduction step. As we can see in Table 3.5b, though the number of rejecting P-t-pairs was reduced severely from 48 to 17, the number of different idempotent transition profiles to consider for the looping part has not changed; it is still* 11*. So the new automaton does not have a smaller number of TMAs in the looping part. The initial part, however, is reduced from 19 to 4 states. This results in a complement automaton with* $4 + 11 \cdot 19 = 213$ *states. The list of rejecting pairs in Table 3.5b now defines which states of the initial part are connected to which automata of the looping part.*

The Empty Transition Profile

Note that, according to the above definition of P-t-Pairs, all pairs of the form $\langle \emptyset, t\rangle$ with t idempotent and all pairs of the form $\langle P, t_\emptyset\rangle$ where $t_\emptyset$ is the empty transition profile $t_\emptyset = (\emptyset, \emptyset)$ are in $\mathrm{Reject}_{\mathrm{P,t}}$. Thus, all ω-words that are described by such a pair will be accepted by the complement automaton. However, all these pairs describe ω-words on which no run of $\mathcal{A}$ exists at all and thus have a prefix leading from $\{q_0\}$ to $\emptyset$ in the initial part $\mathrm{PA}_\mathcal{A}$ of the

Accepting P-t-Pairs
$\langle \{q_0\}, \tau_{\mathcal{A}_{\text{ex}}}(abaab) \rangle$
$\langle \{q_1\}, \tau_{\mathcal{A}_{\text{ex}}}(baaba) \rangle$
$\langle \{q_1\}, \tau_{\mathcal{A}_{\text{ex}}}(abaaba) \rangle$
$\langle \{q_0, q_2\}, \tau_{\mathcal{A}_{\text{ex}}}(baabaa) \rangle$
$\langle \{q_0, q_2\}, \tau_{\mathcal{A}_{\text{ex}}}(abaabaa) \rangle$

(a) The 5 accepting P-t-Pairs of $\mathcal{A}_{\text{ex}}$.

Rejecting P-t-Pairs
$\langle \{q_0\}, \tau_{\mathcal{A}_{\text{ex}}}(ab) \rangle$
$\langle \{q_1\}, \tau_{\mathcal{A}_{\text{ex}}}(aa) \rangle$
$\langle \{q_1\}, \tau_{\mathcal{A}_{\text{ex}}}(ba) \rangle$
$\langle \{q_1\}, \tau_{\mathcal{A}_{\text{ex}}}(aaba) \rangle$
$\langle \emptyset, \tau_{\mathcal{A}_{\text{ex}}}(aa) \rangle$
$\langle \emptyset, \tau_{\mathcal{A}_{\text{ex}}}(ab) \rangle$
$\langle \emptyset, \tau_{\mathcal{A}_{\text{ex}}}(ba) \rangle$
$\langle \emptyset, \tau_{\mathcal{A}_{\text{ex}}}(bb) \rangle$
$\langle \emptyset, \tau_{\mathcal{A}_{\text{ex}}}(aaba) \rangle$
$\langle \emptyset, \tau_{\mathcal{A}_{\text{ex}}}(abaa) \rangle$
$\langle \emptyset, \tau_{\mathcal{A}_{\text{ex}}}(abaab) \rangle$
$\langle \emptyset, \tau_{\mathcal{A}_{\text{ex}}}(baaba) \rangle$
$\langle \emptyset, \tau_{\mathcal{A}_{\text{ex}}}(abaaba) \rangle$
$\langle \emptyset, \tau_{\mathcal{A}_{\text{ex}}}(baabaa) \rangle$
$\langle \emptyset, \tau_{\mathcal{A}_{\text{ex}}}(abaabaa) \rangle$
$\langle \{q_0, q_2\}, \tau_{\mathcal{A}_{\text{ex}}}(aa) \rangle$
$\langle \{q_0, q_2\}, \tau_{\mathcal{A}_{\text{ex}}}(abaa) \rangle$

(b) The 17 rejecting P-t-Pairs of $\mathcal{A}_{\text{ex}}$.

Table 3.5: The set of all P-t-Pairs of $\mathcal{A}_{\text{ex}}$.

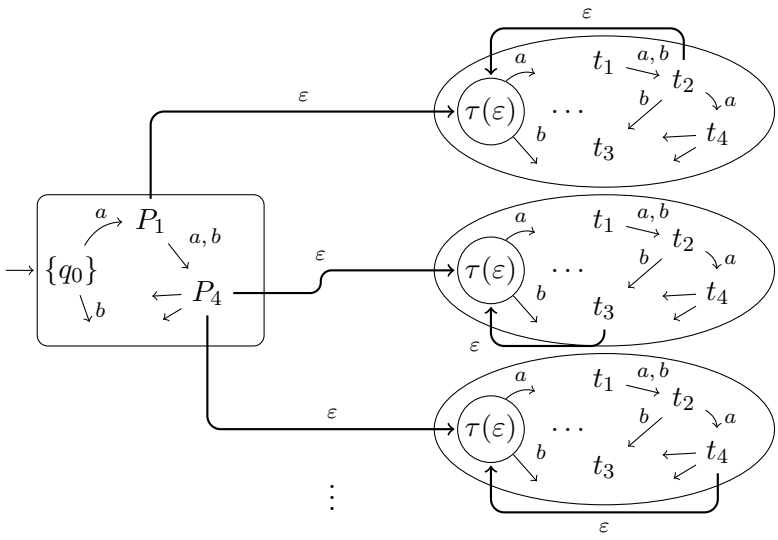


Figure 3.6: A complement with $\text{Reject}_{\text{P,t}} = \{\langle P_1, t_2\rangle, \langle P_4, t_3\rangle, \langle P_4, t_4\rangle, \ldots\}$.

complement automaton. Hence, in our implementation we mark the state $\emptyset$ to be an accepting state and remove all pairs of the form $\langle\emptyset, \ldots\rangle$ from the set of rejecting P-t-pairs. This leads to a simplified complement automaton, in which those ω-words for which there does not exist any run in $\mathcal{A}$ are accepted already early in the initial part of the complement automaton. Removing all pairs of the form $\langle\emptyset, \ldots\rangle$ also removes all pairs of the form $\langle P, t_\emptyset\rangle$, due to the following lemma.

Lemma 12 *Let $\langle P, t\rangle \in$ P-t-Pairs be a P-t-pair. If t is the empty transition profile $t = t_\emptyset$, then it holds $P = \emptyset$.*

Proof If $\langle P, t_\emptyset\rangle$ is in P-t-Pairs for any $P \subseteq Q$, then $t_\emptyset(P) = \emptyset$ and by Lemma 9 we have $t_\emptyset(P) = P$, and therefore $P = \emptyset$. □

Example *In our running example $\mathcal{A}_{\text{ex}}$, reading the finite word bb from any state leads to nothing. Thus the empty transition profile is $t_\emptyset = \tau_{\mathcal{A}_{\text{ex}}}(bb)$. We can see the reduced set of rejecting P-t-pairs of our running example in Table 3.6. Reducing the set of rejecting P-t-pairs also reduces the number of idempotent transition profiles to consider for the looping part from 11 to 5. Thus, we now only have $4 + 5 \cdot 19 = 99$ states in the complement automaton.*

3.3.2 Merging Transition Profiles

The next step towards a smaller complement is to condense the number of transition monoid automata in the looping part. Each of these automata has exactly one final state. Therefore we refer to these automata as *singleton automata*. It seems as an obvious idea to allow not just one state to be final, but to allow an arbitrary subset of the states of a singleton automaton to be marked final. The idea ist that under certain conditions, we can combine several of these singleton automata and obtain a new TMA with a large set of final states. Then, by replacing a set of TMAs with only one TMA, we severely reduce the size of the resulting complement automaton.

More specificly, we merge several TMAs into a new TMA by marking all those states final that were final in one of the original TMAs. The resulting TMA (considered as a $*$-automaton) then accepts the union of all the languages formerly accepted by the singleton automata. Possibly, this resulting TMA can then replace the singleton automata, which are not needed anymore and can be removed from the looping part, resulting in a potentially smaller complement automaton. We cannot guarantee that the new complement automaton really is smaller than the original complement automaton. In the author's opinion, it may on the contrary even grow larger, if more TMAs are generated than singleton automata are removed. However, in all our experiments, the result actually was smaller.

Buckets

The merging cannot be done in an arbitrary fashion, as we will see later. In the following, we state criteria that are sufficient for merging several singleton automata successfully.

Rejecting P-t-Pairs
$\langle \{q_0\}, \tau_{\mathcal{A}_{\text{ex}}}(ab) \rangle$
$\langle \{q_1\}, \tau_{\mathcal{A}_{\text{ex}}}(aa) \rangle$
$\langle \{q_1\}, \tau_{\mathcal{A}_{\text{ex}}}(ba) \rangle$
$\langle \{q_1\}, \tau_{\mathcal{A}_{\text{ex}}}(aaba) \rangle$
$\langle \{q_0, q_2\}, \tau_{\mathcal{A}_{\text{ex}}}(aa) \rangle$
$\langle \{q_0, q_2\}, \tau_{\mathcal{A}_{\text{ex}}}(abaa) \rangle$

Table 3.6: The 6 rejecting P-t-Pairs of $\mathcal{A}_{\text{ex}}$ after removing all pairs of the form $\langle \emptyset, \ldots \rangle$.

Definition 13 *Let $\mathcal{A}$ be an automaton. A set of P-t-pairs is called a* bucket *over $\mathcal{A}$. For a bucket $B = \{\langle P_1, t_1\rangle, \ldots, \langle P_n, t_n\rangle\}$ over $\mathcal{A}$, we define the language $\mathrm{L}_\omega(B) \coloneqq \mathrm{L}^{\mathrm{P}}_*(B) \cdot \left(\mathrm{L}^{\mathrm{t}}_*(B)\right)^\omega$, where*

- $\mathrm{L}^{\mathrm{P}}_*(B) \coloneqq \bigcup_{i=1}^{n} \mathrm{L}_*(P_i)$, *and*
- $\mathrm{L}^{\mathrm{t}}_*(B) \coloneqq \bigcup_{i=1}^{n} \mathrm{L}_*(t_i)$.

By this definition, for any bucket B, it holds $\mathrm{L}_\omega(B) \supseteq \mathrm{L}_\omega(\langle P, t\rangle)$ for each $\langle P, t\rangle \in B$. So by aggregating several P-t-pairs into one bucket, the language accepted by the resulting automaton can only grow larger.

Now we state a condition that allows to merge several rejecting pairs over $\mathcal{A}$ into a bucket B such that $\mathrm{L}_\omega(B)$ has an empty intersection with $\mathrm{L}_\omega(\mathcal{A})$.

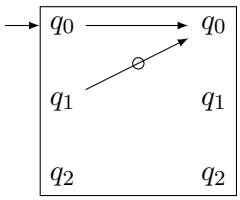
Definition 14 *For a bucket $B = \{\langle P_1, t_1\rangle, \ldots, \langle P_n, t_n\rangle\}$ over $\mathcal{A}$, we define its* join *as the pair $\langle P, t\rangle$ with $P = \bigcup_i P_i$ and $p \to_t q$ iff $\exists i\colon p \to_{t_i} q$, and $p \mathrel{\text{-}\!\!\oslash\!\!\rightarrow}_t q$ iff $\exists i\colon p \mathrel{\text{-}\!\!\oslash\!\!\rightarrow}_{t_i} q$. We say that such a pair $\langle P, t\rangle$* has a lasso *if there is a sequence of states $p_1, \ldots, p_k, \ldots, p_n$, $1 \leq k < n$ such that*

- $p_1 \in P$,
- $p_i \to_t p_{i+1}$ *for all* $1 \leq i < n$,
- $p_k \mathrel{\text{-}\!\!\oslash\!\!\rightarrow}_t p_{k+1}$, *and*
- $p_n \to_t p_k$.

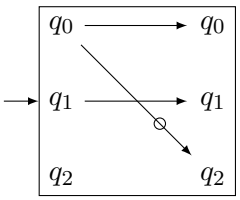
We say that a bucket is mergeable *if its join does not have a lasso.*

For the join $\langle P, t\rangle$ of a bucket, note that P is not necessarily a reachable subset of the state space and that t is not necessarily a reachable transition profile.

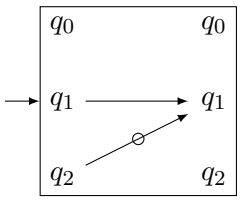
Example *Let us consider the first three P-t-pairs of Table 3.6, namely $p_1 = \langle\{q_0\}, \tau_{\mathcal{A}}(ab)\rangle$, $p_2 = \langle\{q_1\}, \tau_{\mathcal{A}}(aa)\rangle$, and $p_3 = \langle\{q_1\}, \tau_{\mathcal{A}}(ba)\rangle$. These three pairs are depicted in Figures 3.7a to 3.7c. In Figures 3.7d to 3.7f we see the joins of each two of these pairs. Although each two pairs are mergeable, if we merge all three pairs, then we obtain the join depicted in Figure 3.7g, which has a lasso: from q_0 there is an edge to q_2 and from there to q_1 and from there there is an $\mathrel{\text{-}\!\!\oslash\!\!\rightarrow}$-edge back to q_0. So, in general, the "mergeable" relation over buckets is not transitive.*



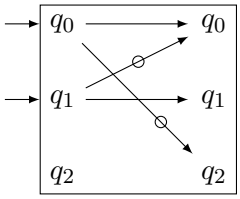
(a) The P-t-pair $p_1 = \langle\{q_0\}, \tau_{\mathcal{A}_{\text{ex}}}(ab)\rangle$.



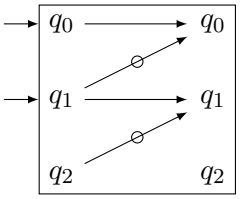
(b) The P-t-pair $p_2 = \langle\{q_1\}, \tau_{\mathcal{A}_{\text{ex}}}(aa)\rangle$.



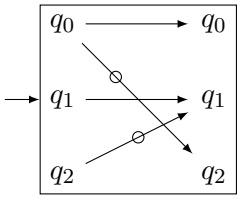
(c) The P-t-pair $p_3 = \langle\{q_1\}, \tau_{\mathcal{A}_{\text{ex}}}(ba)\rangle$.



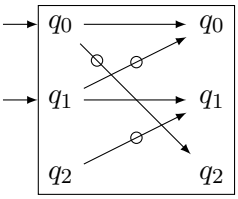
(d) The join of $\{p_1, p_2\}$.



(e) The join of $\{p_1, p_3\}$.



(f) The join of $\{p_2, p_3\}$.



(g) The join of $\{p_1, p_2, p_3\}$.

Figure 3.7: Some P-t-pairs of the automaton $\mathcal{A}_{\text{ex}}$.

As we have mentioned above, the language accepted by the bucket can only grow larger in comparison to the language accepted by the automaton without buckets. Now we prove that we do not grow too large, i.e., that for any bucket over an automaton $\mathcal{A}$, the condition of being mergeable (by the above definition) is indeed a sufficient condition for the resulting automaton to still only recognize the complement of $\mathrm{L}_\omega(\mathcal{A})$.

Lemma 15 *Let B be a mergeable bucket over $\mathcal{A}$. Then $\mathrm{L}_\omega(B) \cap \mathrm{L}_\omega(\mathcal{A}) = \emptyset$.*

Proof Let $B = \{\langle P_1, t_1\rangle, \ldots, \langle P_n, t_n\rangle\}$ and let $\langle P, t\rangle$ be the join of B and let α be a word in $\mathrm{L}_\omega(B)$. Then $\alpha \in \bigcup_{i=1}^{n} \mathrm{L}_*(P_i) \cdot (\bigcup_{i=1}^{n} \mathrm{L}_*(t_i))^\omega$ and there is an infinite sequence $i_0, i_1, i_2, \ldots$ with $1 \leq i_j \leq n$ such that α can be decomposed as $\alpha = uv_1v_2\cdots$ with $u \in \mathrm{L}_*(P_{i_0})$ and $v_j \in \mathrm{L}_*(t_{i_j})$ for each $j \geq 1$.

Assume that $\alpha \in \mathrm{L}_\omega(\mathcal{A})$ and consider an accepting run ρ of $\mathcal{A}$ on α. Consider the form of this run to be $q_0 \xrightarrow{u} q_1 \xrightarrow{v_1} q_2 \xrightarrow{v_2} q_3 \cdots$, and let R be the set of those states which occur infinitely often in this form. Since $\alpha \in \mathrm{L}_\omega(\mathcal{A})$, there is a state $q \in F$ such that q is visited infinitely often in ρ. Now we argue that $\langle P, t\rangle$ has a lasso. We have $q_1 \in P_{i_0}$ and therefore $q_1 \in P$. We have $q_i \rightarrow_t q_{i+1}$ for all $i \geq 1$. Since there are infinitely many accepting states visited in ρ, there must be a state $q_k \in R$ with $q_k \twoheadrightarrow_{t_{i_k}} q_{k+1}$ and therefore $q_k \twoheadrightarrow_t q_{k+1}$. Finally since q_k occurs infinitely often in the above form, there must be a state q_n with $n > k$ and $q_n \rightarrow_{t_{i_n}} q_k$, and therefore $q_n \rightarrow_t q_k$. So $\langle P, t\rangle$ has a lasso and this is a contradiction to the premise that B is mergeable. □

Theorem 16 *Let $\{B_1, \ldots, B_n\}$ be a set of buckets over $\mathcal{A}$ such that each bucket B_i is mergeable, and for each $\langle P, t\rangle \in \mathrm{Reject}_{\mathrm{P,t}}$ there is a bucket B_i with $\langle P, t\rangle \in B_i$. Then $\mathrm{L}_\omega(B_1) \cup \cdots \cup \mathrm{L}_\omega(B_n) = \Sigma^\omega \setminus \mathrm{L}_\omega(\mathcal{A})$.*

Proof Let $\alpha \in \mathrm{L}_\omega(B_i)$ for some $1 \leq i \leq n$. Since B_i is mergeable, by Lemma 15 it follows that $\alpha \notin \mathrm{L}_\omega(\mathcal{A})$.

For the other direction, let $\alpha \in \Sigma^\omega \setminus \mathrm{L}_\omega(\mathcal{A})$. By Lemma 9 and the definition of $\mathrm{Reject}_{\mathrm{P,t}}$, we know that there is a $\langle P, t\rangle \in \mathrm{Reject}_{\mathrm{P,t}}$ such that $\alpha \in \mathrm{L}_\omega(\langle P, t\rangle)$. Then there is a bucket B_i with $\langle P, t\rangle \in B_i$ and since $\mathrm{L}_\omega(B_i) \supseteq \mathrm{L}_\omega(\langle P, t\rangle)$ it follows that $\alpha \in \mathrm{L}_\omega(B_i)$. □

With this result, we now improve the complement construction to use buckets. Given a set $\{B_1, \ldots, B_n\}$ of buckets over $\mathcal{A}$ with properties as in Theorem 16, and automata $\mathcal{A}_i = (Q_i, \Sigma, q_0^i, \delta_i, F_i)$ recognizing the languages $\mathrm{L}_*^{\mathrm{t}}(B_i)$, for each i and for each pair $\langle P, t\rangle \in B_i$, we connect the set P from

the initial $\mathrm{PA}_{\mathcal{A}}$ with the initial state of the automaton $\mathcal{A}_i$ for $\mathrm{L}_*^{\mathrm{t}}(B_i)$, and for each automaton $\mathcal{A}_i$, we connect every final state with the initial state of $\mathcal{A}_i$. This is done formally in the following definition, where we directly connect the automata instead of using ε-transitions.

Definition 17 *Let $\{B_1, \ldots, B_n\}$ be a set of buckets over $\mathcal{A}$ as in Theorem 16 and for each $1 \leq i \leq n$ let $\mathcal{A}_i = (Q_i, \Sigma, q_0^i, \delta_i, F_i)$ be a normalized automaton such that $\mathrm{L}_*(\mathcal{A}_i) = \mathrm{L}_*^{\mathrm{t}}(B_i)$. Then define the automaton $\mathcal{A}' \coloneqq (Q', \Sigma, q_0', \delta', F')$ with*

- $Q' = \mathrm{RSC}_{\mathcal{A}} \cup \bigcup_i Q_i$,
- $q_0' = \{q_0\}$,
- $F' = \{\emptyset\} \cup \{q_0^1, \ldots, q_0^n\}$, *and*
- $\delta'(P, a) = \{\delta^*(P, a)\} \cup \{q \in Q' \mid \exists i(\exists \langle P, t\rangle \in B_i \text{ for some } t \wedge q \in \delta_i(q_0^i, a))\}$ *for $P \subseteq Q$ and $a \in \Sigma$, and*
- $\delta'(q, a) = \begin{cases} \delta_i(q, a) \text{ if } \delta_i(q, a) \cap F_i = \emptyset \\ \{q_0^i\} \cup \delta_i(q, a) \text{ otherwise} \end{cases}$ *for $q \in Q_i$ and $a \in \Sigma$.*

It is easy to see that $\mathrm{L}_\omega(\mathcal{A}') = \mathrm{L}_\omega(B_1) \cup \cdots \cup \mathrm{L}_\omega(B_n)$ and by Theorem 16 we obtain that $\mathcal{A}'$ is an automaton recognizing the complement of $\mathrm{L}_\omega(\mathcal{A})$.

The above definition is parameterized. The automata $\mathcal{A}_i$ for $\mathrm{L}_*^{\mathrm{t}}(B_i)$ are assumed to be given from outside. This will be useful in the next sections, when we replace them by other automata. For now, we can use the transition monoid automaton to generate such an automaton $\mathcal{A}_i$: for a bucket B over $\mathcal{A}$, mark every state t of $\mathrm{TMA}_{\mathcal{A}}$ final for which there is a P-t-pair $\langle P, t\rangle \in B$. We call an automaton generated in such a way a *bucket automaton*.

Definition 18 *Let $B = \{\langle P_1, t_1\rangle, \ldots, \langle P_n, t_n\rangle\}$ be a bucket and let $\mathrm{TMA}_{\mathcal{A}} = (\tilde{Q}, \Sigma, \tilde{q_0}, \tilde{\delta}, \emptyset)$. Define the* bucket automaton $\mathcal{A}^{\mathrm{t}}(B) \coloneqq (\tilde{Q}, \Sigma, \tilde{q_0}, \tilde{\delta}, F_B)$ *with $F_B = \{t \in \mathrm{RTP}_{\mathcal{A}} \mid \exists \langle P_i, t_i\rangle \in B \colon t_i = t\}$.*

Example *In Table 3.7 we see two different bucket distributions obtained from our running example. Each of those buckets is mergeable, and each is "packed" in the following sense. Each bucket B_i cannot be supplemented with any P-t-pair from B_{i+k} for $k \geq 0$ without losing the property of being mergeable. For the bucket distribution of Table 3.7a we obtain a complement*

Bucket B_1	Bucket B_2	Bucket B_3
$\langle\{q_0\}, \tau_{\mathcal{A}_{\text{ex}}}(ab)\rangle$	$\langle\{q_1\}, \tau_{\mathcal{A}_{\text{ex}}}(aa)\rangle$	$\langle\{q_0, q_2\}, \tau_{\mathcal{A}_{\text{ex}}}(abaa)\rangle$
$\langle\{q_1\}, \tau_{\mathcal{A}_{\text{ex}}}(ba)\rangle$	$\langle\{q_0, q_2\}, \tau_{\mathcal{A}_{\text{ex}}}(aa)\rangle$	
	$\langle\{q_1\}, \tau_{\mathcal{A}_{\text{ex}}}(aaba)\rangle$	

(a) Three buckets.

Bucket B_1	Bucket B_2
$\langle\{q_0\}, \tau_{\mathcal{A}_{\text{ex}}}(ab)\rangle$	$\langle\{q_1\}, \tau_{\mathcal{A}_{\text{ex}}}(ba)\rangle$
$\langle\{q_1\}, \tau_{\mathcal{A}_{\text{ex}}}(aa)\rangle$	$\langle\{q_1\}, \tau_{\mathcal{A}_{\text{ex}}}(aaba)\rangle$
$\langle\{q_0, q_2\}, \tau_{\mathcal{A}_{\text{ex}}}(aa)\rangle$	
$\langle\{q_0, q_2\}, \tau_{\mathcal{A}_{\text{ex}}}(abaa)\rangle$	

(b) Two buckets.

Table 3.7: Two different bucket distributions of $\mathcal{A}_{\text{ex}}$.

automaton with 3 TMAs, so in total we have $4 + 3 \cdot 19 = 61$ *states in the complement automaton, whereas for the bucket distribution of Table 3.7b we obtain a complement automaton with only 2 TMAs, so we get only* $4 + 2 \cdot 19 = 42$ *states. The second bucket distribution seems to be better suited for a small complement automaton than the first one.*

3.3.3 Minimizing t-Automata

The bucket automata of Definition 18 recognizing $\mathrm{L}^{\mathrm{t}}_{*}(B)$ are deterministic automata with a complex transition structure. However, some of these automata presumably recognize a very simple $*$-language for various reasons: e.g., there may be unproductive states, from which no final state can be reached or a large amount of final states accepting distinct complex languages could sum up to one single simple language. So an obvious approach to reduce the size of these autoamta is to minimize them using standard techniques. After generating a bucket automaton $\mathcal{A}^{\mathrm{t}}(B)$, our algorithm converts it into the (unique) minimal equivalent deterministic automaton. In our experiments this often led to much smaller automata in the looping part and thus to a smaller complement automaton.

Minimizing a deterministic finite automaton is a very fast operation, as it can be done in time polynomial in the size of the input, see [36, 34, 80, 79]. For our implementation we use the theoretical background of Valmari and

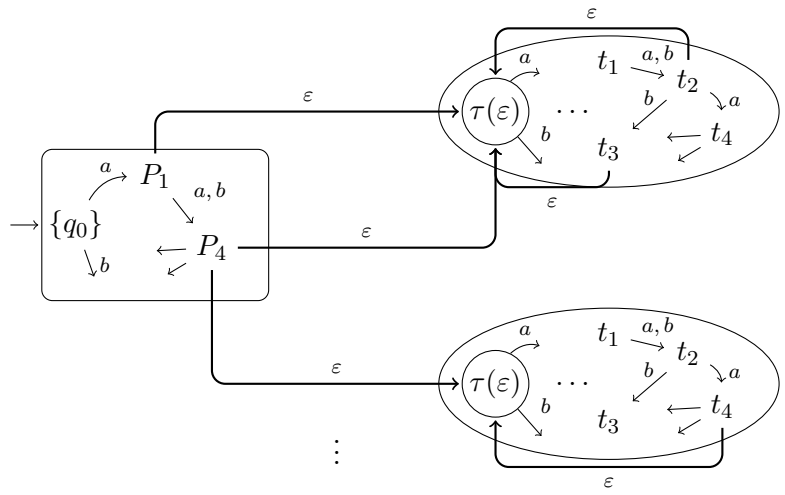


Figure 3.8: A complement automaton with $B_1 = \{\langle P_1, t_2\rangle, \langle P_4, t_3\rangle\}$, $B_2 = \{\langle P_4, t_4\rangle\}$, ...

Lehtinen [80]. They consider DFA minimization for automata with partial transition functions. This is practical for our scenario, as hereby we can omit an additional nonfinal sink state in the bucket automata.

Originally, the minimization of the bucket automata serves as a reduction of the state-space of the final complement automaton. As a nice side benefit, the minimization also is useful for improving the amount of automata that our procedure can complement successfully: as the minimization procedure is called after each single bucket automaton has been created, the DFA minimization procedure frees some of the used memory, which is then available for generating further bucket automata. Note that minimizing any of the DFAs may invalidate the normalized property of it. It can be restored easily by adding a new initial state.

Example *In Figure 3.9 we see the final complement automaton with 12 states. The two bucket automata have been minimized to smaller automata having 3 and 5 states, respectively. After that, both automata have been transformed to ω-automata. Note that this ω-operation does not preserve the minimality of the bucket automaton, e.g., the automaton for B_1 now has two states $(B_1, 0)$ and $(B_1, 2)$ that could be merged into one state, without modifying the resulting ω-language. So the given complement automaton is not a minimal one.*

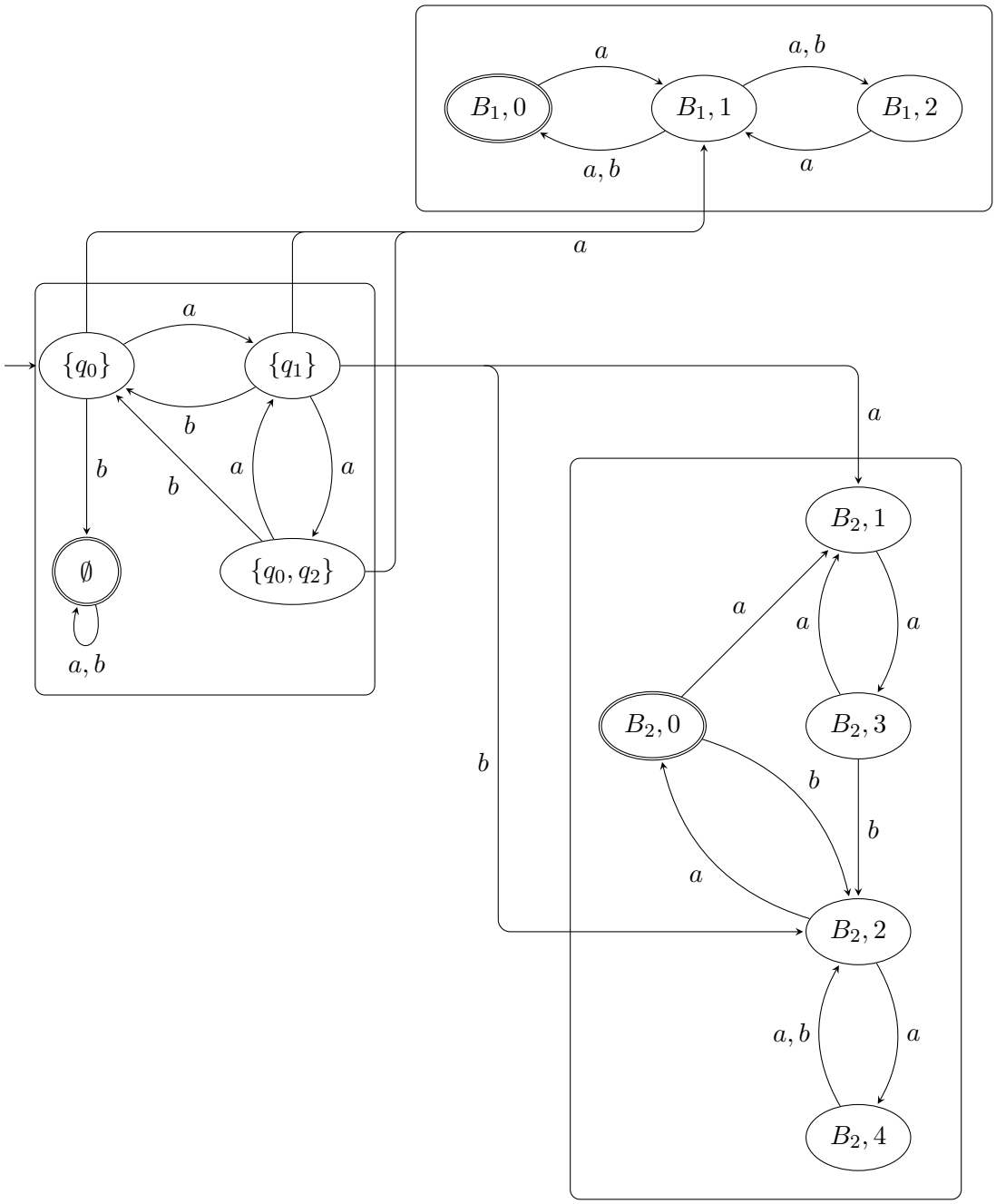


Figure 3.9: The final complement automaton of $\mathcal{A}_{\text{ex}}$.

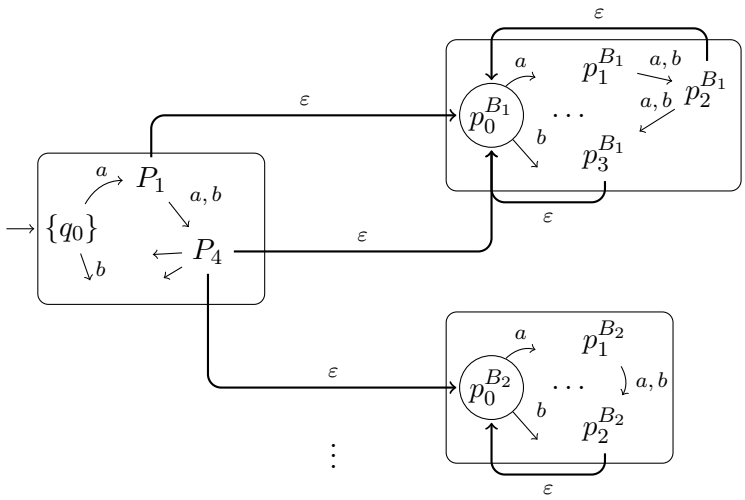


Figure 3.10: A reduced complement with $B_1 = \{\langle P_1, t_2\rangle, \langle P_4, t_3\rangle\}$, $B_2 = \{\langle P_4, t_4\rangle\}$, ...

3.3.4 Experimental Results

We implemented the improvements described in the last section in a computer program, which can be obtained from *The Alekto Project Homepage* [50], including its source code. We tried to resemble as many conditions of the experimental setting of Tsai, Fogarty, Vardi, and Tsay [77] as possible, because we wanted to compare our improved complementation method with the ones described in [77]. Therefore we used the same programming language, namely Java. However, as the source code of the results of Tsai et al. was not available to us, we had to develop our own framework to handle the data structures of automata. We also tried to resemble all the other conditions, as explained in the following.

Tsai et al. implemented four complementation algorithms, which they call `Ramsey`, `Safra-Piterman`, `Rank`, and `Slice`, and which are implementations of the Ramsey-based, the determinization-based, the rank-based, and the slice-based approach, respectively. In order to study the performance of these implementations, they required a collection of input automata. They generated 11 000 input automata randomly. Each of these automata has 15 states, an alphabet with 2 letters, and transitions that were randomly derived from 11 different transition densities and 10 different acceptance densities. For each of these input automata, they generated four complementation tasks by allocating a 2.83 GHz CPU with 1 GiB of memory

and 10 minutes computation time for each complementation algorithm. For `Ramsey` none of these tasks finished successfully, so they excluded this implementation from the next comparison steps. The other three implementations were then improved by various heuristics and for the improved versions, the experiments were repeated. After that, for the improved versions of `Safra-Piterman`, `Rank`, and `Slice`, 4 tasks, 3 383 tasks, and 216 tasks aborted unsuccessfully, respectively (see Table 3.2 on page 21).

As explained in Section 3.3.2, there are in general several possible ways to fill the buckets. So there is no unique distribution of P-t-pairs to buckets, and we have to agree on a concrete deterministic method to fill the buckets. We chose to use the following greedy algorithm: we maintain a list $(B_1, \ldots, B_n)$ of buckets, which can grow if needed, starting with the empty list $()$; for each pair $\langle P, t\rangle \in \mathrm{Reject}_{\mathrm{P,t}}$, we add $\langle P, t\rangle$ to the first bucket B_i that is mergeable with $\langle P, t\rangle$; if no such bucket exists, then we create a new empty bucket B_{n+1}, add it to the list and start over again. The final algorithm that uses all of the above heuristics (the subset construction for the initial part, the merging of transition profiles for the looping part, and the minimization of the bucket automata, together with the greedy bucket filling algorithm) is called `improved-Ramsey`.

For our experiments, we used a computer with a 2.83 GHz CPU and 4 GiB of memory. We imposed the some restrictions on it: we set a timeout of 10 minutes, and we confined the Java heap space to 1 GiB of memory. So the physical preconditions of our experiments and of the experiments by Tsai et al. were roughly the same. We also used the same 11 000 input automata as them, as they were kindly ready to provide us with these.

The results of our experiments are the following. Out of all input tasks, 10 839 finished successfully and 161 did not (152 ran out of memory, and 9 violated the time limit). In terms of successfully finished tasks, this puts `improved-Ramsey` on the second place, between `Safra-Piterman` and `Slice`. The sizes of the complement automata that were computed by `improved-Ramsey` range from 0 to 337 464 states with an average size of 361.09 states (328.97 after removing dead states). Tsai et al. [77] also give some particular average sizes. But we are not able to adequately compare these sizes with our results, as they provide average sizes only for 7 593 of the initial 11 000 automata, namely for those tasks that finished successfully by all of their implementations. Our numbers, on the other hand, base upon all 10 839 finished tasks of our implementation. So a smaller average size could be interpreted as a better performing algorithm (because the algorithm leads to smaller sizes) or as a worse performing algorithm (because fewer input automata are successfully complemented).

3.4 Weak-order-based Complementation

In this section, we introduce a novel, purely theoretical, complementation method. We call this method the "weak-order-based complementation". It complements a Büchi automaton in a similar way as the heuristics in the previous section. But instead of generating the bucket automata by merging transition profiles, it generates a bucket automaton for each subset P of the state set and for each weak order on P. This generalizes the merging of transition profiles. Instead of computing all reachable transition profiles, we decide upfront whether some words belong to mergeable transition profiles on the basis of a weakly-ordered set.

This improves the Ramsey-based construction of the last section in terms of state space: given an input automaton $\mathcal{A}$ of size n, the resulting complement automaton will be of size $2^{\mathcal{O}(n \log n)}$. The complement automaton is composed of the initial part and the looping part. For the initial part, we again take the subset automaton $\mathrm{PA}_{\mathcal{A}}$. For the looping part, we take the automata that recognize languages $\mathrm{L}^{\mathrm{t}}_{*}(B)$ for some suitable bucket B.

In Section 3.3.2 we have seen that under certain condition we can merge transition profiles in the looping part, by putting them into "buckets", and in Section 3.3.4 we used a simple greedy algorithm for generating these buckets. Now we introduce a more intelligent scheme by which the filling of the buckets leads to a small quantity of bucket automata and to a small size of each bucket automaton, more precisely both numbers are bound from above by $2^{\mathcal{O}(n \log n)}$.

The key idea is that we merge all transition profiles that can be embedded into the same total preorder (or weak order) into a bucket in such a way that $\not\to$ edges are strictly increasing in the order and $\to$ edges are not decreasing in the order. By obeying these rules, we can be sure that no accepting cycle will be introduces by the merging, and thus the resulting bucket will be mergeable in the sense of Definition 14. Additionally, we have to make sure that we cover the set $\mathrm{Reject}_{\mathrm{P,t}}$ by mergeable buckets of transition profiles, in order to obtain an automaton for the full complement language and not just for a subset.

Definition 19 *A* total preorder *(or* weak order*) on a set P is a binary relation $\lesssim$ on P that is*

- *total (for all $p, q \in P$ it holds $p \lesssim q$ or $q \lesssim p$), and*
- *transitive (for all $p, q, r \in P$ with $p \lesssim q$ and $q \lesssim r$ it holds $p \lesssim r$).*

For a set Q, we define Pre_Q *to be the set of all pairs* $\langle P, \lesssim \rangle$ *such that* $P \subseteq Q$ *and* $\lesssim$ *is a total preorder on* P*. For an automaton* $\mathcal{A}$ *with state set* Q*, we set* $\mathrm{Pre}_\mathcal{A} := \mathrm{Pre}_Q$ *and we call* $\langle P, \lesssim \rangle$ *a* preordered set *over* $\mathcal{A}$*.*

For any weak order, the corresponding equivalence relation and the notation of equivalence classes are defined in the usual way.

Definition 20 *Let* $\langle P, \lesssim \rangle$ *be a preordered set. Then for* $p, q \in P$*, we define*

$$p \simeq q \;:\Longleftrightarrow\; p \lesssim q \wedge q \lesssim p,$$

and the equivalence class of q *is* $[q] := [q]_{\langle P, \lesssim \rangle} := \{p \in P \mid p \simeq q\}$*. Then*

$$[p] \leq [q] \;:\Longleftrightarrow\; p \lesssim q, \text{ and}$$
$$[p] < [q] \;:\Longleftrightarrow\; [p] \leq [q] \wedge [q] \not\leq [p]$$

are well-defined relations of $P/\!\simeq$*.*

Note that the number of weak orders on a set P with n elements is bounded by n^n, because each weak order on P can be characterized by a mapping that assigns to each element of P a number from 1 to n corresponding to its position in the order (equivalent elements are mapped to the same number). So for an automaton with n states, the number of pairs $\langle P, \lesssim \rangle \in \mathrm{Pre}_\mathcal{A}$ is bounded by $2^n n^n$.

Definition 21 *Let* $\langle P, \lesssim \rangle \in \mathrm{Pre}_\mathcal{A}$ *for an automaton* $\mathcal{A}$*. We say that a transition profile* $t = (\rightarrow, \nrightarrow)$ *over* $\mathcal{A}$ *is* compatible with $\langle P, \lesssim \rangle$ *if*

- $t(P) \subseteq P$*, and*
- *for all* $p, q \in P$ *with* $p \rightarrow q$ *it holds* $[p] \leq [q]$*, and*
- *for all* $p, q \in P$ *with* $p \nrightarrow q$ *it holds* $[p] < [q]$*.*

We define the bucket

$$B_{\langle P, \lesssim \rangle} := \{\langle P, t \rangle \in \mathrm{RSC}_\mathcal{A} \times \mathrm{RTP}_\mathcal{A} \mid t \text{ is compatible with } \langle P, \lesssim \rangle\}.$$

One preordered set $\langle P, \lesssim \rangle$ describes the set of all permitted transitions inside P. For a fixed preordered set, we only allow transitions from smaller elements to greater or equivalent elements. Additionally, for transitions which contain accepting states we only allow those which lead from smaller elements to strictly greater elements. By this means, accepting cycles are ruled out: there cannot be a path through P which contains infinitely many

accepting states. In other words, the join of a bucket $B_{\langle P,\lesssim\rangle}$ does not have a lasso and thus is mergeable in the sense of Definition 14. Conversely, each rejecting P-t-pair $\langle P,t\rangle \in \mathrm{Reject}_{\mathrm{P,t}}$ induces a weak order on P and thus its transition profile t is compatible with a suitable preordered set over $\mathcal{A}$.

Lemma 22 *Let $\mathcal{A}$ be an automaton. Then for each $\langle P,\lesssim\rangle \in \mathrm{Pre}_{\mathcal{A}}$, it holds that $B_{\langle P,\lesssim\rangle}$ is mergeable and for each $\langle P,t\rangle \in \mathrm{Reject}_{\mathrm{P,t}}$, there is a total preorder $\lesssim$ on P with $\langle P,t\rangle \in B_{\langle P,\lesssim\rangle}$.*

Proof We start with the first claim. Assume the join of $B_{\langle P,\lesssim\rangle}$ has a lasso. Then there is a sequence of states $p_1,\ldots,p_k,\ldots,p_n \in Q$, $1 \leq k < n$ with $p_1 \in P$, $p_i \to p_{i+1}$ for all $i < n$, $p_k \mathrel{-\!\!\circledast} p_{k+1}$, and $p_n \to p_k$. Since $t(P) \subseteq P$ for all t compatible with $\langle P,\lesssim\rangle$, it holds $p_i \in P$ for all $1 \leq i \leq n$ by induction on i. Then we have $[p_{k+1}] \leq [p_{k+2}] \leq \cdots \leq [p_n] \leq [p_k]$, and $[p_k] < [p_{k+1}]$, which is a contradiction.

To prove the second claim, note that by definition of $\mathrm{Reject}_{\mathrm{P,t}}$ it holds $t(P) \subseteq P$. Consider the directed graph G with vertex set P and edge relation $\to_t$. The SCCs of G are preordered by $C_1 \; R \; C_2$ iff there is a path from a state $p \in C_1$ to a state $q \in C_2$ in G. We make this preorder total by ordering incomparable SCCs of G in an arbitrary way. We obtain a total preorder R' on the set of SCCs. This induces a total preorder $\lesssim$ on P by $p \lesssim q$ iff $C_p \; R' \; C_q$ for $p \in C_p$ and $q \in C_q$. Clearly $p \to_t q$ implies $[p] \leq [q]$. Let $C \subseteq P$ be an SCC of G. Then for states $p_1, p_2 \in C$, it cannot hold $p_1 \mathrel{-\!\!\circledast}_t p_2$, as otherwise we can construct an accepting run $q_0 \xrightarrow{u} p_1 \xrightarrow[F]{v_1} p_2 \xrightarrow{v_2} p_3 \to \cdots p_n \xrightarrow{v_n} p_1 \cdots$ of $\mathcal{A}$ on a word $u(v_1 \cdots v_n)^\omega$ with $u \in \mathrm{L}_*(P)$ and $v_i \in \mathrm{L}_*(t)$. So $p_1 \mathrel{-\!\!\circledast}_t p_2$ implies $[p_1] < [p_2]$. Altogether, t is compatible with $\langle P,\lesssim\rangle$, and thus $\langle P,t\rangle \in B_{\langle P,\lesssim\rangle}$. □

According to Lemma 22 we have already found a covering of $\mathrm{Reject}_{\mathrm{P,t}}$ by a number of buckets that is bounded by $2^n n^n$. It remains to be shown that for a given preordered set $\langle P,\lesssim\rangle$ the language $\mathrm{L}^{\mathrm{t}}_*(B_{\langle P,\lesssim\rangle})$, i.e., those words whose transition profile is compatible with $\langle P,\lesssim\rangle$, can be recognized by a "small" automaton $\mathcal{A}_{\langle P,\lesssim\rangle}$.

For a given preordered set $\langle P,\lesssim\rangle$, the automaton works as follows. When reading a word v, the automaton tracks for each $q \in Q$ the maximal equivalence classes $[p']$ and $[p'']$ of $\langle P,\lesssim\rangle$ such that in $\tau_{\mathcal{A}}(v)$ there is an $\to$ edge from an element of $[p']$ to q and there is an $\mathrel{-\!\!\circledast}$ edge from and element of $[p'']$ to q. This information can easily be updated when appending a new letter to v. We will show (Lemma 26) that tracking only this information suffices to deduce whether $\tau_{\mathcal{A}}(v)$ is compatible with $\langle P,\lesssim\rangle$.

Definition 23 *For a preordered set $\langle P, \lesssim \rangle$, let $P_\perp \coloneqq \{\perp\} \cup (P/\simeq)$ and let $\mathcal{M}_{\langle P,\lesssim\rangle}$ be the set of all functions $f\colon Q \to P_\perp$. We extend the linear order $\leq$ on $P/\simeq$ to $\{\perp\} \cup (P/\simeq)$ by setting $\perp < [p]$ for all $[p] \in (P/\simeq)$.*

By convention, the maximum of the empty set is the smallest element of $P_\perp$, i.e., $\max \emptyset = \perp$.

A state of the automaton $\mathcal{A}_{\langle P,\lesssim\rangle}$ is a pair from $\mathcal{M}_{\langle P,\lesssim\rangle} \times \mathcal{M}_{\langle P,\lesssim\rangle}$. The initial state is the pair $(\varphi_\varepsilon, \psi_\varepsilon) \in \mathcal{M}_{\langle P,\lesssim\rangle} \times \mathcal{M}_{\langle P,\lesssim\rangle}$ with

$$\varphi_\varepsilon(q) \coloneqq \begin{cases} [q] & \text{if } q \in P, \\ \perp & \text{else;} \end{cases} \quad \text{and} \quad \psi_\varepsilon(q) \coloneqq \begin{cases} [q] & \text{if } q \in P \cap F, \\ \perp & \text{else.} \end{cases}$$

For two functions $\varphi, \psi \in \mathcal{M}_{\langle P,\lesssim\rangle}$ and for a letter $a \in \Sigma$, we define the *update of φ and ψ with the letter a* to be $(\varphi, \psi) \cdot a \coloneqq (\varphi', \psi')$ with

$$\varphi'(q) \coloneqq \max\{\varphi(r) \in P_\perp \mid r \in Q, r \xrightarrow{a} q\}, \text{ and}$$
$$\psi'(q) \coloneqq \begin{cases} \max\{\varphi(r) \in P_\perp \mid r \in Q, r \xrightarrow{a} q\} & \text{if } q \in F, \\ \max\{\psi(r) \in P_\perp \mid r \in Q, r \xrightarrow{a} q\} & \text{else.} \end{cases}$$

We write (φ_a, ψ_a) for $(\varphi_\varepsilon, \psi_\varepsilon) \cdot a$, and inductively we write $(\varphi_{va}, \psi_{va})$ for $(\varphi_v, \psi_v) \cdot a$. The function φ_v maps every state q to the maximal class in $P/\simeq$ from which one can reach q by reading v; it maps to $\perp$ if no such class exists. Analogously, the function ψ_v maps every state q to the maximal class from which one can reach q passing an accepting state by reading v; it maps to $\perp$ if no such class exists. We formalize this in the following lemma.

Lemma 24 *Let $\mathcal{A}$ be an automaton, $\langle P, \lesssim \rangle \in \mathrm{Pre}_\mathcal{A}$ and $v \in \Sigma^*$. Then for each $q \in Q$ it holds*

$$\varphi_v(q) = \max\{[p] \in P_\perp \mid p \in P, p \xrightarrow{v} q\} \text{ and}$$
$$\psi_v(q) = \max\{[p] \in P_\perp \mid p \in P, p \xrightarrow[F]{v} q\}.$$

Proof It is sufficient to show the following equivalent statement, which we prove by induction on the length of v. Let $\langle P, \lesssim \rangle \in \mathrm{Pre}_\mathcal{A}$ and $v \in \Sigma^*$. For any $p \in P$ and $q \in Q$ it holds

$$[p] \leq \varphi_v(q) \iff \exists \bar{p} \in P\colon [p] \leq [\bar{p}] \wedge \bar{p} \xrightarrow{v} q, \text{ and}$$
$$[p] \leq \psi_v(q) \iff \exists \bar{p} \in P\colon [p] \leq [\bar{p}] \wedge \bar{p} \xrightarrow[F]{v} q.$$

For $v = \varepsilon$, it holds $[p] \leq \varphi_v(q)$ iff $q = p \in P$. But this is the case iff there is a state $\bar{p} = p$ with $\bar{p} \xrightarrow{\varepsilon} q$.

Let now $v = wa$ for $w \in \Sigma^*$ and $a \in \Sigma$. Let $(\varphi, \psi) = (\varphi_\varepsilon, \psi_\varepsilon) \cdot w$ and $(\varphi', \psi') = (\varphi, \psi) \cdot a$. Let the statement be true for w. Let there be a state $\bar{p} \in P$ with $[p] \leq [\bar{p}]$ and a path of wa in $\mathcal{A}$ from $\bar{p}$ to q. Then there is an intermediate state $\bar{r} \in Q$ such that there is a path of w from $\bar{p}$ to $\bar{r}$ and $\bar{r} \xrightarrow{a} q$. By induction hypothesis we have $[\bar{p}] \leq \varphi(\bar{r})$ and thus it follows $[p] \leq [\bar{p}] \leq \varphi(\bar{r}) \leq \max\{\varphi(r) \mid r \xrightarrow{a} q\} = \varphi'(q)$. On the other hand, if $[p] \leq \varphi'(q) = \max\{\varphi(r) \mid r \xrightarrow{a} q\}$, then there is an $\bar{r} \in Q$ with $\bar{r} \xrightarrow{a} q$ and $[p] \leq \varphi(\bar{r})$. By induction hypothesis, there is a state $\bar{p} \in P$ with $[p] \leq [\bar{p}]$ and a path of w in $\mathcal{A}$ from $\bar{p}$ to $\bar{r}$. Altogether we have a state $\bar{p}$ with $[p] \leq [\bar{p}]$ and a path of wa in $\mathcal{A}$ from $\bar{p}$ to q.

The proof for ψ_v works analogously with a case distinction on the position of the accepting state. □

Next we define the set of final states of the automaton $\mathcal{A}_{\langle P, \lesssim \rangle}$. A state is supposed to be final if the word that was read until there has a transition profile that is compatible with $\langle P, \lesssim \rangle$. So the functions φ and ψ must distinguish whether this is the case.

Definition 25 *Let $\langle P, \lesssim \rangle$ be a preordered set and $\varphi, \psi \in \mathcal{M}_{\langle P, \lesssim \rangle}$. We say that the pair (φ, ψ) is* consistent with *$\langle P, \lesssim \rangle$ if*

- *for all $q \in Q \setminus P$ it holds $\varphi(q) = \bot$, and*
- *for all $q \in P$ it holds $\varphi(q) \leq [q]$, and*
- *for all $q \in P$ it holds $\psi(q) < [q]$.*

Lemma 26 *Let $\mathcal{A}$ be an automaton, $\langle P, \lesssim \rangle \in \mathrm{Pre}_\mathcal{A}$ and $v \in \Sigma^*$. Then the transition profile $\tau_\mathcal{A}(v)$ is compatible with $\langle P, \lesssim \rangle$ if and only if (φ_v, ψ_v) is consistent with $\langle P, \lesssim \rangle$.*

Proof Let $t = \tau_\mathcal{A}(v)$. With Lemma 24 we obtain

$$
\begin{aligned}
& t(P) \subseteq P \\
& \qquad \iff \neg\exists p \in P\ \exists q \in Q \setminus P \colon p \xrightarrow{v} q \\
& \qquad \iff \forall q \in Q \setminus P\ \neg\exists p \in P \colon p \xrightarrow{v} q \\
& \qquad \iff \forall q \in Q \setminus P \colon \varphi_v(q) = \bot, \text{ as well as}
\end{aligned}
$$

$$
\begin{aligned}
& \forall p, q \in P \colon (p \xrightarrow{v} q \Rightarrow [p] \leq [q]) \\
& \qquad \iff \forall q \in P \colon \max\{[p] \mid p \in P, p \xrightarrow{v} q\} \leq [q] \\
& \qquad \iff \forall q \in P \colon \varphi_v(q) \leq [q], \text{ and}
\end{aligned}
$$

$$\begin{aligned}
&\forall p, q \in P\colon (p \xrightarrow[F]{v} q \Rightarrow [p] < [q]) \\
&\qquad \iff \forall q \in P\colon \max\{[p] \mid p \in P, p \xrightarrow[F]{v} q\} < [q] \\
&\qquad \iff \forall q \in P\colon \psi_v(q) < [q].
\end{aligned}$$ □

We are now ready to define the automaton $\mathcal{A}_{\langle P,\lesssim\rangle}$.

Definition 27 *Given a set P and a weak order $\lesssim$ on P, we define the automaton $\mathcal{A}_{\langle P,\lesssim\rangle} \coloneqq (Q'', \Sigma, q_0'', \delta'', F'')$ with*

- $Q'' = \{(P, \lesssim, \varphi, \psi) \mid \varphi, \psi \in \mathcal{M}(P)\}$
- $q_0'' = (P, \lesssim, \varphi_\varepsilon, \psi_\varepsilon)$
- $\delta''((P, \lesssim, \varphi, \psi), a) = \{(P, \lesssim, \varphi', \psi')\}$ *where* $(\varphi', \psi') = (\varphi, \psi) \cdot a$
- $F'' = \{(P, \lesssim, \varphi, \psi) \mid (\varphi, \psi)$ *is consistent with* $(P, \lesssim)\}$

The following lemma states that the automaton $\mathcal{A}_{\langle P,\lesssim\rangle}$ accepts those words for which the transition profile is compatible with $\langle P, \lesssim\rangle$. This proves the correctness of our construction.

Lemma 28 *For every preordered set $\langle P, \lesssim\rangle$, it holds*

$$\mathrm{L}_*(\mathcal{A}_{\langle P,\lesssim\rangle}) = \mathrm{L}_*^{\mathrm{t}}(B_{\langle P,\lesssim\rangle}).$$

Proof From Lemma 26 we obtain

$$\begin{aligned}
\mathrm{L}_*(\mathcal{A}_{\langle P,\lesssim\rangle}) &= \{v \in \Sigma^* \mid (\varphi_v, \psi_v) \text{ is consistent with } \langle P, \lesssim\rangle\} \\
&= \{v \in \Sigma^* \mid \tau_{\mathcal{A}}(v) \text{ is compatible with } \langle P, \lesssim\rangle\} \\
&= \bigcup\{\mathrm{L}_*(t) \mid t \text{ is compatible with } \langle P, \lesssim\rangle\} \\
&= \mathrm{L}_*^{\mathrm{t}}(B_{\langle P,\lesssim\rangle}).
\end{aligned}$$ □

Starting from an automaton $\mathcal{A}$, recognizing $\mathrm{L}_\omega(\mathcal{A})$. For each $\langle P, \lesssim\rangle$, the automaton $\mathcal{A}_{\langle P,\lesssim\rangle}$ can now be normalized using an extra initial state. After that we use the construction from Definition 17 and obtain a complement automaton, which by Theorem 16 indeed recognizes the language $\Sigma^\omega \setminus \mathrm{L}_\omega(\mathcal{A})$.

Theorem 29 *Given a Büchi automaton with n states, the Ramsey-based complementation method in combination with weak order bucket merging yields a complement automaton with at most $2^n + 2^n n^n((n+1)^{2n}+1)$ states.*

Proof As mentioned above, the number of pairs $\langle P, \lesssim \rangle$ is bounded by $2^n n^n$. The number of states of each automaton $\mathcal{A}_{\langle P,\lesssim\rangle}$ is bounded by $(n+1)^{2n}$. To normalize these automata, a new initial state has to be introduced for each of them. The initial part of the complement automaton consists of a subset automaton, so it is bounded by 2^n. Altogether this gives the claimed bound

$$|\mathrm{PA}_{\mathcal{A}}| + \sum_{\langle P,\lesssim\rangle \in \mathrm{Pre}_{\mathcal{A}}} |Q_{\langle P,\lesssim\rangle}| \;\leq\; 2^n + 2^n \cdot n^n \cdot ((n+1)^{2n} + 1). \qquad \square$$

So the number of states in the resulting complement is in $2^{\mathcal{O}(n \log n)}$. For now, the weak-order-based complementation approach is a purely theoretical method. The heuristics of Section 3.3 allows us to restrict buckets to only contain transition profiles that are reachable. The weak-order-based approach does not seem to allow such an optimization, so it actually has to iterate over all possible weak orders. The quantity of all weak orders is given by the ordered Bell numbers, which is a sequence that grows so rapidly that it is just not feasible to actually generate all possible weak orders.

3.5 Conclusion

In this chapter, we considered the Ramsey-based approach for complementing Büchi automata. We started by summing up the well-established descriptions [12, 67] of the Ramsey-based approach, and after that we proposed three main heuristics to improve the method. First, we shortened the initial part of the complement automaton by introducing a subset construction, second, we reduced the number of automata in the looping part of the complement automaton by merging a couple of them, and third, we reduced the number of states of each automaton in the looping part by applying a DFA minimization. In order to show the practical feasibility of these heuristics, we implemented our method in a Java program, and we showed in an experimental study that, in practice, it can compete with implementations of other, more modern methods [77]. Then we developed a novel complementation method, based upon weak orders, which uses the same principles as our improved Ramsey-based method, and which exhibits an upper bound of $2^{\mathcal{O}(n \log n)}$ on the state space. We conclude that the Ramsey-based approach is still an interesting approach to study on the theoretical side and a powerful method on the practical side, despite its degradation by previous results [77].

Comparing the heuristics of Section 3.3 with the weak-order-based complementation method of Section 3.4, we see that the heuristics provide a good practical result with a poor theoretical upper bound of $2^{\mathcal{O}(n^2)}$, while on the other hand, the weak-order-based complementation gives us an unreasonable practical outcome with a decent upper bound of $2^{\mathcal{O}(n \log n)}$. The poor upper bound of our heuristics are due to the reachable part of the transition monoid automaton: we are not aware of any upper bound on its size that is smaller than $2^{\mathcal{O}(n^2)}$. However, the experiments suggest that in practice, it actually is much smaller.

Regarding future work, the first and obvious question is, whether the heuristics of Section 3.3 can be enhanced such that they have a $2^{\mathcal{O}(n \log n)}$ upper bound on the state space. Second, Fogarty, Kupferman, Vardi, and Wilke [28] developed a complementation method, which unifies two of the main methods. They combined the rank-based approach with the slice-based approach by using a weak order on the nodes of a run DAG. We think that a similar unification should also be possible with the methods developed in this chapter. The third direction of future work leads to universality checking and inclusion checking of Büchi automata [29, 2, 1]. The question here is, whether our heuristics can be helpful for optimizing the existing methods.

Chapter 4

Strategies as Languages

The motivation for the present chapter is the Büchi-Landweber Theorem, which we already discussed in Chapter 1. It says that ω-regular games are determined with regular winning strategies. In order to refine this result for other language classes, we need to consider how the theory of regular ω-languages is related to the theory of regular $*$-languages. We require some framework such that, starting from a class $\mathcal{K}$ of $*$-languages we obtain a class $\mathcal{L}$ of ω-languages that is connected to $\mathcal{K}$ in the above-mentioned game-theoretical way. There are at least two different ways to connect $*$-languages with ω-languages. The first one has already been dealt with in Chapter 3. It is the one described by Büchi (see Characterization 1 on page 10) with his model of nondeterministic automata: all regular ω-languages can be obtained as finite unions of languages of the form $U \cdot V^\omega$ where U and V are regular $*$-languages. The second one is the one implicated by the model of deterministic Muller automata: a regular ω-language can be represented as a Boolean combination of limit ω-languages $\lim(U_i)$, where U_i are regular $*$-languages (see Characterization 3 on page 11). For solving infinite games, we require a deterministic model of automata, therefore we utilize the deterministic Muller automaton. As a weaker variant of the Muller model, we also use SW-automata (see Characterization 2). This leads to a more restricted class of ω-languages, which are described by extensions $\mathrm{ext}(U_i)$ of $*$-languages U_i.

Studying the connection between the complexity of winning conditions and the complexity of winning strategies corresponds to relating $*$-languages to ω-languages. Büchi and Landweber showed that games defined by ω-languages in $\mathrm{BC}(\lim(\mathrm{REG}))$ can be solved with strategies that are defined by $*$-languages in REG. In this chapter, we replace REG with subclasses of REG, e.g., with the class of piecewise testable languages or with levels within concatenation hierarchies of the star-free languages. More specifically, for a class $\mathcal{K}$ of $*$-languages, we consider the classes $\mathrm{BC}(\lim(\mathcal{K}))$ and $\mathrm{BC}(\mathrm{ext}(\mathcal{K}))$ of ω-languages. We show how ω-languages represent "winning

conditions", and $*$-languages can be interpreted as "winning strategies", and we show how we can solve games with winning conditions in a class $\mathrm{BC}(\mathrm{ext}(\mathcal{K}))$ or $\mathrm{BC}(\lim(\mathcal{K}))$ with winning strategies close to $\mathcal{K}$.

The framework for infinite games that we use in this chapter differs from the framework of games on graphs, which is more popular in the literature. The setting of infinite games in this chapter is based on Gale-Stewart games [32]. These games are played between two players, namely Player 1 and Player 2. In each round, first Player 1 picks a letter from an alphabet Σ_1 and then Player 2 a letter from an alphabet Σ_2. An infinite play of the game can then be stated as an ω-word over $\Sigma := \Sigma_1 \times \Sigma_2$. The winner of this play is decided by consulting an ω-language $L \subseteq \Sigma^\omega$, which is called the *winning condition*: If the play belongs to L, then Player 2 is the winner, otherwise Player 1 is. Games whose winning conditions belong to the class $\mathrm{BC}(\mathrm{ext}(\mathcal{K}))$ are referred to as *weak games* while those whose winning conditions belong to $\mathrm{BC}(\lim(\mathcal{K}))$ are called *strong games*.

A strategy for either player gives the choice of an appropriate letter $a \in \Sigma_1$, respectively $x \in \Sigma_2$, for each possible play prefix where it is Player 1's, respectively Player 2's, turn. We can capture a strategy for a player by collecting, for each letter a, the set K_a of those finite play prefixes that induce the choice of a. For example, for a strategy of Player 1 we have $K_a \subseteq \Sigma^*$ for each $a \in \Sigma_1$. We say that a strategy is in $\mathcal{K}$ if each such language K_a is in $\mathcal{K}$.

The fundamental Büchi-Landweber Theorem [14] states that for a winning condition defined by any regular ω-language $L \in \mathrm{BC}(\lim(\mathrm{REG}))$, one of the two players has a winning strategy, that one can decide who is the winner, and that one can present a *regular* winning strategy (in the sense mentioned above) for the winner. In short, we say that *regular games are determined with regular winning strategies*. An analogous result for the class SF of star-free languages was shown in [63, 64] and later, by a different method, also in [59]: Star-free games are determined with star-free winning strategies. For classes above regular languages, there are analogous results for several types of pushdown languages [30]. In this chapter, we focus on subclasses of SF, where the situation is more complicated. We consider piecewise testable languages and two hierarchies below the star-free languages, namely the dot-depth hierarchy [23], whose levels are denoted by DD_i, and the Straubing-Thérien hierarchy [54], whose levels are denoted by ST_i. We show that for the class DD_1 of languages of dot-depth one, games with winning conditions in classes $\mathrm{BC}(\mathrm{ext}(\mathrm{DD}_1))$ and $\mathrm{BC}(\lim(\mathrm{DD}_1))$ are, in general, not determined with winning strategies in DD_1, but only with those in classes DD_2 and DD_3 respectively. In contrast to this, we

show that for games in the more restricted class BC(ext(pos-DD_1)), we have determinacy with winning strategies in DD_1. The class pos-DD_1 is the closure of languages $w_0\Sigma^* w_1 \dots \Sigma^* w_n$, where $w_i \in \Sigma^*$, under union and intersection. The Boolean closure of pos-DD_1 is DD_1. We obtain similar results for the Straubing-Thérien hierarchy.

This chapter integrates results from [20] and [21]. It is structured as follows. In Section 4.1 we give precise definitions of infinite games played on languages, we explain how we use the theory of formal languages to measure the complexity of games, and we show that both kinds of games are actually interchangable by reducing language-games to games on graphs. In Section 4.2 we recall the definitions of the well known subclasses of star-free languages, and the subclasses of infinite languages that we consider in this chapter. Subsequently, in Sections 4.3 to 4.5 we consider games over these classes of infinite languages and present results pertaining to winning strategies in these games. We conclude with some open questions and perspectives.

4.1 The Complexity of Strategies

There does not exist an unambiguous approach for measuring the complexity of winning strategies in infinite games. In Chapter 5, when we tackle games on graphs, we will employ a quantitative method to determine the grade of a winning strategy. In this chapter, we choose a different concept as explained in the following.

The complexity of a formal language can be rated by determining membership of that language to a class of "simple" languages. For example, in the Chomsky hierarchy, we can identify four language classes REG $\subset$ CFL $\subset$ CSL $\subset$ RE, the classes of regular, context-free, context-sensitive, and recursively enumerable languages, respectively. Each smaller class is "simpler" than the next one in the sense that both, a simpler kind of formal grammar, and a simpler automata model exists for it.

Now we want to transfer this kind of complexity hierarchy to games. For this purpose, in the following setting, we express winning strategies as $*$-languages. Then it is natural to denote the complexity of strategies by the complexity of their corresponding languages.

4.1.1 Games defined by Languages

Any ω-language $L \subseteq \Sigma^\omega$ induces a game with winning condition L. Formally, a *game* is a triple (Σ_1, Σ_2, L), where Σ_1 and Σ_2 are alphabets and L is an

ω-language over $\Sigma_1 \times \Sigma_2$. For this chapter, we fix an alphabet Σ_1 for Player 1 and an alphabet Σ_2 for Player 2. Then the compound alphabet for the played game is $\Sigma := \Sigma_1 \times \Sigma_2$. Since the alphabets for both players are fixed, we simply speak of the "game L" instead of the triple (Σ_1, Σ_2, L).

A *play* of the game L is an infinite word $\alpha = (a_i, x_i)_{i \in \mathbb{N}} \in \Sigma^\omega$. A play α is called *winning for Player 2* if $\alpha \in L$. In every game over Σ, a *strategy for Player 1* is a mapping $\sigma\colon \Sigma^* \to \Sigma_1$ and a *strategy for Player 2* is a mapping $\tau\colon \Sigma^* \to \Theta$, where $\Theta := {\Sigma_2}^{\Sigma_1}$ is the (finite) set of all mappings from Σ_1 to Σ_2. A play α is said to be *consistent with* σ, if for all positions $i \in \mathbb{N}$ we have $\sigma(\alpha[0,i)) = a_i$. Analogously, α is *consistent with* τ if for all $i \in \mathbb{N}$ we have $\tau(\alpha[0,i))(a_i) = x_i$. For two strategies σ and τ there is a (uniquely determined) play $\alpha(\sigma, \tau)$ that is consistent with both σ and τ.

If $\alpha(\sigma, \tau) \in L$ for every Player 1 strategy σ, then τ is called a *winning strategy for Player 2*. The other way around, if $\alpha(\sigma, \tau) \notin L$ for all Player 2 strategies τ, then σ is a *winning strategy for Player 1*.

4.1.2 Complexity Measures

Let $\mathcal{K}$ be a class of $*$-languages. We say that a strategy σ for Player 1 *belongs to the class* $\mathcal{K}$ if for every $a \in \Sigma_1$ the language

$$K_a := \{w \in \Sigma^* \mid \sigma(w) = a\}$$

is in $\mathcal{K}$. A strategy τ for Player 2 *belongs to* $\mathcal{K}$ if for every $(a, x) \in \Sigma$ the language

$$K_{a \mapsto x} := \{w \in \Sigma^* \mid \tau(w)(a) = x\}$$

is in $\mathcal{K}$.

Given a game L and a class $\mathcal{K}$ of $*$-languages, we say that L *is determined with winning strategies in* $\mathcal{K}$ if one of the players has a winning strategy in $\mathcal{K}$.

Since we only regard regular languages in this thesis, each language class $\mathcal{K}$ will be a subclass of REG. In particular, each language $K \in \mathcal{K}$ corresponds to a "finite-state strategy" realized by a Moore machine. A Moore machine M_σ implementing a finite-state strategy σ for Player 1, is given by $M_\sigma = (Q, \Sigma, \Sigma_1, q_0, \delta, \lambda)$ with $\lambda\colon Q \to \Sigma_1$ such that for all $w \in \Sigma^*$ it holds $\lambda(\delta^*(q_0, w)) = \sigma(w)$. Analogously, a Moore machine M_τ for Player 2 is given by $M_\tau = (Q, \Sigma, \Theta, q_0, \delta, \lambda)$ with $\lambda\colon Q \to \Theta$ such that for all $w \in \Sigma^*$ it holds $\lambda(\delta^*(q_0, w)) = \tau(w)$. For all ω-regular games, winning strategies in REG suffice to solve these games. This result is called the Büchi-Landweber Theorem.

Theorem 30 (Büchi-Landweber [14]) *Games in* BC(lim(REG)) *are determined with winning strategies in* REG.

4.1.3 Translating Language-Games to Graph-Games

The description of games over languages is not far away from the description of games on graphs that is usually used in the literature. In the literature [33], two-player games are usually considered over a graph as defined in Chapter 2 with two different types of nodes: one type belonging to Player 1, the other to Player 2. However, for our purposes it is more convenient to consider a game graph $G = (Q, \Sigma, q_0, \delta)$ with only one type of nodes, where in a single move from a node, we first let Player 1 choose an action from Σ_1 and after that let Player 2 choose from Σ_2. We call this model of game graphs the "unified model" and define "unified game graphs" accordingly. With this unified model, the conversions between ω-automata, game graphs, and Moore machines are straightforward.

In this section, we give a simple reduction from games over languages (as they are defined in Section 4.1) to games on graphs (as they are defined in Chapter 2).

We proceed in two steps. In the first step, from a given automaton we construct a unified game graph, whose edges represent combined moves, i.e., moves of Player 1 and Player 2. In the second step, from this graph we construct a classical game graph, whose edges represent single moves, i.e., moves from either Player 1 or Player 2. After the second step, we obtain the kind of game graph that is used in the literature.

Given a Muller automaton $\mathcal{A} = (Q, \Sigma, q_0, \delta, \mathcal{F})$ recognizing an ω-regular language L.

Step 1. We construct a game on the graph of $\mathcal{A}$. A *play* of this game is an infinite sequence $\rho = q_0 q_1 \dots$ that starts with the designated state q_0, such that for all $i \in \mathbb{N}$ there is an $a \in \Sigma$ with $\delta(q_i, a) = \{q_{i+1}\}$. A play is won by Player 2 if $\mathrm{Inf}(\rho) \in \mathcal{F}$, otherwise it is won by Player 1. A strategy for Player 1 is a mapping $\sigma\colon Q^+ \to \Sigma_1$, and a strategy for Player 2 is a mapping $\tau\colon Q^+ \to \Theta$. A play ρ is consistent with a strategy σ for Player 1 if for all $i \in \mathbb{N}$ there is an $x \in \Sigma_2$ such that $\delta(q_i, (\sigma(\gamma), x)) = q_{i+1}$ where $\gamma = \rho[0, i]$. Analogously, a play ρ is consistent with a strategy τ for Player 2 if for all $i \in \mathbb{N}$ and for all $a \in \Sigma_1$ it holds $\delta(q_i, (a, \vartheta(a))) = q_{i+1}$ where $\vartheta = \tau(\gamma)$ and $\gamma = \rho[0, i]$.

Step 2. A classical game graph can be obtained from the unified model by expanding the state space and splitting the moves by letters of Σ into moves via Σ_1 and Σ_2.

Formally, we construct the game graph $G' = (Q, Q \times \Sigma, E)$ where

$$E \coloneqq \{(q,(q,a)) \in Q \times (Q \times \Sigma) \mid q \in Q, a \in \Sigma_1\} \\ \cup \{((q,a),p) \in (Q \times \Sigma) \times Q \mid p,q \in Q, \exists b \in \Sigma_2 \colon \delta(q,(a,b)) = p\}.$$

The Muller game $\mathcal{G}' = (G', \mathcal{F}')$ is then defined by setting

$$\mathcal{F}' \coloneqq \{P \subseteq Q \cup (Q \times \Sigma) \mid P \cap Q \in \mathcal{F}\}.$$

4.2 Preliminaries: Language Classes below Regular Languages

In the subsequent definitions we recall some basic subclasses of the regular languages, in particular of the star-free languages. For more background see [54, 70].

Piecewise Testable Languages A *basic* PT-*set* is a $*$-language K which can be written as

$$K = \Sigma^* a_1 \Sigma^* a_2 \cdots \Sigma^* a_n \Sigma^*$$

for $a_1, a_2, \ldots, a_n \in \Sigma$. A $*$-language $K \subseteq \Sigma^*$ is *piecewise testable* if it is a Boolean combination of basic PT-sets. We denote the class of piecewise testable languages by PT, and the class of *positive* Boolean combinations of basic PT-sets (in which only $\cup$ and $\cap$ are used) by pos-PT.

For the ω-regular counterparts of the piecewise testable languages there are already some characterizations by Pin [53].

Proposition 31 (Pin [53]) *Let $L \subseteq \Sigma^\omega$. Then the following conditions are equivalent:*

- $L \in \mathrm{BC}(\mathrm{ext}(\text{pos-PT}))$,
- $L = \lim(K)$ *for some* $K \in \mathrm{PT}$,
- $L \in \mathrm{BC}(\lim(\mathrm{PT}))$.

The Dot-Depth Hierarchy A $*$-language $K \subseteq \Sigma^*$ is *generalized definite* if it is a Boolean combination of sets $w\Sigma^*$ and $\Sigma^* w$ with $w \in \Sigma^*$. We denote the class of generalized definite languages by GDEF.

The dot-depth hierarchy, introduced by Cohen and Brzozowski [23], is a sequence of language classes $\mathrm{DD}_0, \mathrm{DD}_1, \ldots$ (over Σ) where

- $\mathrm{DD}_0 \coloneqq \mathrm{GDEF}$, and
- DD_{n+1} contains all Boolean combinations of languages $K = K_1 \cdot K_2 \cdot \ldots \cdot K_\ell$ with $K_1, \ldots, K_\ell \in \mathrm{DD}_n$.

As a special case let us mention the languages of *dot-depth one*. They consist of all Boolean combinations of generalized definite languages, but they can also be characterized as the Boolean combinations of basic DD_1-sets. A *basic* DD_1*-set* is a $*$-language K which can be written as

$$K = w_0\Sigma^* w_1 \Sigma^* \cdots w_{n-1}\Sigma^* w_n$$

for $w_0, w_1, \ldots, w_n \in \Sigma^*$. Note in particular that $\mathrm{PT} \subsetneq \mathrm{DD}_1$. In analogy to the class pos-PT we define pos-DD_1 as the class of positive Boolean combinations of basic DD_1-sets.

For $|\Sigma| > 1$, the dot-depth hierarchy is strict, and it exhausts the class SF of star-free languages [11]. The study of these classes is based on corresponding congruences on Σ^*. We recall these congruences for the case of languages of dot-depth one.

For $k, m \in \mathbb{N}$ and an m-tuple $\nu = (w_1, \ldots, w_m)$ of words w_i of length $|w_i| = k+1$, we say that ν *appears* in a word $u \in \Sigma^*$ if for $1 \leq i \leq m$, the word u can be written as $u = u_i w_i v_i$ with suitable words u_i, v_i such that $1 \leq i < j \leq m$ implies $|u_i| < |u_j|$. With $\mu_{m,k}(w)$ we denote the set of all m-tuples of words of length $k+1$ that appear in w.

Two words $u, v \in \Sigma^*$ are (m,k)*-equivalent* ($u \sim_{m,k} v$) if

1. u and v have the same k first letters,
2. the same m-tuples of words of length $k+1$ appear in u and v, i.e., $\mu_{m,k}(u) = \mu_{m,k}(v)$, and
3. u and v have the same k last letters.

Then we have the following characterization of DD_1 $*$-languages.

Lemma 32 (Simon [65]) *A $*$-language $K \subseteq \Sigma^*$ is of dot-depth one iff it is a union of $\Sigma^*/\!\sim_{m,k}$ equivalence classes for some $m, k \in \mathbb{N}$.*

Note that in the definition of $\sim_{m,k}$ we refer to possibly overlapping infixes, whereas in the definition of basic DD_1-sets the infixes may not overlap. This does not affect the correctness of Lemma 32.

Analogously to the finite word case, we define (m,k)-equivalency for infinite words. We say that ν *appears* in $\alpha \in \Sigma^\omega$ if for $1 \leq i \leq m$, the word α can be written as $u = u_i w_i \alpha_i$ with suitable words u_i, α_i such that $1 \leq i < j \leq m$ implies $|u_i| < |u_j|$. With $\mu_{m,k}(\alpha)$ we denote the set of all m-tuples of words of length $k+1$ that appear in α. Two words $\alpha, \beta \in \Sigma^\omega$ are (m,k)-*equivalent* ($\alpha \sim_{m,k} \beta$) if

1. α and β have the same k first letters, and
2. the same m-tuples of words of length $k+1$ appear in α and β, i.e., $\mu_{m,k}(u) = \mu_{m,k}(v)$.

Analogous to Lemma 32, we obtain a characterization for this new equivalence relation.

Lemma 33 *An ω-language $L \subseteq \Sigma^\omega$ is in* $\mathrm{BC}(\mathrm{ext}(\text{pos-}\mathrm{DD}_1))$ *iff it can be written as a union of $\Sigma^\omega/\sim_{m,k}$ equivalence classes for some $m, k \in \mathbb{N}$.*

Proof Let L be in $\mathrm{BC}(\mathrm{ext}(\text{pos-}\mathrm{DD}_1))$. Then L can be written in disjunctive normal form as

$$L = \bigcup_i L_i = \bigcup_i \bigcap_j L_{ij}$$

where each L_{ij} is of the form $L_{ij} = \mathrm{ext}(K_{ij})$ or $L_{ij} = \overline{\mathrm{ext}(K_{ij})}$ and each K_{ij} is in pos-DD_1, i.e., it is a positive Boolean combination of basic DD_1-sets, i.e., K_{ij} can be written in disjunctive normal form as

$$K_{ij} = \bigcup_r \bigcap_s \left(w_0^{ijrs} \Sigma^* w_1^{ijrs} \Sigma^* \cdots \Sigma^* w_{n_{ijrs}}^{ijrs} \right).$$

One can cast each of the languages L_i into a union of finitely many $\sim_{m,k}$ classes by choosing a division into equivalence classes which is finer than the one imposed by all the basic DD_1-sets. This is done by setting

$$m \coloneqq \max_i \max_j \max_r \max_s (n_{ijrs} + 1), \text{ and}$$
$$k \coloneqq \max_i \max_j \max_r \max_s \max_{0 \leq x \leq n_{ijrs}} |w_x^{ijrs}|.$$

To show the reverse direction, it is sufficient to show that any ω-language $L = [\alpha]_{\sim_{m,k}}$ is in $\mathrm{BC}(\mathrm{ext}(\text{pos-}\mathrm{DD}_1))$. Define

$$K^{\mathrm{pos}} = \bigcup\{[w]_{\sim_{m,k}} \mid w \text{ and } \alpha \text{ have the same prefix of length } k+1 \text{ and } \mu_{m,k}(w) = \mu_{m,k}(\alpha)\}, \text{ and}$$
$$K^{\mathrm{neg}} = \bigcup\{[w]_{\sim_{m,k}} \mid \mu_{m,k}(w) \supsetneq \mu_{m,k}(\alpha)\}$$

Both sets are finite unions, so

$$\begin{aligned} L &= \mathrm{ext}(K^{\mathrm{pos}}) \cap \overline{\mathrm{ext}(K^{\mathrm{neg}})} \\ &= \mathrm{ext}(K_1^{\mathrm{pos}} \cup \ldots \cup K_m^{\mathrm{pos}}) \cap \overline{\mathrm{ext}(K_1^{\mathrm{neg}} \cup \ldots \cup K_n^{\mathrm{neg}})} \\ &= \mathrm{ext}(K_1^{\mathrm{pos}}) \cup \ldots \cup \mathrm{ext}(K_m^{\mathrm{pos}}) \cap \overline{\mathrm{ext}(K_1^{\mathrm{neg}}) \cup \ldots \cup \mathrm{ext}(K_n^{\mathrm{neg}})} \end{aligned}$$

is in $\mathrm{BC}(\mathrm{ext}(\text{pos-}\mathrm{DD}_1))$. □

Many of the classes of special regular languages have natural characterizations in the framework of logic (see e.g. [70]). The starting point of the study of these characterizations is the classical result of Büchi, Elgot, and Trakhtenbrot [13, 76] on the expressive equivalence of finite automata and monadic second-order logic. Here we deal with sublogics for classes of star-free languages; in this case we have an equivalence with sublogics of first-order logic over finite words.

We recall that a language $K \subseteq \Sigma^+$ belongs to DD_n iff it can be defined by a first-order sentence that is a Boolean combination of "Σ_n-sentences" (first-order sentences in prenex normal form with n alternating quantifier blocks starting with an existential block) in $\mathrm{FO}[<, \mathrm{suc}, \mathrm{pre}, \mathrm{min}, \mathrm{max}]$ [73].

The Straubing-Thérien Hierarchy Closely related to the dot-depth hierarchy of star-free languages is the Straubing-Thérien hierarchy [54]. We denote the class of languages at level n of this hierarchy as ST_n. This hierarchy is recursively defined over a given alphabet Σ by

- $\mathrm{ST}_0 \coloneqq \{\emptyset, \Sigma^*\}$, and
- ST_{n+1} contains all Boolean combinations of languages $K_0 a_1 K_1 \ldots a_n K_n$ with $a_i \in \Sigma$ for $i = 1, \ldots, n$ and $K_i \in \mathrm{ST}_n$ for $i = 0, \ldots, n$.

In particular, we have $\mathrm{ST}_1 = \mathrm{PT}$. Analogous to the dot-depth hierarchy, the Straubing-Thérien hierarchy is strict, infinite, and exhausts the class of all star-free languages. It is known that for all $n \in \mathbb{N}, \mathrm{ST}_n \subsetneq \mathrm{DD}_n$, and for all $n > 0$, $\mathrm{DD}_n \subsetneq \mathrm{ST}_{n+1}$ [54].

Henceforth, we prefer to speak of the language class PT instead of ST_1 to emphasize the fact that is also an independently interesting subclass of DD_1 languages.

Similarly to the characterization of the dot-depth hierarchy by formulas in $\mathrm{FO}[<, \mathrm{suc}, \mathrm{pre}, \min, \max]$, there is also a characterization of the Straubing-Thérien hierarchy [51]. A language $K \subseteq \Sigma^*$ belongs to ST_n iff it is definable by a Σ_n-sentence in $\mathrm{FO}[<]$.

4.3 Winning Strategies in Restricted Weak Games

We start with games in $\mathrm{BC}(\mathrm{ext}(\text{pos-}\mathrm{DD}_1))$ which coincides with the class of Boolean combinations of sets $\mathrm{ext}(K)$ where K is a basic DD_1-set, or in other words: Boolean combinations of sets $w_0\Sigma^* w_1\Sigma^* \cdots w_{n-1}\Sigma^* w_n\Sigma^\omega$.

Theorem 34 *Games in* $\mathrm{BC}(\mathrm{ext}(\text{pos-}\mathrm{DD}_1))$ *are determined with winning strategies in* DD_1.

Proof By Lemma 33, we can write an ω-language L in $\mathrm{BC}(\mathrm{ext}(\text{pos-}\mathrm{DD}_1))$ as a union of $\Sigma^\omega/\!\sim_{m,k}$ equivalence classes $L = \bigcup_{i=1}^{n} [\alpha_i]$ where each $\alpha_i \in \Sigma^\omega$. We show how to obtain a game graph with a parity winning condition that captures the game with winning condition L.

In the graph, the play prefix w will lead to the $\sim_{m,k}$-class $[w]$ of w. The game graph consists of the set of nodes $Q = \Sigma^*/\!\sim_{m,k}$. For every $(a, x) \in \Sigma$, we have edges from $[w]$ to $[w(a, x)]$. Note that this relation is well-defined, as from the set of m-tuples of length $k+1$ occurring in w, the suffix of length k of w, and the new letter (a, x), one can determine the set of m-tuples of length $k + 1$ occurring in $w(a, x)$. We designate $q_0 = [\varepsilon]$ as the starting node of a play. For the winning condition, we assign a priority $\chi(q)$ to every node q, namely $\chi([w]) = 2 \cdot |\mu_{m,k}(w)|$ if there is an $\alpha \in L$ such that the prefix of α of length k equals the prefix of w of the same length and $\mu_{m,k}(\alpha) = \mu_{m,k}(w)$; we set $\chi([w]) = 2 \cdot |\mu_{m,k}(w)| - 1$ otherwise. Note that χ is increasing since for $w \in \Sigma^*$, and $(a, x) \in \Sigma$ we have $\chi([w]) \leq \chi([w(a, x)])$. A play is won by Player 2 in the game for L iff the corresponding play in the graph game reaches ultimately an even priority (and stays there).

By the memoryless determinacy of parity games (see Chapter 2) the parity game is determined, and the winning player has a memoryless winning strategy from the starting node q_0. We show that she also has a DD_1 winning strategy in the original game.

Let $\lambda \colon Q \to \Sigma_1$ be a memoryless winning strategy of Player 1 in the parity game. Define $\sigma \colon \Sigma^* \to \Sigma_1$ to be $\sigma(w) = \lambda([w])$. The strategy σ is in DD_1,

because for each $a \in \Sigma_1$ we know that $\sigma^{-1}(a) = \bigcup_{\lambda(w)=a}[w]$ is in DD_1. We still have to show that σ is winning for Player 1 in the game with winning condition L. For this purpose, let $\alpha = (a_0, x_0)(a_1, x_1)(a_2, x_2) \cdots \in \Sigma^\omega$ be consistent with σ. We have to show that $\alpha \notin L$. Then

$$\rho = [\varepsilon], [(a_0, x_0)], [(a_0, x_0)(a_1, x_1)], \ldots$$

is a play in the parity game that is consistent with λ. So Player 1 wins ρ and thus the maximal priority p that occurs infinitely often in ρ is odd. Let $i \in \mathbb{N}$ such that $\chi(\rho(i)) = p$. Then all following positions must have the same priority $p = \chi(\rho(i)) = \chi(\rho(i+1)) = \ldots$, because χ is increasing. This means the set $\mu_{m,k}(w)$ of m-tuples appearing in a word w from $\rho(i)$ does not change from i onwards. So the set of m-tuples of α is $\mu_{m,k}(\alpha) = \mu_{m,k}(w)$ for any $w \in \rho(i)$. Furthermore the prefix of α of length k is equal to the length k prefix of w for any $w \in \rho(i)$. Since p is odd, and by the definition of χ there does not exist such a word $\alpha \in L$, so $\alpha \notin L$. This proves that σ is winning for Player 1.

In the analogous way it is shown that if Player 2 has a memoryless winning strategy in the parity game from q_0, then Player 2 has a DD_1 winning strategy in the game L. □

Next we turn to pos-PT, the class of positive combinations of basic piecewise testable languages; basic piecewise testable languages are languages of the form $\Sigma^* a_1 \Sigma^* a_2 \ldots a_n \Sigma^*$. We show that in this case we can proceed with a much simpler approach that avoids the formation of equivalence classes.

As a preparation we recall a result of Simon [66] about the transition structure of automata that accept piecewise testable languages. For a DFA $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ and any $\Gamma \subseteq \Sigma$, let $G(\mathcal{A}, \Gamma)$ denote the directed graph underlying the automaton $\mathcal{A}$, such that it only retain edges of $\mathcal{A}$ that are labeled with elements of Γ. Note that $G(\mathcal{A}, \Gamma)$ may consist of several unconnected components. Moreover, order all states in Q by letting $p \leq q$ iff q is reachable from p.

Proposition 35 (Simon [66]) *Let $\mathcal{A}$ be the minimal DFA accepting the $*$-language K. Then K is piecewise testable iff*

1. *$G(\mathcal{A}, \Sigma)$ is acyclic, and*
2. *for every $\Gamma \subseteq \Sigma$, each component of $G(\mathcal{A}, \Gamma)$ has a unique maximal state.*

Theorem 36 *Games in* BC(ext(pos-PT)) *are determined with winning strategies in* PT.

Proof For every ω-language $L \in \mathrm{BC}(\mathrm{ext}(\text{pos-PT}))$, there exists a regular language $K \in \mathrm{PT}$ such that $L = \lim(K)$. This is shown easily by induction over Boolean combinations (cf. [53]). The minimal DFA $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ accepting K can be considered as an ω-automaton accepting L. We thus obtain a game graph with a Büchi winning condition. Since Büchi games are determined with memoryless winning strategies, the strategy of the winning player only depends on the current state in the play. Assume, without loss of generality, that Player 2 has a winning strategy. Then for every mapping $\vartheta\colon \Sigma_1 \to \Sigma_2$, let $F_\vartheta \subseteq Q$ be the set of states that induce a choice of ϑ. Consider the automaton $\mathcal{A}_\vartheta = (Q, \Sigma, q_0, \delta, F_\vartheta)$. Since $G(\mathcal{A}_\vartheta, \Sigma) = G(\mathcal{A}, \Sigma)$, we conclude from Proposition 35 that the language accepted by $\mathcal{A}_\vartheta$ is a piecewise testable language. □

It is worth noting that the game graphs for games in $\mathrm{BC}(\mathrm{ext}(\text{pos-PT}))$ are obtained directly from those finite automata that recognize the piecewise testable languages in question, and that piecewise testable winning strategies are obtained by observing a certain property of the associated transition graphs. For games in $\mathrm{BC}(\mathrm{ext}(\text{pos-DD}_1))$ we had to resort to the domain of congruences or, in algebraic terms, to the concept of syntactic monoids. This results in exponentially larger game graphs. In order to avoid this blow-up one might try to apply a result of Stern [69] that gives a property of transition graphs of (minimal) automata which characterizes the languages in DD_1; however, it seems that the necessary step towards obtaining parity games (as done in Theorem 34) spoils this property – which prevents a direct approach like that for pos-PT.

4.4 Winning Strategies in Weak Games

So far we considered winning specifications that were constructed only from positive combinations of $*$-languages. In this section, we study the general setting of weak games. This means that for every class of $*$-languages, we allow arbitrary Boolean combinations of languages from this class as seeds for winning specifications.

Theorem 37 *There are games in* $\mathrm{BC}(\mathrm{ext}(\mathrm{PT}))$*, and hence in* $\mathrm{BC}(\mathrm{ext}(\mathrm{DD}_1))$ *and in* $\mathrm{BC}(\mathrm{ext}(\mathrm{ST}_1))$*, that do not admit* DD_1 *winning strategies.*

Proof Let $\Sigma \coloneqq \{a, b, c, d\} \times \{0, 1\}$, and define $*$-languages $K_1 \coloneqq (a, 0)^*(b, 0)$, $K_2 \coloneqq \Sigma^*(d, 1)\Sigma^*$, and $K_d \coloneqq \Sigma^*(d, 0)\Sigma^* \cup \Sigma^*(d, 1)\Sigma^*$, and for every letter

$x \in \{a,b,c\}$ define $K_x \coloneqq \Sigma^*(x,1)\Sigma^*$. Let L be the ω-language over Σ, that contains all ω-words α such that

$$[\alpha \in \mathrm{ext}(K_d) \to (\alpha \in \mathrm{ext}(K_1) \leftrightarrow \alpha \in \mathrm{ext}(K_2))] \wedge \bigwedge_{x \in \{a,b,c\}} \alpha \in \overline{\mathrm{ext}(K_x)}$$

All the $*$-languages above are in PT, so L is in $\mathrm{BC}(\mathrm{ext}(\mathrm{PT}))$. Since $\mathrm{PT} = \mathrm{ST}_1 \subsetneq \mathrm{DD}_1$, it follows that L is also in $\mathrm{BC}(\mathrm{ext}(\mathrm{DD}_1))$ and in $\mathrm{BC}(\mathrm{ext}(\mathrm{ST}_1))$. We see that L is won by Player 2: she remembers whether a prefix in K_1 has occurred; if so, then she responds to a later occurrence of d with 1, otherwise with 0. We claim there is no DD_1 winning strategy (and a fortiori no PT winning strategy) for Player 2. Assume there is such a winning strategy $\tau\colon \Sigma^* \to \Theta$, which is implemented by a DD_1 Moore machine. Then there are $*$-languages $K_{\vartheta_1}, \ldots, K_{\vartheta_n}$ of dot-depth one, implementing this strategy. In particular, $K_{d \mapsto 1} \coloneqq \{w \mid \tau(w)(d) = 1\}$ is a dot-depth one language as a finite union of K_{ϑ_i} languages. So it is a finite union of equivalence classes $[w_i]_{\sim_{m,k}}$.

Let $w \in \{a,b,c\}^*$ be a word in which all possible m-tuples of length $k+1$ over the alphabet $\{a,b,c\}$ appear. Such a word exists, since there are only finitely many of such tuples, and concatenating words w_1 and w_2 with sets T_1 and T_2 of tuples yields a word containing $T_1 \cup T_2$. Let Player 1 play a strategy σ_1 that chooses $a^k b \cdot w \cdot d \cdot a^\omega$. Consider the unique word α_1 that is consistent with both σ_1 and τ. Since τ is a winning strategy for Player 2 and σ_1 plays $a^k b$ in the beginning, we have $(d,1)$ occurring in α_1. So the word $a^k bw \times \{0\}$ is in K_{d1}.

Now let Player 1 play the strategy σ_2 which chooses $a^k c \cdot w \cdot d \cdot a^\omega$. The word $a^k c \cdot w$ contains all possible m-tuples of length $k+1$ over the alphabet $\{a,b,c\}$, as well. Then we have $a^k bw \sim_{m,k} a^k cw$ and thus $a^k cw \times \{0\} \in K_{d1}$. Then the unique word α_2 that is consistent with both σ_2 and τ contains $(d,1)$ as an infix. This contradicts that τ is a winning strategy for Player 2.□

Theorem 38 *For each $i \in \mathbb{N}$, games in $\mathrm{BC}(\mathrm{ext}(\mathrm{DD}_i))$ are determined with winning strategies in DD_{i+1}.*

Proof Any regular language $K_\ell \in \mathrm{DD}_i$ is recognized by a deterministic automaton $A_\ell = (Q, \Sigma, q_0, \delta, F)$ such that for every $q \in Q$, the language $[w]_q \coloneqq \{w \in \Sigma^* \mid q_0 \xrightarrow{w} q\}$ is in DD_i. Then, preserving the transition structure of A_ℓ, we obtain a Staiger-Wagner automaton $\mathcal{A}_\ell$ which accepts the ω-language $L_\ell = \mathrm{ext}(K_\ell)$. For any Boolean combination of languages $\mathrm{ext}(K_\ell)$, we obtain a Staiger-Wagner automaton $\mathcal{A}$ by constructing the product automaton from the automata for $\mathrm{ext}(K_\ell)$. Since DD_i is closed

under Boolean combinations, for $\mathcal{A}$, it again holds that $[w]_q \in \mathrm{DD}_i$ for each state q.

Therefore, for a winning condition $L \in \mathrm{BC}(\mathrm{ext}(\mathrm{DD}_i))$, we obtain a game graph where every node is a DD_i equivalence class (and hence a $*$-language in DD_i). The game graph is equipped with a Staiger-Wagner winning condition.

As explained in Chapter 2, we transform the game graph via the AR construction to a new game graph with the parity winning condition. A state in the new game graph is a pair (q, R) consisting of a state $q \in \Sigma^*/\sim_{\mathrm{DD}_i}$ of the original graph and an AR R. Over this game graph (with the parity winning condition), the winner has a memoryless winning strategy. We have to show that each node (q, R) corresponds to a DD_{i+1} language in the sense that the play prefixes leading to (q, R) form a language $K_{(q,R)}$ in DD_{i+1}. Then, it will immediately follow that the play prefixes that cause the winner to choose a fixed letter a are obtained as a union of languages $K_{(q,R)}$ and hence in DD_{i+1} as desired.

To reason about $K_{(q,R)}$, it is convenient to apply the logical characterization of DD_i-languages as mentioned in Section 4.2. Each vertex q corresponds to the language K_q consisting of play prefixes leading to q. Each K_q is a language in DD_i, defined by a Boolean combination ψ_q of Σ_i-sentences in $\mathrm{FO}[<, \mathrm{suc}, \mathrm{pre}, \min, \max]$. We now have to express the restriction that a play prefix leads to state q *and* to the AR $R \subseteq Q$ for the state space Q. This is formalized by the sentence

$$\varphi_{(q,R)} = \psi_q \wedge \bigwedge_{r \in R} \exists x \psi_r[x] \wedge \bigwedge_{r \notin R} \neg \exists x \psi_r[x],$$

where, for every $r \in Q$, the formula $\psi_r[x]$ is obtained from ψ_r by relativisation with respect to the segment $[0, x]$ (see Chapter 2). Thus $\exists x \psi_r[x]$ describes all words that have a prefix that satisfies ψ_r.

Since ψ_r is a Boolean combination of Σ_i-sentences, we obtain (in prenex form) a Boolean combination of Σ_{i+1}-sentences. In this way we obtain the membership of $K_{(q,R)}$ in DD_{i+1}. □

Theorem 39 *For each $i \in \mathbb{N}$, games in $\mathrm{BC}(\mathrm{ext}(\mathrm{ST}_i))$ are determined with winning strategies in ST_{i+1}.*

Proof Analogously to Theorem 38, for a language $L \in \mathrm{BC}(\mathrm{ext}(\mathrm{ST}_i))$, we can obtain a game graph with Staiger-Wagner winning condition where every node is a ST_i equivalence class (and hence a $*$-language in ST_i). For each node, this language can be described by a Boolean combination of Σ_i-sentences in $\mathrm{FO}[<]$. The remainder of the proof is similar to that of Theorem 38. □

4.5 Winning Strategies in Strong Games

In [63, 59] the general result by Büchi and Landweber was refined to the class of star-free games, i.e., games with star-free winning conditions are determined with star-free winning strategies. In this section, we refine this result even further, as the games considered here constitute the class of all star-free games.

Theorem 40 *Games in* $\mathrm{BC}(\lim(\mathrm{PT}))$ *are determined with winning strategies in* PT.

Proof For every ω-language $L \in \mathrm{BC}(\lim(\mathrm{PT}))$, there exists a regular language $K \in \mathrm{PT}$ such that $L = \lim(K)$ (see [53]). The remainder of this proof is identical to that of Theorem 36. □

Theorem 41 *There are games in the class* $\mathrm{BC}(\lim(\mathrm{DD}_0))$ *that do not admit* DD_1 *winning strategies.*

Proof Let $\Sigma \coloneqq \{a, b\} \times \{0, 1\}$. Consider DD_0 languages $K_{a,0} \coloneqq \Sigma^*(a, 0)$, $K_{a,1} \coloneqq \Sigma^*(a, 1)$, $K_{b,0} \coloneqq \Sigma^*(b, 0)$, and $K_{b,1} \coloneqq \Sigma^*(b, 1)$. The winning condition is an ω-language

$$L \coloneqq \overline{\lim(K_{a,0})} \cap \big[\overline{\lim(K_{b,0})} \cup \lim(K_{b,1})\big] \cap \big[\lim(K_{b,0}) \cup \overline{\lim(K_{b,1})}\big],$$

which states that a word $\alpha \in \Sigma^\omega$ belongs to L if and only if, first, α contains only finitely many occurrences of $(a, 0)$, and second, $(b, 0)$ occurs infinitely often in α if and only if $(b, 1)$ does. Thus $L \in \mathrm{BC}(\lim(\mathrm{DD}_0))$.

Player 2 has a winning strategy which can be described in two parts as follows. First, whenever Player 1 plays a, respond with 1. Second, remember the previous response to Player 1's choice of b, and flip that response if Player 1 chooses to play b again. This ensures that for any play $\alpha \in \Sigma^\omega$ consistent with this strategy, first, $\alpha \notin \lim(K_{a,0})$, and second, $(b, 0)$ and $(b, 1)$ appear alternately in α. Indeed, Player 2's strategy maps finite play prefixes to functions $\vartheta_0, \vartheta_1 \in \Theta$, where $\vartheta_0 = \{a \mapsto 1, b \mapsto 0\}$, $\vartheta_1 = \{a \mapsto 1, b \mapsto 1\}$, and

$$K_{\vartheta_0} = \Sigma^*(b, 1) \cup \big(\Sigma^*(b, 1) \cdot \overline{\Sigma^*(b, 0)\Sigma^* \cup \Sigma^*(b, 1)\Sigma^*} \cdot (a, 1)\big)$$
$$K_{\vartheta_1} = \Sigma^*(b, 0) \cup \big(\Sigma^*(b, 0) \cdot \overline{\Sigma^*(b, 0)\Sigma^* \cup \Sigma^*(b, 1)\Sigma^*} \cdot (a, 1)\big)$$

It is straightforward to verify that $K_{\vartheta_0}, K_{\vartheta_1} \in \mathrm{DD}_2$. Now we show that Player 2 has no DD_1 winning strategy. On the contrary, assume that Player 2 has such a strategy $\tau\colon \Sigma^* \to \Theta$. In particular, for $\vartheta \in \Theta$ there is a

language $K_\vartheta \in \mathrm{DD}_1$ such that for every play prefix $u \in K_\vartheta, \tau(u) = \vartheta$, and K_ϑ is described by a finite set of m_ϑ-tuples of words of length $k_\vartheta + 1$ for $m_\vartheta, k_\vartheta \in \mathbb{N}$. We define $k_\tau \coloneqq \max\{k_\vartheta \mid \vartheta \in \Theta\}$.

For Player 1, we fix the strategy σ that involves playing ba^{k_τ} ad infinitum. Now we consider the play $\alpha \in \Sigma^\omega$ that is consistent with σ and τ. Since τ is a winning strategy, α must be winning for Player 2.

Since $\alpha \notin \lim(K_{a,0})$, there must be a smallest position $\ell_a \in \mathbb{N}$ such that for any $j \geq \ell_a$, $\alpha[\ell_a, j]$ does not contain the letter $(a, 0)$. For each $\vartheta \in \Theta$, since there are only finitely many m_ϑ-tuples of words of length $k_\vartheta + 1$, there exists a smallest position $\ell_\vartheta \in \mathbb{N}$ after which no new m_ϑ-tuple is seen in α. That is, the same tuples appear in the prefix $\alpha[0, j]$ for every $j \geq \ell_\vartheta$.

Finally, we define ℓ_τ as the smallest multiple of $k_\tau+1$ greater than or equal to $\max\{\ell_a, \ell_\vartheta \mid \vartheta \in \Theta\}$, and consider the prefix $w_\tau \coloneqq \alpha[0, \ell_\tau]$. Without loss of generality, we assume that $\tau(w_\tau) = \vartheta$ and $\vartheta(b) = 0$. It is easy to see that $w_\tau \sim_{m_\vartheta, k_\vartheta} w_\tau(b,0)(a,1)^{k_\tau} \sim_{m_\vartheta, k_\vartheta} w_\tau(b,0)(a,1)^{k_\tau}(b,0)(a,1)^{k_\tau} \cdots \in K_\vartheta$. Therefore, after w_τ, τ always responds to b's with 0's, leading to infinite occurrences of $(b, 0)$ but not of $(b, 1)$. This contradicts α being a winning play for Player 2. □

While one may construct strong games at any level of the dot-depth hierarchy such that there are no winning strategies at the same level, the final result shows that there are winning strategies at most two levels higher in the hierarchy. Whether winning strategies are also located in the level between remains open.

Theorem 42 *For each $i \in \mathbb{N}$, games in $\mathrm{BC}(\lim(\mathrm{DD}_i))$ are determined with winning strategies in DD_{i+2}.*

Proof We proceed as in the proof of Theorem 38. We first construct a graph where every node is a DD_i equivalence class – a $*$-language in DD_i. Now, for languages $K \in \mathrm{DD}_i$, we are given a game over the ω-language $L \in \mathrm{BC}(\lim(K))$. We obtain the game graph for L when we equip the graph constructed above with a Muller winning condition. As explained in Section 2.6, we transform this game graph via the LAR construction into a new game graph with a parity winning condition. A state in the new game graph is a pair (q, R) consisting of a state $q \in \Sigma^*/\sim_{\mathrm{DD}_i}$ and an LAR R. Over this parity game graph, the winner has a memoryless winning strategy.

We know that each vertex q collects the play prefixes that belong to a language $K_q \in \mathrm{DD}_i$, defined by a Boolean combination ψ_q of Σ_i-sentences (cf. Section 2.7). Using these formulas, we now express the fact that each

play prefix leading to any state (q, R) in the parity game graph forms a language $K_{(q,R)} \in \mathrm{DD}_{i+2}$.

Given a permutation *perm* of the state space of the original Muller game, and an index h, an LAR can be defined as $R = (perm, h)$. Let the sentence φ_R express the fact that a play prefix has led to R. It is evident that $K_{(q,R)}$ is the set of words that satisfy the formula φ_R. In order to avoid overloaded notation, we only provide a description for an example: the most recent prefix types in *perm* are q, r, s, in that order; the index value is $h = 3$. With φ_R, we express that the most recent prefix types are q, r, s in this order and that for the previous prefix this sequence is r, s, q: (1) the current play prefix (at position max) is q, at the previous position is r, and any preceding position that is not occupied by r is occupied by s, and (2) for the play prefix at position pre(max) the most recent play prefixes are in r, s, q in this order. This can be described as

$$\begin{aligned}\varphi_R = \psi_q[\max] \wedge \psi_r[\mathrm{pre}(\max)] \\ \wedge \exists x, y, z\big(\max > x > y > z \wedge \psi_r[x] \wedge \psi_s[y] \wedge \psi_q[z] \\ \wedge \forall x_1(\max > x_1 > x \to \psi_r[x_1]) \\ \wedge \forall y_1(x > y_1 > y \to \psi_s[y_1])\big),\end{aligned}$$

where, for every $q \in Q$, the formula $\psi_q[x]$ is obtained from ψ_q by relativisation with respect to the segment $[0, x]$ (see Chapter 2).

Since ψ_q, ψ_r, and ψ_s are Boolean combinations of Σ_i-sentences, we obtain a Boolean combination of Σ_{i+2}-sentences (in prenex normal form) which is equivalent to φ_R. Thus, we obtain languages $K_{(q,R)} \in \mathrm{DD}_{i+2}$. □

Theorem 43 *For each $i \in \mathbb{N}$, games in $\mathrm{BC}(\lim(\mathrm{ST}_i))$ are determined with winning strategies in ST_{i+2}.*

Proof As with Theorem 39, for a language $L \in \mathrm{BC}(\lim(\mathrm{ST}_i))$, we start with constructing a game graph with Muller winning conditions where every node is a ST_i equivalence class. For each node, this language can be described by a Boolean combination of Σ_i-sentences in FO[<]. Then the sentence describing the LAR as in the proof of Theorem 42 can be written analogously, without the constant "max", as follows. Note that this constant can be defined in terms of $\leq$, and, for the sake of readability, we use variables x_{max} and x_{max-1} to stand in place of max and pre(max)

respectively.

$$\begin{aligned}\varphi_R = \exists x_{\max}, x_{\max-1}(&\forall y(y \leq x_{\max}) \wedge \forall y(y < x_{\max} \rightarrow y \leq x_{\max-1}) \\ &\wedge \psi_q[x_{\max}] \wedge \psi_r[x_{\max-1}] \\ &\wedge \exists x, y, z(x_{\max} > x > y > z \wedge \psi_r[x] \wedge \psi_s[y] \wedge \psi_q[z] \\ &\qquad \wedge \forall x_1(x_{\max} > x_1 > x \rightarrow \psi_r[x_1]) \\ &\qquad \wedge \forall y_1(x > y_1 > y \rightarrow \psi_s[y_1]))).\end{aligned}$$

This sentence is a Boolean combination of Σ_{i+2} sentences, and we infer the existence of a winning strategy in ST_{i+2} as in Theorem 42. □

4.6 Conclusion

In the present chapter, we bridged the gap between languages and strategies. We extended the original question by Büchi and Landweber how simple winning strategies for a given class of games are, by connecting the complexity of strategies to the complexity of languages. Such connections already exist for the class of regular languages [14], and the class of star-free languages [59, 63], in particular on the level of logical characterizations. We continued these studies for subclasses of the star-free languages. We used the operators ext() and lim() in order to reach a class of ω-languages from the starting point of a class of $*$-languages. Although this approach leads to language classes of lower complexity than the already mentioned classes of regular and star-free languages, the results we obtained are more intricate than the results on regular languages.

We showed that weak games in the dot-depth hierarchy as well as in the Straubing-Thérien hierarchy, i.e., games with winning conditions in $\mathrm{BC}(\mathrm{ext}(\mathrm{DD}_i))$ and $\mathrm{BC}(\mathrm{ext}(\mathrm{ST}_i))$, are determined with winning strategies in DD_{i+1} and ST_{i+1}, respectively. For the special cases of winning conditions in $\mathrm{BC}(\mathrm{ext}(\mathcal{K}))$ where $\mathcal{K}$ is the class PT of piecewise testable languages or the class DD_1 of languages of dot-depth one, this bound is strict, i.e., these games do not admin winning strategies in $\mathcal{K}$ but only in DD_2.

If we restrict the input of the ext()-operator to only positive combinations of basic piecewise testable languages or to positive combinations of basic dot-depth one languages, then we obtain determinacy of these games with piecewise testable strategies or DD_1-strategies, respectively.

For the lim()-operator, we obtained the general result that strong games in the considered hierarchies, i.e., games definable in $\mathrm{BC}(\mathrm{lim}(\mathrm{DD}_i))$ and $\mathrm{BC}(\mathrm{lim}(\mathrm{ST}_i))$, are determined with winning strategies in DD_{i+2} and ST_{i+2},

respectively. For the particular case of strong games in $\mathrm{BC}(\lim(\mathrm{DD}_1))$ then follows that these games have winning strategies in DD_3. We additionally showed that they do not admit winning strategies in DD_1. The class of piecewise testable languages exhibits a special property for strong games, as games in $\mathrm{BC}(\lim(\mathrm{PT}))$ have winning strategies in PT again.

Among the problems remaining open is obviously the question whether the bound of $i + 2$ of Theorems 42 and 43 can be improved to $i + 1$. The author's personal opinion is that this is not the case, but this is difficult to prove: e.g., even if there would exist a game $L \in \mathrm{BC}(\lim(\mathrm{DD}_1))$ such that each winning strategy of L would not be in DD_2, this would be hard to show. To date, there are no effective characterizations known, neither for DD_2 nor for any dot-depth level beyond that. Another, rather general problem is to study complexity issues of variant other types, e.g., one could compare the sizes of automata for game representations and for strategy representations. Finally, the results of this chapter motivate other abstract frameworks for switching between classes of $*$-languages to corresponding classes of ω-languages and back. These could be used primarily for the purpose of studying classes beyond the star-free languages.

Chapter 5

Permissive Strategies

In Chapter 4 we applied the language-theoretic framework to games for the purpose of measuring their complexity. More precisely, we assessed the complexity of a game belonging to class $\mathcal{K}$ by putting its winning strategy into a class closely related to $\mathcal{K}$. This approach led to a qualitative notion for the complexity of a game carried over from the complexity of its winning strategy. The purpose of the present chapter is to present a quantitative notion for the complexity of a strategy. We choose an approach based on the popular model of parity games (see Section 2.6).

We consider a kind of non-deterministic strategy (cf. [3, 5]) that we name *multi-strategy*. A multi-strategy is a generalization of the notion of a strategy in the following sense: while a strategy selects exactly one action to be played in response to the current play prefix, a multi-strategy can retain one or more possible actions. This yields several possible outcomes for a given multi-strategy of the controller (Player 1) and a given strategy of the environment (Player 2). Utilizing multi-strategies has two advantageous applications for the synthesis of winning strategies. First, it makes the multi-strategy robust against possible errors or inaccuracies in the model of the system: if a certain action is disabled, possibly for a short time, the multi-strategy does not fail but instead it inherently proposes an alternative action, which also leads to a winning play. Second, it offers a modular approach for the problem of solving parity games: when we combine several multi-strategies, we obtain a refined multi-strategy, which is ideally winning for all of the original specifications. For both applications, non-deterministic strategies prove useful only for Player 1 and not for Player 2, since we assume that the (possibly hostile) environment plays perfectly in all situations. That is why we allow multi-strategies only for Player 1.

The quantitative complexity measure we are after is the one of *permissiveness*. We consider a strategy to be more permissive if it allows more behaviors. The more permissive a multi-strategy is, i.e., the more behaviors it allows, the better it is suited for the above-mentioned applications.

In the prevalent literature, a non-deterministic strategy is more *permissive* than a second one if it allows a superset of outcomes compared to the second non-deterministic strategy. Bernet, Janin, and Walukiewicz [3] showed that under this qualitative notion, a most permissive winning strategy is not guaranteed to exist. Hence, we follow the quantitative approach by defining a measure that specifies how permissive a given multi-strategy is. To that end, we equip each edge of the game graph with a weight, which we treat as a *penalty* that is imposed when Player 1 blocks this edge. The sum of these penalties, accumulated in each step, clearly diverges when the play coninues infinitely long. Therefore, we define the penalty of a multi-strategy as the average sum of accumulated penalties in the limit (or, alternatively, as a discounted sum of accumulated penalties). The lower this penalty is, the more permissive is the given multi-strategy. Our aim is to answer the following questions: Does a most permissive multi-strategy exist for every game? Can we compute the maximal permissiveness of a given game?

In this chapter, we integrate results from [7, 8, 9]. We solve a game with penalties by transforming it into a *mean-payoff game* [26, 84] with deterministic strategies. By this game reduction, a move in the latter game corresponds to a set of moves in the former, and is assigned a (negative) *payoff* depending on the penalty of the original move. In a previous work, Bouyer, Duflot, Markey, and Renault [5] introduced the notion of penalties and showed how to compute permissive strategies for games with reachability objectives. In the present chapter, we extend the study of [5] to parity objectives. Using the above transformation, we reduce the problem of finding a most permissive strategy in a parity game with penalties to that of computing an optimal strategy in a *mean-payoff parity game*, which combines a mean-payoff objective with a parity objective.

In Section 5.1 we consider mean-payoff parity games. While these games have already been studied [18, 4, 16], we propose a modern proof that these games are determined and that both players have optimal strategies. Moreover, we prove that the second player does not only have an optimal strategy with finite memory, but one that uses no memory at all. Finally, we provide a new algorithm for computing the values of a mean-payoff parity game, which is faster than the best known algorithms for this problem.

In Section 5.2 we define the model of parity games with penalties, which we call *mean-penalty parity games*. We prove the existence of a most permissive multi-strategy, and we show that the existence of a multi-strategy whose penalty is less than a given threshold can be decided in $\mathrm{NP} \cap \mathrm{coNP}$. Finally, we adapt our deterministic algorithm for mean-payoff parity games to parity games with penalties. Our algorithm computes the penalties

of a most permissive multi-strategy in time exponential in the number of priorities and polynomial in the size of the game graph and the largest penalty.

There are also other notions of permissiveness [3, 55], but for these a most permissive strategy is not guaranteed to exist. Multi-strategies have also been used for improving strategy iteration algorithms [44].

Parity games admit optimal memoryless strategies for both players, and the problem of deciding the winner is in NP ∩ coNP. As of this writing, it is not known whether parity games can be solved in polynomial time; the best known algorithms run in time polynomial in the size of the game graph but exponential in the number of priorities.

Another fundamental class of games are games with quantitative objectives. Mean-payoff games, where the aim is to maximize the average weight of the transitions taken in a play, are also in NP ∩ coNP and admit memoryless optimal strategies [26, 84]. The same is true for *energy games*, where the aim is to always keep the sum of the weights above a given threshold [15, 6]. In fact, parity games can easily be reduced to mean-payoff or energy games [39]. Several other game models have recently appeared in the literature that mix several qualitative or quantitative objectives: apart from mean-payoff parity games, these include generalized parity games [19], energy parity games [16] and lexicographic mean-payoff (parity) games [4] as well as generalized energy and mean-payoff games [17].

5.1 A new Algorithm for Mean-payoff Parity Games

In this section, we establish that mean-payoff parity games are determined, that both players have optimal strategies, that for Player 2 even memoryless strategies suffice, and that the value problem for mean-payoff parity games is in NP ∩ coNP. Furthermore, we present a deterministic algorithm which computes the values in time exponential in the number of priorities, and runs in pseudo-polynomial time when the number of priorities is bounded.

5.1.1 Definitions

A *weighted game graph* is a tuple $G = (Q_1, Q_2, E, \text{weight})$, where (Q_1, Q_2, E) is a game graph, and $\text{weight}\colon E \to \mathbb{R}$ is a function assigning a *weight* to every transition. All notions for game graphs, like *subarena*, *play*, *strategy*, carry over to weighted game graphs.

When weighted game graphs are subject to algorithmic processing, we assume that all weights are integers, and we set $W := \max\{1, |\text{weight}(e)| \mid$

$e \in E\}$. Moreover, we define the *size* of G, denoted by $\|G\|$, as $|Q| + |E| \cdot \lceil \log_2 W \rceil$. (Up to a linear factor, $\|G\|$ is the length of a binary encoding of G). In the same spirit, the size $\|x\|$ of a rational number x equals the total length of the binary representations of its numerator and its denominator.

Formally, a *mean-payoff parity game* is a tuple $\mathcal{G} = (G, \chi)$, where G is a weighted game graph, and $\chi \colon Q \to \mathbb{N}$ is a priority function assigning a *priority* to every state. A play $\rho = \rho(0)\rho(1)\cdots$ is *parity-winning* if the minimal priority occurring infinitely often in ρ is even, i.e., if $\min\{\chi(q) \mid q \in \mathrm{Inf}(\rho)\} \equiv 0 \pmod 2$. All notions that we have defined for weighted game graphs carry over to mean-payoff parity games. In particular, a play of $\mathcal{G}$ is just a play of G and a strategy for Player i in $\mathcal{G}$ is nothing but a strategy for Player i in G. Hence, we write $\mathrm{Out}^{\mathcal{G}}(\sigma, q)$ for $\mathrm{Out}^{G}(\sigma, q)$, and so on. As for weighted games graphs, we often omit the superscript if $\mathcal{G}$ is clear from the context. Finally, for a mean-payoff parity game $\mathcal{G} = (G, \chi)$ and a subarena S of G, we write $\mathcal{G} \restriction S$ for the mean-payoff parity game $(G \restriction S, \chi \restriction S)$.

We say that a mean-payoff parity game $\mathcal{G} = (G, \chi)$ is a *mean-payoff game* if $\chi(q)$ is even for all $q \in Q$. In particular, given a weighted game graph G, we obtain a mean-payoff game by assigning priority 0 to all states. We denote this game by $(G, 0)$.

If $\chi(Q) \subseteq \{0, 1\}$, then we say that $\mathcal{G}$ is a *mean-payoff Büchi game*; if $\chi(Q) \subseteq \{1, 2\}$, we call it a *mean-payoff co-Büchi game*. Hence, in a Büchi game Player 1 needs to visit the set $\chi^{-1}(0)$ infinitely often, whereas in a co-Büchi game he has to visit the set $\chi^{-1}(1)$ only finitely often.

For a play ρ of $\mathcal{G}$, we define its *payoff* as

$$\mathrm{payoff}^{\mathcal{G}}(\rho) = \begin{cases} \liminf_{n\to\infty} \mathrm{payoff}^{\mathcal{G}}_n(\rho) & \text{if } \rho \text{ is parity-winning,} \\ -\infty & \text{otherwise,} \end{cases}$$

where for $n \in \mathbb{N}$

$$\mathrm{payoff}^{\mathcal{G}}_n(\rho) = \begin{cases} \dfrac{1}{n} \displaystyle\sum_{i=0}^{n-1} \mathrm{weight}(\rho(i), \rho(i+1)) & \text{if } n > 0, \\ -\infty & \text{if } n = 0. \end{cases}$$

If σ is a strategy for Player 1 in $\mathcal{G}$, we define its *value* from $q_0 \in Q$ as

$$\mathrm{val}^{\mathcal{G}}(\sigma, q_0) = \inf_\tau \mathrm{payoff}^{\mathcal{G}}(\rho(\sigma, \tau, q_0)) = \inf\{\mathrm{payoff}^{\mathcal{G}}(\rho) \mid \rho \in \mathrm{Out}^{\mathcal{G}}(\sigma, q_0)\},$$

where τ ranges over all strategies of Player 2 in $\mathcal{G}$. Analogously, the value of a strategy τ for Player 2 from q_0 is defined as

$$\mathrm{val}^{\mathcal{G}}(\tau, q_0) = \sup_\sigma \mathrm{payoff}^{\mathcal{G}}(\rho(\sigma, \tau, q_0)) = \sup\{\mathrm{payoff}^{\mathcal{G}}(\rho) \mid \rho \in \mathrm{Out}^{\mathcal{G}}(\tau, q_0)\},$$

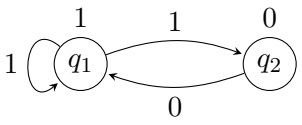


Figure 5.1: A mean-payoff parity game for which infinite memory is necessary.

where σ ranges over all strategies of Player 1 in $\mathcal{G}$. The *lower* and *upper value* of a state $q_0 \in Q$ are defined by

$$\underline{\mathrm{val}}^{\mathcal{G}}(q_0) = \sup_\sigma \mathrm{val}^{\mathcal{G}}(\sigma, q_0) \qquad \text{and} \qquad \overline{\mathrm{val}}^{\mathcal{G}}(q_0) = \inf_\tau \mathrm{val}^{\mathcal{G}}(\tau, q_0),$$

respectively. Intuitively, $\underline{\mathrm{val}}^{\mathcal{G}}(q_0)$ and $\overline{\mathrm{val}}^{\mathcal{G}}(q_0)$ are the maximal (respectively minimal) payoff that Player 1 (respectively Player 2) can ensure (in the limit). We say that a strategy σ of Player 1 is *optimal from* q_0 if $\mathrm{val}^{\mathcal{G}}(\sigma, q_0) = \underline{\mathrm{val}}^{\mathcal{G}}(q_0)$. Analogously, we call a strategy τ of Player 2 optimal from q_0 if $\mathrm{val}^{\mathcal{G}}(\tau, q_0) = \overline{\mathrm{val}}^{\mathcal{G}}(q_0)$. A strategy is *(globally) optimal* if it is optimal from every state $q \in Q$. It is easy to see that $\underline{\mathrm{val}}^{\mathcal{G}}(q_0) \leq \overline{\mathrm{val}}^{\mathcal{G}}(q_0)$. If $\underline{\mathrm{val}}^{\mathcal{G}}(q_0) = \overline{\mathrm{val}}^{\mathcal{G}}(q_0)$, we say that q_0 has a *value*, which we denote by $\mathrm{val}^{\mathcal{G}}(q_0)$.

In the next section, we will see that mean-payoff games are *determined*, i.e., that every state has a value. The *value problem* is the following decision problem: Given a mean-payoff parity game $\mathcal{G}$ (with integral weights), a designated state $q_0 \in Q$, and a number $x \in \mathbb{Q}$, decide whether $\mathrm{val}^{\mathcal{G}}(q_0) \geq x$.

Example *Consider the mean-payoff parity game $\mathcal{G}$ depicted in Figure 5.1, where a state or an edge is labeled with its priority, respectively weight; all states belong to Player 1. Note that* $\mathrm{val}^{\mathcal{G}}(q_1) = 1$ *since Player 1 can delay visiting q_2 longer and longer while still ensuring that this vertex is seen infinitely often. However, there is no finite-memory strategy that achieves this value.*

Let σ be a finite-memory strategy of Player 1 in $\mathcal{G}$, and let ρ be the unique play of $\mathcal{G}$ that starts in q_1 and is consistent with σ. Assume furthermore that ρ visits q_2 infinitely often (otherwise $\mathrm{val}^{\mathcal{G}}(\sigma, q_1) = -\infty$*). Then $\rho = {q_1}^{k_1} q_2 {q_1}^{k_2} q_2 \cdots$, where each $k_i \in \mathbb{N} \setminus \{0\}$. Since σ is a finite-memory strategy, there exists $m \in \mathbb{N}$ such that $k_i \leq m$ for all $i \in \mathbb{N}$. Hence,* $\mathrm{val}^{\mathcal{G}}(\sigma, q_q) = \mathrm{payoff}(\rho) \leq m/(m+1) < 1$.

5.1.2 Strategy Complexity

It follows from Martin's determinacy theorem [45] that mean-payoff parity games are determined. Moreover, Chatterjee, Henzinger, and Jurdziński [18] gave an algorithmic proof for the existence of optimal strategies. Finally, it can be shown that for every $x \in \mathbb{R} \cup \{-\infty\}$ the set $\{\rho \in Q^\omega \mid \text{payoff}(\rho) \geq x\}$ is closed under *combinations*. By Theorem 4 in [43], this property implies that Player 2 even has a memoryless optimal strategy. We give here a purely inductive proof of these facts that does not rely on Martin's theorem. We start by proving that Player 1 has an optimal strategy in games where Player 2 is absent.

Lemma 44 *Let $\mathcal{G}$ be a mean-payoff parity game with $Q_2 = \emptyset$. Then Player 1 has an optimal strategy in $\mathcal{G}$.*

Proof Since $Q_2 = \emptyset$, every state q_0 has a value $\text{val}(q_0) = \underline{\text{val}}(q_0)$, so it suffices to construct for each $q_0 \in Q$ a strategy σ with $\text{val}^{\mathcal{G}}(\sigma, q_0) \geq \text{val}^{\mathcal{G}}(q_0)$. If $\text{val}^{\mathcal{G}}(q_0) = -\infty$, we can choose an arbitrary strategy σ. Otherwise, by the definition of $\underline{\text{val}}^{\mathcal{G}}(q_0)$, for each $\varepsilon > 0$ there exists a play $\rho_\varepsilon \in \text{Out}^{\mathcal{G}}(q_0)$ with $\text{payoff}(\rho_\varepsilon) \geq \text{val}^{\mathcal{G}}(q_0) - \varepsilon$. Consider the sets $\text{Inf}(\rho_\varepsilon)$ of states occurring infinitely often in ρ_ε. Since there are only finitely many such sets, we can find a set $P \subseteq Q$ such that for each $\varepsilon > 0$ there exists $0 < \varepsilon' < \varepsilon$ with $P = \text{Inf}(\rho_{\varepsilon'})$. Let $q_{\min} \in P$ be a vertex of lowest priority. (This priority must be even since each ρ_ε fulfils the parity condition).

Let σ_1 be an optimal memoryless strategy in the mean-payoff game $\mathcal{G}_P = (G \restriction P, 0)$ (the strategy σ_1 just leads the play to a simple cycle with maximum average weight), and let σ_2 be the memoryless attractor strategy in the game $\mathcal{G}_P$ that ensures a visit to $q_{\min}$ from all states $q \in P$; we extend both strategies to a strategy in $\mathcal{G}$ by combining them with a memoryless attractor strategy for P. (In particular, σ_2 enforces a visit to $q_{\min}$ from q_0.) Note that $\text{val}^{\mathcal{G}_P}(q) \geq \text{val}^{\mathcal{G}}(q_0)$ for all $q \in P$ since each of the plays $\rho_{\varepsilon'}$ visits each vertex in P and has payoff $\geq \text{val}^{\mathcal{G}}(q_0) - \varepsilon'$.

Player 1's optimal strategy σ is played in rounds: in the ith round, Player 1 first forces a visit to $q_{\min}$ by playing according to σ_2; once $q_{\min}$ has been visited, Player 1 plays σ_1 for i steps before proceeding to the next round. Note that $\text{val}^{\mathcal{G}_P}(\sigma, q_{\min}) = \text{val}^{\mathcal{G}_P}(\sigma_1, q_{\min})$. Moreover, the unique play $\rho \in \text{Out}^{\mathcal{G}}(\sigma, q_0)$ satisfies $q_{\min} \in \text{Inf}(\rho) \subseteq P$ and therefore fulfils the parity condition. To sum up, we have $\text{val}^{\mathcal{G}}(\sigma, q_0) = \text{val}^{\mathcal{G}}(\sigma, q_{\min}) = \text{val}^{\mathcal{G}_P}(\sigma, q_{\min}) = \text{val}^{\mathcal{G}_P}(\sigma_1, q_{\min}) = \text{val}^{\mathcal{G}_P}(q_{\min}) \geq \text{val}^{\mathcal{G}}(q_0)$. □

Using Lemma 44, we can prove that mean-payoff-parity games are not only determined, but also that Player 1 has an optimal strategy and that

Player 2 has a memoryless optimal strategy.

We use the *loop factorization* technique (cf. [83]): Let γ be a play prefix and let $\hat{q} \in Q$. The *loop factorization of γ relative to $\hat{q}$* is the unique factorization of the form $\gamma = \gamma_0\gamma_1 \cdots \gamma_l$, where γ_0 does not contain $\hat{q}$, and each factor γ_i, $1 \leq i \leq l$, is of the form $\gamma_i = \hat{q} \cdot \gamma_i'$ where γ_i' does not contain $\hat{q}$. Analogously, for a play ρ which has infinitely many occurrences of $\hat{q}$ the *loop factorization of ρ relative to $\hat{q}$* is the unique factorization $\rho = \gamma_0\gamma_1 \cdots$ where each γ_i has the same properties as in the above case.

For a state $\hat{q}$ with m successors, $\hat{q}E = \{q_1, \ldots, q_m\}$, we define an operator $\pi_i \colon Q^* \to Q^*$ for each $1 \leq i \leq m$ by setting

$$\pi_i(\gamma) \coloneqq \begin{cases} \gamma & \text{if either } \gamma = \hat{q}q_i\gamma' \text{ for some } \gamma' \in Q^* \text{ or } \gamma = q_i = \hat{q}, \\ \varepsilon & \text{otherwise.} \end{cases}$$

The operator π_i induces another operator $\Pi_i \colon Q^* \to Q^*$ by setting

$$\Pi_i(\gamma) = \pi_i(\gamma_0)\pi_i(\gamma_1) \cdots \pi_i(\gamma_l),$$

where $\gamma = \gamma_0\gamma_1 \cdots \gamma_l$ is the loop factorization of γ relative to $\hat{q}$. The operator Π_i operates on play prefixes, but it can easily be extended to operate on infinite plays with infinitely many occurrences of $\hat{q}$.

Theorem 45 *Let $\mathcal{G}$ be a mean-payoff parity game.*

1. *$\mathcal{G}$ is determined;*
2. *Player 1 has an optimal strategy in $\mathcal{G}$;*
3. *Player 2 has a memoryless optimal strategy in $\mathcal{G}$.*

Proof We proceed by an induction over the size of $S \coloneqq \{q \in Q_2 \mid |qE| > 1\}$, the set of all Player 2 states with more than one successor. If $S = \emptyset$, all statements follow from Lemma 44. Let 1.–3. be fulfilled for all games with $|S| < n$ and let $\mathcal{G} = (G, \chi)$ be a mean-payoff parity game with $|S| = n$. We prove that the statements also hold for $\mathcal{G}$. Let $\hat{q} \in S$ with $\hat{q}E = \{q_1, \ldots, q_m\}$. For each $1 \leq j \leq m$, we define a new game $\mathcal{G}_j = (G_j, \chi)$ by setting $E_j = E \setminus (\{\hat{q}\} \times Q) \cup \{(\hat{q}, q_j)\}$, and $G_j = (Q_1, Q_2, E_j, \text{weight} \restriction E_j)$. Note that the induction hypothesis applies to each $\mathcal{G}_j$. W.l.o.g. assume that $\mathrm{val}^{\mathcal{G}_1}(\hat{q}) \leq \mathrm{val}^{\mathcal{G}_j}(\hat{q})$ for all $1 \leq j \leq m$. We will construct a memoryless strategy τ for Player 2 and a strategy σ for Player 1 such that $\mathrm{val}^{\mathcal{G}}(\tau, q_0) \leq \mathrm{val}^{\mathcal{G}_1}(q_0)$ and $\mathrm{val}^{\mathcal{G}}(\sigma, q_0) \geq \mathrm{val}^{\mathcal{G}_1}(q_0)$ for every $q_0 \in Q$. Hence,

$$\mathrm{val}^{\mathcal{G}_1}(q_0) \leq \mathrm{val}^{\mathcal{G}}(\sigma, q_0) \leq \underline{\mathrm{val}}^{\mathcal{G}}(q_0) \leq \overline{\mathrm{val}}^{\mathcal{G}}(q_0) \leq \mathrm{val}^{\mathcal{G}}(\tau, q_0) \leq \mathrm{val}^{\mathcal{G}_1}(q_0),$$

and all these numbers are equal. In particular, we have $\mathrm{val}^{\mathcal{G}}(q_0) = \underline{\mathrm{val}}^{\mathcal{G}}(q_0) = \overline{\mathrm{val}}^{\mathcal{G}}(q_0)$, $\mathrm{val}^{\mathcal{G}}(\sigma, q_0) = \mathrm{val}^{\mathcal{G}}(q_0)$ and $\mathrm{val}^{\mathcal{G}}(\tau, q_0) = \mathrm{val}^{\mathcal{G}}(q_0)$, which proves 1.–3.

By the induction hypothesis, Player 2 has a memoryless optimal strategy τ in $\mathcal{G}_1$. Clearly, τ is also a memoryless strategy for Player 2 in $\mathcal{G}$, and $\mathrm{val}^{\mathcal{G}}(\tau, q_0) = \mathrm{val}^{\mathcal{G}_1}(\tau, q_0) = \mathrm{val}^{\mathcal{G}_1}(q_0)$ for all $q_0 \in Q$.

It remains to construct a strategy σ for Player 1 in $\mathcal{G}$ such that $\mathrm{val}^{\mathcal{G}}(\sigma, q_0) \geq \mathrm{val}^{\mathcal{G}_1}(q_0)$ for all $q_0 \in Q$.

First, we devise a strategy $\hat{\sigma}$ such that $\mathrm{val}^{\mathcal{G}}(\hat{\sigma}, \hat{q}) \geq \mathrm{val}^{\mathcal{G}_1}(\hat{q})$. If $\mathrm{val}^{\mathcal{G}_1}(\hat{q}) = -\infty$, we can take an arbitrary strategy. Hence, assume that $\mathrm{val}^{\mathcal{G}_1}(\hat{q})$ is finite. By the induction hypothesis, for each $j = 1, \ldots, m$ there exists a strategy σ_j for Player 1 in $\mathcal{G}_j$ with $\mathrm{val}^{\mathcal{G}_j}(\sigma_j, \hat{q}) = \mathrm{val}^{\mathcal{G}_j}(\hat{q})$. We define $\hat{\sigma}$ to be the *interleaving strategy*, defined by

$$\hat{\sigma}(\gamma) = \hat{\sigma}(\gamma_0 \cdots \gamma_l) = \begin{cases} \sigma_1(\Pi_1(\gamma)) & \text{if } \gamma_l = \hat{q} q_1 \gamma' \text{ for some } \gamma' \in Q^*, \\ \vdots & \vdots \\ \sigma_m(\Pi_m(\gamma)) & \text{if } \gamma_l = \hat{q} q_m \gamma' \text{ for some } \gamma' \in Q^*, \end{cases}$$

for all play prefixes γ whose loop factorization relative to $\hat{q}$ equals $\gamma_0 \cdots \gamma_l$. We claim that $\mathrm{val}^{\mathcal{G}}(\hat{\sigma}, \hat{q}) \geq \mathrm{val}^{\mathcal{G}_1}(\hat{q})$.

Let $\rho \in \mathrm{Out}^{\mathcal{G}}(\hat{\sigma}, \hat{q})$. If ρ has only finitely many occurrences of $\hat{q}$, then ρ is equivalent to a play in $\mathcal{G}_j$ that is consistent with σ_j for some j. Since $\mathrm{val}^{G_j}(\hat{q}) \geq \mathrm{val}^{G_1}(\hat{q})$ and σ_j is optimal, $\mathrm{payoff}(\rho) \geq \mathrm{val}^{G_1}(\hat{q})$, and we are done. Otherwise, consider the loop factorization $\rho = \gamma_0 \gamma_1 \cdots$ and set

$$\Gamma = \{j \in \{1, \ldots, m\} \mid \gamma_i \cdot \hat{q} \text{ is a loop in } \mathcal{G}_j \text{ for infinitely many } i \in \mathbb{N}\}.$$

Since the mean-payoff parity condition is prefix-independent, we can assume w.l.o.g. that every loop in ρ is a loop in $\mathcal{G}_j$ for $j \in \Gamma$. For each $j \in \Gamma$, denote by $\rho_j = \Pi_j(\rho)$ the corresponding play in $\mathcal{G}_j$. By definition of $\hat{\sigma}$, we have $\rho_j \in \mathrm{Out}^{\mathcal{G}_j}(\sigma_j, \hat{q})$ for each $j \in \Gamma$. Since $\mathrm{val}^{\mathcal{G}_1}(\hat{q})$ is finite and $\mathrm{val}^{\mathcal{G}_1}(\hat{q}) \leq \mathrm{val}^{\mathcal{G}_j}(\hat{q})$, each ρ_j fulfils the parity condition. As the minimal priority occurring infinitely often in ρ also occurs infinitely often in one ρ_j, this implies that ρ fulfils the parity condition.

We claim that for each $n > 0$, $\mathrm{payoff}_n(\rho)$ is a weighted average of $\mathrm{payoff}_{n_j}(\rho_j)$ for some $n_j > 0$. To see this, consider the loop factorization $\gamma'_0 \cdots \gamma'_k$ of $\rho[0, n]$. (Note that $\gamma'_i = \gamma_i$ for all $i < k$.) For each $j \in \Gamma$, set

$$n_j = \begin{cases} |\Pi_j(\rho[0, n])| - 1 & \text{if } \gamma'_k \text{ is a history of } \mathcal{G}_j \text{ and either } \gamma'_k \neq \hat{q} \text{ or } q_j = \hat{q}. \\ |\Pi_j(\rho[0, n])| & \text{otherwise.} \end{cases}$$

Intuitively, n_j is the number of transitions in $\rho[0,n]$ that correspond to a transition in ρ_j. Hence,

$$\{(\rho(i),\rho(i+1)) \mid 0 \le i < n\} = \bigcup_{j\in\Gamma} \{(\rho_j(i),\rho_j(i+1)) \mid 0 \le i < n_j\}.$$

In particular, $\sum_{j\in\Gamma} n_j = n$ and $\sum_{j\in\Gamma} n_j/n = 1$. We have

$$\begin{aligned}
\operatorname{payoff}_n(\rho) &= \frac{1}{n}\sum_{i=0}^{n-1} \operatorname{weight}(\rho(i),\rho(i+1)) \\
&= \frac{1}{n}\sum_{\substack{j\in\Gamma\\ n_j>0}} \sum_{i=0}^{n_j-1} \operatorname{weight}(\rho_j(i),\rho_j(i+1)) \\
&= \sum_{\substack{j\in\Gamma\\ n_j>0}} \frac{n_j}{n}\cdot\frac{1}{n_j}\sum_{i=0}^{n_j-1} \operatorname{weight}(\rho_j(i),\rho_j(i+1)) \\
&= \sum_{\substack{j\in\Gamma\\ n_j>0}} \frac{n_j}{n}\cdot \operatorname{payoff}_{n_j}(\rho_j).
\end{aligned}$$

Since a weighted average is always bounded from below by the minimum element, we can conclude that

$$\operatorname{payoff}_n(\rho) \ge \min_{\substack{j\in\Gamma\\ n_j>0}} \operatorname{payoff}_{n_j}(\rho_j) \ge \min_{j\in\Gamma} \operatorname{payoff}_{n_j}(\rho_j).$$

Taking the lower limit on both sides, we obtain

$$\begin{aligned}
\operatorname{payoff}(\rho) &= \liminf_{n\to\infty} \operatorname{payoff}_n(\rho) \\
&\ge \liminf_{n\to\infty} \min_{j\in\Gamma} \operatorname{payoff}_{n_j}(\rho_j) \\
&= \min_{j\in\Gamma} \liminf_{n\to\infty} \operatorname{payoff}_{n_j}(\rho_j) \\
&= \min_{j\in\Gamma} \liminf_{n_j\to\infty} \operatorname{payoff}_{n_j}(\rho_j) \\
&= \min_{j\in\Gamma} \operatorname{payoff}(\rho_j).
\end{aligned}$$

Since each ρ_j is consistent with σ_j and σ_j is optimal, we have $\operatorname{payoff}(\rho_j) \ge \operatorname{val}^{\mathcal{G}_j}(\hat{q}) \ge \operatorname{val}^{\mathcal{G}_1}(\hat{q})$ for each $j \in \Gamma$ and therefore also $\operatorname{payoff}(\rho) \ge \operatorname{val}^{\mathcal{G}_1}(\hat{q})$. Since this holds for all $\rho \in \operatorname{Out}^{\mathcal{G}}(\hat{\sigma},\hat{q})$, we can conclude that $\operatorname{val}^{\mathcal{G}}(\hat{\sigma},\hat{q}) \ge \operatorname{val}^{\mathcal{G}_1}(\hat{q})$.

Finally, we construct a strategy σ for Player 1 in $\mathcal{G}$ such that $\mathrm{val}^{\mathcal{G}}(\sigma, q_0) \geq \mathrm{val}^{\mathcal{G}_1}(q_0)$ for all $q_0 \in Q$. Let

$$\sigma(\gamma) = \begin{cases} \sigma_1(\gamma) & \text{if } \hat{q} \text{ does not occur in } \gamma, \\ \hat{\sigma}(\hat{q}\gamma_2) & \text{if } \gamma = \gamma_1\hat{q}\gamma_2 \text{ with } \gamma_1 \in (Q \setminus \{\hat{q}\})^*. \end{cases}$$

Then for each play $\rho \in \mathrm{Out}^{\mathcal{G}}(\sigma, q_0)$ where $\hat{q}$ does not occur, it holds $\mathrm{payoff}^{\mathcal{G}}(\rho) = \mathrm{payoff}^{\mathcal{G}_1}(\rho) \geq \mathrm{val}^{\mathcal{G}_1}(\sigma_1, q_0) = \mathrm{val}^{\mathcal{G}_1}(q_0)$. If $\hat{q}$ occurs in at least one play consistent with σ, then in the game $\mathcal{G}_1$ (where σ_1 is optimal), we have $\mathrm{val}^{\mathcal{G}_1}(q_0) = \mathrm{val}^{\mathcal{G}_1}(\sigma_1, q_0) \leq \mathrm{val}^{\mathcal{G}_1}(\hat{q})$. Hence, for each play $\rho \in \mathrm{Out}^{\mathcal{G}}(\sigma, q_0)$ where $\hat{q}$ occurs (say at position j), it holds $\mathrm{payoff}^{\mathcal{G}}(\rho) = \mathrm{payoff}^{\mathcal{G}}(\rho[j, \infty)) \geq \mathrm{val}^{\mathcal{G}}(\hat{\sigma}, \hat{q}) \geq \mathrm{val}^{\mathcal{G}_1}(\hat{q}) \geq \mathrm{val}^{\mathcal{G}_1}(q_0)$. Altogether we have $\mathrm{payoff}^{\mathcal{G}}(\rho) \geq \mathrm{val}^{\mathcal{G}_1}(q_0)$ for every play $\rho \in \mathrm{Out}^{\mathcal{G}}(\sigma, q_0)$ and therefore it holds $\mathrm{val}^{\mathcal{G}}(\sigma, q_0) \geq \mathrm{val}^{\mathcal{G}_1}(q_0)$. □

5.1.3 Computational Complexity

In this section, we prove that the value problem for mean-payoff parity games lies in NP ∩ coNP. Although this has already been proved by Chatterjee and Doyen [16], our proof has the advantage that it works immediately on mean-payoff parity games, and not on energy parity games as in [16].

In order to put the value problem for mean-payoff parity games into coNP, we first show that the value can be decided in polynomial time in games where Player 2 is absent.

Proposition 46 *The problem of deciding, given a mean-payoff parity game $\mathcal{G}$ with $Q_2 = \emptyset$, a state $q_0 \in Q$, and $x \in \mathbb{Q}$, whether* $\mathrm{val}^{\mathcal{G}}(q_0) \geq x$, *is in P.*

Proof Deciding whether $\mathrm{val}^{\mathcal{G}}(q_0) \geq x$ is achieved by Algorithm 1, which employs as subroutines Tarjan's linear-time algorithm [24] for SCC decomposition and Karp's polynomial-time algorithm [41] for computing the minimum/maximum cycle weight, (i.e., the minimum/maximum average weight on a cycle) in a given strongly connected graph.

The algorithm is sound: If the algorithm accepts, then there is an even priority p and a reachable SCC C in G_p with $p \in \chi(C)$ that has maximum cycle weight $w \geq x$. We construct a strategy σ for Player 1 with $\mathrm{val}^{\mathcal{G}}(\sigma, q_0) = w$. Let $q \in C$ be a state with priority p. Since q is reachable from q_0 and C is strongly connected, both q_0 and C lie inside $\mathrm{Attr}_1(\{q\})$. Let σ_q be the memoryless attractor strategy for $\{q\}$. Now, since w is the maximum cycle weight in C, there exists a simple cycle $\gamma = q_1 \cdots q_n q_1$ in C with cycle weight w. We construct a (memoryless) strategy σ_γ on C by

Input : mean-payoff parity game $\mathcal{G}$ with $Q_2 = \emptyset$, $q_0 \in Q$, $x \in \mathbb{Q}$.
Output : whether $\mathrm{val}^{\mathcal{G}}(q_0) \geq x$.

$G' = G \restriction \{q \in Q \mid q \text{ is reachable from } q_0\}$
for each even $p \in \chi(Q)$ **do**
  $G_p = G' \restriction \{q \in Q \mid \chi(q) \geq p\}$
  decompose G_p into SCCs
  for each SCC C of G_p with $p \in \chi(C)$ **do**
    compute maximum cycle weight w in C
    if $w \geq x$ **then accept**
  end
end
reject

Algorithm 1: A polynomial-time algorithm for deciding the value of a state in a one-player mean-payoff parity game.

setting $\sigma_\gamma(q_n) = q_1$ and $\sigma_\gamma(q_i) = q_{i+1}$ for every $1 \leq i < n$; this strategy is extended to the whole game by combining it with an attractor strategy for $\{q_1, \ldots, q_n\}$. The strategies σ_q and σ_γ are then combined to a strategy σ, which is played in rounds: in the ith round, Player 1 first forces a visit to $\chi^{-1}(p) \cap C$ by playing according to σ_q; once $\chi^{-1}(p) \cap C$ has been reached, Player 1 plays σ_γ for i steps before proceeding to the next round. Note that σ fulfils the parity condition because q is visited infinitely often and all other priorities that appear infinitely often obey $\chi(q) \geq p$. Finally, the payoff of $\rho(\sigma, q_0)$ equals the cycle weight of γ, i.e., $\mathrm{val}^{\mathcal{G}}(q_0) \geq \mathrm{val}^{\mathcal{G}}(\sigma, q_0) = w \geq x$.

The algorithm is complete: Assume that $\mathrm{val}^{\mathcal{G}}(q_0) = v \geq x$ and let $\rho \in \mathrm{Out}^{\mathcal{G}}(q_0)$ be a play with $\mathrm{payoff}^{\mathcal{G}}(\rho) = v$; such a play exists due to Lemma 44. Consider the set $\mathrm{Inf}(\rho)$ and let $p = \min \chi(\mathrm{Inf}(\rho))$ (which is even since $\mathrm{payoff}(\rho)$ is finite). Since $\mathrm{Inf}(\rho)$ is strongly connected, $\mathrm{Inf}(\rho) \subseteq C$ for an SCC C of G_p with $p \in \chi(C)$. Since optimal memoryless strategies exist in mean-payoff games [26], there exists a simple cycle with average weight $\geq v$ in C. Hence the algorithm accepts.

Since SCC decomposition and maximum cycle weight computation both take polynomial time, the whole algorithm runs in polynomial time. □

It follows from Theorem 45 and Proposition 46 that the value problem for mean-payoff parity games is in coNP: to decide whether $\mathrm{val}^{\mathcal{G}}(q_0) < x$, a nondeterministic algorithm can guess a memoryless strategy τ for Player 2

and check whether $\mathrm{val}^{\mathcal{G}}(\tau, q_0) < x$ in polynomial time.

Corollary 47 *The value problem for mean-payoff parity games is in coNP.*

Following ideas from [16], we prove that the value problem is not only in coNP, but also in NP. The core of Algorithm 2 is the procedure Check that

Input : mean-payoff parity game $\mathcal{G}$, state $q_0 \in Q$, $x \in \mathbb{Q}$.
Output : whether $\mathrm{val}^{\mathcal{G}}(q_0) \geq x$.

guess 2-trap T in $\mathcal{G}$ with $q_0 \in T$
Check(T)
accept

procedure Check(S)
  if $S \neq \emptyset$ **then**
    $p \coloneqq \min\{\chi(q) \mid q \in S\}$
    if p is even **then**
      guess memoryless strategy σ_{M} for Player 1 in $G \restriction S$
      if $\mathrm{val}^{(G\restriction S,0)}(\sigma_{\mathrm{M}}, q) < x$ for some $q \in S$ **then reject**
      Check($S \setminus \mathrm{Attr}_1^{\mathcal{G}\restriction S}(\chi^{-1}(p))$)
    else
      guess 2-trap $T \neq \emptyset$ in $\mathcal{G} \restriction (S \setminus \mathrm{Attr}_2^{\mathcal{G}\restriction S}(\chi^{-1}(p)))$
      Check(T)
      Check($S \setminus \mathrm{Attr}_1^{\mathcal{G}\restriction S}(T)$)
    end
  end
end procedure

Algorithm 2: A nondeterministic algorithm for deciding the value of a state in a mean-payoff parity game.

on input S checks whether the value of all states in the game $\mathcal{G} \restriction S$ is at least x. If the least priority p in S is even, this is witnessed by a strategy in the mean-payoff game $(G \restriction S, 0)$ that ensures payoff $\geq x$ and the fact that the values of all states in the game $\mathcal{G} \restriction S \setminus \mathrm{Attr}_1^{\mathcal{G}\restriction S}(\chi^{-1}(p))$ are greater than x, which we can check by calling Check recursively. If, on the other hand, the least priority p in S is odd, then $\mathrm{val}^{\mathcal{G}\restriction S}(q) \geq x$ for all $q \in S$ is witnessed by a 2-trap T inside $S \setminus \mathrm{Attr}_2^{\mathcal{G}\restriction S}(\chi^{-1}(p))$ such that both the values in the game $\mathcal{G} \restriction T$ and the values in the game $\mathcal{G} \restriction S \setminus \mathrm{Attr}_1^{\mathcal{G}\restriction S}(T)$ are

bounded from below by x; the latter two properties can again be checked by calling Check recursively. The correctness of the algorithm relies on the following two lemmas.

Lemma 48 *Let $\mathcal{G}$ be a mean-payoff parity game with least priority p even, $T = Q \setminus \mathrm{Attr}_1(\chi^{-1}(p))$, and $x \in \mathbb{R}$. If $\mathrm{val}^{(G,0)}(q) \geq x$ for all $q \in Q$ and $\mathrm{val}^{\mathcal{G}\restriction T}(q) \geq x$ for all $q \in T$, then $\mathrm{val}^{\mathcal{G}}(q) \geq x$ for all $q \in Q$.*

Proof Assume that $\mathrm{val}^{(G,0)}(q) \geq x$ for all $q \in Q$ and $\mathrm{val}^{\mathcal{G}\restriction T}(q) \geq x$ for all $q \in T$, and let $q^* \in Q$. By Theorem 45, it suffices to show that for every memoryless strategy τ of Player 2 there exists a strategy σ of Player 1 such that $\mathrm{payoff}(\rho(\sigma, \tau, q^*)) \geq x$. Hence, assume that τ is a memoryless strategy of Player 2 in $\mathcal{G}$. Moreover, let σ_{M} be a memoryless strategy for Player 1 in $(G, 0)$ with $\mathrm{val}^{(G,0)}(\sigma_{\mathrm{M}}, q) \geq x$ for all $q \in Q$, let σ_T be a strategy for Player 1 in $\mathcal{G} \restriction T$ with $\mathrm{val}^{\mathcal{G}\restriction T}(\sigma_T, q) \geq x$ for all $q \in T$, and let σ_{A} be a memoryless attractor strategy of Player 1 on $\mathrm{Attr}_1(\chi^{-1}(p))$ that ensures to reach $\chi^{-1}(p)$. We combine these three strategies to a new strategy σ, which is played in rounds. In the kth round, the strategy behaves as follows:

1. while the play stays inside T, play σ_T;
2. as soon as the play reaches $\mathrm{Attr}_1(\chi^{-1}(p))$, switch to strategy σ_{A} and play σ_{A} until the play reaches $\chi^{-1}(p)$;
3. when the play reaches $\chi^{-1}(p)$, play σ_M for exactly k steps and proceed to the next round.

Let $\rho \coloneqq \rho(\sigma, \tau, q^*)$. To complete the proof, we need to show that $\mathrm{payoff}(\rho) \geq x$. We distinguish whether ρ visits $\mathrm{Attr}_1(\chi^{-1}(p))$ infinitely often or not.

In the first case, we divide ρ into $\rho = \gamma_0\gamma_1\gamma_2\cdots$ where each $\gamma_i = \gamma_i^T\gamma_i^{\mathrm{A}}\gamma_i^{\mathrm{M}}$ consists of a part consistent with σ_T (thus staying inside T), a part consistent with σ_{A} (thus staying in $\mathrm{Attr}_1(\chi^{-1}(p))$), and one that starts with a state in $\chi^{-1}(p)$ and is consistent with σ_{M}. Since τ is a memoryless strategy, there can only be $|T|$ many different γ_i^T, and the length of each γ_i^T is bounded by some constant k. Since each γ_i^{A} is consistent with an attractor strategy, the length of each γ_i^{A} is bounded by $|Q|$. Hence, the length of γ_i^{M} grows continuously while the length of $\gamma_i^T\gamma_i^{\mathrm{A}}$ is bounded. Therefore, $\liminf_{n\to\infty} \mathrm{payoff}_n(\rho) = \liminf_{n\to\infty} \mathrm{payoff}_n(\gamma_1^{\mathrm{M}}\gamma_2^{\mathrm{M}}\cdots)$. Since $\mathrm{val}^{(G,0)}(\sigma_{\mathrm{M}}, q) \geq x$ for all $q \in Q$ and priority p is visited infinitely often, we have $\mathrm{payoff}(\rho) = \liminf_{n\to\infty} \mathrm{payoff}_n(\rho) \geq x$.

In the second case, $\rho = \gamma \cdot \rho'$, where ρ' is a play of $\mathcal{G} \restriction T$ that is consistent with σ_T. Hence, $\mathrm{payoff}(\rho) = \mathrm{payoff}(\rho') \geq \mathrm{val}^{\mathcal{G}\restriction T}(\sigma_T, \rho'(0)) \geq x$. □

Lemma 49 *Let $\mathcal{G}$ be a mean-payoff parity game with least priority p odd, $T = Q \setminus \mathrm{Attr}_2(\chi^{-1}(p))$, and $x \in \mathbb{R}$. If $\mathrm{val}^{\mathcal{G}}(q) \geq x$ for some $q \in Q$, then $T \neq \emptyset$ and $\mathrm{val}^{\mathcal{G}\restriction T}(q) \geq x$ for some $q \in T$.*

Proof Let $q^* \in Q$ be a state with value $\mathrm{val}^{\mathcal{G}}(q^*) \geq 0$. If $T = \emptyset$, then $\mathrm{Attr}_2(\chi^{-1}(p)) = Q$ and there is a memoryless attractor strategy τ for Player 2 in $\mathcal{G}$ that ensures to visit $\chi^{-1}(p)$ infinitely often. This implies $\mathrm{val}^{\mathcal{G}}(\tau, q^*) = -\infty$, a contradiction to $\mathrm{val}^{\mathcal{G}}(q^*) \geq x$. Thus $T \neq \emptyset$.

Now assume that $\mathrm{val}^{\mathcal{G}\restriction T}(q) < x$ for all $q \in T$, and let τ be a (w.l.o.g. memoryless) strategy for Player 2 in $\mathcal{G} \restriction T$ that ensures $\mathrm{val}^{\mathcal{G}\restriction T}(\tau, q) < x$ for all $q \in T$. We extend τ to a strategy τ' in $\mathcal{G}$ by combining it with a memoryless attractor strategy for $\chi^{-1}(p)$ on the states in $Q \setminus T$. Let $\rho \in \mathrm{Out}^{\mathcal{G}}(\tau', q^*)$. Either ρ reaches $\chi^{-1}(p)$ infinitely often, in which case $\mathrm{payoff}^{\mathcal{G}}(\rho) = -\infty$, or there is a position i from which onwards ρ stays in T, in which case $\mathrm{payoff}^{\mathcal{G}}(\rho) = \mathrm{payoff}^{\mathcal{G}\restriction T}(\rho[i,\infty)) \leq \mathrm{val}^{\mathcal{G}\restriction T}(\tau, \rho(i))$. In any case, $\mathrm{val}^{\mathcal{G}}(\tau', q^*) \leq \max_{q \in T} \mathrm{val}^{\mathcal{G}\restriction T}(\tau, q) < x$, a contradiction to $\mathrm{val}^{\mathcal{G}}(q^*) \geq x$. □

Finally, Algorithm 2 runs in polynomial time because the value of a memoryless strategy in a mean-payoff game can be computed in polynomial time [84] and because recursive calls are limited to disjoint subarenas.

Theorem 50 *The value problem for mean-payoff parity games is in NP.*

Proof We claim that Algorithm 2 is a nondeterministic polynomial-time algorithm for the value problem. To analyse the running time, denote by $T(n)$ the worst-case running time of the procedure Check on a subarena S of size n. Since the value of a memoryless strategy for Player 1 in a mean-payoff game can be computed in polynomial time [84] and attractor computations take linear time, there exists a polynomial $f\colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ such that the numbers $T(n)$ satisfy the following recurrence:

$$T(1) \leq f(\|G\|, \|x\|),$$
$$T(n) \leq \max_{1 \leq k < n} T(k) + T(n-k) + f(\|G\|, \|x\|)\,.$$

Solving this recurrence, we get that $T(n) \leq (2n-1) f(\|G\|, \|x\|)$ for all $n \geq 1$, again a polynomial. Consequently, the algorithm runs in polynomial time.

To prove the correctness of the algorithm, we need to prove that the algorithm is both sound and complete. We start by proving soundness: If the algorithm accepts its input, then $\mathrm{val}^{\mathcal{G}}(q_0) \geq x$. In fact, we prove the following stronger statement. We say that Check(S) *succeeds* if the procedure terminates without rejection (for at least one sequence of guesses).

Claim 51 *Let $S \subseteq Q$. If S is a subarena of $\mathcal{G}$ and* $\mathrm{Check}(S)$ *does succeed, then* $\mathrm{val}^{\mathcal{G}\restriction S}(q) \geq x$ *for all* $q \in S$.

Assume that the claim is true and that the algorithm accepts its input. Then there exists a 2-trap T with $q_0 \in T$ such that $\mathrm{val}^{\mathcal{G}\restriction T}(q) \geq x$ for all $q \in T$. Since T is a 2-trap, it follows that $\mathrm{val}^{\mathcal{G}}(q_0) \geq x$.

To prove the claim, we proceed by induction over the cardinality of S. If $|S| = 0$, the claim is trivially fulfilled. Hence, assume that $|S| > 0$ and that the claim is true for all sets $S' \subseteq Q$ with $|S'| < |S|$. Let $p = \min\{\chi(q) \mid q \in S\}$. We distinguish two cases:

1. The minimal priority p is even. Since $\mathrm{Check}(S)$ succeeds, there exists a memoryless strategy σ_{M} of Player 1 in $\mathcal{G} \restriction S$ such that $\mathrm{val}^{(G\restriction S,0)}(\sigma_{\mathrm{M}}, q) \geq x$ for all $q \in S$, i.e., $\mathrm{val}^{(G\restriction S,0)}(q) \geq x$ for all $q \in S$. Let $A = \mathrm{Attr}_1^{\mathcal{G}\restriction S}(\chi^{-1}(p))$. Since $\mathrm{Check}(S)$ succeeds, so does $\mathrm{Check}(S \setminus A)$. Hence, by the induction hypothesis, $\mathrm{val}^{\mathcal{G}\restriction(S\setminus A)}(q) \geq x$ for all $q \in S \setminus A$. By Lemma 48, these two facts imply that $\mathrm{val}^{\mathcal{G}\restriction S}(q) \geq x$ for all $q \in S$.

2. The minimal priority p is odd. Since $\mathrm{Check}(S)$ succeeds, there exists a 2-trap $T \neq \emptyset$ in $\mathcal{G} \restriction (S \setminus \mathrm{Attr}_2^{\mathcal{G}\restriction S}(\chi^{-1}(p)))$ such that both $\mathrm{Check}(T)$ and $\mathrm{Check}(S \setminus \mathrm{Attr}_1^{\mathcal{G}\restriction S}(T))$ succeed. Let $A = \mathrm{Attr}_1^{\mathcal{G}\restriction S}(T))$. By the induction hypothesis, Player 1 has a strategy σ_T in $\mathcal{G} \restriction T$ such that $\mathrm{val}^{\mathcal{G}\restriction T}(\sigma_T, q) \geq x$ for all $q \in T$ and a strategy σ_S in $\mathcal{G} \restriction S \setminus A$ such that $\mathrm{val}^{\mathcal{G}\restriction S\setminus A}(\sigma_S, q) \geq x$ for all $q \in S \setminus A$. We extend σ_T to a strategy σ_A in $\mathcal{G} \restriction A$ such that $\mathrm{val}^{\mathcal{G}\restriction A}(\sigma_A, q) \geq x$ for all $q \in A$ by combining σ_T with a suitable attractor strategy. By playing σ_S as long as the play stays in $S \setminus A$ and switching to σ_A as soon as the play enters A, Player 1 can ensure that $\mathrm{val}^{\mathcal{G}\restriction S}(q) \geq x$ for all $q \in S$.

Finally, we prove that the algorithm is complete: if $\mathrm{val}^{\mathcal{G}}(q_0) \geq x$, then the algorithm accepts the input $\mathcal{G}, q_0, x$. Since the set $\{q \in Q \mid \mathrm{val}^{\mathcal{G}}(q) \geq x\}$ is a trap for Player 2, it suffices to prove the following claim.

Claim 52 *Let $S \subseteq Q$. If S is a subarena of $\mathcal{G}$ and* $\mathrm{val}^{\mathcal{G}\restriction S}(q) \geq x$ *for all* $q \in S$, *then* $\mathrm{Check}(S)$ *succeeds.*

As the previous claim, we prove this claim by an induction over the cardinality of S. Clearly, $\mathrm{Check}(S)$ succeeds if $|S| = 0$. Hence, assume that $|S| > 0$ and that the claim is correct for all sets $S' \subseteq Q$ with $|S'| < |S|$. Moreover, assume that S is a subarena of $\mathcal{G}$ such that $\mathrm{val}^{\mathcal{G}\restriction S}(q) \geq x$ for all $q \in S$ (otherwise the claim is trivially fulfilled). Again, we distinguish whether $p \coloneqq \min\{\chi(q) \mid q \in S\}$ is even or odd.

1. The minimal priority p is even. Since $\mathrm{val}^{\mathcal{G}\restriction S}(q) \geq x$ for all $q \in S$, also $\mathrm{val}^{(G\restriction S,0)}(q) \geq x$ for all $q \in S$, which is witnessed by a memoryless strategy σ_{M}. Let $A = \mathrm{Attr}_1^{\mathcal{G}\restriction S}(\chi^{-1}(p))$. Since $S \setminus A$ is a 1-trap and $\mathrm{val}^{\mathcal{G}\restriction S}(q) \geq x$ for all $q \in S$, we must also have $\mathrm{val}^{\mathcal{G}\restriction (S\setminus A)}(q) \geq x$ for all $q \in S \setminus A$. Hence, by the induction hypothesis, $\mathrm{Check}(S \setminus A)$ succeeds. Therefore, in order to succeed, $\mathrm{Check}(S)$ only needs to guess a suitable memoryless strategy σ_{M}.

2. The minimal priority p is odd. Let $A \coloneqq \mathrm{Attr}_2^{\mathcal{G}\restriction S}(\chi^{-1}(p))$. We claim that $\mathrm{Check}(S)$ succeeds if it guesses $T \coloneqq \{q \in S \setminus A \mid \mathrm{val}^{\mathcal{G}\restriction (S\setminus A)}(q) \geq x\}$. By Lemma 49, the set T is nonempty. Note that T is a 2-trap and that $\mathrm{val}^{\mathcal{G}\restriction T}(q) \geq x$ for all $q \in T$. Hence, by the induction hypothesis, $\mathrm{Check}(T)$ succeeds. It remains to be shown that $\mathrm{Check}(S\setminus \mathrm{Attr}_1^{\mathcal{G}\restriction S}(T))$ succeeds as well. Note that $S \setminus \mathrm{Attr}_1^{\mathcal{G}\restriction S}(T)$ is a 1-trap, which together with $\mathrm{val}^{\mathcal{G}\restriction S}(q) \geq x$ for all $q \in S$ implies that $\mathrm{val}^{\mathcal{G}\restriction (S\setminus \mathrm{Attr}_1^{\mathcal{G}\restriction S}(T))}(q) \geq x$ for all $q \in S \setminus \mathrm{Attr}_1^{\mathcal{G}\restriction S}(T)$. Hence, the induction hypothesis yields that $\mathrm{Check}(S \setminus \mathrm{Attr}_1^{\mathcal{G}\restriction S}(T))$ succeeds. □

5.1.4 A Deterministic Algorithm

In this section, we present a deterministic algorithm for computing the values of a mean-payoff parity game, which runs faster than all known algorithms for solving these games. Algorithm SolveMPP is based on the classical algorithm for solving parity games, due to Zielonka [82]. The algorithm employs as a subprocedure an algorithm SolveMP for solving mean-payoff games. By [84], such an algorithm can be implemented to run in time $\mathcal{O}(n^3 \cdot m \cdot W)$ for a game with n states and m edges. We denote by $f \sqcup g$ and $f \sqcap g$ the pointwise maximum, respectively minimum, of two (partial) functions $f, g \colon Q \to \mathbb{R} \cup \{\pm\infty\}$ (where $(f \sqcup g)(q) = (f \sqcap g)(q) = f(q)$ if $g(q)$ is undefined).

The algorithm works as follows: If the least priority p in $\mathcal{G}$ is even, the algorithm first identifies the least value of $\mathcal{G}$ by computing the values of the mean-payoff game $(G, 0)$ and (recursively) the values of the game $\mathcal{G} \restriction Q \setminus \mathrm{Attr}_1(\chi^{-1}(p))$, and taking their minimum x. All states from where Player 2 can enforce a visit to a state with value x in one of these two games must have value x in $\mathcal{G}$. In the remaining subarena, the values can be computed by calling SolveMPP recursively. If the least priority is odd, we can similarly compute the greatest value of $\mathcal{G}$ and proceed by recursion.

Algorithm SolveMPP($\mathcal{G}$)

Input : mean-payoff parity game $\mathcal{G} = (G, \chi)$
Output : $\mathrm{val}^{\mathcal{G}}$

if $Q = \emptyset$ **then return** $\emptyset$

$p \coloneqq \min\{\chi(q) \mid q \in Q\}$
if p is even **then**
  $g \coloneqq \mathrm{SolveMP}(G, 0)$
  if $\chi(q) = p$ for all $q \in Q$ **then return** g

  $T \coloneqq Q \setminus \mathrm{Attr}_1^{\mathcal{G}}(\chi^{-1}(p))$
  $f \coloneqq \mathrm{SolveMPP}(\mathcal{G} \restriction T)$
  $x \coloneqq \min(f(T) \cup g(Q))$
  $A \coloneqq \mathrm{Attr}_2^{\mathcal{G}}(f^{-1}(x) \cup g^{-1}(x))$
  return $(Q \to \mathbb{R} \cup \{-\infty\} \colon q \mapsto x) \sqcup \mathrm{SolveMPP}(\mathcal{G} \restriction Q \setminus A)$
else
  $T \coloneqq Q \setminus \mathrm{Attr}_2^{\mathcal{G}}(\chi^{-1}(p))$
  if $T = \emptyset$ **then return** $(Q \to \mathbb{R} \cup \{-\infty\} \colon q \mapsto -\infty)$

  $f \coloneqq \mathrm{SolveMPP}(\mathcal{G} \restriction T)$
  $x \coloneqq \max f(T)$
  $A \coloneqq \mathrm{Attr}_1^{\mathcal{G}}(f^{-1}(x))$
  return $(Q \to \mathbb{R} \cup \{-\infty\} \colon q \mapsto x) \sqcap \mathrm{SolveMPP}(\mathcal{G} \restriction Q \setminus A)$
end

Algorithm 3: A deterministic algorithm for computing the values of a mean-payoff parity game.

Theorem 53 *The values of a mean-payoff parity game with d priorities can be computed in time $\mathcal{O}(|Q|^{d+2} \cdot |E| \cdot W)$.*

Proof We claim that SolveMPP computes, given a mean-payoff parity game $\mathcal{G}$, the function $\mathrm{val}^{\mathcal{G}}$ in the given time bound. Denote by $T(n, m, d)$ the worst-case running time of the algorithm on a game with n states, m edges and d priorities. Note that, if $\mathcal{G}$ has only one priority, then there are no recursive calls to SolveMPP. Since attractors can be computed in time $\mathcal{O}(n + m)$ and the running time of SolveMP is $\mathcal{O}(n^3 \cdot m \cdot W)$, there exists a constant c such that the numbers $T(n, m, d)$ satisfy the following recurrence:

$$\begin{aligned}
&T(1, m, d) \leq c,\\
&T(n, m, 1) \leq c \cdot n^3 \cdot m \cdot W,\\
&T(n, m, d) \leq T(n-1, m, d-1) + T(n-1, m, d) + c \cdot n^3 \cdot m \cdot W\,.
\end{aligned}$$

We claim that $T(n, m, d) \leq c \cdot (n+1)^{d+2} \cdot m \cdot W \in \mathcal{O}(n^{d+2} \cdot m \cdot W)$. The claim is clearly true if $n = 1$. Hence, assume that $n \geq 2$ and that the claim is true for all lower values of n. If $d = 1$, the claim follows from the second inequality. Otherwise,

$$\begin{aligned}
T(n, m, d) &\leq T(n-1, m, d-1) + T(n-1, m, d) + c \cdot n^3 \cdot m \cdot W\\
&\leq c \cdot n^{d+1} \cdot m \cdot W + c \cdot n^{d+2} \cdot m \cdot W + c \cdot n^3 \cdot m \cdot W\\
&\leq c \cdot (n^{d+1} + n \cdot n^{d+1} + n^{d+1}) \cdot m \cdot W\\
&\leq c \cdot ((n+1)^{d+1} + n \cdot (n+1)^{d+1}) \cdot m \cdot W\\
&= c \cdot (n+1)^{d+2} \cdot m \cdot W
\end{aligned}$$

It remains to be proved that the algorithm is correct, i.e., that SolveMPP($\mathcal{G}$) returns $\mathrm{val}^{\mathcal{G}}$. We prove the claim by induction over the number of states. If there are no states, the claim is trivial. Hence, assume that $Q \neq \emptyset$ and that the claim is true for all games with less than $|Q|$ states. Let $p \coloneqq \min\{\chi(q) \mid q \in Q\}$. We only consider the case that p is even. If p is odd, the proof is similar, but relies on Lemma 49 instead of Lemma 48.

Let T, f, g, x and A be defined as in the corresponding case of the algorithm, and let $f^* = \text{SolveMPP}(\mathcal{G})$. If $\chi(Q) = \{p\}$, then $f^* = g = \mathrm{val}^{(G,0)} = \mathrm{val}^{\mathcal{G}}$, and the claim is fulfilled. Otherwise, by the definition of x and applying the induction hypothesis to the game $\mathcal{G} \restriction T$, we have $\mathrm{val}^{(G,0)}(q) \geq x$ for all $q \in Q$ and $\mathrm{val}^{\mathcal{G}\restriction T}(q) = f(q) \geq x$ for all $q \in T$. Hence, Lemma 48 yields that $\mathrm{val}^{\mathcal{G}}(q) \geq x$ for all $q \in Q$. On the other hand, from

any state $q \in A$ Player 2 can play an attractor strategy to $f^{-1}(x) \cup g^{-1}(x)$, followed by an optimal strategy in the game $\mathcal{G} \restriction T$, respectively in the mean-payoff game $(G, 0)$, which ensures that Player 1's payoff does not exceed x. Hence, $\mathrm{val}^{\mathcal{G}}(q) = x = f^*(q)$ for all $q \in A$.

Now, let $q \in Q \setminus A$. We already know that $\mathrm{val}^{\mathcal{G}}(q) \geq x$. Moreover, since $Q \setminus A$ is a 2-trap and applying the induction hypothesis to the game $\mathcal{G} \restriction Q \setminus A$, we have $\mathrm{val}^{\mathcal{G}}(q) \geq \mathrm{val}^{\mathcal{G} \restriction Q \setminus A}(q) = \mathrm{SolveMPP}(\mathcal{G} \restriction Q \setminus A)(q)$. Hence, $\mathrm{val}^{\mathcal{G}}(q) \geq f^*(q)$. To see that $\mathrm{val}^{\mathcal{G}}(q) \leq f^*(q)$, consider the strategy τ of Player 2 that mimics an optimal strategy in $\mathcal{G} \restriction Q \setminus A$ as long as the play stays in $Q \setminus A$ and switches to an optimal strategy in $\mathcal{G}$ as soon as the play reaches A. We have $\mathrm{val}^{\mathcal{G}}(\tau, q) \leq \max\{\mathrm{val}^{\mathcal{G} \restriction Q \setminus A}(q), x\} = f^*(q)$. □

Algorithm SolveMPP is faster and conceptually simpler than the original algorithm proposed for solving mean-payoff parity games [18]. Compared to the recent algorithm proposed by Chatterjee and Doyen [16], which uses a reduction to energy parity games and runs in time $\mathcal{O}(|Q|^{d+4} \cdot |E| \cdot d \cdot W)$, our algorithm has three main advantages: 1. it is faster; 2. it operates directly on mean-payoff parity games, and 3. it is more flexible since it computes the values exactly instead of just comparing them to an integer threshold.

5.2 Mean-penalty Parity Games

In this section, we define multi-strategies and *mean-penalty parity games*. We reduce these games to mean-payoff parity games, show that their value problem is in NP ∩ coNP, and propose a deterministic algorithm for computing the values, which runs in pseudo-polynomial time if the number of priorities is bounded.

5.2.1 Definitions

Syntactically, a *mean-penalty parity game* is a mean-payoff parity game with non-negative weights, i.e., a tuple $\mathcal{G} = (G, \chi)$, where $G = (Q_1, Q_2, E, \mathrm{weight})$ is a weighted game graph with $\mathrm{weight}\colon E \to \mathbb{R}^{\geq 0}$ (or $\mathrm{weight}\colon E \to \mathbb{N}$ for algorithmic purposes), and $\chi\colon Q \to \mathbb{N}$ is a priority function assigning a priority to every state. As for mean-payoff parity games, a play ρ is parity-winning if the minimal priority occurring infinitely often ($\min\{\chi(q) \mid q \in \mathrm{Inf}(\rho)\}$) is even.

Since we are interested in controller synthesis, we define multi-strategies only for Player 1 (who represents the controller). Formally, a *multi-strategy* (for Player 1) in $\mathcal{G}$ is a function $\sigma\colon Q^*Q_1 \to 2^Q \setminus \{\emptyset\}$ such that $\sigma(\gamma q) \subseteq qE$

for all $\gamma \in Q^*$ and $q \in Q_1$. A play ρ of $\mathcal{G}$ is *consistent* with a multi-strategy σ if $\rho(k+1) \in \sigma(\rho[0,k])$ for all $k \in \mathbb{N}$ with $\rho(k) \in Q_1$, and we denote by $\mathrm{Out}^{\mathcal{G}}(\sigma, q_0)$ the set of all plays ρ of $\mathcal{G}$ that are consistent with σ and start in $\rho(0) = q_0$.

Note that, unlike for deterministic strategies, there is, in general, no unique play consistent with a multi-strategy σ for Player 1 and a (deterministic) strategy τ for Player 2 from a given initial state. Additionally, note that every deterministic strategy can be viewed as a multi-strategy.

Let $\mathcal{G}$ be a mean-penalty parity game, and let σ be a multi-strategy. We inductively define $\mathrm{penalty}^{\mathcal{G}}_{\sigma}(\gamma)$ (the *total penalty* of γ w.r.t. σ) for all $\gamma \in Q^*$ by setting $\mathrm{penalty}^{\mathcal{G}}_{\sigma}(\varepsilon) = 0$ as well as $\mathrm{penalty}^{\mathcal{G}}_{\sigma}(\gamma q) = \mathrm{penalty}^{\mathcal{G}}_{\sigma}(\gamma)$ if $q \in Q_2$ and

$$\mathrm{penalty}^{\mathcal{G}}_{\sigma}(\gamma q) = \mathrm{penalty}^{\mathcal{G}}_{\sigma}(\gamma) + \sum_{q' \in qE \setminus \sigma(\gamma q)} \mathrm{weight}(q, q')$$

if $q \in Q_1$. Hence, $\mathrm{penalty}^{\mathcal{G}}_{\sigma}(\gamma)$ is the total weight of transitions blocked by σ along γ. The *mean penalty* of an infinite play ρ is then defined as the average penalty that is incurred along this play in the limit, i.e.,

$$\mathrm{penalty}^{\mathcal{G}}_{\sigma}(\rho) = \begin{cases} \limsup_{n \to \infty} \frac{1}{n} \mathrm{penalty}^{\mathcal{G}}_{\sigma}(\rho[0,n)) & \text{if } \rho \text{ is parity-winning,} \\ \infty & \text{otherwise.} \end{cases}$$

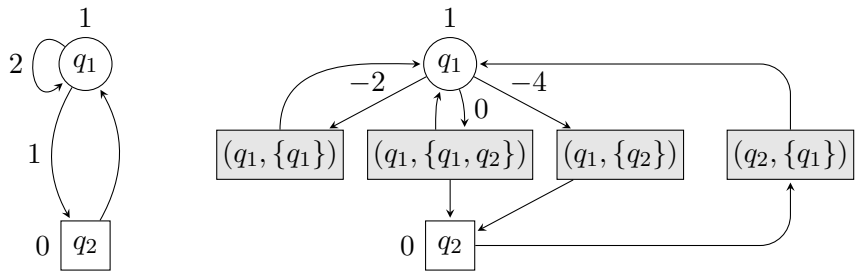
The mean penalty of a multi-strategy σ from a given initial state q_0 is defined as the supremum over the mean penalties of all plays that are consistent with σ, i.e.,

$$\mathrm{penalty}^{\mathcal{G}}(\sigma, q_0) = \sup\{\mathrm{penalty}^{\mathcal{G}}_{\sigma}(\rho) \mid \rho \in \mathrm{Out}^{\mathcal{G}}(\sigma, q_0)\}.$$

The *value* of a state q_0 in a mean-penalty parity game $\mathcal{G}$ is the least mean penalty that a multi-strategy of Player 1 can achieve, i.e., $\mathrm{val}^{\mathcal{G}}(q_0) = \inf_{\sigma} \mathrm{penalty}^{\mathcal{G}}(\sigma, q_0)$, where σ ranges over all multi-strategies of Player 1. A multi-strategy σ is called *optimal* if $\mathrm{penalty}^{\mathcal{G}}(\sigma, q_0) = \mathrm{val}^{\mathcal{G}}(q_0)$ for all $q_0 \in Q$.

Finally, the *value problem for mean-penalty parity games* is the following decision problem: Given a mean-penalty parity game $\mathcal{G} = (G, \chi)$, an initial state $q_0 \in Q$, and a number $x \in \mathbb{Q}$, decide whether $\mathrm{val}^{\mathcal{G}}(q_0) \leq x$.

Example *Figure 5.2a represents a mean-penalty parity game. Note that weights of transitions out of Player 2 states are not indicated as they are irrelevant for the mean penalty. In this game, Player 1 (controlling circle*



(a) A mean-penalty parity game.

(b) The corresponding mean-payoff parity game.

Figure 5.2: A reduction.

states) has to regularly block *the self-loop if she wants to enforce infinitely many visits to the state with priority* 0*. This comes with a penalty of* 2*. However, the multi-strategy in which she blocks no transition can be played safely for an arbitrary number of times. Hence Player 1 can win with mean-penalty* 0 *(but infinite memory) by blocking the self-loop once every* k *moves, where* k *grows with the number of visits to* q_2*.*

5.2.2 Strategy Complexity

In order to solve mean-penalty games, we reduce them to mean-payoff parity games. We construct from a given mean-penalty parity game $\mathcal{G}$ an exponential-size mean-payoff parity game $\mathcal{G}'$, similar to [5] but with an added priority function. Formally, for a mean-penalty parity game $\mathcal{G} = (G, \chi)$ with weighted game graph $G = (Q_1, Q_2, E, \text{weight})$, the weighted game graph $G' = (Q_1', Q_2', E', \text{weight}')$ of the corresponding mean-payoff parity game $\mathcal{G}'$ is defined as follows:

- $Q_1' = Q_1$ and $Q_2' = Q_2 \cup \bar{Q}$, where $\bar{Q} \coloneqq \{(q, F) \mid q \in Q,\ \emptyset \neq F \subseteq qE\}$;
- E' is the (disjoint) union of three kinds of transitions:
  (1) transitions of the form $(q, (q, F))$ for each $q \in Q_1$ and $\emptyset \neq F \subseteq qE$,
  (2) transitions of the form $(q, (q, \{q'\}))$ for each $q \in Q_2$ and $q' \in qE$,
  (3) transitions of the form $((q, F), q')$ for each $q' \in F$;
- the weight function weight' assigns 0 to transitions of type (2) and (3), but $\text{weight}'(q, (q, F)) = -2 \sum_{q' \in qE \setminus F} \text{weight}(q, q')$ to transitions of type (1).

Finally, the priority function χ' of $\mathcal{G}'$ coincides with χ on Q and assigns priority $M \coloneqq \max\{\chi(q) \mid q \in Q\}$ to all states in $\bar{Q}$.

Example *Figure 5.2b depicts the mean-payoff parity game obtained from the mean-penalty parity game from Example 11, depicted in Figure 5.2a.*

The correspondence between $\mathcal{G}$ and $\mathcal{G}'$ is expressed in the following lemma.

Lemma 54 *Let $\mathcal{G}$ be a mean-penalty parity game, $\mathcal{G}'$ the corresponding mean-payoff parity game, and $q_0 \in Q$.*

1. *For every multi-strategy σ in $\mathcal{G}$ there exists a strategy σ' for Player 1 in $\mathcal{G}'$ such that* $\mathrm{val}(\sigma', q_0) \geq -\mathrm{penalty}(\sigma, q_0)$.
2. *For every strategy σ' for Player 1 in $\mathcal{G}'$ there exists a multi-strategy σ in $\mathcal{G}$ such that* $\mathrm{penalty}(\sigma, q_0) \leq -\mathrm{val}(\sigma', q_0)$.
3. $\mathrm{val}^{\mathcal{G}'}(q_0) = -\mathrm{val}^{\mathcal{G}}(q_0)$.

Proof Clearly, 3. is implied by 1. and 2., and we only need to prove the first two statements. To prove 1., let σ be a multi-strategy in $\mathcal{G}$. For a play prefix $\gamma = q_0(q_0, F_0) \cdots q_n(q_n, F_n)$ in $\mathcal{G}'$, let $\tilde{\gamma} \coloneqq q_0 \cdots q_n$ be the corresponding play prefix in $\mathcal{G}$. We set $\sigma'(\gamma q) = (q, F)$ if $q \in Q_1$ and $\sigma(\tilde{\gamma} q) = F$. Clearly, for each $\rho' \in \mathrm{Out}(\sigma', q_0)$ there exists a play $\rho \in \mathrm{Out}(\sigma, q_0)$ with $-\mathrm{penalty}_\sigma(\rho) = \mathrm{payoff}(\rho')$ (namely $\rho(i) = \rho'(2i)$ for all $i \in \mathbb{N}$). Hence,

$$\begin{aligned}
\mathrm{val}^{\mathcal{G}'}(\sigma', q_0) &= \inf\{\mathrm{payoff}(\rho') \mid \rho' \in \mathrm{Out}(\sigma', q_0)\} \\
&\geq \inf\{-\mathrm{penalty}_\sigma(\rho) \mid \rho \in \mathrm{Out}(\sigma, q_0)\} \\
&= -\sup\{\mathrm{penalty}_\sigma(\rho) \mid \rho \in \mathrm{Out}(\sigma, q_0)\} \\
&= -\mathrm{penalty}(\sigma, q_0)\,.
\end{aligned}$$

To prove 2., let σ' be a strategy for Player 1 in $\mathcal{G}'$. For a play prefix $\gamma = q_0 \cdots q_n$ in $\mathcal{G}$, we inductively define the corresponding play prefix $\tilde{\gamma}$ in $\mathcal{G}'$ by setting $\tilde{q} = q$ and $\tilde{\gamma q} = \tilde{\gamma} \cdot \sigma'(\tilde{\gamma}) \cdot q$. We set $\sigma(\gamma) = F$ if $\sigma'(\tilde{\gamma}) = (q, F)$. For each $\rho \in \mathrm{Out}(\sigma, q_0)$ there exists a play $\rho' \in \mathrm{Out}(\sigma', q_0)$ with $\mathrm{penalty}_\sigma(\rho) = -\mathrm{payoff}(\rho')$, namely the play ρ' defined by $\rho'(2i) = \rho(i)$ and

$$\rho'(2i+1) = \begin{cases} (\rho(i), \sigma(\rho[0,i])) & \text{if } \rho(i) \in Q_1, \\ (\rho(i), \{\rho(i+1)\}) & \text{if } \rho(i) \in Q_2, \end{cases}$$

for all $i \in \mathbb{N}$. Hence,

$$\begin{aligned}\text{penalty}(\sigma, q_0) &= \sup\{\text{penalty}_\sigma(\rho) \mid \rho \in \text{Out}(\sigma, q_0)\} \\ &\leq \sup\{-\text{payoff}(\rho') \mid \rho' \in \text{Out}(\sigma', q_0)\} \\ &= -\inf\{\text{payoff}(\rho') \mid \rho' \in \text{Out}(\sigma', q_0)\} \\ &= -\text{val}^{\mathcal{G}'}(\sigma', q_0)\,. \end{aligned}$$

□

It follows from Theorem 45 and Lemma 54 that every mean-penalty parity game admits an optimal multi-strategy.

Corollary 55 *In every mean-penalty parity game, Player 1 has an optimal multi-strategy.*

We now show that Player 2 has a memoryless optimal strategy of a special kind in the mean-payoff parity game derived from a mean-penalty parity game. This puts the value problem for mean-penalty parity games into coNP, and is also a crucial point in the proof of Lemma 57 below.

Lemma 56 *Let $\mathcal{G}$ be a mean-penalty parity game and $\mathcal{G}'$ the corresponding mean-payoff parity game. Then in $\mathcal{G}'$ there is a memoryless optimal strategy τ' for Player 2 such that for every $q \in Q$ there exists a total order $\leq_q$ on the set qE with $\tau'((q,F)) = \min_{\leq_q} F$ for every state $(q,F) \in \bar{Q}$.*

Proof Let τ be a memoryless optimal strategy for Player 2 in $\mathcal{G}'$. For a state q, we consider the set qE and order it in the following way. We inductively define $F_1 = qE$, $q_i = \tau((q,F_i))$ and $F_{i+1} = F_i \setminus \{q_i\}$ for every $1 \leq i \leq |qE|$. Note that $\{q_1, \ldots, q_{|qE|}\} = qE$. We set $q_1 \leq_q q_2 \leq_q \cdots \leq_q q_{|qE|}$ and define a new memoryless strategy τ' for Player 2 in $\mathcal{G}'$ by $\tau'((q,F)) = \min_{\leq_q} F$ for $(q,F) \in \bar{Q}$ and $\tau'(q) = \tau(q)$ for all $q \in Q_2$.

To prove the lemma, we have to show that τ' is at least as good as τ and thus optimal.

Let $q_0 \in Q$ and $\rho' \in \text{Out}^{\mathcal{G}'}(\tau', q_0)$. We construct a play $\rho \in \text{Out}^{\mathcal{G}'}(\tau, q_0)$ with $\text{payoff}^{\mathcal{G}'}(\rho) \geq \text{payoff}^{\mathcal{G}'}(\rho')$ in the following way. For every position i with $\rho'(i) = (q, F')$, let $F = \{q' \in qE \mid \tau'((q,F')) \leq_q q'\}$ (then $\tau((q,F)) = \tau'((q,F'))$ by the definition of τ') and set $\rho(i) = (q,F)$. For every other position i, let $\rho(i) = \rho'(i)$. Note that $\rho \in \text{Out}(\tau, q_0)$ and $\min \chi(\text{Inf}(\rho)) = \min \chi(\text{Inf}(\rho'))$. Moreover, we have $F' \subseteq F$ and therefore $\text{weight}'(q,(q,F')) \leq \text{weight}'(q,(q,F))$ whenever $\rho'(i) = (q,F')$ and $\rho(i) = (q,F)$ (because weights in $\mathcal{G}$ are nonnegative). Hence, $\text{payoff}(\rho) \geq \text{payoff}(\rho')$.

Since ρ' was chosen arbitrarily, it follows that

$$\begin{aligned}\mathrm{val}(\tau, q_0) &= \sup\{\mathrm{payoff}(\rho) \mid \rho \in \mathrm{Out}(\tau, q_0)\} \\ &\geq \sup\{\mathrm{payoff}(\rho') \mid \rho' \in \mathrm{Out}(\tau', q_0)\} \\ &= \mathrm{val}(\tau', q_0)\,.\end{aligned}$$

Hence, τ' is optimal. □

5.2.3 Computational Complexity

In order to put the value problem for mean-penalty parity games into NP ∩ coNP, we propose a more sophisticated reduction from mean-penalty parity games to mean-payoff parity games, which results in a polynomial-size mean-payoff parity game. Intuitively, in a state $q \in Q_1$ we ask Player 1 *consecutively* for each outgoing transition whether he wants to block that transition. If he allows a transition, then Player 2 has to decide whether she wishes to explore this transition. Finally, after all transitions have been processed in this way, the play proceeds along the *last* transition that Player 2 has desired to explore.

The construction uses two counters, i and m, where i represents the edge-number that is currently dealt with, and m represents the last edge that Player 2 wants to explore.

Formally, let us fix a mean-penalty parity game $\mathcal{G} = (G, \chi)$ with weighted game graph $G = (Q_1, Q_2, E, \mathrm{weight})$, and denote by $k \coloneqq \max\{|qE| \mid q \in Q\}$ the maximal out-degree of a state. Then the polynomial-size mean-payoff parity game $\mathcal{G}''$ has vertices of the form q and (q, a, i, m), where $q \in Q$, $a \in \{\mathrm{choose}, \mathrm{allow}, \mathrm{block}\}$, $i \in \{1, \ldots, k+1\}$ and $m \in \{0, \ldots, k\}$; vertices of the form q and $(q, \mathrm{choose}, i, m)$ belong to Player 1, while vertices of the form $(q, \mathrm{allow}, i, m)$ or $(q, \mathrm{block}, i, m)$ belong to Player 2. To describe the transition structure of $\mathcal{G}$, let $q \in Q$ and assume that $qE = \{q_1, \ldots, q_k\}$ (a state may occur more than once in this list). Then the following transitions originate in a state of the form q or (q, a, i, m):

1. a transition from q to $(q, \mathrm{choose}, 1, 0)$ with weight 0,
2. for all $1 \leq i \leq k$ and $0 \leq m \leq k$ a transition from $(q, \mathrm{choose}, i, m)$ to $(q, \mathrm{allow}, i, m)$ with weight 0,
3. if $q \in Q_1$ then for all $1 \leq i \leq k$ and $0 \leq m \leq k$ a transition from $(q, \mathrm{choose}, i, m)$ to $(q, \mathrm{block}, i, m)$ with weight 0, *except* if $i = k$ and $m = 0$;

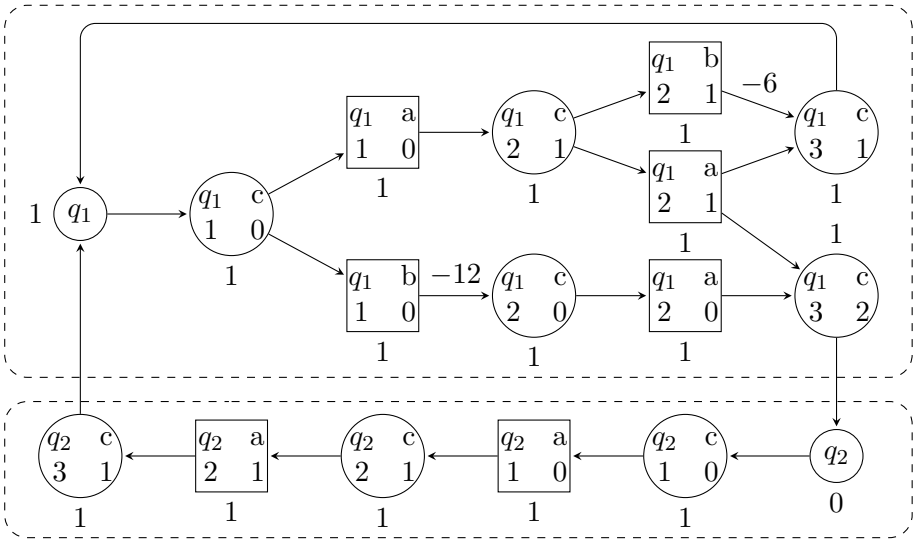


Figure 5.3: The game $\mathcal{G}''$ associated with the game $\mathcal{G}$ of Figure 5.2a.

4. for all $0 \leq m \leq k$ a transition from $(q, \text{choose}, k+1, m)$ to q_m with weight 0 (where q_0 can be chosen arbitrarily),

5. for all $1 \leq i \leq k$ and $0 \leq m \leq k$ a transition from (q, allow, i, m) to $(q, \text{choose}, i+1, i)$ with weight 0,

6. for all $1 \leq i \leq k$ and $1 \leq m \leq k$ a transition from (q, allow, i, m) to $(q, \text{choose}, i+1, m)$ with weight 0,

7. for all $1 \leq i \leq k$ and $0 \leq m \leq k$ a transition from (q, block, i, m) to $(q, \text{choose}, i+1, m)$ with weight $-2(k+1) \cdot \text{weight}(q, q_i)$.

Finally, the priority of a state $q \in Q$ equals the priority of the same state in $\mathcal{G}$, whereas all states of the form (q, a, i, m) have priority $M = \max\{\chi(q) \mid q \in Q\}$.

Example *For the game of Figure 5.2a, this transformation would yield the game depicted in Figure 5.3. In this picture,* a, b *and* c *stand for* allow, block *and* choose*, respectively; zero weights are omitted.*

It is easy to see that the game $\mathcal{G}''$ has polynomial size and can, in fact, be constructed in polynomial time from the given mean-penalty parity game $\mathcal{G}$. The following lemma relates the game $\mathcal{G}''$ to the mean-payoff parity game $\mathcal{G}'$ of exponential size constructed in Section 5.2.2 and to the original game $\mathcal{G}$.

Lemma 57 *Let $\mathcal{G}$ be a mean-penalty parity game, $\mathcal{G}'$ the corresponding mean-payoff parity game of exponential size, $\mathcal{G}''$ the corresponding mean-payoff parity game of polynomial size, and $q_0 \in Q$.*

1. *For every multi strategy σ in $\mathcal{G}$ there exists a strategy σ' for Player 1 in $\mathcal{G}''$ such that $\mathrm{val}(\sigma', q_0) \geq -\mathrm{penalty}(\sigma, q_0)$.*
2. *For every strategy τ for Player 2 in $\mathcal{G}'$ there exists a strategy τ' for Player 2 in $\mathcal{G}''$ such that $\mathrm{val}(\tau', q_0) \leq \mathrm{val}(\tau, q_0)$.*
3. $\mathrm{val}^{\mathcal{G}''}(q_0) = -\mathrm{val}^{\mathcal{G}}(q_0)$.

Proof To prove 1., let σ be a multi-strategy in $\mathcal{G}$. For any play prefix γ in $\mathcal{G}''$, let $\tilde{\gamma}$ be the projection to states in $\mathcal{G}$ (i.e., all states of the form (q, a, i, m) are omitted). Assuming that $q_1, \ldots, q_k$ is the enumeration of qE used in the definition of $\mathcal{G}''$, we set $\sigma'(\gamma \cdot (q, \mathrm{choose}, i, m)) = (q, \mathrm{allow}, i, m)$ if (and only if) either $q \in Q_1$ and $q_i \in \sigma(\tilde{\gamma})$ or $q \in Q_2$. It is easy to see that for each $\rho' \in \mathrm{Out}(\sigma', q_0)$ there exists a play $\rho \in \mathrm{Out}(\sigma, q_0)$ with $-\mathrm{penalty}_\sigma(\rho) = \mathrm{payoff}(\rho')$. Hence,

$$\begin{aligned}
\mathrm{val}(\sigma', q_0) &= \inf\{\mathrm{payoff}(\rho') \mid \rho' \in \mathrm{Out}(\sigma', q_0)\} \\
&\geq \inf\{-\mathrm{penalty}_\sigma(\rho) \mid \rho \in \mathrm{Out}(\sigma, q_0)\} \\
&= -\sup\{\mathrm{penalty}_\sigma(\rho) \mid \rho \in \mathrm{Out}(\sigma, q_0)\} \\
&= -\mathrm{penalty}(\sigma, q_0)\,.
\end{aligned}$$

To prove 2., let τ be a strategy for Player 2 in $\mathcal{G}'$. By Lemma 56, there exists a memoryless strategy τ^* for Player 2 in $\mathcal{G}'$ such that $\mathrm{val}(\tau^*, q_0) \leq \mathrm{val}(\tau, q_0)$ and for all $q \in Q$ there exists a total order $\leq_q$ on qE with $\tau^*((q, F)) = \min_{\leq_q} F$ for all $(q, F) \in \bar{Q}$. We define a memoryless strategy τ' for Player 2 in $\mathcal{G}''$ as follows: Assume that $q_1, \ldots, q_k$ is the enumeration of qE used in the definition of $\mathcal{G}''$. Then we set $\tau'((q, \mathrm{allow}, i, m)) = (q, \mathrm{choose}, i+1, i)$ if (and only if) one of the following three conditions is fulfilled: 1. $m = 0$, or 2. $q \in Q_1$ and $q_i \leq_q q_m$, or 3. $q \in Q_2$ and $\tau^*(q) = (q, \{q_i\})$. Now it is easy to see that for each $\rho' \in \mathrm{Out}(\tau', q_0)$ there exists a play $\rho \in \mathrm{Out}(\tau^*, q_0)$ with $\mathrm{payoff}(\rho) = \mathrm{payoff}(\rho')$. Hence,

$$\begin{aligned}
\mathrm{val}(\tau', q_0) &= \sup\{\mathrm{payoff}(\rho') \mid \rho' \in \mathrm{Out}(\tau', q_0)\} \\
&\leq \sup\{\mathrm{payoff}(\rho) \mid \rho \in \mathrm{Out}(\tau^*, q_0)\} \\
&= \mathrm{val}(\tau^*, q_0) \\
&\leq \mathrm{val}(\tau, q_0)\,.
\end{aligned}$$

Finally, we prove 3. It follows from 1. that $\mathrm{val}^{\mathcal{G}''}(q_0) \geq -\mathrm{val}^{\mathcal{G}}(q_0)$, and it follows from 2. that $\mathrm{val}^{\mathcal{G}''}(q_0) \leq \mathrm{val}^{\mathcal{G}'}(q_0)$. But $\mathrm{val}^{\mathcal{G}'}(q_0) = -\mathrm{val}^{\mathcal{G}}(q_0)$ by Lemma 54, and therefore $\mathrm{val}^{\mathcal{G}''}(q_0) = -\mathrm{val}^{\mathcal{G}}(q_0)$. □

Since the mean-payoff game $\mathcal{G}''$ can be computed from $\mathcal{G}$ in polynomial time, we obtain a polynomial-time many-one reduction from the value problem for mean-penalty parity games to the value problem for mean-payoff parity games. By Corollary 47 and Theorem 50, the latter problem belongs to NP ∩ coNP.

Theorem 58 *The value problem for mean-penalty parity games belongs to NP ∩ coNP.*

5.2.4 A Deterministic Algorithm

Naturally, we can use the polynomial translation from mean-penalty parity games to mean-payoff parity games to solve mean-penalty parity games deterministically. Note that the mean-payoff parity game $\mathcal{G}''$ derived from a mean-penalty parity game has $\mathcal{O}(|Q| \cdot k^2)$ states and $\mathcal{O}(|Q| \cdot k^2)$ edges, where k is the maximum out-degree of a state in $\mathcal{G}$; the number of priorities remains constant. Moreover, if weights are given in integers and W is the highest absolute weight in $\mathcal{G}$, then the highest absolute weight in $\mathcal{G}''$ is $\mathcal{O}(k \cdot W)$. Using Theorem 53, we thus obtain a deterministic algorithm for solving mean-penalty parity games that runs in time $\mathcal{O}(|Q|^{d+3} \cdot k^{2d+7} \cdot W)$. If k is a constant, the running time is $\mathcal{O}(|Q|^{d+3} \cdot W)$, which is acceptable. In the general case however, the best upper bound on k is the number of states, and we get an algorithm that runs in time $\mathcal{O}(|Q|^{3d+10} \cdot W)$. Even if the numbers of priorities is small, this running time would not be acceptable in practical applications.

The goal of this section is to show that we can do better; namely we will give an algorithm that runs in time $\mathcal{O}(|Q|^{d+3} \cdot |E| \cdot W)$, independently of the maximum out-degree. The idea is as follows: we use Algorithm SolveMPP on the mean-payoff parity game $\mathcal{G}'$ of exponential size, but we show that we can run it *on* $\mathcal{G}$, i.e., by handling the extra states of $\mathcal{G}'$ symbolically during the computation. As a first step, we adapt the pseudo-polynomial algorithm by Zwick and Paterson [84] to compute the values of a mean-penalty parity game with a trivial parity objective.

Lemma 59 *The values of a mean-penalty parity game with priority function $\chi \equiv 0$ can be computed in time $\mathcal{O}(|Q|^4 \cdot |E| \cdot W)$.*

Proof Let $\mathcal{G} = (G, \chi)$ with $G = (Q_1, Q_2, E, \text{weight})$, and $\mathcal{G}' = (G', \chi')$ with $G' = (Q_1', Q_2', E', \text{weight}')$. For a state $q \in Q'$, we let $v_0(q) = 0$, and for $k > 0$, we define

$$v_k(q) = \begin{cases} \max\limits_{q' \in qE'} \text{weight}'(q, q') + v_{k-1}(q') & \text{if } q \in Q_1', \\ \min\limits_{q' \in qE'} \text{weight}'(q, q') + v_{k-1}(q') & \text{if } q \in Q_2'. \end{cases}$$

If $q \in Q$, then the definition of $\mathcal{G}'$ yields that

$$v_k(q) = \begin{cases} \max\limits_{F \subseteq qE} \text{weight}'(q, (q, F)) + \min\limits_{q' \in F} v_{k-2}(q') & \text{if } q \in Q_1, \\ \min\limits_{q' \in qE} v_{k-2}(q') & \text{if } q \in Q_2, \end{cases}$$

In the first case, a naïve computation would require the examination of an exponential number of transitions. In order to avoid this blow-up, we use the same idea as in the proof of Lemma 56: Let $qE = \{q_1, \ldots, q_r\}$ be sorted in such a way that $i \leq j$ implies $v_{k-2}(q_i) \leq v_{k-2}(q_j)$. Since $\text{weight}'(q, (q, F)) \leq \text{weight}'(q, (q, F'))$ if $F \subseteq F'$, we have

$$v_k(q) = \max_i \text{weight}'(q, (q, \{q_i, \ldots, q_r\})) + v_{k-2}(q_i).$$

Hence the sequence v_{2k} can be computed in time $\mathcal{O}(k \cdot |E|)$ on Q. Now, despite the exponential size of $\mathcal{G}'$, the length of a simple cycle in $\mathcal{G}'$ is at most $2|Q|$. Hence, Theorem 2.2 in [84] becomes

$$2k \cdot \text{val}^{\mathcal{G}'}(q) - 4|Q| \cdot W' \leq v_{2k}(q) \leq 2k \cdot \text{val}^{\mathcal{G}'}(q) + 4|Q| \cdot W'$$

for all $q \in Q$, where W' is the maximal absolute weight in $\mathcal{G}'$. Since $W' \leq |Q| \cdot 2W$, it follows from [84] that $\text{val}^{\mathcal{G}} = -\text{val}^{\mathcal{G}'} \restriction Q$ can be computed in time $\mathcal{O}(|Q|^4 \cdot |E| \cdot W)$. □

Now, given a mean-penalty parity game $\mathcal{G}$ with associated mean-payoff parity game $\mathcal{G}'$ and a set T of states of $\mathcal{G}$, we define

$$\triangle^{\mathcal{G}}(T) = T \cup \{(q, F) \in \bar{Q} \mid F \subseteq T\};$$
$$\blacktriangle^{\mathcal{G}}(T) = T \cup \{(q, F) \in \bar{Q} \mid F \cap T \neq \emptyset\}.$$

We usually omit to mention the superscript $\mathcal{G}$ when it is clear from the context.

Lemma 60 *If S is a subarena of $\mathcal{G}$, then $\triangle(S)$ and $\blacktriangle(S)$ are subarenas of $\mathcal{G}'$.*

Proof Assume that S is a subarena of $\mathcal{G}$, and pick a state q in $\triangle(S)$. If $q \in Q$, then it also belongs to S and, as a state of $\mathcal{G}$, has a successor q' in S. Then $\triangle(S)$ contains $(q, \{q'\})$, which is a successor of q. If q belongs to $\bar{Q}$, then $qE' \subseteq S$ by definition of $\triangle(S)$; hence it has at least one successor in S. A similar argument shows that $\blacktriangle(S)$ is also a subarena of $\mathcal{G}'$. □

Lemma 61 *Let $\mathcal{G}$ be a mean-penalty parity game with associated mean-payoff parity game $\mathcal{G}'$, and let $A, B \subseteq Q$. Then*

$$\triangle(A \cap B) = \triangle(A) \cap \triangle(B), \qquad \triangle(A \cup B) \supseteq \triangle(A) \cup \triangle(B),$$
$$\blacktriangle(A \cup B) = \blacktriangle(A) \cup \blacktriangle(B), \qquad \blacktriangle(A \cap B) \subseteq \blacktriangle(A) \cap \blacktriangle(B),$$
$$\triangle(Q \setminus A) = Q' \setminus \blacktriangle(A), \qquad \blacktriangle(Q \setminus A) = Q' \setminus \triangle(A)\,.$$

Proof Straightforward. □

Lemma 62 *Let $\mathcal{G}$ be a mean-penalty parity game with associated mean-payoff parity game $\mathcal{G}'$, and let $F \subseteq Q$. Then*

$$\triangle(\mathrm{Attr}_1^{\mathcal{G}}(F)) = \mathrm{Attr}_1^{\mathcal{G}'}(F) = \mathrm{Attr}_1^{\mathcal{G}'}(\triangle(F)),$$
$$\blacktriangle(\mathrm{Attr}_2^{\mathcal{G}}(F)) = \mathrm{Attr}_2^{\mathcal{G}'}(F) = \mathrm{Attr}_2^{\mathcal{G}'}(\blacktriangle(F))\,.$$

Proof We only prove the first statement; the second can be proved using similar arguments. Clearly, $\mathrm{Attr}_1^{\mathcal{G}'}(F) = \mathrm{Attr}_1^{\mathcal{G}'}(\triangle(F))$, so we only need to prove that $\triangle(\mathrm{Attr}_1^{\mathcal{G}}(F)) = \mathrm{Attr}_1^{\mathcal{G}'}(F)$. First pick $q \in \triangle(\mathrm{Attr}_1^{\mathcal{G}}(F))$. If $q \in Q$, then the attractor strategy for reaching F can be mimicked in $\mathcal{G}'$, and therefore $q \in \mathrm{Attr}_1^{\mathcal{G}'}(F)$. On the other hand, if $q \in \bar{Q}$, then all successors of q lie in $\mathrm{Attr}_1^{\mathcal{G}}(F)$ and therefore also in $\mathrm{Attr}_1^{\mathcal{G}'}(F)$. Hence, $q \in \mathrm{Attr}_1^{\mathcal{G}'}(F)$. Now pick $q \in \mathrm{Attr}_1^{\mathcal{G}'}(F)$. If $q \in Q$, then the attractor strategy for reaching F yields a multi-strategy σ in $\mathcal{G}$ such that all plays $\rho \in \mathrm{Out}^{\mathcal{G}}(\sigma, q)$ visit F. Hence, $q \in \mathrm{Attr}_1^{\mathcal{G}}(F) \subseteq \triangle(\mathrm{Attr}_1^{\mathcal{G}}(F))$. On the other hand, if $q \in \bar{Q}$, then all successors of q lie in $Q \cap \mathrm{Attr}_1^{\mathcal{G}'}(F)$ (since q is a Player 2 state) and therefore also in $\mathrm{Attr}_1^{\mathcal{G}}(F)$. Hence, $q \in \triangle(\mathrm{Attr}_1^{\mathcal{G}}(F))$. □

Algorithm SymbSolveMPP is our algorithm for computing the values of a mean-penalty parity game. The algorithm employs as a subroutine an algorithm SymbSolveMP for computing the values of a mean-penalty parity with a trivial priority function (see Lemma 59). Since SymbSolveMP can be implemented to run in time $\mathcal{O}(|Q|^4 \cdot |E| \cdot W)$, the running time of the procedure SymbSolveMPP is $\mathcal{O}(|Q|^{d+3} \cdot |E| \cdot W)$. Notably, the algorithm runs in polynomial time if the number of priorities is bounded and we are only interested in the average *number* of edges blocked by a strategy in each step (i.e., if all weights are equal to 1).

Algorithm SymbSolveMPP($\mathcal{G}$)

Input : mean-penalty parity game $\mathcal{G} = (G, \chi)$
Output : $\mathrm{val}^{\mathcal{G}}$

if $Q = \emptyset$ **then return** $\emptyset$

$p \coloneqq \min\{\chi(q) \mid q \in Q\}$
if p is even **then**
  $g \coloneqq \mathrm{SymbSolveMP}(G, 0)$
  if $\chi(q) = p$ for all $q \in Q$ **then return** g

  $T \coloneqq Q \setminus \mathrm{Attr}_1^{\mathcal{G}}(\chi^{-1}(p))$
  $f \coloneqq \mathrm{SymbSolveMPP}(\mathcal{G} \restriction T)$
  $x \coloneqq \max(f(T) \cup g(Q))$
  $A \coloneqq \mathrm{Attr}_2^{\mathcal{G}}(f^{-1}(x) \cup g^{-1}(x))$
  return $(Q \to \mathbb{R} \cup \{\infty\} \colon q \mapsto x) \sqcap \mathrm{SymbSolveMPP}(\mathcal{G} \restriction Q \setminus A)$
else
  $T \coloneqq Q \setminus \mathrm{Attr}_2^{\mathcal{G}}(\chi^{-1}(p))$
  if $T = \emptyset$ **then return** $(Q \to \mathbb{R} \cup \{\infty\} \colon q \mapsto \infty)$

  $f \coloneqq \mathrm{SymbSolveMPP}(\mathcal{G} \restriction T)$
  $x \coloneqq \min f(T)$
  $A \coloneqq \mathrm{Attr}_1^{\mathcal{G}}(f^{-1}(x))$
  return $(Q \to \mathbb{R} \cup \{\infty\} \colon q \mapsto x) \sqcup \mathrm{SymbSolveMPP}(\mathcal{G} \restriction Q \setminus A)$
end

Algorithm 4: A deterministic algorithm for computing the values of a mean-penalty parity game.

Theorem 63 *The values of a mean-penalty parity game with d priorities can be computed in time $\mathcal{O}(|Q|^{d+3} \cdot |E| \cdot W)$.*

Proof From Lemma 59 and with the same runtime analysis as in the proof of Theorem 53, we get that SymbSolveMPP runs in time $\mathcal{O}(|Q|^{d+3} \cdot |E| \cdot W)$. We now prove that the algorithm is correct, by proving that there is a correspondence between the values the algorithm computes on a mean-penalty parity game $\mathcal{G}$ and the values computed by Algorithm SolveMPP on the mean-payoff parity game $\mathcal{G}'$. More precisely, we show that $\text{SolveMPP}(\mathcal{G}') \restriction Q = -\text{SymbSolveMPP}(\mathcal{G})$. The correctness of the algorithm thus follows from Lemma 54, which states that $\text{val}^{\mathcal{G}'} \restriction Q = -\text{val}^{\mathcal{G}}$.

The proof is by induction on the number of states in $\mathcal{G}$. The result holds trivially if $Q = \emptyset$. Otherwise, assume that the result is true for all games with less than $|Q|$ states and let $p = \min\{\chi(q) \mid q \in Q\}$. By construction, p is also the minimal priority in $\mathcal{G}'$. We only consider the case that p is even; the other case is proved using the same arguments.

Write g', T', f', x' and A' for the items computed by SymbSolveMPP on $\mathcal{G}'$, while q, T, f, x and A are the corresponding items computed by SolveMPP on $\mathcal{G}$. Then $g'(q) = -g(q)$ for all $q \in Q$, and $g'((q, F)) = \min_{q' \in F} g'(q')$ for all $(q, F) \in \bar{Q}$ (since such states belongs to Player 2). If $\mathcal{G}$ has only one priority, the result follows. Otherwise, by Lemmas 61 and 62, we have $T' = \blacktriangle(T)$. However, any state $(q, F) \in T'$ that is not a state of the game $(\mathcal{G} \restriction T)'$ has no predecessor in $\mathcal{G}' \restriction T'$: if $q \in T'$ then $q \in T \cap Q_1$ and $qE \setminus T \neq \emptyset$, i.e., $qE \cap \text{Attr}_1(\chi^{-1}(p)) \neq \emptyset$; but then $q \in \text{Attr}_1(\chi^{-1}(p))$ and thus $q \notin T$, a contradiction. It follows that $\text{SolveMPP}(\mathcal{G}' \restriction T') \restriction T = \text{SolveMPP}((\mathcal{G} \restriction T)') \restriction T$.

Now, since T is a strict subset of Q, the induction hypothesis applies, so that $f'(t) = -f(t)$ for all $t \in T$. It follows that $x' = -x$. Let $S \coloneqq Q \setminus A$ and $S' \coloneqq Q' \setminus A'$. By Lemma 62, $A' = \blacktriangle(A)$, and by Lemma 61, $S' = \triangle(S)$. Again, any state $(q, F) \in S'$ that is not a state of the game $(\mathcal{G} \restriction S)'$ has no predecessor in $\mathcal{G}' \restriction S'$. Hence, $\text{SolveMPP}(\mathcal{G}' \restriction S') \restriction S = \text{SolveMPP}((\mathcal{G} \restriction S)') \restriction S$ Applying the induction hypothesis to the game $G \restriction S$, we get that $\text{SolveMPP}((\mathcal{G} \restriction S)') \restriction S = -\text{SymbSolveMPP}(G \restriction S)$, and the result follows for $\mathcal{G}$. □

5.3 Conclusion

In this chapter, we studied mean-payoff parity games and mean-penalty parity games. We specified a reduction from the latter type of game to the former, and by this showed that for both kind of games, the value problem

is in NP $\cap$ coNP. We showed that deterministically solving a mean-payoff parity game can be done in time $\mathcal{O}(n^{d+2} \cdot m \cdot W)$, and that solving a mean-penalty parity game can be accomplished in time $\mathcal{O}(n^{d+3} \cdot m \cdot W)$, where n is the number of states, d is the number of priorities, m is the number of edges and W is the maximum weight of the game graph.

As for both kinds of games, optimal strategies for Player 1 require infinite memory, it is not clear how to represent these strategies and how to synthesize them. A suitable alternative to optimal strategies are *ε-optimal* strategies, which achieve the value of a game by a divergence of at most ε. Finite-memory ε-optimal strategies are guaranteed to exist [4]. So a challenge for future work could be to extend the algorithms of this chapter, such that they compute a finite-memory ε-optimal (multi-) strategy for Player 1.

Chapter 6

Concluding Remarks

The scope of this thesis was to establish a tight connection between the theory of games and formal language theory, particularly with regards to measuring the complexity of strategies.

We started in Chapter 3 by improving a 50 years old algorithm, the Ramsey-based Büchi automata complementation method, both on the practical as well as on the theoretical side. We showed that this oldest of today's existing complementation algorithms can compete with the more modern ones on the practical side by conducting experimental studies on a set of 11,000 sample automata. Furthermore, we introduced a novel complementation construction, based on weak-orders, and we showed that the Ramsey-based approach is tightly connected to the weak-order-based approach, which yields an $2^{\mathcal{O}(n \log n)}$ upper bound on the size of the complement. In Chapter 4 we embedded the concept of games into the domain of formal languages. By doing this, we were able to give a qualitative measure of the complexity of a winning strategy, as well as of the complexity of the corresponding winning condition. In this way, we were able to extend and refine the fundamental Büchi-Landweber Theorem to subclasses of the class of regular languages. We considered language classes inside the dot-depth hierarchy and inside the Straubing-Thérien hierarchy. For solving weak games, winning strategies lie one level above winning condition inside the hierarchy. For strong games on level i, we could show that winning strategies on level $i + 2$ suffice. We were not able to determine whether also level $i + 1$ is enough to express winning strategies. In Chapter 5 we introduced another measure for the complexity of strategies, but this time on graph-games. This measure is of quantitative nature and it determines the *permissiveness* of a given multi-strategy. The permissiveness is measured by assigning to each strategy a penalty, namely the average sum (in the limit) of the weight of edges that are to be disallowed. We reduced the value problem of such a game to the value problem of a corresponding mean-payoff parity game, showing that both problems are in $\mathrm{NP} \cap \mathrm{coNP}$. We

revisited the study of mean-payoff parity games and obtained deterministic algorithms, which run faster than all previously known algorithms.

The challenges for future research are to be found in all three topics of this thesis. For the complementation of Büchi automata a desirable perspective is a unified approach of two or more complementation methods. As already mentioned, the unification of the rank-based and the slice-based approach has already been achieved [28]. The author's personal opinion is that similar unifications are also possible for the other complementation approaches, in particular for the weak-order-based and the rank-based approach. For universality checking and inclusion checking of Büchi automata, one next step in research could be to investigate how the heuristics of the improved Ramsey-based approach – like the idea of merging transition profiles – can be employed to the quite efficient algorithms of [29, 2, 1].

For the Büchi-Landweber Theorem, one problem that remains open is whether strong games on level i can be solved by winning strategies of level $i+1$, with regards to the dot-depth and the Straubing-Thérien hierarchy. Furthermore, we have seen that our results in some cases exploit the preservation of the transition structure when going from games to transducers, in other cases exploit the logical characterizations of the corresponding language classes. It is desirable to possess a uniform treatment, which can cover all of these results. Such a universal framework is yet to be found.

For permissive strategies, a challenge for future work is to modify our algorithms so that they compute not only the values of the game, but also a finite-memory ε-optimal (multi-) strategy, or even an optimal (multi-) strategy for Player 1. A finite representation of such a strategy can be constructed along the lines of our determinacy proof, although this algorithm could have exponential running time even if the number of priorities and the size of the weights are bounded. Apart from that, it would be interesting to transfer the concept of permissive strategies to the setting of timed games. There, the timing constraints of both players are important for winning the game. A possible notion of permissiveness in timed games could make use of how long such a time span may be.

Bibliography

[1] Parosh A. Abdulla, Yu-Fang Chen, Lorenzo Clemente, Lukáš Holík, Chih-Duo Hong, Richard Mayr, and Tomáš Vojnar. "Advanced Ramsey-Based Büchi Automata Inclusion Testing." In: *CONCUR*. Vol. 6901. LNCS. Springer-Verlag, 2011, pp. 187–202. DOI: 10.1007/978-3-642-23217-6_13.

[2] Parosh A. Abdulla, Yu-Fang Chen, Lorenzo Clemente, Lukáš Holík, Chih-Duo Hong, Richard Mayr, and Tomáš Vojnar. "Simulation Subsumption in Ramsey-Based Büchi Automata Universality and Inclusion Testing." In: *CAV*. Vol. 6174. LNCS. Springer-Verlag, 2010, pp. 132–147. DOI: 10.1007/978-3-642-14295-6_14.

[3] Julien Bernet, David Janin, and Igor Walukiewicz. "Permissive strategies: from parity games to safety games." In: *RAIRO – ITA* 36.3 (2002), pp. 261–275. DOI: 10.1051/ita:2002013.

[4] Roderick Bloem, Krishnendu Chatterjee, Thomas A. Henzinger, and Barbara Jobstmann. "Better Quality in Synthesis through Quantitative Objectives." In: *CAV*. Ed. by Ahmed Bouajjani and Oded Maler. Vol. 5643. LNCS. Springer-Verlag, 2009, pp. 140–156. ISBN: 978-3-642-02657-7. DOI: 10.1007/978-3-642-02658-4_14.

[5] Patricia Bouyer, Marie Duflot, Nicolas Markey, and Gabriel Renault. "Measuring Permissivity in Finite Games." In: *CONCUR*. Ed. by Mario Bravetti and Gianluigi Zavattaro. Vol. 5710. LNCS. Springer-Verlag, 2009, pp. 196–210. ISBN: 978-3-642-04080-1. DOI: 10.1007/978-3-642-04081-8_14.

[6] Patricia Bouyer, Uli Fahrenberg, Kim G. Larsen, Nicolas Markey, and Jiří Srba. "Infinite Runs in Weighted Timed Automata with Energy Constraints." In: *FORMATS*. Ed. by Franck Cassez and Claude Jard. Vol. 5215. LNCS. Springer-Verlag, 2008, pp. 33–47. DOI: 10.1007/978-3-540-85778-5_4.

[7] Patricia Bouyer, Nicolas Markey, Jörg Olschewski, and Michael Ummels. *Measuring Permissiveness in Parity Games: Mean-Payoff Parity Games Revisited*. Research Report LSV-11-17. Laboratoire Spécification et Vérification, ENS Cachan, France, June 2011.

[8] Patricia Bouyer, Nicolas Markey, Jörg Olschewski, and Michael Ummels. *Measuring Permissiveness in Parity Games: Mean-Payoff Parity Games Revisited*. Tech. rep. arXiv, 2011. arXiv: 1102.3615.

[9] Patricia Bouyer, Nicolas Markey, Jörg Olschewski, and Michael Ummels. "Measuring Permissiveness in Parity Games: Mean-Payoff Parity Games Revisited." In: *ATVA*. Ed. by Tevfik Bultan and Pao-Ann Hsiung. Vol. 6996. LNCS. Springer-Verlag, 2011, pp. 135–149. ISBN: 978-3-642-24371-4. DOI: 10.1007/978-3-642-24372-1_11.

[10] Stefan Breuers, Christof Löding, and Jörg Olschewski. "Improved Ramsey-based Büchi Complementation." In: *FoSSaCS*. Vol. 7213. LNCS. Springer-Verlag, 2012, pp. 150–164. DOI: 10.1007/978-3-642-28729-9_10.

[11] Janusz A. Brzozowski and Robert Knast. "The Dot-Depth Hierarchy of Star-Free Languages is Infinite." In: *Journal of Computer and System Sciences* 16.1 (1978), pp. 37–55. DOI: 10.1016/0022-0000(78)90049-1.

[12] J. Richard Büchi. "On a decision method in restricted second order arithmetic." In: *Logic, Methodology and Philosophy of Science*. Stanford Univerity Press, 1962, pp. 1–11.

[13] J. Richard Büchi and Calvin C. Elgot. "Decision problems of weak second-order arithmetics and finite automata." In: *Notices of the American Mathematical Society* 5.834 (1958).

[14] J. Richard Büchi and Lawrence H. Landweber. "Solving Sequential Conditions by Finite-State Strategies." In: *Transactions of the American Mathematical Society* 138 (1969), pp. 295–311. ISSN: 00029947. DOI: 10.2307/1994916.

[15] Arindam Chakrabarti, Luca de Alfaro, Thomas A. Henzinger, and Mariëlle Stoelinga. "Resource Interfaces." In: *EMSOFT*. Ed. by Rajeev Alur and Insup Lee. Vol. 2855. LNCS. Springer-Verlag, 2003, pp. 117–133. DOI: 10.1007/978-3-540-45212-6_9.

[16] Krishnendu Chatterjee and Laurent Doyen. "Energy Parity Games." In: *ICALP (2)*. Ed. by Samson Abramsky, Cyril Gavoille, Claude Kirchner, Friedhelm Meyer auf der Heide, and Paul G. Spirakis. Vol. 6199. LNCS. Springer-Verlag, 2010, pp. 599–610. ISBN: 978-3-642-14161-4. DOI: 10.1007/978-3-642-14162-1_50.

[17] Krishnendu Chatterjee, Laurent Doyen, Thomas A. Henzinger, and Jean-François Raskin. "Generalized Mean-payoff and Energy Games." In: *FSTTCS*. Ed. by Kamal Lodaya and Meena Mahajan. Vol. 8. LIPIcs. Schloss Dagstuhl, 2010, pp. 505–516. ISBN: 978-3-939897-23-1. DOI: 10.4230/LIPIcs.FSTTCS.2010.505.

[18] Krishnendu Chatterjee, Thomas A. Henzinger, and Marcin Jurdziński. "Mean-Payoff Parity Games." In: *LICS*. IEEE Computer Society Press, 2005, pp. 178–187. ISBN: 0-7695-2266-1. DOI: 10.1109/LICS.2005.26.

[19] Krishnendu Chatterjee, Thomas A. Henzinger, and Nir Piterman. "Generalized Parity Games." In: *FoSSaCS*. Ed. by Helmut Seidl. Vol. 4423. LNCS. Springer-Verlag, 2007, pp. 153–167. DOI: 10.1007/978-3-540-71389-0_12.

[20] Namit Chaturvedi, Jörg Olschewski, and Wolfgang Thomas. "Languages vs. ω-Languages in Regular Infinite Games." In: *DLT*. Ed. by Giancarlo Mauri and Alberto Leporati. Vol. 6795. LNCS. Springer-Verlag, 2011, pp. 180–191. ISBN: 978-3-642-22320-4. DOI: 10.1007/978-3-642-22321-1_16.

[21] Namit Chaturvedi, Jörg Olschewski, and Wolfgang Thomas. "Languages vs. ω-Languages in Regular Infinite Games." In: *International Journal of Foundations of Computer Science* 23.5 (2012), pp. 985–1000. DOI: 10.1142/S0129054112400412.

[22] Alonzo Church. "Applications of recursive arithmetic to the problem of circuit synthesis." In: *Summaries of the Summer Institute of Symbolic Logic*. Vol. 1. Cornell University. Ithaca, N.Y., 1957, pp. 3–50.

[23] Rina S. Cohen and Janusz A. Brzozowski. "Dot-depth of star-free events." In: *Journal of Computer and System Sciences* 5 (1 Feb. 1971), pp. 1–16. ISSN: 0022-0000. DOI: 10.1016/S0022-0000(71)80003-X.

[24] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. 3rd ed. MIT Press, 2009.

[25] Heinz-Dieter Ebbinghaus, Jörg Flum, and Wolfgang Thomas. *Mathematical logic*. 2nd ed. Springer-Verlag, 1994. ISBN: 978-3-540-94258-0.

[26] Andrzej Ehrenfeucht and Jan Mycielski. "Positional strategies for mean payoff games." In: *International Journal of Game Theory* 8.2 (1979), pp. 109–113. ISSN: 0020-7276. DOI: 10.1007/BF01768705.

[27] E. Allen Emerson and Charanjit S. Jutla. "Tree Automata, Mu-Calculus and Determinacy." In: *FOCS*. IEEE Computer Society Press, 1991, pp. 368–377. DOI: 10.1109/SFCS.1991.185392.

[28] Seth Fogarty, Orna Kupferman, Moshe Y. Vardi, and Thomas Wilke. "Unifying Büchi Complementation Constructions." In: *CSL*. Vol. 12. LIPIcs. Schloss Dagstuhl, 2011, pp. 248–263. DOI: 10.4230/LIPIcs.CSL.2011.248.

[29] Seth Fogarty and Moshe Y. Vardi. "Efficient Büchi Universality Checking." In: *TACAS*. Ed. by Javier Esparza and Rupak Majumdar. Vol. 6015. LNCS. Springer-Verlag, 2010, pp. 205–220. ISBN: 978-3-642-12001-5. DOI: 10.1007/978-3-642-12002-2_17.

[30] Wladimir Fridman. "Formats of Winning Strategies for Six Types of Pushdown Games." In: *GandALF*. Ed. by Angelo Montanari, Margherita Napoli, and Mimmo Parente. Vol. 25. Electronic Proceedings in Theoretical Computer Science. 2010, pp. 132–145. DOI: 10.4204/EPTCS.25.14.

[31] Ehud Friedgut, Orna Kupferman, and Moshe Y. Vardi. "Büchi Complementation Made Tighter." In: *International Journal of Foundations of Computer Science* 17.4 (2006), pp. 851–868. DOI: 10.1142/S0129054106004145.

[32] David Gale and Frank M. Stewart. "Infinite Games with Perfect Information." In: *Annals of Mathematical Studies*. Contributions to the Theory of Games 28.2 (1953). Ed. by Harold W. Kuhn and Albert W. Tucker, pp. 245–266.

[33] Erich Grädel, Wolfgang Thomas, and Thomas Wilke, eds. *Automata, Logics, and Infinite Games. A Guide to Current Research*. Vol. 2500. LNCS. Springer-Verlag, 2002. ISBN: 3-540-00388-6.

[34] David Gries. "Describing an Algorithm by Hopcroft." In: *Acta Informatica* 2 (2 1973), pp. 97–109. ISSN: 0001-5903. DOI: 10.1007/BF00264025.

[35] Yuri Gurevich and Leo Harrington. "Trees, Automata, and Games." In: *STOC*. Ed. by Harry R. Lewis, Barbara B. Simons, Walter A. Burkhard, and Lawrence H. Landweber. ACM Press, 1982, pp. 60–65. ISBN: 0-89791-070-2. DOI: 10.1145/800070.802177.

[36] John E. Hopcroft. *An n log n algorithm for minimizing states in a finite automaton*. Tech. rep. Stanford, CA, USA: Stanford University, 1971.

[37] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. 2nd ed. Addison-Wesley series in computer science. Addison-Wesley-Longman, 2001. ISBN: 978-0-201-44124-6.

[38] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages and Computation*. 1st ed. Addison-Wesley, 1979. ISBN: 0-201-02988-X.

[39] Marcin Jurdziński. "Deciding the Winner in Parity Games is in UP ∩ co-UP." In: *Information Processing Letters* 68.3 (1998), pp. 119–124. DOI: 10.1016/S0020-0190(98)00150-1.

[40] Detlef Kähler and Thomas Wilke. "Complementation, Disambiguation, and Determinization of Büchi Automata Unified." In: *ICALP (1)*. Vol. 5125. LNCS. Springer-Verlag, 2008, pp. 724–735. DOI: 10.1007/978-3-540-70575-8_59.

[41] Richard M. Karp. "A characterization of the minimum cycle mean in a digraph." In: *Discrete Mathematics* 23.3 (1978), pp. 309–311. ISSN: 0012-365X. DOI: 10.1016/0012-365X(78)90011-0.

[42] Nils Klarlund. "Progress Measures for Complementation of ω-Automata with Applications to Temporal Logic." In: *FOCS*. IEEE Computer Society Press, 1991, pp. 358–367. DOI: 10.1109/SFCS.1991.185391.

[43] Eryk Kopczyński. "Half-Positional Determinacy of Infinite Games." In: *ICALP (2)*. Ed. by Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener. Vol. 4052. LNCS. Springer-Verlag, 2006, pp. 336–347. ISBN: 3-540-35907-9. DOI: 10.1007/11787006_29.

[44] Michael Luttenberger. *Strategy Iteration using Non-Deterministic Strategies for Solving Parity Games*. Tech. rep. arXiv, 2008. arXiv: 0806.2923.

[45] Donald A. Martin. "Borel Determinacy." In: *Annals of Mathematics* 102 (1975), pp. 363–371. DOI: 10.2307/1971035.

[46] Robert McNaughton. "Finite-state infinite games." In: *Project MAC Rep.* (Sept. 1965).

[47] Robert McNaughton. "Testing and Generating Infinite Sequences by a Finite Automaton." In: *Information and Control* 9.5 (1966), pp. 521–530. DOI: 10.1016/S0019-9958(66)80013-X.

[48] M. Michel. *Complementation is more difficult with automata on infinite words*. Tech. rep. CNET, Paris, 1988.

[49] Andrzej W. Mostowski. *Games with forbidden positions*. Tech. rep. 78. Instytut Matematyki, Uniwersytet Gdański, Poland, 1991.

[50] Jörg Olschewski. *The Alekto Project Homepage*. 2011. URL: http://www.automata.rwth-aachen.de/research/Alekto/.

[51] Dominique Perrin and Jean-Éric Pin. "First-order logic and star-free sets." In: *Journal of Computer and System Sciences* 32.3 (June 1986), pp. 393–406. ISSN: 0022-0000. DOI: 10.1016/0022-0000(86)90037-1.

[52] Dominique Perrin and Jean-Éric Pin. *Infinite Words*. Amsterdam: Elsevier, 2004.

[53] Jean-Éric Pin. "Positive Varieties and Infinite Words." In: *LATIN*. Ed. by Claudio L. Lucchesi and Arnaldo V. Moura. Vol. 1380. LNCS. Springer-Verlag, 1998, pp. 76–87. ISBN: 3-540-64275-7. DOI: 10.1007/BFb0054312.

[54] Jean-Éric Pin. *Varieties of Formal Languages*. North Oxford, London and Plenum, New-York, 1986.

[55] Sophie Pinchinat and Stéphane Riedweg. "You Can Always Compute Maximally Permissive Controllers Under Partial Observation When They Exist." In: *ACC*. Vol. 4. 2005, pp. 2287–2292. DOI: 10.1109/ACC.2005.1470310.

[56] Nir Piterman. "From Nondeterministic Büchi and Streett Automata to Deterministic Parity Automata." In: *Logical Methods in Computer Science* 3.3 (2007). DOI: 10.2168/LMCS-3(3:5)2007.

[57] Amir Pnueli. "Applications of Temporal Logic to the Specification and Verification of Reactive Systems: A Survey of Current Trends." In: *Current Trends in Concurrency*. Ed. by J. W. de Bakker, Willem P. de Roever, and Grzegorz Rozenberg. Vol. 224. LNCS. Springer-Verlag, 1986, pp. 510–584. ISBN: 3-540-16488-X. DOI: 10.1007/BFb0027047.

[58] Michael O. Rabin. *Automata on Infinite Objects and Church's Problem*. Boston, MA, USA: American Mathematical Society, 1972. ISBN: 978-0-8218-1663-9.

[59] Alexander Rabinovich and Wolfgang Thomas. "Logical Refinements of Church's Problem." In: *CSL*. Ed. by Jacques Duparc and Thomas A. Henzinger. Vol. 4646. LNCS. Springer-Verlag, 2007, pp. 69–83. ISBN: 978-3-540-74914-1. DOI: 10.1007/978-3-540-74915-8_9.

[60] Frank P. Ramsey. "On a problem of formal logic." In: *Proceedings of the London Mathematical Society* 2.1 (1930), pp. 264–286. ISSN: 0024-6115. DOI: 10.1112/plms/s2-30.1.264.

[61] Shmuel Safra. "On the Complexity of ω-Automata." In: *FOCS*. IEEE Computer Society Press, 1988, pp. 319–327. DOI: 10.1109/SFCS.1988.21948.

[62] Sven Schewe. "Büchi Complementation Made Tight." In: *STACS*. Vol. 3. LIPIcs. Schloss Dagstuhl, 2009, pp. 661–672. DOI: 10.4230/LIPIcs.STACS.2009.1854.

[63] Victor L. Selivanov. "Fine Hierarchy of Regular Aperiodic ω-Languages." In: *DLT*. Ed. by Tero Harju, Juhani Karhumäki, and Arto Lepistö. Vol. 4588. LNCS. Springer-Verlag, 2007, pp. 399–410. ISBN: 978-3-540-73207-5. DOI: 10.1007/978-3-540-73208-2_37.

[64] Victor L. Selivanov. "Fine Hierarchy of Regular Aperiodic ω-Languages." In: *International Journal of Foundations of Computer Science* 19.3 (2008), pp. 649–675. DOI: 10.1142/S0129054108005875.

[65] Imre Simon. "Hierarchies of events with dot-depth one." PhD thesis. University of Waterloo, 1972.

[66] Imre Simon. "Piecewise testable events." In: *Automata Theory and Formal Languages*. Ed. by Helmut Brakhage. Vol. 33. LNCS. Springer-Verlag, 1975, pp. 214–222. ISBN: 3-540-07407-4. DOI: 10.1007/3-540-07407-4_23.

[67] A. Prasad Sistla, Moshe Y. Vardi, and Pierre Wolper. "The Complementation Problem for Büchi Automata with Applications to Temporal Logic." In: *Theoretical Computer Science* 49 (1987), pp. 217–237. DOI: 10.1016/0304-3975(87)90008-9.

[68] Ludwig Staiger and Klaus W. Wagner. "Automatentheoretische und automatenfreie Charakterisierungen topologischer Klassen regulärer Folgenmengen." German. In: *Elektronische Informationsverarbeitung und Kybernetik* 10.7 (1974), pp. 379–392.

[69] Jacques Stern. "Characterizations of some classes of regular events." In: *Theoretical Computer Science* 35 (1985), pp. 17–42. ISSN: 0304-3975. DOI: 10.1016/0304-3975(85)90003-9.

[70] Howard Straubing. *Finite Automata, Formal Logic, and Circuit Complexity*. Basel, Switzerland: Birkhäuser Verlag, 1994. ISBN: 3-7643-3719-2.

[71] Wolfgang Thomas. "Automata on Infinite Objects." In: *Handbook of Theoretical Computer Science*. Vol. B: *Formal Models and Semantics*. Ed. by Jan van Leeuwen. Amsterdam: Elsevier, 1990, pp. 133–192. ISBN: 978-0-444-88074-1.

[72] Wolfgang Thomas. "Church's Problem and a Tour through Automata Theory." In: *Pillars of Computer Science*. Ed. by Arnon Avron, Nachum Dershowitz, and Alexander Rabinovich. Vol. 4800. LNCS. Springer-Verlag, 2008, pp. 635–655. ISBN: 978-3-540-78126-4. DOI: 10.1007/978-3-540-78127-1_35.

[73] Wolfgang Thomas. "Classifying regular events in symbolic logic." In: *Journal of Computer and System Sciences* 25.3 (1982), pp. 360–376. ISSN: 0022-0000.

[74] Wolfgang Thomas. "Languages, automata, and logic." In: *Handbook of Formal Languages*. Vol. 3: *Beyond Words*. Ed. by Grzegorz Rozenberg and Arto Salomaa. New York, NY, USA: Springer-Verlag, 1997, pp. 389–455. ISBN: 978-3540-60649-9.

[75] Wolfgang Thomas. "On the Synthesis of Strategies in Infinite Games." In: *STACS*. 1995, pp. 1–13. DOI: 10.1007/3-540-59042-0_57.

[76] Boris A. Trakhtenbrot. "Finite automata and logic of monadic predicates." In: *Doklady Akademii Nauk SSSR* 140 (1961), pp. 326–329.

[77] Ming-Hsien Tsai, Seth Fogarty, Moshe Y. Vardi, and Yih-Kuen Tsay. "State of Büchi Complementation." In: *CIAA*. Vol. 6482. LNCS. Springer-Verlag, 2010, pp. 261–271. DOI: 10.1007/978-3-642-18098-9_28.

[78] Ming-Hsien Tsai and Yih-Kuen Tsay. personal communication. 2011.

[79] Antti Valmari. "Fast brief practical DFA minimization." In: *Information Processing Letters* 112.6 (Mar. 2012), pp. 213–217. ISSN: 0020-0190. DOI: 10.1016/j.ipl.2011.12.004.

[80] Antti Valmari and Petri Lehtinen. "Efficient Minimization of DFAs with Partial Transition Functions." In: *STACS*. Ed. by Susanne Albers and Pascal Weil. Vol. 1. LIPIcs. Schloss Dagstuhl, 2008, pp. 645–656. ISBN: 978-3-939897-06-4. DOI: 10.4230/LIPIcs.STACS.2008.1328.

[81] Qiqi Yan. "Lower Bounds for Complementation of ω-Automata Via the Full Automata Technique." In: *Logical Methods in Computer Science* 4.1 (2008). DOI: 10.2168/LMCS-4(1:5)2008.

[82] Wiesław Zielonka. "Infinite games on finitely coloured graphs with applications to automata on infinite trees." In: *Theoretical Computer Science* 200.1–2 (1998), pp. 135–183. DOI: 10.1016/S0304-3975(98)00009-7.

[83] Wiesław Zielonka. "Perfect-Information Stochastic Parity Games." In: *FoSSaCS*. Ed. by Igor Walukiewicz. Vol. 2987. LNCS. Springer-Verlag, 2004, pp. 499–513. ISBN: 3-540-21298-1. DOI: 10.1007/978-3-540-24727-2_35.

[84] Uri Zwick and Mike Paterson. "The Complexity of Mean Payoff Games on Graphs." In: *Theoretical Computer Science* 158.1&2 (1996), pp. 343–359. DOI: 10.1016/0304-3975(95)00188-3.