Parity and Generalised Büchi Automata

Determinisation and Complementation

Thesis submitted in accordance with the requirements of the University of Liverpool for the degree of

Doctor in Philosophy

by

Praveen Thomas Methrayil Varghese

November 2014

Department of Computer Science

UNIVERSITY OF LIVERPOOL

Thesis committee: Prof. Dr. Wolfgang Thomas and Dr. Dominik Wojtczak
Supervisory team: Dr. Sven Schewe (Primary) and Prof. Dr. Frank Wolter (Secondary)
Date of examination: 20 January 2015

Abstract

In this thesis, we study the problems of determinisation and complementation of finite automata on infinite words. We focus on two classes of automata that occur naturally: generalised Büchi automata and nondeterministic parity automata. Generalised Büchi and parity automata occur naturally in model-checking, realisability checking and synthesis procedures. We first review a tight determinisation procedure for Büchi automata, which uses a simplification of Safra trees called *history trees*. As Büchi automata are special types of both generalised Büchi and parity automata, we adjust the data structure to arrive at suitably tight determinisation constructions for both generalised Büchi and parity automata.

As the parity condition describes combinations of Büchi and CoBüchi conditions, instead of immediately modifying the data structure to handle parity automata,we arrive at a suitable data structure by first looking at a special case, Rabin automata with one accepting pair. One pair Rabin automata correspond to parity automata with three priorities and serve as a starting point to modify the structures that result from Büchi determinisation: we then nest these structures to reflect the standard parity condition and describe a direct determinisation construction.

The generalised Büchi condition is characterised by an accepting family with k accepting sets. It is easy to extend classic determinisation constructions to handle generalised Büchi automata by incorporating the degeneralization algorithm in the determinisation construction. We extend the tight Büchi construction to do exactly this.

Our determinisation constructions go to deterministic Rabin automata. It is known that one can determinise to the more convenient parity condition by incorporating the standard *Latest Appearance Record* construction in the determinisation procedure. We determinise to parity automata using this technique.

We prove lower bounds on these constructions. In the case of determinisation to Rabin automata, our constructions are tight to the state. In the case of determinisation to parity, there is a constant factor ≤ 1.5 between upper and lower bounds reducing to optimal(to the state) in the case of Büchi and 1-pair Rabin.

We also reconnect tight determinisation and complementation and provide constructions for complementing generalised Büchi and parity automata by starting with our data structure for determinisation. We introduce suitable data structures for the complementation procedures based on the data structure used for determinisation. We prove lower bounds for both constructions that are tight upto an $O(n)$ factor where n is the number of states of the nondeterministic automaton that is complemented.

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Sven Schewe. I started my research based on his work about determinising Büchi automata and I am amazed that this particular problem evolved into the papers we published and finally became the backbone of this thesis. I owe my progress as a researcher to his supervision. I am also thankful to him for his friendship, pleasant company and the coffee breaks that brightened up the great British weather and that led to my coffee addiction.

My introduction to this line of research was Wolfgang Thomas' excellent survey on 'Languages, Automata, and Logic'. It is my great privilege to have had such a distinguished scientist as Wolfgang on my thesis examination committee along with Dominik Wojtczak. Their examination of my thesis gave me more perspective on my research and I thank them for their helpful comments which have improved my thesis greatly.

It was good fun to work with Ashutosh Trivedi on strategy improvement. I learnt a lot about a topic I had not worked on before and this work with Ashu and Sven led to a paper at ICALP. I am especially grateful for his great company on several evenings out in Liverpool.

I want to thank my colleagues in Liverpool – Petar Iliev, John Fearnley, Amir Kermani, Anshul Gupta, Ana Ozaki, Will Gatens, Julio Lemos and André Hernich. Thank you for all the inspiring, and even the random discussions over coffee, for the reading group and for making my time as a Ph.D. student more pleasurable.

Nir Piterman got me started with ω-automata when I was a Master's student at Leicester. I would not have been a researcher were it not for his supervision of my Master's thesis and for this I am eternally indebted.

I wish to thank the EPSRC and the Department of Computer Science for supporting my Ph.D. studies financially, especially the Department of Computer Science for providing a good research environment.

My time in Liverpool would have been a lot less enjoyable had I not met Gary, Hannah, Olivia and Nahum. Thank you Worralls for providing a place for me to stay, for feeding me, and for all the good times.

Finally, to my family. I dedicate this thesis to them. Mum, Dad and Monica, thank you for everything.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise, that this work was undertaken during my period of study at the University of Liverpool and has not been submitted for any other degree or professional qualification except as specified.

The first section of chapter 3 contains a construction from [Sch09b]. This construction is modified to suit transition-labeled automata as input but the technical details are almost the same.

The technical parts of this thesis are contained in Chapters 3, 4 and 5. These are a result of joint research with Sven Schewe. Most of these results were published in ATVA 2012([SV12]) and MFCS 2014 ([SV14a] and [SV14b]).

Contents

Chapter 1

Introduction

Finite automata on infinite structures and finite games of infinite duration are two theories that have often influenced each other, both having been inspired by Church's realisability problem [Chu62]. The term "ω-automata" generally refers to finite automata that accept or reject words of infinite length, or, ω-words. ω-automata were first introduced in Büchi's decidability proof for the monadic second-order logic of one successor (S1S) [Büc62]. Following from Büchi's result, they have formed the basis for the theories of model checking, realisability checking, and synthesis procedures for linear time temporal logic (LTL) [Pnu77].

The ω-automata introduced by Büchi (that were subsequently named after him) extend the well known theory of finite automata on finite words to languages over infinite words. *Büchi* automata are the most studied form of ω-automata. They extend finite automata in the sense that, while finite runs of finite automata are accepting if an accepting state is visited at the end of the run, an infinite run of a Büchi automaton is accepting if a final state is visited (or a final transition is taken) infinitely many times during the course of the run. The Büchi acceptance condition thus specifies a set of states(or transitions) that have to be visited(respectively, taken) infinitely often.

Although the connection to finite automata on finite words might seem to suggest that automata manipulations for Büchi automata are equally simple, this is un-

fortunately not the case. In particular, Büchi automata are not closed under determinisation. While finite automata can simply be determinised with the subset construction [RS59], it turns out that deterministic Büchi automata do not recognise the same languages as nondeterministic Büchi automata. For example, deterministic Büchi automata cannot recognise the simple ω-regular language that consists of all infinite words that contain only finitely many 0's over an alphabet $\{0, 1\}$.

As a consequence of McNaughton's result, there arose a need for more specific acceptance conditions, such as Muller's subset condition, Rabin's *pairs* condition [Rab69] or its dual, the Streett condition [Str82], or the *parity* condition. Yan established an $n^{\Omega(n)}$ lower bound for the determinisation of Büchi automata [Yan08] even for determinising to Muller automata, which implies that the standard subset, or a breakpoint construction is not enough.

A related acceptance condition to the Büchi condition is the *generalised* Büchi condition, which comprises of an accepting family. It requires that a final state (or final transition) from each accepting set is visited (respectively, is taken) infinitely many times. The standard translation from LTL to ω-automata [GPVW95] goes to generalised Büchi automata with the acceptance condition on the transitions.

In this thesis, we study the problems of the determinisation and complementation of nondeterministic Büchi, generalised Büchi, and parity automata. We have already outlined the importance of Büchi automata in the context of verification. The standard translation from LTL to automata results in generalised Büchi automata. Parity automata are also particularly important, given that the parity condition naturally recognizes languages specified by fixed-point expressions. Although nondeterministic parity automata are as equally expressive as Büchi automata, as they are more succinct, they can easily encode other conditions, such as the intersection of Büchi and co-Büchi conditions.

We study *transition-labeled* automata in this thesis. Transition-based acceptance mechanisms have proven to be a more natural target of automata transformations. A generalisation of the ranking-based complementation procedures quoted above to

transition-based acceptance is straight forward, and the Safra-style determinisation procedures from the literature [Saf88, Saf92, Pit07, Sch09b] have a natural representation with an acceptance condition on transitions. Their translation to state-based acceptance is by multiplying the acceptance from the last transition to the statespace.

A similar observation can be made for other automata transformations, like the removal of ε-transitions from translations of μ-calculi [Wil01, SF06a] and the treatment of asynchronous systems [SF06b], where the state-space grows by multiplication with the acceptance information (e.g., maximal priority on a finite sequence of transitions), while it cannot grow in case of transition-based acceptance. Similarly, tools like SPOT [Dur14] offer more concise automata with transition-based acceptance mechanism as a translation from LTL. Using state-based acceptance in the automaton that we want to determinise or complement would also complicate the presentation of the complementation procedure. But first and foremost, using transition-based acceptance provides cleaner results.

This is the case because in state-based acceptance, the role of the states is overloaded. In finite automata over infinite structures, each state represents the class of tails of the word that can be accepted from this state. In state-based acceptance, they have to account for the acceptance mechanism itself, too, while they are relieved from this burden in transition-based acceptance. In complementation techniques based on rankings, this results in a situation where states with certain properties, such as final states for Büchi automata, can only occur with some ranks, but not with all.

As transition-based acceptance separates these concerns, the presentation becomes cleaner. The natural downside is that, for example, we lose the $n^{O(n)}$ bound [CZ11b] for parity complementation, as the number of priorities in a parity automaton with transition-based acceptance can grow arbitrarily. But in return, we do get a clean and simple complementation procedure.

1.1 Determinisation of ω-automata

The determinisation of ω-automata was a key step in Rabin's extension of Büchi's proof to the case of trees [Rab69]. Rabin's proof built on McNaughton's doubly exponential determinisation construction [McN66]. Safra was the first to introduce singly-exponential determinisation constructions for Büchi [Saf88] and Streett [Saf92] automata and current determinisation techniques [Pit07, Sch09b] build on Safra's work. For instance, Schewe's determinisation technique for Büchi automata yielding deterministic Rabin automata[Sch09b] builds on Safra's [Saf88] and Piterman's [Pit07] determinisation procedures. It uses a separation of concern, where the main acceptance mechanism, represented by *history trees*, is separated from the formal acceptance condition, e.g., a Rabin (used by [Saf88, Saf92]) or parity condition (used by [Pit07]). History trees can be seen as a simplification of Safra trees [Saf88]. In a nutshell, they represent a family of breakpoint constructions: sufficiently many to identify an accepting run, and sufficiently few to be concise.

There are also other constructions that tackle determinisation from a different viewpoint. An example is Muller and Schupp's [MS95] presentation of a nondeterminisation technique for alternating tree automata shows a connection to the determinisation of finite automata on infinite words. Kähler and Wilke build on this construction in [KW08]. Although they start from a different viewpoint, their method seems to converge with the Safra-based constructions.

In this thesis, we adapt Schewe's determinisation procedure [Sch09b] to determinise parity automata. We investigate what is required to handle such an adaptation and devise a similar determinisation procedure for nondeterministic parity automata.

We also consider generalised Büchi automata. There are several ways to determinise a generalised Büchi automaton with n states and k accepting sets. One could start with translating the resulting generalised Büchi automaton first to an ordinary nondeterministic Büchi automaton with nk states and a single accepting set, result-

ing in a determinisation complexity of roughly $(nk)^{O(nk)}$ states, or one could treat it as a Streett automaton, which is equally expensive and has a more complex determinisation construction.

Schewe's determinisation procedure [Sch09b] again proves to be an easy target for generalisation, because it separates the representation of the history of a run from the acceptance condition. To extend this technique from ordinary to generalised Büchi, it suffices to apply a round-robin approach to all breakpoints under consideration. That is, each subset is enriched by a natural number identifying the accepting set, for which we currently seek to see the following breakpoint. Each time a breakpoint is reached, we turn to the next accepting set. Note that this algorithm is a generalisation in the narrower sense: in case that there is exactly one accepting set, it behaves exactly as the determinisation procedure for Büchi automata in [Sch09b]. An algorithm to determinise generalised Büchi automata to deterministic parity automata using this method was used in [KPV06], similarly extending Piterman's construction [Pit07, LW09].

The constructions in this thesis provide Rabin automata as output. We give estimations of the size of the deterministic automata. In the case of nondeterministic generalised Büchi automata, we find that for an automaton with n states and k accepting sets, we get a deterministic Rabin automaton with $\mathsf{ght}_k(n)$ states and $2^n - 1$ Rabin pairs. The function $\mathsf{ght}_k(n)$(enumerating the number of possible states of the deterministic automaton) is approximately $(1.65\,n)^n$ for $k = 1$, $(3.13\,n)^n$ for $k = 2$, and $(4.62n)^n$ for $k = 3$, and converges against $(1.47kn)^n$ for large k. These bounds can also be used to establish smaller maximal sizes of minimal models, which is useful for Safraless determinisation procedures [KV05, FS13, KPV06]. For the transformation to deterministic parity automata from [Sch09b], we obtain an automaton with $O(n!^2\, k^n)$ states and $2n + 1$ priorities.

In the case of nondeterministic parity automata, we do not provide a similar estimation for the size of the deterministic automata (except for the simple cases of

Büchi and 1-pair Rabin automata), but we show tightness up to a constant factor of 1.5.

Our constructions lead to deterministic Rabin automata as targets. While these constructions are tight with respect to the number of states of the deterministic automaton, the acceptance condition is exponential in the number of Rabin pairs. Determinising to parity automata seems to be an even more attractive target since emptiness games for parity automata [Sch07, JPZ08] have a lower computational complexity compared to emptiness games for Streett or Rabin automata [PP06]. We therefore show how we can obtain deterministic parity automata in each case by using our construction to deterministic Rabin automata as an intermediate step.

Colcombet and Zdanowski [CZ09] showed that Schewe's determinisation procedure for Büchi automata is optimal. We extend this lower bound to generalised Büchi automata and parity automata generalising their techniques, showing that the determinisation procedures are optimal.

1.1.1 Timeline of complexity results

We give here a timeline of the sequence of results with respect to the determinisation problem for Büchi automata.

	Target automaton	**Complexity**
1966 - McNaughton	Deterministic Muller	$2^{2^{O(n)}}$
1988 - Safra	Deterministic Rabin	$n^{O(n)}$, $\approx 12^n\, n^{2n}$
2006 - Piterman	Deterministic parity	$\approx n!\, n^n$
2009 - Liu and Wang, separately -Schewe	Deterministic parity	$\approx n!^2$
2009 - Schewe	Deterministic Rabin trace	$\approx (1.65\, n)^n$
2009 - Colcombet and Zdanowski	Deterministic Rabin trace	$\theta(1.65\, n)^n$

Table 1.1: Determinisation timeline for Büchi automata

In the case of Büchi determinisation, the best current determinisation procedure that determinises to Rabin automata is Schewe's construction [Sch09b]. A tight matching bound for this construction is shown by Colcombet and Zdanowski in [CZ09]. The best current determinisation procedure that produces parity automata

as output is Piterman's construction [Pit07]. We show in this thesis that the construction in [Pit07] is optimal.

For generalised Büchi automata, a construction that is similar to Piterman's Büchi determinisation construction is shown in [KPV06]. This is the construction with the best upper bound. We provide a matching lower bound for this construction. We also provide matching upper and lower bounds for the determinisation of generalised Büchi automata to deterministic Rabin automata.

There are two ways to determinise parity automata. One could convert them first to Büchi automata and then apply one of the above constructions on them. The other way is to directly determinise. We provide the first direct determinisation procedure for parity automata along with tight bounds for the construction.

1.2 Complementation of ω-automata

The earliest results on ω-automata involved the complementation problem. Büchi's seminal paper [Büc62] on the correspondence between logic and automata showed that Büchi automata are closed under complementation and introduced a complementation algorithm that uses Ramsey theory. Later results in the field of formal verification established the importance of the complementation problem for ω-automata, for e.g., the language inclusion problem uses complementation [Kur94]. Another important use of complementation is in checking the correctness of translation techniques [Var07, TTH13a]. The GOAL tool [TTH13a] shows this with a test suite that incorporates recent algorithms [Saf88, Tho99, KV01, Pit07] for Büchi complementation.

Given the importance of this problem, it is not surprising that the complementation of ω-automata is much researched. In particular, there has been a long and fruitful quest for the exact complexity of complementation algorithms for Büchi automata [Büc62, SS78, Péc86, PSVW87, Saf88, Mic88, Tho99, Löd99, KV01, GKSV03,

FKV06, Pit07, Var07, Yan08]. We will highlight some of the important results with respect to this problem now.

The first important result with respect to the complementation of $\omega-$ automata is Büchi's proof that nondeterministic Büchi automata are closed under complementation. Using a Ramsey-based argument Büchi came up with a doubly-exponential complementation algorithm [Büc62]. This is much harder than complementing finite automata on finite words — an exponential determinisation procedure followed by a reversal of final states yields an efficient complementation algorithm that is exponential. The complexity of the determinisation problem for ω-automata rules out an analogous algorithm with similar complexity.

Further results on the complexity of the complementation problem followed in the late 1980s, starting with the discovery of an EXPTIME upper bound [Péc86, PSVW87]. There was a caveat: the complementation techniques used produced automata with up to $2^{O(n^2)}$ states, meaning that there was still an exponential gap to the lower bounds of Sakoda and Sipser for the case of automata on finite words.

Safra's brilliant algorithm for the determinisation of Büchi automata suggested a $n^{O(n)}$ bound for Büchi complementation, an upper bound that was seemingly matched by Max Michel's [Mic88] $\Omega(n!)$ lower bound implying that Büchi complementation is in $n^{\theta(n)}$, suggesting that Safra's determinisation procedure also resulted in tight complementation.

However, Vardi [Var07] rightly pointed out that this impression of tightness was in fact not true, because the *big-Oh* notation in the exponent hides an $n^{\theta(n)}$ gap between upper and lower bounds.

It turns out that complementation procedures for ω-automata are more efficient than determinisation, contrary to the case of finite words. By studying alternating automata, complementation procedures that bypassed determinisation were found. The gap between upper ($(6n)^n$) and lower bounds (from Michel) was then narrowed down to $2^{\theta(n)}$ by techniques that observe the run graphs of alternating automata [KV01].

A further refinement of the above complementation technique builds on "tight level rankings" [FKV06, Yan08] improving the upper bound to $O\big((0.96n)^n\big)$. Yan [Yan08] using *full automata* improved the lower bound for complementation to $\Omega\big((0.76n)^n\big)$. Finally, Schewe [Sch09a] provided a matching upper bound, showing tightness up to an $O(n^2)$ factor reducing to an $O(n)$ factor for trace languages.

For Rabin, Streett, and parity automata, there has been much progress [CZL09, CZ11b, CZ11a], in particular establishing an $n^{\theta(n)}$ bound for parity complementation with state-based acceptance, which has been a great improvement and pushed tightness of parity complementation to the level known from Büchi complementation since the late 80s [Saf88, Mic88].

In this thesis, we discuss a bridge between optimal determinisation and tight complementation. We show how the nondeterministic power of an automaton can be exploited by using a more concise data structure compared to determinisation (flat trees instead of general ones). In the case of generalised Büchi automata, this bridge again results in a generalisation of the Büchi complementation procedure discussed in [Sch09a] in the narrower sense: for one accepting set, the resulting automata coincide. We also provide a matching lower bound: we show for alphabets L_n^k that the size of a generalised Büchi automaton that recognises the complement of a full generalised Büchi automaton with n states and k accepting sets must be larger than $|L_n^k|$, while the ordinary Büchi automaton we construct is smaller than $|L_{n+1}^{k+1}|$. For large k – that is, if k is not small compared to n – $|L_n^k|$ is approximately $\big(\frac{kn}{e}\big)^n$. This improves significantly over the $\big(\Omega(nk)\big)^n$ bound established by Yan [Yan08].

We then establish tight bounds for the complementation of parity automata with transition-based acceptance. We get a clean and simple complementation procedure based on a data structure we call flattened nested history trees (FNHTs), which is inspired by a generalisation of history trees to multiple levels, one for each even priority ≥ 2. We show that any procedure that determinises full parity automata with states Q and maximal priority π has at least $\mathsf{fnht}(Q,\pi)/2$ states, where $\mathsf{fnht}(Q,\pi)$ is the set of FNHTs for a given set Q of states and maximal priority π of the parity

automaton that is to be complemented. Our complementation construction uses a marker in addition for its acceptance mechanism. Essentially, it is used to mark some position of interest in an FNHT. It accounts for the $O(n)$ gap between the upper and lower bound. We show that, for $\pi \geq 2$ (and hence for Büchi automata upwards) the number of states of our complementation construction is bounded by $4n + 1$ times the lower bound.

1.2.1 Timeline of complexity results

We show here the timeline of results with respect to Büchi complementation.

	Complexity
1966 - Büchi	$2^{2^{O(n)}}$
1986 - Pecuchet	$2^{O(n^2)}$
1987 - Sistla, Vardi, Volper	$2^{O(n^2)}$
1988 - Safra	$n^{O(n)}$
1988 - Michel	$n^{\Omega(n)}$
1991 - Klarlund	$O\big((6n)^n\big)$
1997 - Kupferman, Vardi	$O\big((6n)^n\big)$
2004 - Friedgut, Kupferman, Vardi	$o\big((1.06\,n)^n\big)$
2006 - Yan	$\omega\big((0.76\,n)^n\big)$
2006 - Friedgut, Kupferman, Vardi	$o\big((0.97\,n)^n\big)$
2009 - Schewe	$o\big((0.76\,n)^n\big)$

Table 1.2: Timeline of complementation complexity results

In this thesis, we introduce direct complementation procedures for generalised Büchi and parity automata and provide tight bounds.

1.3 Structure of thesis

In **Chapter 2**, we define the necessary ideas needed for the problems of determinisation and complementation. We describe different acceptance types of automata and mention the folk results in this field.

In **Chapter 3**, we tackle the determinisation of several types of automata. We first review the tight Büchi construction. We then take a step towards the determinisation of parity automata by looking at the determinisation of a special case: Rabin

automata with one accepting pair. We then expand the data structure required to describe the state of the deterministic automaton for the case of parity determinisation and provide a construction to directly determinise parity automata. Next, we provide a construction to determinise generalised Büchi automata. All these constructions provide Rabin automata as output. We give estimations of the size of the deterministic automata. We then show how we can determinise to deterministic parity automata and we give estimations for these procedures.

In **Chapter 4**, we prove lower bounds for our constructions.

In **Chapter 5**, we discuss a bridge between optimal determinisation and tight complementation and show tight bounds for our complementation procedures.

In **Chapter 6**, we discuss our results.

Chapter 2

Preliminaries & problem statements

2.1 Preliminaries

We denote the set of non-negative integers by ω, i.e. $\omega = \{0, 1, 2, 3, ...\}$. For a finite alphabet Σ, an infinite *word* α is an infinite sequence $\alpha_0\alpha_1\alpha_2\cdots$ of letters from Σ. We sometimes interpret ω-words as functions $\alpha : i \mapsto \alpha_i$, and use Σ^ω to denote the ω-words over Σ.

Automata on infinite words. ω-automata are finite automata that are interpreted over infinite words and recognise ω-regular languages $L \subseteq \Sigma^\omega$. Nondeterministic ω-automata are quintuples $\mathcal{N} = (Q, \Sigma, I, T, \mathcal{F})$, where,

- Q is a finite set of states,
- I is a non-empty subset $I \subseteq Q$ of initial states,
- Σ is a finite alphabet,
- $T : Q \times \Sigma \times Q$ is a transition relation that maps states and input letters to sets of successor states and,
- $\mathcal{F}$ is an acceptance component. In this thesis, we consider Rabin, Streett, parity, Büchi and generalised Büchi acceptance.

Runs and transitions. A *run* ρ of a nondeterministic ω-automaton $\mathcal{N}$ on an input word α is an infinite sequence $\rho : \omega \to Q$ of states of $\mathcal{N}$, also denoted $\rho = q_0 q_1 q_2 \cdots \in Q^\omega$, such that the first symbol of ρ is an initial state $q_0 \in I$ and, for all $i \in \omega$, $(q_i, \alpha_i, q_{i+1}) \in T$ is a valid *transition*.

For a run ρ on a word α, we denote with $\overline{\rho} : i \mapsto \big(\rho(i), \alpha(i), \rho(i{+}1)\big)$ the transitions of ρ. Let $\mathsf{infin}(\rho) = \{q \in Q \mid \forall i \in \omega\ \exists j > i$ such that $\rho(j) = q\}$ denote the set of all states that occur infinitely often during the run ρ. Likewise, let $\mathsf{infin}(\overline{\rho}) = \{t \in T \mid \forall i \in \omega\ \exists j > i$ such that $\overline{\rho}(j) = t\}$ denote the set of all transitions that are taken infinitely many times in ρ.

For technical convenience we also allow for finite runs $q_0 q_1 q_2 \dots q_n$ with $T \cap \{q_n\} \times \{\alpha(n)\} \times Q = \emptyset$. Naturally, no finite run satisfies any ω-acceptance condition. Finite runs are rejecting, and have no influence on the language of an automaton.

Acceptance conditions. In this thesis, we use acceptance conditions over transitions. Acceptance mechanisms over states can be defined accordingly. *Rabin* automata are ω-automata, whose acceptance is referring to a family of pairs $\{(A_i, R_i) \mid i \in J\}$, with $A_i, R_i \subseteq T$, of accepting and rejecting transitions for all indices i of some index set J. A run ρ of a Rabin automaton is *accepting* if there is an index $i \in J$, such that infinitely many accepting transitions $t \in A_i$, but only finitely many rejecting transitions $t \in R_j$ occur in $\overline{\rho}$. That is, if there is an $i \in J$ such that $\mathsf{infin}(\overline{\rho}) \cap A_i \neq \emptyset = \mathsf{infin}(\overline{\rho}) \cap R_i$.

ω-automata that use the complementary *Streett* condition are called Streett automata. Their acceptance is defined by a family of pairs $\{(G_i, B_i) \mid i \in J\}$, with $G_i, B_i \subseteq T$, of good and bad transitions for all indices i of some index set J. A run ρ of a Streett automaton is *accepting* if, for all indices $i \in J$, some good transition $t \in G_i$ or no bad transition $t \in B_j$ occur infinitely often in $\overline{\rho}$. That is, if, for all $i \in J$, $\mathsf{infin}(\overline{\rho}) \cap G_i \neq \emptyset$ or $\mathsf{infin}(\overline{\rho}) \cap B_i = \emptyset$ holds.

One-pair Rabin automata $\mathcal{R}_1 = \big(Q, \Sigma, I, T, (A, R)\big)$, are Rabin automata with a

singleton index set, such that we directly refer to the only pair (A, R). They are of special technical interest in this paper.

Parity automata (first introduced by Mostowski in [Mos84]) are ω-automata, whose acceptance is defined by a priority function $\mathsf{pri} : T \to \Pi$ that maps transitions to a finite set $\Pi \subset \omega$ of non-negative integers. A run ρ of a parity automaton is *accepting* if $\limsup_{n\to\infty} \mathsf{pri}\big(\overline{\rho}(n)\big)$ is even, that is, if the highest priority that occurs infinitely often in the transitions of ρ is even. Parity automata can be viewed as special Rabin, or as special Streett automata. In older works, the parity condition was referred to as Rabin chain condition—because one can represent them by choosing A_i as the set of states with priority $\leq 2i$ and R_i as the sets of states with priorities $\leq 2i-1$, resulting in a chain $A_i \subseteq R_i \subseteq A_{i+1} \subseteq \ldots$—or a Streett chain condition—where G_i is the set of states with priority $\geq 2i$, and B_i is the set of states with priority $\geq 2i-1$. Some works also refer to the parity condition as the *Mostowski* condition.

Parity automata with $\Pi \subseteq \{1, 2\}$ are called *Büchi* automata. They can also be viewed as one-pair Rabin automata with an empty set of rejecting states $R = \emptyset$. Büchi automata are denoted $\mathcal{B} = (Q, \Sigma, I, T, F)$, where $F \subseteq T$ are called the final or accepting transitions. A run is accepting if it contains infinitely many accepting transitions. $\mathcal{B}$ is thus a rendering of the parity automaton, where $\mathsf{pri} : t \mapsto 2$ if $t \in F$ and $\mathsf{pri} : t \mapsto 1$ if $t \notin F$.

We assume without loss of generality that the set Π of priorities satisfies that $\min \Pi \in \{0, 1\}$. If this is not the case, we can simply change pri accordingly to $\mathsf{pri}' : t \mapsto \mathsf{pri}(t) - 2$ several times until this constraint is satisfied. We likewise assume that Π has no holes, that is, $\Pi = \{i \in \omega \mid \max \Pi \geq i \geq \min \Pi\}$. If there is a hole $h \notin \Pi$ with $\max \Pi > h > \min \Pi$, we can change pri to $\mathsf{pri}' : t \mapsto \mathsf{pri}(t)$ if $\mathsf{pri}(t) < h$ and $\mathsf{pri}' : t \mapsto \mathsf{pri}(t) - 2$ if $\mathsf{pri}(t) > h$. Obviously, these changes do not affect the acceptance of any run, and applying finitely many of these changes brings Π into this normal form.

The different priorities have a natural order $\succcurlyeq$, where $i \succ j$ if i is even and j is odd; i is even and $i > j$; or j is odd and $i < j$. For a non-empty set $\Pi' \subseteq \Pi$

of priorities, $\mathsf{opt}\Pi' = \{i \in \Pi' \mid \forall j \in \Pi'.\ i \succcurlyeq j\}$ denotes the optimal priority for acceptance.

Finally we define *generalised Büchi* automata that include the acceptance component F_i, where F_i is a family of accepting (or final) sets. A run ρ of a generalised Büchi automaton is accepting if it contains infinitely many transitions from all final sets ($\forall i \in [k]$ (where k is the cardinality of F_i), $inf(\overline{\rho}) \cap F_i \neq \emptyset$).

For all types of automata, a word α is accepted by an automaton $\mathcal{A}$ if, and only if, it has an accepting run, and its language $\mathcal{L}(\mathcal{A})$ is the set of words it accepts.

Deterministic automata. We call an automaton $(Q, \Sigma, I, T, \mathcal{F})$ *deterministic* if I is singleton and T contains at most one target node for all pairs of states and input letters, that is, if $(q, \alpha, r), (q, \alpha, s) \in T$ implies $r = s$. Deterministic automata are denoted $(Q, \Sigma, q_0, \delta, \mathcal{F})$, where q_0 is the only initial state and δ is the partial function with $\delta : (q, \alpha) \mapsto r \Leftrightarrow (q, \alpha, r) \in T$.

As deterministic automata can block, we also allow them to accept immediately. Technically, one can use a state $\top$ which every automaton has. From $\top$, all transitions go back to $\top$, and sequences that contain one (and thus almost only) $\top$ states are accepting. This state is not counted to the state-space Q. If we want to include it, we explicitly write $Q^\top$.

Theorem 2.1.1 (McNaughton's Theorem) *Let Σ be an alphabet and $L \subseteq \Sigma^\omega$ be a Büchi recognisable language. Then L is recognised by a deterministic Muller automaton (and therefore by a deterministic Rabin or parity automaton).* *[McN66]*

Determinisation. For ω-automata, determinisation is a consequence of McNaughton's theorem. Given a nondeterministic automaton $\mathcal{A}$ recognising a language $\mathcal{L}(\mathcal{A})$, determinisation is a procedure performed on $\mathcal{A}$ that returns a deterministic automaton $\mathcal{D}$ recognising the language $\mathcal{L}(\mathcal{A})$.

Complementation. Given a nondeterministic automaton $\mathcal{A}$ recognising a language $\mathcal{L}(\mathcal{A})$, complementation is a procedure performed on $\mathcal{A}$ that returns a an automaton $\mathcal{C}$ recognising the complementary language $\mathcal{L}(\overline{\mathcal{A}})$.

2.2 Current results for determinisation and complementation

Theorem 2.2.1 *Given a nondeterministic Büchi automaton with n states, we obtain a deterministic Rabin automaton with $2^{O(n\ log\ n)}$ states and n Rabin pairs.* *[Saf88]*

The above result establishes asymptotic upper bounds for both determinisation and complementation of Büchi automata.

Theorem 2.2.2 *Given a nondeterministic Büchi automaton with n states, we obtain a deterministic parity automaton with $O(n!^2)$ states and a $2n$ parity index.* *[Pit07]*

Theorem 2.2.3 *Given a nondeterministic Büchi automaton with n states, we obtain a deterministic Rabin automaton with approximately $(1.65\,n)^n$ states and $2^n - 1$ Rabin pairs.* *[Sch09b]*

Theorem 2.2.4 *The determinisation construction described in [Sch09b] that takes a nondeterministic Büchi automaton with n states and returns a eterministic Rabin automaton with approximately $(1.65\,n)^n$ states and $2^n - 1$ Rabin pairs is tight with respect to the number of states.* *[CZ09]*

Theorem 2.2.5 *The problem of complementing a Büchi automaton is in $\Omega((0.76\,n)^n)$.* *[Yan08]*

Theorem 2.2.6 *Given a nondeterministic Büchi automaton with n states, we construct a Büchi automaton with $o\big((0.76\,n)^n\big)$ states that accepts the complement language. This construction meets the lower bound in [Yan08] modulo a factor of $O(n^2)$.* *[Sch09a]*

Note that the parameter n stands for the number of states in the above statements. Note also that we are primarily concerned with state complexity in this thesis. Optimising for two parameters, i.e., states and transitions is another question that is interesting from a technical standpoint.

2.3 Problem statements

In this thesis, we will study the following problems:

1. We are given a nondeterministic parity automaton $\mathcal{P}$. Can we directly determinise $\mathcal{P}$ again to a parity automaton without first translating to nondeterministic Büchi automata?

2. Can we prove tight lower bounds for this construction?

3. Can we devise a tight complementation construction to complement $\mathcal{P}$?

4. Given a nondeterministic generalised Büchi automaton $\mathcal{GB}$, can we devise similarly tight determinisation and complementation constructions?

Chapter 3

Determinisation constructions

In this chapter, we study the problem of determinisation of parity automata and generalised Büchi automata. We first review the tight construction of Büchi automata from [Sch09b]. We then take a step towards the determinisation of parity automata by extending the Büchi determinisation algorithm to a slightly more succinct acceptance condition. We then generalise this algorithm to determinise parity automata. In a different extension, we extend the Büchi determinisation algorithm to determinise generalised Büchi automata. The results in Sections 3.2, 3.3, 3.4 and 3.5 are published in [SV14a] and [SV12].

3.1 Determinising Büchi automata

In this section, we outline a variant ([Sch09b]) of Safra's determinisation construction[Saf88] for Büchi automata. This construction was proved to be tight in [CZ09]. It is a variant of Safra's construction in the sense that there is a slight difference in the data structure used to describe the states of the deterministic automaton. The construction itself does not differ from Safra's except with respect to the changes the difference in the data structure brings to the construction.

3.1.1 History trees

Omega automata can have infinitely many possible runs on a given word. Safra trees introduced in [Saf88] are a succinct representation of the possible initial prefixes of these runs of a Büchi automaton $\mathcal{B}$ on an input word α. *History trees*, introduced in [Sch09b] are slightly modified Safra trees and achieve the same objectives. The main difference between Safra trees and history trees is the omission of explicit node names.

Definition 3.1.1 *A* history tree *is an ordered labelled tree $(\mathcal{T}, l)$, where $\mathcal{T}$ is a finite, prefix closed subset of finite sequences of natural numbers ω. Every element $v \in \mathcal{T}$ is called a* node*. Prefix closedness implies that, if a node $v = n_1 \dots n_j n_{j+1} \in \mathcal{T}$ is in $\mathcal{T}$, then $v' = n_1 \dots n_j$ is also in $\mathcal{T}$. We call v' the predecessor of v, denoted $\mathsf{pred}(v)$. The empty sequence $\epsilon \in \mathcal{T}$ is called the* root *of the ordered tree $\mathcal{T}$. Obviously, ϵ has no predecessor.*

We further require $\mathcal{T}$ to be order closed *with respect to siblings: if a node $v = n_1 \dots n_j$ is in $\mathcal{T}$, then $v' = n_1 \dots n_{j-1} i$ is also in $\mathcal{T}$ for all $i \in \omega$ with $i < n_j$. In this case, we call v' an* older sibling *of v (and v a* younger sibling *of v'). We denote the set of older siblings of v by $\mathsf{os}(v)$.*

A history tree is a tree labelled with sets of automata states. That is, $l : \mathcal{T} \to 2^Q \setminus \{\emptyset\}$ is a labelling function, which maps nodes of $\mathcal{T}$ to non-empty sets of automata states. For Büchi automata, the labelling is subject to the following criteria.

1. The label of each node is a subset of the label of its predecessor:
   $l(v) \subseteq l(\mathsf{pred}(v))$ holds for all $\varepsilon \neq v \in \mathcal{T}$.

2. The intersection of the labels of two siblings is disjoint:
   $\forall v, v' \in \mathcal{T}.\ v \neq v' \wedge \mathsf{pred}(v) = \mathsf{pred}(v') \Rightarrow l(v) \cap l(v') = \emptyset$.

3. The union of the labels of all siblings is *strictly* contained in the label of their predecessor:
   $\forall v \in \mathcal{T}\ \exists q \in l(v)\ \forall v' \in \mathcal{T}.\ v = \mathsf{pred}(v') \Rightarrow q \notin l(v')$.

We denote the number of history trees for Büchi automata having n states with $\mathsf{ht}(n)$. An estimation for the number of history trees produced when determinising a Büchi automaton is given in Subsection 3.5.1.

3.1.2 Construction

Let $\mathcal{B} = (Q, \Sigma, I, T, F)$ be a nondeterministic Büchi automaton with $|Q| = n$ states. We will construct an equivalent deterministic Rabin automaton $\mathcal{D} = (D, \Sigma, d_0, \delta, \{(A_i, R_i) \mid i \in J\})$ where,

- D is the set of history trees over Q,
- d_0 is the history tree $(\{\varepsilon\}, l : \varepsilon \mapsto I)$,
- J is the set of nodes that occur in some ordered tree of size n,
- for every tree $d \in D$ and letter $\sigma \in \Sigma$, the transition $d' = \Delta(d, \sigma)$ is the result of the transition mechanism described below.

Transition mechanism

We will illustrate in parallel the transition mechanism using the example fragment from Figure 3.1 on reading a letter σ. The starting history tree is shown in Figure 3.2.

We determine $\Delta{:}\big((\mathcal{T}, l), \sigma\big) \mapsto (\mathcal{T}', l')$ as follows:

1. *Update of node labels (subset constructions).*

   We update l to the function l_1 by assigning, for all $v \in \mathcal{T}$, $l_1 : v \mapsto \{q \in Q \mid \exists q' \in l(v).\ (q', \sigma, q) \in T\}$, i.e., to the σ successors of $l(v)$.

2. *Splitting of run threads / spawning new children.*

   In this step, we spawn new children for every node in the history tree. For all nodes, we spawn a child labelled with the set of states reached through accepting transitions.

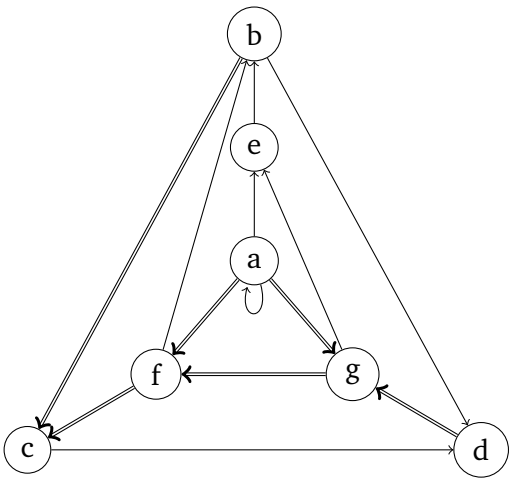

Figure 3.1: Transition graph for the example fragment of a Büchi automaton on reading the letter σ. Double arrows represent accepting transitions

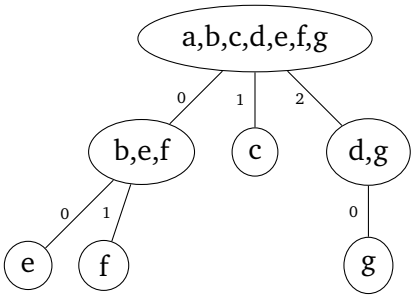

Figure 3.2: Starting example history tree

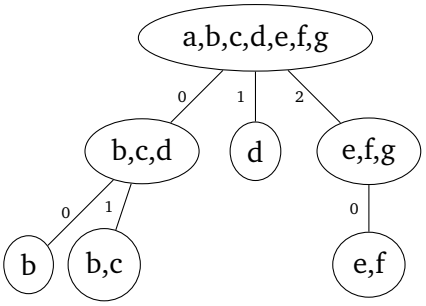

Figure 3.3: In Step 1, we update the labels of all nodes in a history tree by performing subset constructions on every node.

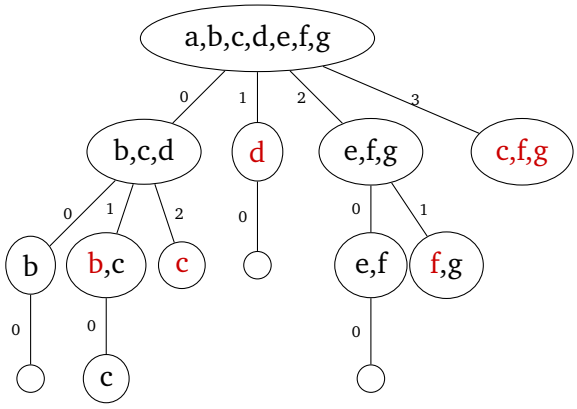

Figure 3.4: In Step 2, we spawn new children for every node in the history tree. A child node is labelled with it's parent's label minus the set of states that do not have an incoming accepting transition. If there is no state in the label of the parent that can be reached through an accepting transition, then the label of the child will be empty, eg. node 010. Although not part of this step, we show in red the states that are duplicated in younger siblings that are marked for removal in the next step

Thus, for every node $\varepsilon \in d$ with c children, we spawn a new child vc and expand l_1 to vc by assigning $l_1 : vc \mapsto \{q \in Q \mid \exists q' \in l(v).\ (q', \sigma, q) \in A\}$. We use $\mathcal{T}_n$ to denote the extended tree that includes the new children.

3. *Removing states from labels – horizontal pruning.*

   We obtain a function l_2 from l_1 by removing, for every node v with label $l(v) = Q'$ and all states $q \in Q'$, q from the labels of all younger siblings of v and all of their descendants.

4. *Identifying breakpoints – vertical pruning.*

   We denote with $\mathcal{T}_e \subseteq \mathcal{T}_n$ the set of all nodes v whose label $l_2(v)$ is now equal to the union of the labels of its children. We obtain $\mathcal{T}_v$ from $\mathcal{T}_n$ by removing all descendants of nodes in $\mathcal{T}_e$, and restrict the domain of l_2 accordingly.

   During the run of an automaton, a point is reached from which accepting states are visited again and again. We call such a point a *breakpoint*. Nodes in $\mathcal{T}_v \cap \mathcal{T}_e$ represent the breakpoints reached during the infinite run ρ and are called *accepting*, that is, the transition of $\mathcal{D}$ will be in A_v for exactly the $v \in \mathcal{T}_v \cap \mathcal{T}_e$.

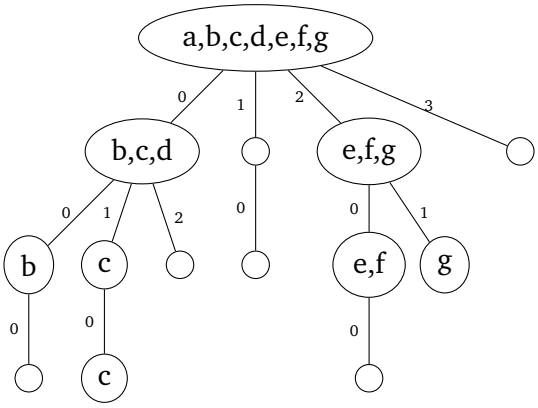

Figure 3.5: In Step 3, we remove from the label of younger siblings and their descendants, those states that are already present in an older sibling.

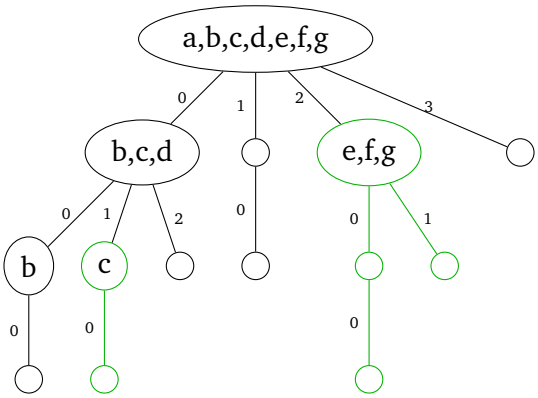

Figure 3.6: In Step 4, for every node whose label is equal to the union of its descendants, we remove the descendants of such a node and mark them (in green) as breakpoints.

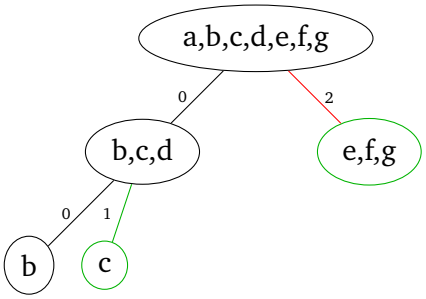

Figure 3.7: In Step 5, we remove the empty nodes. The resulting tree may not be ordered.

5. *Removing nodes with empty label.*

   We denote with $\mathcal{T}_r = \{v \in \mathcal{T}_v \mid l_2(v) \neq \emptyset\}$ the subtree of $\mathcal{T}_v$ that consists of the nodes with non-empty label and restrict the domain of l_2 accordingly.

6. *Reordering.*

   To repair the orderedness, we call $\|v\| = |\mathsf{os}(v) \cap \mathcal{T}_r|$ the number of (still existing) older siblings of v, and map $v = n_1 \dots n_j$ to $v' = \|n_1\| \, \|n_1 n_2\| \, \|n_1 n_2 n_3\| \dots \|v\|$, denoted $\mathsf{rename}(v)$.

   For $\mathcal{T}' = \mathsf{rename}(\mathcal{T}_r)$, we update a pair $(\mathcal{T}_r, l_2)$ from Step 5 to $d' = \left(\mathcal{T}', l'\right)$ with $l' : \mathsf{rename}(v) \mapsto l_2(v)$.

   We call a node $v \in \mathcal{T}' \cap \mathcal{T}$ *stable* if $v = \mathsf{rename}(v)$, and we call all nodes in J *rejecting* if they are not stable. That is, the transition will be in R_v exactly for those $v \in J$, such that v is not a stable node in $\mathcal{T} \cap \mathcal{T}'$.

3.1.3 Correctness

In order to establish the correctness of our determinisation construction, we need to prove that $L(\mathcal{B}) = L(\mathcal{D})$, that is, we need to ascertain that the ω-language accepted by the nondeterministic Büchi automaton is equivalent to the ω-language accepted by the deterministic Rabin automaton.

Lemma 3.1.2 *[$L(\mathcal{B}) \subseteq L(\mathcal{D})$] Given that there is an accepting run of $\mathcal{B}$ on an ω-word*

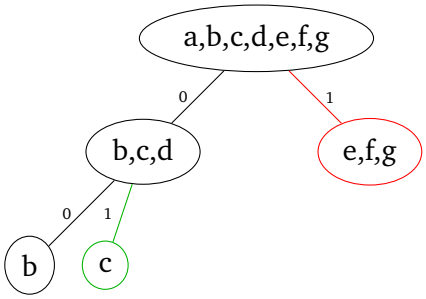

Figure 3.8: In Step 6, we reorder the tree and consider only those breakpoints that have not been reordered to characterise the current history tree in the Rabin acceptance condition. For example, this history tree is added to the accepting set with index 001, A_{001}. As the node 01 was reordered from 02, it is considered rejecting and indicates that this tree will be added to the rejecting set R_{01}.

α, there is a node $v \in J$ that is stable infinitely often from some point in the run and always accepting eventually during the run of $\mathcal{D}$ on α.

Notation. For a state q of $\mathcal{B}$ and a history tree $d = (\mathcal{T}, l)$, we call a node v the *host node of* q, denoted $\mathsf{host}(q, d)$, if $q \in l(v)$, but not in $l(vc)$ for any child vc of v.

Proof idea. The idea is that the state of each accepting run is eventually 'trapped' in the same node of the history tree, and this node must be accepting infinitely often. Let $d_0, d_1 \ldots$ be the run of $\mathcal{D}$ on α and $q_0, q_1, \ldots$ an accepting run of $\mathcal{B}$ on α. Then we can define a sequence $v_0, v_1, \ldots$ with $v_i = \mathsf{host}(q_i, d_i)$, and there must be a longest eventually stable prefix v in this sequence.

An inductive argument can then be exploited to show that, once this prefix v is henceforth stable, the index v cannot be rejecting. The assumption that there is a point in time where v is stable but never again accepting leads to a contradiction. Once the transition $(q_i, \alpha(i), q_{i+1})$ is accepting, $q_{i+1} \in l_{i+1}(vc)$ for some $c \in \omega$ and for $d_{i+1} = (\mathcal{T}_{i+1}, l_{i+1})$. As v is never again accepting *or* rejecting, we can show for all $j > i$ that, if $q_j \in l_j(vc_j)$, then $q_{j+1} \in l_{j+1}(vc_{j+1})$ for some $c_{j+1} \leq c_j$. This monotonicity leads to a contradiction with the assumption that v is the *longest* stable prefix.

Proof. We first fix a run that is accepting $\rho = q_0 q_1 \ldots$ of $\mathcal{B}$ on an input word α, and

let $\rho_{\mathcal{D}} = d_0 d_1 \ldots$ be the run of $\mathcal{D}$ on α. We then define the related sequence of host nodes $\vartheta = v_0 v_1 v_2 \ldots = \mathsf{host}(q_0, d_0)\mathsf{host}(q_1, d_1)\mathsf{host}(q_2, d_2)\ldots$.

We define $s = \liminf_{n\to\infty} |v_n|$ to be the shortest length occurring infinitely often of those host nodes. We follow this run and argue that the initial sequence of length s of the nodes in ϑ eventually stabilises. Let $i_0 < i_1 < i_2 < \ldots$ be an infinite ascending chain of indices such that

1. $(q_j, \alpha(j), q_{j+1}) \in T$ is a transition for any $j \geq i_0$,
2. the length $|v_j| \geq s$ of the j-th node is not smaller than s for all $j \geq i_0$, and
3. the length $|v_j| = s$ is equal to s for all indices $j \in \{i_0, i_1, i_2, \ldots\}$ in this chain.

(1), (2) and (3) together imply that when we follow the run of the deterministic automaton in positions $i_0, i_1, i_2, \ldots$, the host nodes $v_{i_0}, v_{i_1}, v_{i_2}, \ldots$ form a descending chain when the single nodes v_i are compared by lexicographic order. As the domain is finite, almost all elements of the descending chain are equal, say $v_i := \pi$. In particular, $\pi \in J$ is stable infinitely often from some point onwards.

We now assume for contradiction that this stable prefix π is accepting only finitely often. We choose an index i from the above defined ascending chain $i_0 < i_1 < i_2 < \ldots$ such that

1. π is stable for all $j \geq i$ and
2. π is not accepting for any $j \geq i$.

Note that π is the host of q_i for d_i, and $q_j \in l_j(\pi)$ holds for all $j \geq i$.

As ρ is accepting, there is a smallest index $j > i$ such that $(q_{j-1}, \alpha(j-1), q_j) \in A$. Now, as π is stable but not accepting for all $k \geq i$ (and hence for all $k \geq j$), q_k must henceforth be in the label of a child of π in d_k, which contradicts the assumption that infinitely many nodes in ϑ have length $s = |\pi|$.

Thus, π is eventually stable infinitely often and always accepting eventually. □

Lemma 3.1.3 *[$L(\mathcal{D}) \subseteq L(\mathcal{B})$] Given that there is a node $v \in J$, which is eventually stable infinitely often and always accepting eventually for an ω-word α, then there is an accepting run of $\mathcal{B}$ on α.*

Notation. For an ω-word α and $j \geq i$, we denote with $\alpha[i, j[$ the word $\alpha(i)\alpha(i+1)\alpha(i+2)\dots\alpha(j-1)$.

We denote with $Q_1 \to^{\alpha} Q_2$ for a finite word $\alpha = \alpha_1 \dots \alpha_{j-1}$ that there is, for all $q_j \in Q_2$ a sequence $q_1 \dots q_j$ with $q_1 \in Q_1$ and $(q_i, \alpha_i, q_{i+1}) \in T$ for all $1 \leq i < j$. If, for all $q_j \in Q_2$, there is such a sequence that contains a transition in F, we write $Q_1 \Rightarrow^{\alpha} Q_2$.

Proof. Let $\alpha \in L(\mathcal{D})$. Then there is a v that is eventually stable stable infinitely often from some point and always accepting eventually in the run $\rho_{\mathcal{D}}$ of $\mathcal{D}$ on α. We pick such a v.

The *proof idea* is the usual way of building a tree of initial sequences of runs. Let $i_0 < i_1 < i_2 < \dots$ be an infinite ascending chain of indices such that

- v is stable for all transitions $(d_{j-1}, \alpha(j-1), d_j)$ with $j \geq i_0$, and
- the chain $i_0 < i_1 < i_2 < \dots$ contains exactly those indices $i \geq i_0$ such that $(d_{i-1}, \alpha(i-1), d_i)$ is accepting.

Let $d_i = (\mathcal{T}_i, l_i)$ for all $i \in \omega$. By construction, we have

- $I \to^{\alpha[0,i_0[} l_{i_0}(v)$, and
- $l_{i_j}(v) \Rightarrow^{\alpha[i_j,i_{j+1}[} l_{i_{j+1}}(v)$.

Using this observation, we can build a tree of initial sequences of runs as follows: we build a tree of initial sequences of runs of $\mathcal{B}$ that contains a sequence $q_0 q_1 q_2 \dots q_{i_j}$ for any $j \in \omega$ if, and only if,

- $(q_i, \alpha(i), q_{i+1}) \in T$ is a transition of $\mathcal{B}$ for all $i < i_j$, and
- for all $k < j$ there is an $i \in [i_k, i_{k+1}[$ such that $(q_i, \alpha(i), q_{i+1}) \in F$ is an accepting transition.

By construction, this tree has the following properties:

- it is infinite,
- it is finitely branching,
- for all $j \in \omega$, a branch of length $> i_j$ contains at least j accepting transitions.

Exploiting König's lemma, the first two properties provide us with an infinite path, which is a run of $\mathcal{B}$ on α. The third property then implies that this run is accepting. α is therefore in the language of $\mathcal{B}$. □

The lemmata 3.1.2 and 3.1.3 together provide the following corollary.

Corollary 3.1.4 $L(\mathcal{R}) = L(\mathcal{D})$.

3.2 Towards the determinisation of parity automata

We will now start to extend the construction in the previous section to directly determinise parity automata.

Since the parity condition is in fact, a nested condition, it makes sense to nest history trees to handle the parity condition. However, the parity condition is a nesting of Büchi and coBüchi conditions, and for this reason, we will need to nest trees that handle the coBüchi condition in addition to the Büchi condition. The simplest acceptance condition that is a combination of Büchi and coBüchi is the Rabin condition with one pair. This also corresponds to a parity condition with 3 colours. We will describe how to modify history trees to handle this acceptance condition and also show a determinisation construction for this acceptance condition.

3.2.1 Root history trees

For one-pair Rabin automata, it suffices to adjust this data structure slightly.

Definition 3.2.1 (Root History Tree) *A* root history tree *is a tree labelled with sets of automata states. That is,* $l : \mathcal{T} \to 2^Q \setminus \{\emptyset\}$ *is a labelling function, which maps nodes of* $\mathcal{T}$ *to non-empty sets of automata states.*

For 1-pair Rabin automata, the labelling is subject to the following criteria.

1. The label of each node is a subset of the label of its predecessor:
   $l(v) \subseteq l(\mathsf{pred}(v))$ holds for all $\varepsilon \neq v \in \mathcal{T}$.

2. The intersection of the labels of two siblings is disjoint:
   $\forall v, v' \in \mathcal{T}.\ v \neq v' \wedge \mathsf{pred}(v) = \mathsf{pred}(v') \Rightarrow l(v) \cap l(v') = \emptyset$.

3. The union of the labels of all siblings is contained, but not necessarily strictly contained in the label of their predecessor:
   $\forall v \in \mathcal{T} \setminus \{\varepsilon\}\ \exists q \in l(v)\ \forall v' \in \mathcal{T}.\ v = \mathsf{pred}(v') \Rightarrow q \notin l(v')$.

4. The label of the root ε *equals* the union of its children's labels:
   $l(\varepsilon) = \bigcup \{l(v) \mid v \in \mathcal{T} \cap \omega\}$.

Thus, a *root history tree* (RHT) satisfies (1) and (2) from the definition of history trees, and a relaxed version of (3) that allows for non-strict containment of the label of the root.

Note that the 1-pair Rabin condition has an accepting and a rejecting component. Our modification allows for a transition step where only the youngest child of the root contains states which are reachable through rejecting transitions. All other children will contain successors reachable only through accepting and neutral transitions as in the Büchi construction.

3.2.2 Construction

Let $\mathcal{R}_1 = (Q, \Sigma, I, T, (A, R))$ be a nondeterministic one-pair Rabin automaton with $|Q| = n$ states. We first construct a language equivalent deterministic Rabin automaton $\mathcal{D}_1 = (D, \Sigma, d_0, \Delta, \{(A_i, R_i) \mid i \in J\})$ where,

- D is the set of RHTs over Q,
- d_0 is the history tree $(\{\varepsilon, 0\},\ l : \varepsilon \mapsto I,\ l : 0 \mapsto I)$,
- J is the set of nodes $\neq \varepsilon$ that occur in some RHT of size $n+1$ (due to the exemption for the root in Rule (4) in the definition of RHTs, an RHT can contain at most $n+1$ nodes), and
- for every tree $d \in D$ and letter $\sigma \in \Sigma$, the transition $d' = \Delta(d, \sigma)$ is the result of the sequence of the transition mechanism described below.

  The index set is the set of nodes, and, for each index, the accepting and rejecting sets refer to this node.

Transition mechanism for determinising one-pair Rabin Automata

We determine $\Delta\colon\big((\mathcal{T}, l), \sigma\big) \mapsto (\mathcal{T}', l')$ as follows:

1. *Update of node labels (subset constructions).*

   The root of a history tree d collects the momentarily reachable states $Q_r \subseteq Q$ of the automaton $\mathcal{R}_1$. In the first step of the construction, we update the label of the root to the set of reachable states upon reading a letter $\sigma \in \Sigma$, using the classical subset construction. We update the label of every other node of the RHT d to reflect the successors reachable through accepting or neutral transitions.

   For ε, we update l to the function l_1 by assigning $l_1 : \varepsilon \mapsto \{q' \in Q \mid \exists q \in l(\varepsilon).\ (q, \sigma, q') \in T\}$, and for all $\varepsilon \neq v \in \mathcal{T}$, we update l to the function l_1 by assigning $l_1 : v \mapsto \{q' \in Q \mid \exists q \in l(v).\ (q, \sigma, q') \in T \setminus R\}$.

2. *Splitting of run threads / spawning new children.*

   In this step, we spawn new children for every node in the RHT. For nodes other than the root ε, we spawn a child labelled with the set of states reachable through accepting transitions; for the root ε, we spawn a child labelled like the root.

Thus, for every node $\varepsilon \neq v \in d$ with c children, we spawn a new child vc and expand l_1 to vc by assigning $l_1 : vc \mapsto \{q \in Q \mid \exists q' \in l(v).\ (q', \sigma, q) \in A\}$. If ε has c children, we spawn a new child c of the root ε and expand l_1 to c by assigning $l_1 : c \mapsto l_1(\varepsilon)$. We use $\mathcal{T}_n$ to denote the extended tree that includes the new children.

3. *Removing states from labels – horizontal pruning.*

   We obtain a function l_2 from l_1 by removing, for every node v with label $l(v) = Q'$ and all states $q \in Q'$, q from the labels of all younger siblings of v and all of their descendants.

4. *Identifying breakpoints – vertical pruning.*

   We denote with $\mathcal{T}_e \subseteq \mathcal{T}_n$ the set of all nodes $v \neq \varepsilon$ whose label $l_2(v)$ is now equal to the union of the labels of its children. We obtain $\mathcal{T}_v$ from $\mathcal{T}_n$ by removing all descendants of nodes in $\mathcal{T}_e$, and restrict the domain of l_2 accordingly.

   Nodes in $\mathcal{T}_v \cap \mathcal{T}_e$ represent the breakpoints reached during the infinite run ρ and are called *accepting*, that is, the transition of $\mathcal{D}_1$ will be in A_v for exactly the $v \in \mathcal{T}_v \cap \mathcal{T}_e$. Note that the root cannot be accepting.

5. *Removing nodes with empty label.*

   We denote with $\mathcal{T}_r = \{v \in \mathcal{T}_v \mid l_2(v) \neq \emptyset\}$ the subtree of $\mathcal{T}_v$ that consists of the nodes with non-empty label and restrict the domain of l_2 accordingly.

6. *Reordering.*

   To repair the orderedness, we call $\|v\| = |\mathsf{os}(v) \cap \mathcal{T}_r|$ the number of (still existing) older siblings of v, and map $v = n_1 \dots n_j$ to $v' = \|n_1\|\ \|n_1 n_2\|\ \|n_1 n_2 n_3\| \dots \|v\|$, denoted $\mathsf{rename}(v)$.

   For $\mathcal{T}' = \mathsf{rename}(\mathcal{T}_r)$, we update a pair $(\mathcal{T}_r, l_2)$ from Step 5 to $d' = \left(\mathcal{T}', l'\right)$ with $l' : \mathsf{rename}(v) \mapsto l_2(v)$.

   We call a node $v \in \mathcal{T}' \cap \mathcal{T}$ *stable* if $v = \mathsf{rename}(v)$, and we call all nodes in J

rejecting if they are not stable. That is, the transition will be in R_v exactly for those $v \in J$, such that v is not a stable node in $\mathcal{T} \cap \mathcal{T}'$.

Note that this construction is a generalisation of the same construction for Büchi automata: if $R = \emptyset$, then the label of 0 is always the label of ε in this construction, and the node 1 is not part of any reachable RHT. (We would merely write 0 in front of every node of a history tree.)

Correctness

The correctness proof of this construction follows the same lines as the correctness proof of the Büchi construction.

Lemma 3.2.2 *[$L(\mathcal{R}_1) \subseteq L(\mathcal{D}_1)$] Given that there is an accepting run of $\mathcal{R}_1$ on an ω-word α, there is a node $v \in J$ that is eventually always stable and always eventually accepting in the run of $\mathcal{D}_1$ on α.*

Proof idea. The *proof idea* and notation are the same as for Büchi determinisation: the state of each accepting run is eventually 'trapped' in the same node of the RHT, and this node must be accepting infinitely often. Let $d_0, d_1 \ldots$ be the run of $\mathcal{D}_1$ on α and $q_0, q_1, \ldots$ an accepting run of $\mathcal{R}_1$ on α. Then we can define a sequence $v_0, v_1, \ldots$ with $v_i = \mathsf{host}(q_i, d_i)$, and there must be a longest eventually stable prefix v in this sequence.

An inductive argument can then be exploited to show that, once this prefix v is henceforth stable, the index v cannot be rejecting. The assumption that there is a point in time where v is stable but never again accepting can lead to a contradiction. Once the transition $(q_i, \alpha(i), q_{i+1})$ is accepting, $q_{i+1} \in l_{i+1}(vc)$ for some $c \in \omega$ and for $d_{i+1} = (\mathcal{T}_{i+1}, l_{i+1})$. As v is never again accepting *or* rejecting, we can show for all $j > i$ that, if $q_j \in l_j(vc_j)$, then $q_{j+1} \in l_{j+1}(vc_{j+1})$ for some $c_{j+1} \leq c_j$. This

monotonicity leads to a contradiction with the assumption that v is the *longest* stable prefix.

Proof. We first fix a run that is accepting $\rho = q_0 q_1 \ldots$ of $\mathcal{R}_1$ on an input word α, and let $\rho_{\mathcal{D}_1} = d_0 d_1 \ldots$ be the run of $\mathcal{D}_1$ on α. We then define the related sequence of host nodes $\vartheta = v_0 v_1 v_2 \ldots = \mathsf{host}(q_0, d_0)\mathsf{host}(q_1, d_1)\mathsf{host}(q_2, d_2)\ldots$.

We define $s = \liminf_{n\to\infty} |v_n|$ to be the shortest length occurring infinitely often of those host nodes. Note that the root cannot be the host node of any state, as it is always labelled by the union of the labels of its children.

We follow the run and argue that the initial sequence of length s of the nodes in ϑ eventually stabilises. Let $i_0 < i_1 < i_2 < \ldots$ be an infinite ascending chain of indices such that

1. $(q_j, \alpha(j), q_{j+1}) \in T \setminus R$ is a neutral or accepting transition for any $j \geq i_0$,
2. the length $|v_j| \geq s$ of the j-th node is not smaller than s for all $j \geq i_0$, and
3. the length $|v_j| = s$ is equal to s for all indices $j \in \{i_0, i_1, i_2, \ldots\}$ in this chain.

(1), (2) and (3) together imply that when we follow the run of the deterministic automaton in positions $i_0, i_1, i_2, \ldots$, the host nodes $v_{i_0}, v_{i_1}, v_{i_2}, \ldots$ form a descending chain when the single nodes v_i are compared by lexicographic order. As the domain is finite, almost all elements of the descending chain are equal, say $v_i := \pi$. In particular, $\pi \in J$ is stable infinitely often from some point onwards.

We now assume for contradiction that this stable prefix π is accepting only finitely many times. We choose an index i from the chain $i_0 < i_1 < i_2 < \ldots$ such that

1. π is stable for all $j \geq i$ and
2. π is not accepting for any $j \geq i$.

Note that π is the host of q_i for d_i, and $q_j \in l_j(\pi)$ holds for all $j \geq i$.

As ρ is accepting, there is a smallest index $j > i$ such that $(q_{j-1}, \alpha(j-1), q_j) \in A$. Now, as π is stable but not accepting for all $k \geq i$ (and hence for all $k \geq j$), q_k must

henceforth be in the label of a child of π in d_k, which contradicts the assumption that infinitely many nodes in ϑ have length $s = |\pi|$.

Thus, π is eventually stable infinitely often and always accepting eventually. □

Lemma 3.2.3 *[$L(\mathcal{D}_1) \subseteq L(\mathcal{R}_1)$] Given that there is a node $v \in J$, which is eventually stable infinitely often and always accepting eventually for an ω-word α, then there is an accepting run of $\mathcal{R}_1$ on α.*

Notation. The notation is the same as for Büchi determinisation except for the following minor adjustment. If, for all $q_j \in Q_2$, there is such a sequence that contains a transition in A but no transition in R, we write $Q_1 \Rightarrow^{\alpha} Q_2$.

Proof. Let $\alpha \in L(\mathcal{D}_1)$. Then there is a v that is eventually always stable and always eventually accepting in the run $\rho_{\mathcal{D}_1}$ of $\mathcal{D}_1$ on α. We pick such a v.

Let $1 < i_0 < i_1 < i_2 < \ldots$ be an infinite ascending chain of indices such that

- v is stable for all transitions $(d_{j-1}, \alpha(j-1), d_j)$ with $j \geq i_0$, and
- the chain $i_0 < i_1 < i_2 < \ldots$ contains exactly those indices $i \geq i_0$ such that $(d_{i-1}, \alpha(i-1), d_i)$ is accepting.

Let $d_i = (\mathcal{T}_i, l_i)$ for all $i \in \omega$. By construction, we have

- $I \rightarrow^{\alpha[0,i_0[} l_{i_0}(v)$, and
- $l_{i_j}(v) \Rightarrow^{\alpha[i_j,i_{j+1}[} l_{i_{j+1}}(v)$.

Using this observation, we can build a tree of initial sequences of runs as follows: we build a tree of initial sequences of runs of $\mathcal{R}_1$ that contains a sequence $q_0 q_1 q_2 \ldots q_{i_j}$ for any $j \in \omega$ iff

- $(q_i, \alpha(i), q_{i+1}) \in T$ is a transition of $\mathcal{R}_1$ for all $i < i_j$,
- $(q_i, \alpha(i), q_{i+1}) \notin R$ is not rejecting for all $i \geq i_0 - 1$, and
- for all $k < j$ there is an $i \in [i_k, i_{k+1}[$ such that $(q_i, \alpha(i), q_{i+1}) \in A$ is an accepting transition.

By construction, this tree has the following properties:

- it is infinite,
- it is finitely branching,
- no branch contains more than i_0 rejecting transitions, and,
- for all $j \in \omega$, a branch of length $> i_j$ contains at least j accepting transitions.

Exploiting König's lemma, the first two properties provide us with an infinite path, which is a run of $\mathcal{R}_1$ on α. The last two properties then imply that this run is accepting. α is therefore in the language of $\mathcal{R}_1$. □

The lemmata 3.2.2 and 3.2.3 together provide the following corollary.

Corollary 3.2.4 $L(\mathcal{R}_1) = L(\mathcal{D}_1)$.

3.3 Determinising parity automata

Having outlined a determinisation construction for one-pair Rabin automata using root history trees, we proceed to define *nested history trees* (NHTs), the data structure we use for determinising parity automata.

We assume that we have a parity automaton $\mathcal{P} = (Q, \Sigma, I, T, \mathsf{pri} : T \to \Pi)$. Note that the priority function pri can also be expressed as $\mathsf{pri} : T \to [\pi]$, where $[\pi] \in \Pi$ and we select $e = 2\lfloor 0.5\pi \rfloor$.

Definition 3.3.1 (Nested history trees) *A* nested history tree *is a triple $(\mathcal{T}, l, \lambda)$, where $\mathcal{T}$ is a finite, prefix closed subset of finite sequences of natural numbers and a special symbol $\mathfrak{s}$ (for* stepchild*), $\omega \cup \{\mathfrak{s}\}$. We refer to all other children vc, $c \in \omega$ of a node v as its* natural children*. We call $l(v)$ the label of the node $v \in \mathcal{T}$, and $\lambda(v)$ its* level*.*

A node $v \neq \varepsilon$ is called a Rabin root*, if, and only if it ends in $\mathfrak{s}$. The root ε is called a Rabin root if, and only if $\pi > e$. A node $v \in \mathcal{T}$ is called a* base node *if, and only if it is not a Rabin root and $\lambda(v) = 2$. The set of base nodes is denoted* $\mathsf{base}(\mathcal{T})$.

- *The label $l(v)$ of each node $v \neq \varepsilon$ is a subset of the label of its predecessor:*

  $l(v) \subseteq l(\mathsf{pred}(v))$ *holds for all* $\varepsilon \neq v \in \mathcal{T}$.

- *The intersection of the labels of two siblings is disjoint:*

  $\forall v, v' \in \mathcal{T}.\ v \neq v' \wedge \mathsf{pred}(v) = \mathsf{pred}(v') \Rightarrow l(v) \cap l(v') = \emptyset$.

- *For all* base nodes*, the union of the labels of all siblings is* strictly *contained in the label of their predecessor:*

  $\forall v \in \mathsf{base}(\mathcal{T})\ \exists q \in l(v)\ \forall v' \in \mathcal{T}.\ v = \mathsf{pred}(v') \Rightarrow q \notin l(v')$.

- *A node $v \in \mathcal{T}$ has a stepchild if, and only if v is neither a base-node, nor a Rabin root.*

- *The union of the labels of all siblings of a non-base node* equals *the union of its children's labels:*

  $\forall v \in \mathcal{T} \setminus \mathsf{base}(\mathcal{T})$, $l(v) = \{q \in l(v') \mid v' \in \mathcal{T}$ *and* $v = \mathsf{pred}(v')\}$ *holds.*

- *The level of the root is $\lambda(\varepsilon) = e$.*

- *The level of a stepchild is 2 smaller than the level of its parent:*

  for all $v\mathfrak{s} \in \mathcal{T}$, $\lambda(v\mathfrak{s}) = \lambda(v) - 2$ *holds.*

- *The level of all other children equals the level of its parent:*

  for all $i \in \omega$ *and* $vi \in \mathcal{T}$, $\lambda(vi) = \lambda(v)$ *holds.*

While the definition sounds rather involved, it is (for odd π) a nesting of RHTs. Indeed, for $\pi = 3$, we simply get the RHTs, and λ is the constant function with domain $\{2\}$. For odd $\pi > 3$, removing all nodes that contain an $\mathfrak{s}$ somewhere in the sequence again resemble RHTs, while the sub-trees rooted in a node $v\mathfrak{s}$ such that v does not contain a $\mathfrak{s}$ resemble NHTs whose root has level $\pi - 3$.

The transition mechanism from the previous section is adjusted accordingly. For each level a (note that levels are always even), we define three sets of transitions for the parity automaton $\mathcal{P}$: the rejecting transitions $R_a = \{t \in T \mid \mathsf{pri}(t) > a$ and $\mathsf{pri}(t)$

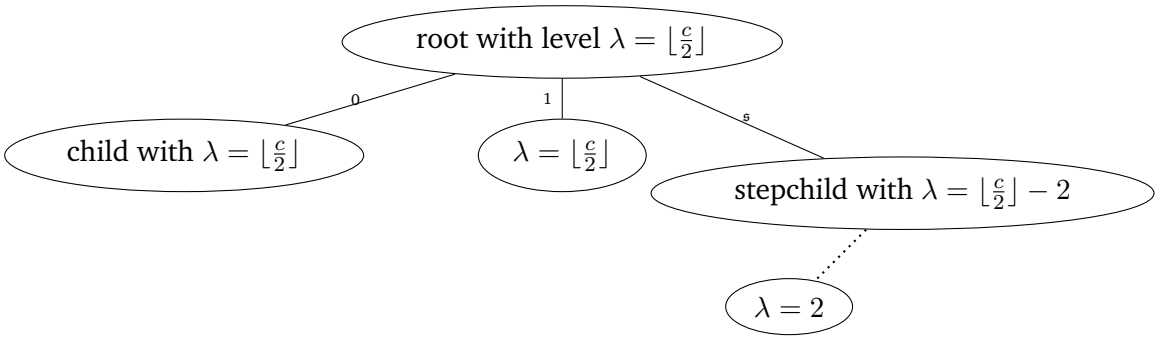

Figure 3.9: Abstract illustration of an NHT showing the differences in levels between children and stepchildren.

is odd}; the accepting transitions $A_a = \{t \in T \mid \mathsf{pri}(t) \geq a \text{ and } \mathsf{pri}(t) \text{ is even}\}$, and the (at least) neutral transitions, $N_a = T \setminus R_a$.

3.3.1 Construction.

Let $\mathcal{P} = \big(P, \Sigma, I, T, \{\mathsf{pri} : P \to [\pi]\big)$ be a nondeterministic parity automaton with $|P| = n$ states.

We construct a language equivalent deterministic Rabin automaton $\mathcal{R}^\pi = (D, \Sigma, d_0, \Delta, \{(A_i, R_i) \mid i \in J\})$ where,

- D is the set of NHTs over P (i.e., with $l(\varepsilon) \subseteq P$) whose root has level e, where $e = \pi$ if π is even, and $e = \pi - 1$ if π is odd,
- d_0 is the NHT we obtain by starting with $(\{\varepsilon\},\ l : \varepsilon \mapsto I,\ \lambda : \varepsilon \mapsto e)$, and performing Step 7 from the transition construction until an NHT is produced.
- J is the set of nodes v that occur in some NHT of level e over P, and
- for every tree $d \in D$ and letter $\sigma \in \Sigma$, the transition $d' = \Delta(d, \sigma)$ is the result of the sequence of transformations described below.

Transition mechanism for determinising parity automata.

Note that we do not define the update of λ, but use λ. This can be done because the level of the root always remains $\lambda(\varepsilon) = e$; the level $\lambda(v)$ of all nodes v is therefore

defined by the number of $\mathfrak{s}$ occurring in v. Likewise, the property of v being a base-node or a Rabin root is, for a given π, a property of v and independent of the labelling function.

Starting from an NHT $d = (\mathcal{T}, l, \lambda)$, we define the transitions $\Delta : (d, \sigma) \mapsto d'$ as follows:

1. *Update of node labels (subset constructions).*

   For the root, we continue to use $l_1(\varepsilon) = \{q' \in Q \mid \exists q \in l(\varepsilon).\ (q, \sigma, q') \in T\}$.

   For other nodes $v \in \mathcal{T}$ that are *not Rabin roots*, we use $l_1(v) = \{q' \in Q \mid \exists q \in l(v).\ (q, \sigma, q') \in N_{\lambda(v)}\}$.

   For the remaining Rabin roots $v\mathfrak{s} \in \mathcal{T}$, we use $l_1(v\mathfrak{s}) = \{q' \in Q \mid \exists q \in l(v\mathfrak{s}).\ (q, \sigma, q') \in N_{\lambda(v)}\}$. That is, we use the neutral transition of the higher level of the *parent* of the Rabin node.

2. *Splitting of run threads / spawning new children.*

   In this step, we spawn new children for every node in the NHT. For nodes $v \in \mathcal{T}$ that are not Rabin roots, we spawn a child labelled with the set of states reachable through accepting transitions. For a Rabin root $v \in \mathcal{T}$, we spawn a new child labelled like the root.

   Thus, for every node $v \in \mathcal{T}$ which is not a Rabin root and has c *natural* children, we spawn a new child vc and expand l_1 to vc by assigning $l_1 : vc \mapsto \{q \in Q \mid \exists q' \in l(v).\ (q', \sigma, q) \in A_{\lambda(v)}\}$. If a Rabin root v has c natural children, we spawn a new child vc of the Rabin root v and expand l_1 to vc by assigning $l_1 : vc \mapsto l_1(v)$. We use $\mathcal{T}_n$ to denote the extended tree that includes the new children.

3. *Removing states from labels – horizontal pruning.*

   We obtain a function l_2 from l_1 by removing, for every node v with label $l(v) = Q'$ and all states $q \in Q'$, q from the labels of all younger siblings of v and all of their descendants.

Stepchildren are always treated as the *youngest* sibling, irrespective of the order of creation.

4. *Identifying breakpoints – vertical pruning.*

   We denote with $\mathcal{T}_e \subseteq \mathcal{T}_n$ the set of all nodes $v \neq \varepsilon$ whose label $l_2(v)$ is now equal to the union of the labels of its *natural* children. We obtain $\mathcal{T}_v$ from $\mathcal{T}_n$ by removing all descendants of nodes in $\mathcal{T}_e$, and restrict the domain of l_2 accordingly.

   Nodes in $\mathcal{T}_v \cap \mathcal{T}_e$ represent the breakpoints reached during the infinite run ρ and are called *accepting*. That is, the transition of $\mathcal{R}^\pi$ will be in A_v for exactly the $v \in \mathcal{T}_v \cap \mathcal{T}_e$. Note that Rabin roots cannot be accepting as this would destroy the nestedness of the data structure.

5. *Removing nodes with empty label.*

   We denote with $\mathcal{T}_r = \{v \in \mathcal{T}_v \mid l_2(v) \neq \emptyset\}$ the subtree of $\mathcal{T}_v$ that consists of the nodes with non-empty label and restrict the domain of l_2 accordingly.

6. *Reordering.*

   To repair the orderedness, we call $\|v\| = |\mathsf{os}(v) \cap \mathcal{T}_r|$ the number of (still existing) older siblings of v, and map $v = n_1 \dots n_j$ to $v' = \|n_1\| \, \|n_1 n_2\| \, \|n_1 n_2 n_3\| \dots \|v\|$, denoted $\mathsf{rename}(v)$.

   For $\mathcal{T}_o = \mathsf{rename}(\mathcal{T}_r)$, we update a pair $(\mathcal{T}_r, l_2)$ from Step 5 to $d' = \left(\mathcal{T}_o, l'\right)$ with $l' : \mathsf{rename}(v) \mapsto l_2(v)$.

   We call a node $v \in \mathcal{T}_o \cap \mathcal{T}$ *stable* if $v = \mathsf{rename}(v)$, and we call all nodes in J *rejecting* if they are not stable. That is, the transition will be in R_v exactly for those $v \in J$, such that v is not a stable node in $\mathcal{T} \cap \mathcal{T}'$.

7. *Repairing nestedness.*

   We initialise $\mathcal{T}'$ to $\mathcal{T}_o$ and then add recursively for

- Rabin roots v without children a child $v0$ to $\mathcal{T}'$ and expand l' by assigning $l' : v0 \mapsto l'(v)$, and for
- nodes v, which are neither Rabin roots nor base-nodes, without children a child $v\mathfrak{s}$ to $\mathcal{T}'$ and expand l' by assigning $l' : v\mathfrak{s} \mapsto l'(v)$

until we have constructed an NHT $d' = (\mathcal{T}', l', \lambda')$.

3.3.2 Correctness

Lemma 3.3.2 $L(\mathcal{P}) \subseteq L(\mathcal{R}^\pi)$

Notation. For a state q of $\mathcal{P}$, an NHT $d = (\mathcal{T}, l, \lambda)$ and an even number $a \leq e$, we call a node v' the *a host node of q*, denoted $\mathsf{host}_a(q, d)$, if $q \in l(v')$, but not in $l(v'c)$ for any natural child $v'c$ of v', and $\lambda(v') = a$.

Let $\rho = q_0, q_1, q_2 \ldots$ be an accepting run of $\mathcal{P}$ with even level $a = \liminf_{i\to\infty} \mathsf{pri}\big(q_i, \alpha(i), q_{i+1}\big)$ on an ω-word α, let $d_0 d_1 d_2 \ldots$ be the run of $\mathcal{R}^\pi$ on α, and let $v_i = \mathsf{host}_a(q_i, d_i)$ for all $i \in \omega$.

Proof idea. The core idea of the proof is again that the state of each accepting run is eventually 'trapped' in a maximal initial sequence v of a-hosts, with the additional constraint that neither v nor any of its ancestors are infinitely often rejecting, and the transitions of the run of $\mathcal{P}$ are henceforth in N_a.

We show by contradiction that v is accepting infinitely often. For $\lambda(v) = a$, the proof is essentially the same as for one-Rabin determinisation. For $\lambda(v) > a$, the proof is altered by a case distinction, where one case assumes that, for some index $i > 0$ such that, for all $j \geq i$, v is a prefix of all v_j, $(q_{j-1}, \alpha(j-1), q_j) \in N_a$, and $(d_{j-1}, \alpha(j-1), d_j) \notin R_v \cup A_v$, q_i is in the label of a natural child vc of v. This provides the induction basis – in the one-pair Rabin case, the basis is provided through the accepting transition of the one-pair Rabin automaton, and we have no corresponding transition with even priority $\geq \lambda(v)$ – by definition. If no such i exists, we choose an i that satisfies the above requirements except that q_i is in the label of a natural child

vc of v. We can then infer that the label of $v\mathfrak{s}$ also henceforth contains q_i. As a Rabin root whose parent is not accepting or rejecting, $v\mathfrak{s}$ is not rejecting either.

Proof. We fix an accepting run $\rho = q_0q_1\ldots$ of $\mathcal{P}$ on an input word α, and use $a = \liminf_{i\to\infty}\big((q_i,\alpha(i),q_{i+1})\big)$ to refer to the dominating even priority of its transitions $\overline{\rho}$. We also let $\rho_{\mathcal{R}^\pi} = d_0d_1\ldots$ be the run of $\mathcal{R}^\pi$ on α. We then define the related sequences of host nodes $\vartheta = v_0v_1v_2\ldots = \mathsf{host}_a(q_0,d_0)\mathsf{host}_a(q_1,d_1)\mathsf{host}_a(q_2,d_2)\ldots$.

Note that Rabin roots cannot be the a^{th} host node of any state, as it is always labelled by the union of the labels of its children, and its children have the same level as the Rabin root itself.

- Let v' be the longest sequence, which is the initial sequence of almost all v_i, and
- Let v the longest initial sequence of v', such that, for no initial sequence v'' of v (including v itself), infinitely many transitions $(d_i,\alpha(i),d_{i+1})$ are in R_a.

We first observe that such a node v exists: as $q_i \in l_i(\varepsilon)$ for $d_i = (\mathcal{T}_i, l_i, \lambda_i)$ for all $i \in \omega$, ε satisfies all requirements except for maximality, such that a maximal element v exists. We now distinguish two cases.

Case 1: '$a = \lambda(v)$'

The first case is that the level of the node v equals the dominating priority of $\overline{\rho}$. For this case, we can argue as in the one-Rabin pair case: if the transition is infinitely often in the set A_v of $\mathcal{R}^\pi$, then $\rho_{\mathcal{R}^\pi}$ is accepting. Otherwise we choose a point $i \in \omega$ with the following properties:

- for all $j \geq i$, $(q_j,\alpha(j),q_{j+1}) \in N_a$,
- for all $j \geq i$ and all initial sequences w of v, $(d_j,\alpha(j),d_{j+1}) \notin R_w$,
- for all $j \geq i$, $(d_j,\alpha(j),d_{j+1}) \notin A_v$, and
- $\mathsf{pri}(q_i,\alpha(i),q_{i+1}) = a$.

We can now build a simple inductive argument with the following ingredients.

Induction basis:

There is a $k \in \omega$ such that $q_{i+1} \in l_{i+1}(vk)$.

The induction basis holds as the transition $(q_i, \alpha(i), q_{i+1})$ is in A_a and the node v is stable and non-accepting in $(d_i, \alpha(i), d_{i+1})$.

Induction step:

if, for some $k \in \omega$ and $j > i$, $q_j \in l_j(vk)$, then

- there is a $k' \leq k$ such that $q_{j+1} \in l_{j+1}(vk')$, and
- if $k = k'$ then $(d_j, \alpha(j), d_{j+1}) \notin R_{vk}$.

To see this, q_{j+1} is added to the '$l_1(vk)$' from Step 1 of the transition mechanism of the transition $(d_j, \alpha(j), d_{j+1})$. As v is stable but not accepting, the only two reasons for $q_{j+1} \notin l_{j+1}(vk)$ are that

- there is, for some $k'' < k$, a $q \in l_j(vk'')$ with and $(q, \alpha(j), q_{j+1}) \in N_{\lambda(v)}$ (note that $\lambda(v) = \lambda(vk) = \lambda(vk'') = a$), or
- for some $k'' < k$, the node vk'' is removed in Step 5 of the transition mechanism of the transition $(d_j, \alpha(j), d_{j+1})$.

In both cases (and their combination), we have $k' < k$. If neither is the case, then $(d_j, \alpha(j), d_{j+1}) \notin R_{vk}$ (as $\mathsf{rename}(vk) = vk$ holds in the transition mechanism).

The position $k \in \omega$ of the child vk with $q_j \in l(vk)$ can thus only be decreased finitely many times (and $\lambda(vk) = a$ for all $k \in \omega$). For some $k \in \omega$, vk is therefore a prefix of almost all v_i of ϑ. Once stable, it is henceforth no more rejecting. This contradicts the assumption that v is the longest such sequence.

Case 2: '$a > \lambda(v)$':

The second case is that the level of v is strictly greater than the dominating priority of $\overline{\rho}$. We argue along similar lines. If the transition is infinitely often in the set A_v of $\mathcal{R}^\pi$, then $\rho_{\mathcal{R}^\pi}$ is accepting. Otherwise we choose a point $i \in \omega$ with the following properties:

- for all $j \geq i$, $(q_j, \alpha(j), q_{j+1}) \in N_a$„
- for all $j \geq i$ and all initial sequences w of v, $(d_j, \alpha(j), d_{j+1}) \notin R_w$, and
- for all $j \geq i$, $(d_j, \alpha(j), d_{j+1}) \notin A_v$.

The difference to the previous argument is that the third prerequisite, '$\mathsf{pri}(q_i, \alpha(i), q_{i+1}) = a$', holds no longer. This was used for the induction basis. We replace this by a distinction of two sub-cases.

The first one is, that we do have an induction basis: we can choose the i such that there is a $k \in \omega$ such that $q_{i+1} \in l_{i+1}(vk)$. The rest of the argument can be copied for this case:

Induction step:

if, for some $k \in \omega$ and $j > i$, $q_j \in l_j(vk)$, then

- there is a $k' \leq k$ such that $q_{j+1} \in l_{j+1}(vk')$, and
- if $k = k'$ then $(d_j, \alpha(j), d_{j+1}) \notin R_{vk}$.

To see this, q_{j+1} is added to the '$l_1(vk)$' from Step 1 of the transition mechanism of the transition $(d_j, \alpha(j), d_{j+1})$. As v is stable but not accepting, the two only reason for $q_{j+1} \notin l_{j+1}(vk)$ are that

- there is, for some $k'' < k$, a $q \in l_j(vk'')$ with and $(q, \alpha(j), q_{j+1}) \in N_{\lambda(v)}$ (note that $\lambda(v) = \lambda(vk) = \lambda(vk'') = a$), or
- for some $k'' < k$, the node vk'' is removed in Step 5 of the transition mechanism of the transition $(d_j, \alpha(j), d_{j+1})$.

In both cases (and their combination), we have $k' < k$. If neither is the case, then $(d_j, \alpha(j), d_{j+1}) \notin R_{vk}$ (as $\mathsf{rename}(vk) = vk$ holds in the transition mechanism).

The position $k \in \omega$ of the child vk with $q_j \in l(vk)$ can thus only be decreased finitely many times (and $\lambda(vk) = a$ for all $k \in \omega$). For some $k \in \omega$, vk is therefore

a prefix of almost all v_i of ϑ. Once stable, it is henceforth no more rejecting. This contradicts the assumption that v is the longest such sequence.

The other sub-case is that no such i exists. We then choose i such that the two remaining conditions are met. As $\lambda(v) > a \geq 2$ holds, the union of the labels of the children of v must be the same as the label of v. Consequently, we have $q_j \in l(v\mathfrak{s})$ for all $j > i$. It remains to show that $v\mathfrak{s}$ is not rejecting infinitely many times. But the only ways a Rabin root can be rejecting are that either its parent node is accepting (the breakpoint of Step 4 from the transition mechanism) or not stable (Step 5 with Step 3, removing states from the label that occur in younger siblings) in a transition. But both are excluded in the definition of i.

Finally, we note that, for all v_i in ϑ, $\lambda(v_i) = a$ holds by construction. Consequently, $\lambda(v_i') \geq a$ holds for all initial sequences v_i' of v_i. In particular, we have $\lambda(v) \geq a$, such that the above case distinction is complete. □

Lemma 3.3.3 $L(\mathcal{R}^\pi) \subseteq L(\mathcal{P})$

The proof of this lemma is essentially the proof of Lemma 3.2.3 where, for the priority $a = \lambda(v)$ chosen to be the level of the accepting index v, A_a takes the role of the accepting set A from the one-pair Rabin automaton.

Notation. We denote with $Q_1 \Rightarrow_a^\alpha Q_2$ for a finite word $\alpha = \alpha_1 \dots \alpha_{j-1}$ that there is, for all $q_j \in Q_2$, a sequence $q_1 \dots q_j$ with

- $q_1 \in Q_1$,
- $(q_i, \alpha_i, q_{i+1}) \in N_a$ for all $1 \leq i < j$, and
- $(q_i, \alpha_i, q_{i+1}) \in A_a$ for some $1 \leq i < j$.

Proof. Let $\alpha \in L(\mathcal{R}^\pi)$. Then there is a v that is eventually always stable and always eventually accepting in the run $\rho_{\mathcal{R}^\pi}$ of $\mathcal{R}^\pi$ on α. We pick such a v.

Let $1 < i_0 < i_1 < i_2 < \dots$ be an infinite ascending chain of indices such that

- v is stable for all transitions $(d_{j-1}, \alpha(j-1), d_j)$ with $j \geq i_0$, and
- the chain $i_0 < i_1 < i_2 < \ldots$ contains exactly those indices $i \geq i_0$ such that $(d_{i-1}, \alpha(i-1), d_i)$ is accepting.

Let $d_i = (\mathcal{T}_i, l_i, \lambda_i)$ for all $i \in \omega$. By construction, we have

- $I \rightarrow^{\alpha[0,i_0[} l_{i_0}(v)$, and
- $l_{i_j}(v) \Rightarrow_a^{\alpha[i_j,i_{j+1}[} l_{i_{j+1}}(v)$.

Using this observation, we can build a tree of initial sequences of runs as follows: we build a tree of initial sequences of runs of $\mathcal{P}$ that contains a sequence $q_0 q_1 q_2 \ldots q_{i_j}$ for any $j \in \omega$ if, and only if

- $(q_i, \alpha(i), q_{i+1}) \in T$ is a transition of $\mathcal{P}$ for all $i < i_j$,
- $(q_i, \alpha(i), q_{i+1}) \in N_a$ is not rejecting for all $i \geq i_0 - 1$, and
- for all $k < j$ there is an $i \in [i_k, i_{k+1}[$ such that $(q_i, \alpha(i), q_{i+1}) \in A_a$ is an accepting transition.

By construction, this tree has the following properties:

- it is infinite,
- it is finitely branching,
- no branch contains more than i_0 transitions with odd priority $> a$, and,
- for all $j \in \omega$, a branch of length $> i_j$ contains at least j transitions with even priority $\geq a$.

Exploiting König's lemma, the first two properties provide us with an infinite path, which is a run of $\mathcal{P}$ on α. The last two properties then imply that this run is accepting. α is therefore in the language of $\mathcal{P}$. □

The lemmata 3.3.2 and 3.3.3 together provide the following corollary

Corollary 3.3.4 $L(\mathcal{P}) = L(\mathcal{R}^\pi)$.

3.4 Determinising generalised Büchi automata

Generalised Büchi automata are Büchi automata with multiple accepting sets. The determinisation construction described in this section is a generalisation of the determinisation construction for nondeterministic Büchi automata presented in 3.1, which in turn is a variation of Safra's [Saf88]. We first define the structure that captures the acceptance mechanism of our deterministic Rabin automaton. We adapt *history trees* defined previously to handle the determinisation of generalised Büchi automata.

3.4.1 Generalised history trees

For a generalised history tree $\mathcal{G} = (\mathcal{T}, l, h)$, $(\mathcal{T}, l)$ is the history tree introduced in 3.1. Generalised history trees are obtained by enriching history trees with the second labelling function, h, that is used to relate nodes with a particular accepting set.

Definition 3.4.1 (Generalised History Tree) *A generalised history tree $\mathcal{G}$ over Q for k accepting sets is a triple $\mathcal{G} = (\mathcal{T}, l, h)$ such that:*

- *$\mathcal{T}$ is an ordered tree,*
- *$l : \mathcal{T} \to 2^Q \setminus \{\emptyset\}$ is a labelling function such that*
  1. *$l(v) \subsetneq l(\mathsf{pred}(v))$ holds for all $\varepsilon \neq v \in \mathcal{T}$,*
  2. *the intersection of the labels of two siblings is disjoint*

     ($\forall v, v' \in \mathcal{T}.\ v \neq v' \wedge \mathsf{pred}(v) = \mathsf{pred}(v') \Rightarrow l(v) \cap l(v') = \emptyset$), and
  3. *the union of the labels of all siblings is* strictly *contained in the label of their predecessor*

     ($\forall v \in \mathcal{T}\ \exists q \in l(v)\ \forall v' \in \mathcal{T}.\ v = \mathsf{pred}(v') \Rightarrow q \notin l(v')$), and
- *$h : \mathcal{T} \to [k]$ is a function that labels every node with a natural number from $[k]$.*

  We call $F_{h(v)}$ the active *accepting set of v.*

3.4.2 Determinisation construction

Let $\mathcal{GB} = (Q, \Sigma, I, T, \{F_i \mid i \in [k]\})$ be a generalised Büchi automaton with $|Q| = n$ states and k accepting sets. We will construct an equivalent deterministic Rabin automaton $\mathcal{R}^{\mathcal{G}} = (D, \Sigma, d_0, \delta, \{(A_i, R_i) \mid i \in J\})$ where,

- D is the set of generalised history trees over Q.
- d_0 is the generalised history tree $(\{\varepsilon\}, l : \varepsilon \mapsto I, h : \varepsilon \mapsto 1)$.
- J is the set of nodes that occur in some ordered tree of size n.
- For every tree $d \in D$ and letter $\sigma \in \Sigma$, the transition $d' = \delta(d, \sigma)$ is the result of the following sequence of transformations:

Transition mechanism

We determine $\Delta{:}\big((\mathcal{T}, l, h), \sigma\big) \mapsto (\mathcal{T}', l', h')$ as follows:

1. *Raw update of l.*

   We update l to the function l_1 by assigning, for all $v \in \mathcal{T}$, $l_1 : v \mapsto \{q \in Q \mid \exists q' \in l(v).\ (q', \sigma, q) \in T\}$, i.e., to the σ successors of $l(v)$.

2. *Sprouting new children.*

   For every node $v \in d$ with c children, we sprout a new child vc. Let $\mathcal{T}_n$ be the tree of new children. Then we define, for all v in T_n, $l_1 : v \mapsto \big\{q \in Q \mid \exists q' \in l\big(\mathsf{pred}(v)\big).\ (q', \sigma, q) \in F_{h(\mathsf{pred}(v))}\big\}$, i.e., to the σ successors of the active accepting sets of their parents, and extend h to $T'_n = \mathcal{T} \cup \mathcal{T}_n$ by $h : v \mapsto 1$ for all $v \in \mathcal{T}_n$

3. *Stealing of labels.*

   We obtain a function l_2 from l_1 by removing, for every node v with label $l(v) = Q'$ and all states $q \in Q'$, q from the labels of all younger siblings of v and all of their descendants.

4. *Identifying breakpoints.*

   We denote with $\mathcal{T}_e \subseteq T'_n$ the set of all nodes v whose label $l_2(v)$ is now equal to the union of the labels of its children. We obtain $\mathcal{T}_v$ from $\mathcal{T}'_n$ by removing all descendants of nodes in $\mathcal{T}_e$, and restrict the domain of l_2 and h accordingly. (The resulting tree $\mathcal{T}'$ may no longer be ordered.)

   Nodes in $\mathcal{T}_v \cap \mathcal{T}_e$ are called *accepting*. We obtain h_1 from h by choosing $h : v \mapsto h(v) + 1$ for accepting nodes v with $h(k) \neq k$, $h_1 : v \mapsto 1$ for accepting nodes v with $h(k) = k$, and $h_1 : v \mapsto h(v)$ for all non-accepting nodes.

   The transition is in A_v if, and only if v is accepting.

5. *Removing nodes with empty label.*

   We denote with $\mathcal{T}_r = \{v \in \mathcal{T}_v \mid l_2(v) \neq \emptyset\}$ the subtree of $\mathcal{T}_v$ that consists of the nodes with non-empty label and restrict the domain of l_2 accordingly.

6. *Reordering.*

   To repair the orderedness, we call $\|v\| = |\mathsf{os}(v) \cap \mathcal{T}_r|$ the number of (still existing) older siblings of v, and map $v = n_1 \dots n_j$ to $v' = \|n_1\| \, \|n_1 n_2\| \, \|n_1 n_2 n_3\| \dots \|v\|$, denoted $\mathsf{rename}(v)$.

   For $\mathcal{T}' = \mathsf{rename}(\mathcal{T}_r)$, we update a triple $(\mathcal{T}_r, l_2, h_1)$ from the previous step to $d' = \big(\mathcal{T}', l', h'\big)$ with $l' : \mathsf{rename}(v) \mapsto l_2(v)$ and $h' : \mathsf{rename}(v) \mapsto h_1(v)$.

   We call a node $v \in \mathcal{T}' \cap \mathcal{T}$ *stable* if $v = \mathsf{rename}(v)$, and we call all nodes in J *rejecting* if they are not stable. The transition is in R_v if, and only if v is rejecting.

Correctness

The correctness and completeness proofs of this determinisation construction are similar to the proofs for Büchi determinisation.

Lemma 3.4.2 *[$L(\mathcal{GB}) \subseteq L(\mathcal{R}^{\mathcal{G}})$] Given that there is an accepting run of $\mathcal{GB}$, there is a node that is stable infinitely often from some point in the run and always accepting eventually during the run of $\mathcal{R}^{\mathcal{G}}$ on α*

Proof. We first fix a run that is accepting $\rho = q_0 q_1 \ldots$ of $\mathcal{GB}$ on an input word α, and let $\rho_{\mathcal{D}} = d_0 d_1 \ldots$ be the run of $\mathcal{R}^{\mathcal{G}}$ on α. We then define the related sequence of host nodes $\vartheta = v_0 v_1 v_2 \ldots = \mathsf{host}(q_0, d_0)\mathsf{host}(q_1, d_1)\mathsf{host}(q_2, d_2)\ldots$. We define l to be the shortest length occurring infinitely often $|v_i|$ of those host nodes.

The idea for proving correctness is that we need to show that the states in each accepting run are eventually trapped in the same node of the generalised history tree. This requires an inductive argument to show that there is some sequence induced during the run (in this case, the sequence v of host nodes), which once established that it is the longest stable prefix at some point during the run, then this prefix can be shown to never be rejecting by way of contradiction.

We follow the run ρ and claim that there is a longest eventually stable prefix of the nodes in ϑ of length l. Now, we extract an infinite ascending chain of indices $i_0 < i_1 < i_2 < \ldots$ such that

1. the length $|v_j| \geq l$ of the j-th node is not smaller than l for all $j \geq i_0$, and
2. the length $|v_j| \geq l$ of the j-th node is equal to $l = |v_i|$ for all indices $i \in \{i_0, i_1, i_2, \ldots\}$ in this chain.

(1) and (2) together imply that when we follow the run of the deterministic automaton in positions $i_0, i_1, i_2, \ldots$, the host nodes $v_{i_0}, v_{i_1}, v_{i_2}, \ldots$ form a descending chain when the single nodes v_i are compared by lexicographic order.

As the domain is finite, almost all elements of the descending chain are equal, say $v_i := \pi$. In particular, π is stable infinitely often from some point onwards.

Now that we have a claim that there is a longest eventually stable prefix, we now assume for contradiction, that this stable prefix π is accepting only finitely often and that it becomes rejecting from some point in time. We choose some index i from the

chain $i_0 < i_1 < i_2 < \ldots$ such that the longest stable prefix π is stable for all positions $j \geq i$. (Note that π is the host of q_i for d_i, and $q_j \in l_j(\pi)$ holds for all $j \geq i$.)

As ρ is accepting, there is a smallest index $j > i$ such that $(q_{j-1}, \alpha(j-1), q_j) \in F_{h_i(\pi)}$. Now, as π is not accepting, q_i must henceforth be in the label of a child of π, which contradicts the assumption that infinitely many nodes in ϑ have length $|\pi|$.

Thus, π is eventually stable infinitely often and always accepting eventually. □

Lemma 3.4.3 *[$L(\mathcal{R}^{\mathcal{G}}) \subseteq L(\mathcal{GB})$] Given that there is a node $v \in d$ (where d is a generalised history tree) which is eventually stable infinitely often and always accepting eventually for an ω-word α, then there is an accepting run of $\mathcal{GB}$ on α.*

Notation. The notation is the same as in Section 3.1.3 with the following minor modification. If, for all $q_j \in Q_2$, there is such a sequence that contains a transition in F_a, we write $Q_1 \Rightarrow_a^{\alpha} Q_2$.

Proof. Let $\alpha \in L(\mathcal{R}^{\mathcal{G}})$. Then there is a v that is eventually always stable and always eventually accepting in the run $\rho_{\mathcal{R}^{\mathcal{G}}}$ of $\mathcal{R}^{\mathcal{G}}$ on α. We pick such a v.

Let $i_0 < i_1 < i_2 < \ldots$ be an infinite ascending chain of indices such that

- v is stable for all transitions $(d_j, \alpha(j), d_{j+1})$ with $j \geq i_0$, and
- the chain $i_0 < i_1 < i_2 < \ldots$ contains exactly those indices $i \geq i_0$ such that $(d_{i-1}, \alpha(i-1), d_i)$ is accepting; this implies that h is updated exactly at these indices.

Let $d_i = (\mathcal{T}_i, l_i, h_i)$ for all $i \in \omega$. By construction, we have

- $I \rightarrow^{\alpha[0,i_0[} l_{i_0}(v)$, and
- $l_{i_j}(v) \Rightarrow_{h_{i_j}}^{\alpha[i_j,i_{j+1}[} l_{i_{j+1}}(v)$.

Exploiting König's lemma, this provides us with the existence of a run of $\mathcal{GB}$ on α that visits all accepting sets F_i of $\mathcal{GB}$ infinitely many times. [Note that the value of

h is circulating in the successive sequences of the run.] This run is accepting, and α therefore in the language of $\mathcal{GB}$. □

The lemmata 3.4.2 and 3.4.3 together provide the following corollary

Corollary 3.4.4 $L(\mathcal{GB}) = L(\mathcal{R}^{\mathcal{G}})$

3.5 Estimations

In this subsection, we provide estimations for the number of states of a deterministic automaton arising from our constructions. We provide estimations for the case of determinising 1-pair Rabin automata and generalised Büchi automata. It is easy to extend current results for estimating history trees[Sch09b] to these cases, especially for 1-pair Rabin automata.

3.5.1 Estimation of the number of history trees

Schewe's estimation of the number of history trees for a given Büchi automaton is the current best estimation for this problem. He estimates the number $\#\mathsf{ht}(n)$ of history trees for Büchi automata with n states to be in $o\big((1.65\,n)^n\big)$ [Sch09b].

3.5.2 Estimation of the number of Root History Trees

Let $\#\mathsf{ht}(n)$ and $\#\mathsf{rht}(n)$ be the number of history trees and RHTs, respectively, over sets with n states. First, $\#\mathsf{rht}(n) \geq \#\mathsf{ht}(n)$ holds, because the sub-tree rooted in 0 of an RHT is a history tree. Second, $\#\mathsf{ht}(n+1) \geq \#\mathsf{rht}(n)$, because adding the additional state to $l(\varepsilon)$ turns an RHT into a history tree. With an estimation similar to that of history trees [Sch09b], we get:

Theorem 3.5.1 *The number of root history tree $\#\mathsf{rht}(n)$, grows at a speed such that* $\inf\big\{c \mid \#\mathsf{rht}(n) \in O\big((cn)^n\big)\big\} = \inf\big\{c \mid \#\mathsf{ht}(n) \in O\big((cn)^n\big)\big\} \approx 1.65$.

In Subsection3.5.1, it was shown that $\#\mathsf{ht}(n)$ grows at a speed, such that $\inf\big\{c \mid \#\mathsf{ht}(n) \in O\big((cn)^n\big)\big\} \approx 1.65$. We argue that $\#\mathsf{rht}(n)$ does not only grow in the same

speed, it even holds that there is only a small constant factor between $\#\mathsf{ht}(n)$ and $\#\mathsf{rht}(n)$.

First, there is obviously a bijection between RHTs over Q and the subset of history trees over $Q \cup \{q_d\}$, where $q_d \notin Q$ is a fresh dummy state, and q_d is the only state that is hosted by the root. We estimate this size by the number of history trees, where q_d is hosted by the root ε of the history tree.

To keep the estimation simple, it is easy to see that the share of history trees with $< \frac{1}{3}n$ nodes diminishes to 0, as the number of trees with n nodes grows much faster than the number of trees with $< \frac{1}{3}n$ nodes and the number of functions from $[n]$ onto $[n]$, $n!$, grows much faster than the functions from $[n]$ to $[\frac{1}{3}n]$. So we can assume for our estimation that the tree has at least $\frac{1}{3}n$ nodes, such that the share of trees where q_d is in the root is at most $< \frac{3}{n}$.

The limit $\lim_{n\to\infty} \frac{\#\mathsf{ht}(n+1)}{n\#\mathsf{ht}(n)}$ converges to $(1 + \frac{c}{n})^n = e^c$ for $c \approx 1.65$. Thus, we get the following estimation:

$\lim_{n\to\infty} \frac{\#\mathsf{rht}(n)}{\#\mathsf{ht}(n)} \leq \lim_{n\to\infty} 3\frac{\#\mathsf{ht}(n+1)}{n\#\mathsf{ht}(n)} 3e^c < 3e^c$.

3.5.3 Estimation of the number of generalised history trees

The parameter $\mathsf{ght}_k(n)$ can be estimated in a similar way as the number of history trees for the determinisation of Büchi automata, as generalised history trees are, just like history trees, ordered trees with further functions on the set of nodes of the tree.

From [Sch09b], we have that $\mathsf{ht}(n) \in \sup_{x>0} O\big(m(x) \cdot 4^{\beta(x)}\big) \subset o\big((1.65\,n)^n\big)$, where n is the number of states of the nondeterministic Büchi automaton and m is the size of a history tree.

Using the functions from above, $\mathsf{ght}_k(n) \in \sup_{x>0} O\big(m(x) \cdot k^{\beta(x)} \cdot 4^{\beta(x)}\big)$, providing

$(1.65\,n)^n$, for $k = 1$, $(3.13\,n)^n$ for $k = 2$, and $(4.62n)^n$ for $k = 3$.[1]

This value converges against

$$(1.47kn)^n$$

[1]The values for the constants in the estimates are given from evaluating the equations for different values of k.

for large k.

Note that for all deterministic automata produced by our constructions, there may be up to $2^n - 1$ Rabin pairs.

For estimating nested history trees, simply multiplying the size of the data structure by the number of priorities gives an $(nk)^{O(nk)}$ upper bound, which is comparable to current trivial upper bounds derived from converting parity automata to Büchi automata and then determinising. Finding a tighter estimate for the number of nested history trees is a good question for future work.

3.6 Determinising to parity automata

In this section, we show that starting from our determinisation constructions described previously, we can determinise to deterministic parity automata. We also provide size estimations for the constructions. Most of these results are published in [SV14a] and [SV12].

Deterministic parity automata seem to be a nice target when determinising ω-automata given that algorithms that solve parity games (e.g, for acceptance games of alternating and emptiness games of nondeterministic parity tree automata) have a lower complexity when compared to solving Rabin games. For Büchi and Streett automata, determinisation to parity automata was first shown by Piterman in [Pit07]. For applications that involve co-determinisation, the parity condition also avoids the intermediate Streett condition.

Safra's determinisation construction (and younger variants) intuitively enforces a parity-like order on the nodes of history trees. Index Appearance Records are a variant of Latest Appearance Records first introduced in [McN66]. By storing the order in which nodes are introduced during the construction, we can capture the Index Appearance Records construction that is traditionally used to convert Rabin or Streett automata to parity automata. To achieve this, we augment the states of the deterministic automaton (GHTs, RHTs or NHTs) with a *later introduction record*

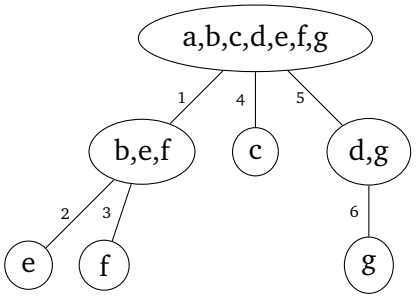

Figure 3.10: Example ordered tree with numbers denoting order of appearance. The corresponding LIR is [0 1 2 3 4 5 6]

(LIR), an abstraction of the order in which the non-Rabin nodes of the ordered trees are introduced. (As Rabin roots are but redundant information, they are omitted in this representation.)

3.6.1 From nondeterministic parity, 1-pair Rabin, and Büchi automata to deterministic parity automata

We show how this construction works for determinising parity automata to deterministic parity automata. As the parity condition subsumes the 1-pair Rabin condition, which in turn subsumes the Büchi condition, this construction works for the determinisation of automata that express all the above acceptance conditions.

For an ordered tree $\mathcal{T}$ with m nodes that are not Rabin roots, an LIR is a sequence $v_1, v_2, \ldots v_m$ that contains the nodes of $\mathcal{T}$ that are not Rabin root nodes, such that, each node appears after its ancestors and older siblings.

For example, consider the ordered tree given in Figure 3.10 where the numbers leading to the states describe its ordering with respect to time of generation of the node. The LIR corresponding to the ordered tree is $[0123456]$. 0 denotes the root.

For convenience in the lower bound proof, we represent a node $v \in \mathcal{T}$ of an NHT $d = (\mathcal{T}, l, \lambda)$ in the LIR by a triple (S_v, π_v, P_v) where $S_v = l(v)$, is the label of v, $\pi_v = \lambda(v)$ the level of v, and $P_v = \{q \in Q \mid v = \mathsf{host}_{\pi_v}(q, d)\}$ is the set of states π_v hosted by v. The v can be reconstructed by the order and level. We call the possible

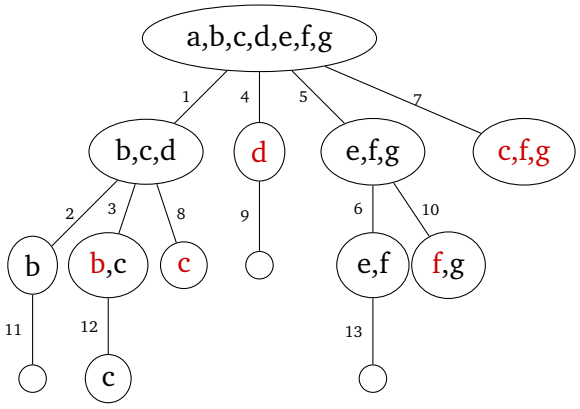

Figure 3.11: This figure shows an example transition on reading a letter σ just like the examples shown for the construction to Rabin automata. The corresponding LIR is now [0 1 2 3 4 5 6 7 8 9 10 11 12 13]. We will skip several steps like deleting duplicates horizontally and vertically and go straight to the next step where nodes are deleted for purposes of illustration of the LIR.

sequences of these triples *LIR-NHTs*. Obviously, each LIR-NHT defines an NHT, but not the other way round.

Definition 3.6.1 (LIR-NHT) *A finite sequence* $(S_1, \pi_1, P_1)(S_2, \pi_2, P_2)(S_3, \pi_3, P_3)\ldots(S_k, \pi_k, P_k)$ *of triples is a LIR-NHT if it satisfies the following requirements for all* $i \in [k]$.

1. $P_i \subseteq S_i$,
2. $\{P_i\} \cup \{S_j \mid j{>}i,\ \pi_i{=}\pi_j$, *and* $S_j{\cap}S_i{\neq}\emptyset\}$ *partitions* S_i.
3. $\{S_j \mid j > i,\ \pi_i = \pi_j + 2$, *and* $S_j \cap P_i \neq \emptyset\}$ *partition* P_i.
4. *If the highest priority of* $\mathcal{P}$ *is even, then* $\pi_i = e$ *implies* $S_i \subseteq S_1$. *(In this case, the lowest level construction is Büchi and the first triple always refers to the root.)*
5. *For* $\pi_i < e$, *there is a* $j < i$ *with* $S_i \subseteq P_j$.

To define the transitions of $\mathcal{D}$, we can work in two steps. First, we identify, for each position i of a state $N = (S_1, \pi_1, P_1)(S_2, \pi_2, P_2)(S_3, \pi_3, P_3)\ldots$ of $\mathcal{D}$, the node v_i of the NHT $d = (\mathcal{T}, l, \lambda)$ for the same input letter. We then perform the transition $\big(d, \sigma, (\mathcal{T}', l', \lambda')\big)$ on this Rabin automaton. We are then first interested in the set of

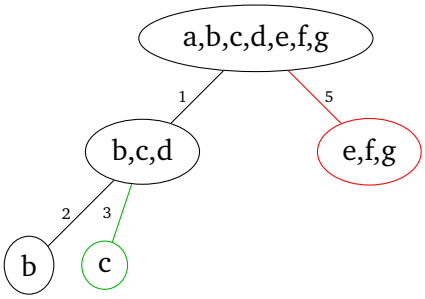

Figure 3.12: Here, some nodes have been deleted. Node 5 still remains and its position in the LIR is advanced. The LIR now looks like this: [0 1 2 3 $\overline{5}$ 4 6 7 8 9 10 11 12 13]. The transition leading to this state is now labelled with priority 6 as the smallest node position in the LIR that is either accepting or rejecting is position 3.

non-rejecting nodes from this transition and their indices. These indices are moved to the left, otherwise maintaining their order. All remaining vertices of $\mathcal{T}'$ are added at the right, maintaining orderedness.

The priority of the transition is determined by the smallest position i in the sequence, where the related node in the underlying tree is accepting or rejecting. It is therefore more convenient to use a min-parity condition, where the parity of $\liminf_{n\to\infty} \mathsf{pri}(\overline{\rho})$ determines acceptance of a run ρ. As this means smaller numbers have higher priority, pri is representing the opposite of a priority function, and we refer to the priority as the *co-priority* for clear distinction.

If the smallest node is rejecting, the transition has co-priority $2i-1$, if it is accepting (and not rejecting), then the transition has co-priority $2i$, and if no such node exists, then the transition has co-priority $ne+1$.

In case the nondeterministic automaton is Büchi or 1-pair Rabin, when we augment a LIR to a history tree/RHT, the LIR-NHT reduces to the order in which the nodes of the history tree/RHT are introduced.

Lemma 3.6.2 *Given a nondeterministic parity automaton $\mathcal{P}$ with $|P| = n$ states and maximal priority π, we can construct a language equivalent deterministic parity automaton $\mathcal{D}$ with $ne+1$ priorities for $e = 2\lfloor 0.5\pi \rfloor$, whose states are the LIR-NHTs described above.*

Proof. We use our determinisation technique from Section 3.3 to construct a deterministic parity automaton, whose states consist of the LIR-NHTs, i.e., the NHTs augmented with the Later Introduction Records, with the parity index on the transitions from the states of the automata.

First, we observe that $\mathcal{P}$ is language equivalent to the deterministic Rabin automaton $\mathcal{R}^\pi$ from the construction of Section 3.3 by Corollary 3.3.4.

Let α be a word in the language $L(\mathcal{R}^\pi)$ of the automaton $\mathcal{R}^\pi$. By definition of acceptance, we have an index v such that the node v is a node, which is eventually always stable and always eventually accepting in the transitions of the run of $\mathcal{R}^\pi$ on α. Note that v cannot be a Rabin root, as Rabin roots cannot be accepting.

Once stable, the position of this node in the LIR is non-increasing, and it decreases exactly when a node at a smaller position is deleted. This can obviously happen only finitely many times, and the position will thus eventually stabilise at some position p. Moreover, all positions $\leq p$ will then be henceforth stable.

Then, by our construction, it is easy to see that henceforth no transition can have a co-priority $< 2p$. At the same time, for each following transition where v is accepting in the deterministic Rabin automaton, the respective transition of the run of $\mathcal{P}$ has a priority $\leq 2p$. (At some node that is represented in a position $\leq 2p$, an accepting or rejecting event happens.) These two observations provide, together with the fact that these priorities $\leq 2p$ must occur infinitely many times by the deterministic Rabin automaton being accepting, that the dominating priority of the run is an even priority $\leq 2p$.

In the other direction, let $2i$ be the dominant priority for a run of our DPA $\mathcal{D}$ on a word α. This leads to a scenario where all positions $\leq i$ eventually maintain their positions in the LIR. The respective nodes they represent remain stable, but not accepting, from then on in the transitions of the run of $\mathcal{R}^\pi$ on α.

Observe that all older siblings (and ancestors, except for the omitted Rabin root) of a node v of an NHT are represented on a smaller position than v. The node

corresponding to the position i is always eventually accepting in the transitions of $\mathcal{R}^\pi$ on α, such that α is accepted by $\mathcal{R}^\pi$. □

Size estimation for determinisation of 1-pair Rabin and Büchi automata to parity automata

Lemma 3.6.3 *A deterministic parity automaton resulting from determinising a one-pair Rabin automaton $\mathcal{R}_1$ has $O(n!^2)$ states*

Proof. Let $|Q| = n$ be the number of states of our nondeterministic one-pair Rabin automaton. We explicitly represent (for the sake of evaluating the state-space) the tree structure of an RHT/LIR pair with m nodes by a sequence of $m-1$ integers $i_1, i_2 \dots i_m$ such that i_j points to the position $< j$ of the parent of the node v_j in the LIR $v_1, v_2, \dots v_m$. There are $(m-1)!$ such sequences. There is an obvious bijection between this representation of an LIR and its original definition. Thus, for an RHT/LIR pair with $n+1$ nodes, we can have up to $n!$ such RHT/LIR pairs just by virtue of the order of introduction of the nodes.

To more accurately evaluate the number of states, we have to consider the way RHTs are labelled. The root is always labelled with the complete set of reachable states.

We first consider the case where the root is labelled with all $|n|$ states of the nondeterministic one-pair Rabin automaton $\mathcal{R}_1$.

For history trees with m nodes over $n = |Q|$ states and n states labelling the root, Let $t(n, m)$ denote the number of such trees augmented with later introduction record.

First of all, $t(n, n+1) = (n! \cdot n!)$ holds : For such a tree, there can be up to $n!$ onto functions that resemble the labelling of states of the deterministic automaton and $n!$ RHTs augmented with LIRs.

For every $m \leq (n+1)$, the following is a coarse estimation from [Sch09b] providing $t(n, m-1) \leq \frac{1}{2} t(n, m)$. Hence, $\sum_{i=1}^{n} t(n, i) \leq 2(n! \cdot n!)$.

We next consider the case where the root is not labelled with all $|n|$ states of the nondeterministic one-pair Rabin automaton $\mathcal{R}_1$. Now, we let $t'(n,m)$ denote the number of root history trees with m nodes over n states augmented with LIRS. We now have $t'(n,n) = (n-1)!n!$ and, by a similar analysis to the case of t, we also have $t'(n,m-1) \leq \frac{1}{2}t'(n,m)$ for every $m \leq n$, and hence $\sum_{i=1}^{n-1} t'(n,i) \leq 2(n-1)!n!$.

Overall, the number of Root History Trees augmented with LIRs is $\sum_{i=1}^{n} t(n,i) + \sum_{i=1}^{n-1} t'(n,i) \leq O(n!^2)$. The number of states of the resulting deterministic parity automaton is $O(n!^2)$, which equates to a linear increase in size when compared with a deterministic parity automaton resulting from the determinisation of a language equivalent nondeterministic Büchi automaton instead of a nondeterministic one-pair Rabin automaton. □

Lemma 3.6.4 *A deterministic parity automaton resulting from determinising a Büchi automaton $\mathcal{B}$ has $O\big(n!(n-1!)\big)$ states.*

A similar estimation for the case of determinising Büchi automata to parity automata would result in $O\big((n-1)!n!\big)$ states, when the acceptance condition is placed on the transitions rather than the states. This is because over a set of n states, an RHT has a redundant root node when compared to a history tree over the same number of states.

3.6.2 From Generalised Büchi automata to deterministic parity automata

LIRs can similarly be augmented to generalised history trees to describe a determinisation construction from generalised Büchi automata to deterministic parity automata. A GHT/LIR pair is again a sequence $v_1, v_2, \ldots v_m$ that contains the nodes of $\mathcal{T}$ from the generalised history tree $\mathcal{G} = (\mathcal{T}, l, h)$. Every node appears after its parents and older siblings.

Theorem 3.6.5 *Given a nondeterministic generalised Büchi automaton $\mathcal{GB}$ with $|Q| =$*

n states and k accepting sets, one can construct a language equivalent deterministic parity automaton $\mathcal{D}$ with $O\big(n!(n-1!)k^n\big)$ states and $2n$ priorities.

Proof. First, we observe that the counting of GHT/LIR pairs is almost identical to the counting of history tree/LIR pairs. The distinguishing component in this case is the function $h : \mathcal{T} \to [k]$ which labels every node of a generalised history tree with a natural number. The inclusion of h multiplies the state-space by k^n. We thus have at most $O\big(n!(n-1!)k^n\big)$ states of the deterministic automaton through our construction.

We use our determinisation technique from Section 3.4 to construct a deterministic parity automaton, whose states are the GHT/LIR pairs and the parity index on the transitions from the states of the automaton. The priority of a transition from a state is determined by the smallest position i in the LIR that is either accepting or rejecting. If the smallest node is rejecting, the transition has co-priority $2i-1$, if it is accepting (and not rejecting), then the transition has co-priority $2i$. If no node is accepting or rejecting, the co-priority is $2n+1$.

We observe that $\mathcal{GB}$ is language equivalent to the deterministic Rabin automaton $\mathcal{R}^{\mathcal{G}}$ from the construction of Section 3.4 by Corollary 3.4.4. Let α be a word in the language $L(\mathcal{R}^{\mathcal{G}})$ of the automaton $\mathcal{R}^{\mathcal{G}}$. By definition of acceptance, we have an index v such that the node v is a node, which is eventually always stable and always eventually accepting in the transitions of the run of $\mathcal{R}^{\mathcal{G}}$ on α.

Once stable, the position of this node in the LIR is non-increasing, and it decreases exactly when a node at a smaller position is deleted. This can obviously happen only finitely many times, and the position will thus eventually stabilise at some position p. Moreover, all positions $\leq p$ will then be henceforth stable.

Then, by our construction, it is easy to see that henceforth no transition can have a co-priority $< 2p$. At the same time, for each following transition where v is accepting in the deterministic Rabin automaton, the respective transition of the run of $\mathcal{GB}$ is accepting infinitely often. These two observations provide, together with the fact that these priorities $\leq 2p$ must occur infinitely many times by the deterministic

Rabin automaton being accepting, that the dominating priority of the run is an even priority $\leq 2p$.

In the other direction, let $2i$ be the dominant priority for a run of our DPA $\mathcal{D}$ on a word α. This leads to a scenario where all positions $\leq i$ eventually maintain their positions in the LIR. The respective nodes they represent remain stable, but not accepting, from then on in the transitions of the run of $\mathcal{R}^{\mathcal{G}}$ on α.

Observe that all older siblings (and ancestors) of a node v of a GHT are represented on a smaller position than v. The node corresponding to the position i is always eventually accepting in the transitions of $\mathcal{R}^{\mathcal{G}}$ on α, such that α is accepted by $\mathcal{R}^{\mathcal{G}}$. □

3.7 Summary

Building on the current tight construction for the determinisation of Büchi automata, we have introduced determinisation constructions for the following cases:

1. Determinising from a nondeterministic parity(consequently, the special-case of 1-pair Rabin) automaton to a deterministic Rabin automaton.

2. Determinising from a nondeterministic generalised Büchi automaton to a deterministic Rabin automaton.

3. Determinising from a nondeterministic parity(consequently, the special-case of 1-pair Rabin) automaton to a deterministic parity automaton.

4. Determinising from a nondeterministic generalised Büchi automaton to a deterministic parity automaton.

In the next chapter, we will show lower bounds for these constructions.

Chapter 4

Lower bounds for determinisation

In this chapter, we provide lower bounds for our constructions. We use a family of automata called *full automata* that can simulate every automaton of the same size. We adapt a technique used to prove an exact lower bound for Büchi determinisation in [CZ09], to prove exact lower bounds for parity automata and generalised Büchi automata. We also provide a tight lower bound for determinisation *to* parity automata. The arguments for these techniques are more involved than for Büchi determinisation. Most of the results presented here were published in [SV14a] and [SV12].

4.1 Technical preliminaries

4.1.1 Full parity automata

Our lower bound proof builds on *full* automata (cf. [Yan08]), like the ones used in [CZ09] to establish a lower bound for the translation from nondeterministic Büchi to deterministic Rabin automata.

A parity automaton $\mathcal{P}_n^\pi = \big(Q, \Sigma_n^\pi, I, T, \mathsf{pri}\big)$ with n states is called *full* if its alphabet $\Sigma_n^\pi = Q \times Q^\top \to 2^{[\pi]}$ is the set of functions from $Q \times Q^\top$ to sets of priorities $[\pi]$, and

- $I = Q$,

- $T = \big\{ (q, \sigma, q') \mid q \in Q,\ q' \in Q^\top,\ \sigma(q, q') \neq \emptyset \big\}$,
- $\mathsf{pri} : (q, \sigma, q') \mapsto \mathsf{opt}\big(\sigma(q, q')\big)$ for all $q, q' \in Q$ with $\sigma(q, q') \neq \emptyset$, where opt returns the highest even number of a set, and the lowest odd number if the set contains no even numbers.

$(q, \sigma, \top)$ encodes immediate acceptance from state q. Every nondeterministic parity automaton with priorities $\leq \pi$ can be viewed as a language restriction (by alphabet restriction) of $\mathcal{P}_n^\pi$. $\mathcal{P}_n^\pi$ therefore recognises the hardest language recognisable by any parity automata with n states and maximal priority π.

4.1.2 Full Generalised Büchi automata

A generalised Büchi automaton $\mathcal{B}_n^k = (\Sigma, Q, I, T, \{F_i \mid i \in [k]\})$ is called *full* if

- $\Sigma_n^k = Q \times Q^\top \to 2^{[k+1]}$, $|Q| = n$, and $I = Q$,
- $T = \{(q, \sigma, q') \mid \exists i \in [k+1].\ (q, i, q') \in \sigma\}$, and
- $F_i = \{(q, \sigma, q') \mid (q, i, q') \in \sigma\}$.

4.1.3 Language games

A language game is an initialised two player game $G = (V, E, v_0, \mathcal{L})$, which is played between a verifier and a spoiler on a star-shaped directed labelled multi-graph (V, E) without self-loops. It has a finite set V of vertices, but a potentially infinite set of edges.

The centre of the star, which we refer to by $c \in V$, is the only vertex of the verifier, while all other vertices are owned by the spoiler. Besides the centre, the game has a second distinguished vertex, the initial vertex v_0, where a play of the game starts. The remaining vertices $W = V \setminus \{v_0, c\}$ are called the working vertices. Like v_0, they are owned by the spoiler.

The edges are labelled by finite words over an alphabet Σ. Edges leaving the centre vertex are labelled by the empty word ε, and there is exactly one edge leaving

via each working vertex, and no outgoing edge to the initial vertex. The set of these outgoing edges is thus $\{(c, \varepsilon, v) \mid v \in W\}$. The edges that lead back to the centre vertex are labelled with non-empty words.

The players play out a run of the game in the usual way by placing a pebble on the initial vertex v_0, letting the owner of that vertex select an outgoing edge, moving the pebble along it, and so forth. This way, an infinite sequence of edges is produced, and concatenating the finite words by which they are labelled provides an infinite word w over Σ. The verifier has the objective to construct a word in $\mathcal{L}$, while the spoiler has the antagonistic objective to construct a word in $\Sigma^\omega \setminus \mathcal{L}$.

Theorem 4.1.1 *[CZ09] If the verifier wins a language game for a language recognised by a deterministic Rabin automaton $\mathcal{R}$ with r states, then he wins the language game using a strategy with memory r.*

This is because he can simply run $\mathcal{R}$ as a witness automaton. Intuitively, the verifier would play on the product of $\mathcal{R}$ and G. This is a Rabin game, and if the verifier wins, then he wins memoryless [Kla94, Zie98]. Thus, the states of $\mathcal{R}$ can serve as the memory in G: the verifier will simply make the decision defined by the decision he made in the product game.

Corollary 4.1.2 *If the verifier wins a language game for a language recognised by a deterministic Rabin automaton $\mathcal{R}$ with $r < |W|$ states, then he wins the language game played on a reduced graph, where the set of his outgoing edges is reduced to r edges of his choice before playing the otherwise unchanged game.*

These are simply the at most r edges chosen by the verifier under the at most r different memory states.

4.1.4 Restricting the reachability set

Estimating the size of deterministic Rabin, Streett, or parity automata that recognise the same language as a nondeterministic automaton (either a parity automaton with

n states and maximal priority π, or a generalised Büchi automaton with n states and k accepting sets) reduces to estimating the size of the deterministic Rabin, Streett, or parity automata that recognises the language of $\mathcal{P}_n^\pi$ or $\mathcal{B}_n^k$. A useful property of this language is that we can focus on states with different sets of reachable states independently. We use $\mathsf{reach}(u)$ to denote the states reachable by a word $u \in \Sigma^*$ in $\mathcal{P}_n^\pi$, resp. $\mathcal{B}_n^k$ that is not immediately accepted by $\mathcal{P}_n^\pi$, resp. $\mathcal{B}_n^k$ (that is, such that $\top$ is not reachable on u). $\mathsf{reach}(u)$ can be defined inductively: $\mathsf{reach}(\varepsilon) = I$, and $\mathsf{reach}(va) = \left\{q' \in Q \mid \exists q \in \mathsf{reach}(v).\ (q, \sigma, q') \in T\right\}$ for all words $v \in \Sigma^*$ and all letters $a \in \Sigma$. This allows us to extend a useful observation from [CZ09]: states refer to reachability sets. Moreover, the states for each reachablity set can be considered independently.

Notation. In this section, we use $\rho(s, u)$ to refer to a finite part of the run of a deterministic automaton that starts in a state s upon reading a word $u \in {\Sigma_n^\pi}^+$ or a word $u \in {\Sigma_n^k}^+$, respectively. If this finite part of the run ends in a state s', we also write $\rho(s, u, s')$. In particular, $\rho(s, u, s')$ implies that s' is reached from s when reading u. For Rabin automata, an index i is accepting resp. rejecting for $\rho(s, u, s')$, if it is accepting resp. rejecting in some transition in this sequence of a run. For parity automata, the co-priority of $\rho(s, u, s')$ is the smallest co-priority that occurs in any transition in the respective sequence of a run.

Lemma 4.1.3 *Let $\mathcal{A}$ be a deterministic Rabin[1], Streett, or parity (or, more generally, Muller) automaton that recognises the language of $\mathcal{P}_n^\pi$ or its complement. Then $\rho(s_0, u, s)$ and $\rho(s_0, v, s)$ imply $\mathsf{reach}(u) = \mathsf{reach}(v)$ or $\mathsf{reach}(u) \ni \top \in \mathsf{reach}(v)$.*

Proof. Assume for contradiction that this is not the case. We select two words $u, v \in \Sigma^*$ with $\rho(s_0, u, s)$ and $\rho(s_0, v, s)$. Let $\sigma_\emptyset : (q, q') \mapsto \emptyset \forall (q, q') \in Q \times Q^\top$.

If $\mathsf{reach}(u) \ni \top \notin \mathsf{reach}(v)$, then $v{\sigma_\emptyset}^\omega$ is accepted, and $u{\sigma_\emptyset}^\omega$ is rejected by $\mathcal{P}_n^\pi$.

[1] for Rabin automata, this has been shown in [CZ09]

If $\top \notin \mathsf{reach}(u) \cup \mathsf{reach}(v)$ and $q \in \mathsf{reach}(u) \smallsetminus \mathsf{reach}(v)$, then we use $\sigma_q : (q, q) \mapsto \{2\}$ and $\sigma_q : (q', q'') \mapsto \emptyset$ if $(q', q'') \neq (q, q)$. Then $u{\sigma_q}^\omega$ is accepted, while $v{\sigma_q}^\omega$ is rejected by $\mathcal{P}_n^\pi$.

Doing the same with u and v reversed provides us with the required contradiction. □

A similar argument can be used for deterministic automata that recognise the language of $\mathcal{B}_n^k$. As a consequence, we can focus on sets of states with the same reachability set.

4.2 Lower bounds for parity determinisation

Now we establish the optimality of our determinisation to Rabin automata. In Sub-Section 3.3, we introduced a construction to determinise parity automata to deterministic Rabin automata. We show that the Rabin automata produced using Corollary 3.3.4 and recognising the same language as $\mathcal{P}_n^\pi$ cannot be minimized further by using the language of $\mathcal{P}_n^\pi$ as the target language.

We also show that our determinisation to parity automata is optimal up to a small constant factor. What is more, this lower bound extends to the more liberal Streett acceptance condition.

The technique we employ is similar to [CZ09], in that we use the states (for Rabin automata) or a large share of the states (for parity automata) of the resulting automaton as memory in a game, and argue that it can be won, but not with less memory. Just as in [CZ09], we use a game where this memory is a lower bound for the size of a deterministic Rabin automaton that recognises the language of a full nondeterministic automaton (see below). To estimate the size of a minimal Streett automaton, we use the complement language instead. Consequently, we get a dual result: a lower bound for a Rabin automaton that recognises the complement language. By duality, this bound is also the lower bound for a deterministic Streett

automaton that recognises the language of this full automaton. As the parity condition is a special Streett condition, this lower bound extends to parity automata.

To establish that the deterministic parity automaton $\mathcal{D}_n^\pi$ from Lemma 3.6.2 cannot be 50% larger than any deterministic Streett – and thus in particular than any deterministic parity – automaton that recognises the language of $\mathcal{D}_n^\pi$, we use the complement language of $\mathcal{P}_n^\pi$ as our target language.

We therefore get a bound on the smallest size of a Rabin automaton that recognises the complement of the language of $\mathcal{P}_n^\pi$, and hence for a Streett automaton that recognises $\mathcal{P}_n^\pi$. Having an upper bound for parity that matches this lower bound for the more general Streett condition, we can infer tightness of our determinisation construction for both classes of automata.

4.2.1 To deterministic Rabin automata.

To establish the lower bound, it is easier to use triples (S_v, π_v, P_v) for each node $v \in \mathcal{T}$ of an NHT $(\mathcal{T}, l, \lambda)$. By abuse of notation, we refer to the triple by $\mathcal{T}(v)$ (and thus to the state of the DRA by $\mathcal{T}$), to label by $T_S(v)$ and to $\{q \in Q \mid v = \mathsf{host}_{\lambda(v)}(q, \mathcal{T})\}$ by $\mathcal{T}_P(v)$.

To define the edges leaving a spoiler vertex $\mathcal{T}$, we refer to the finite part of a run of $\mathcal{R}_n^\pi$ that starts in $\mathcal{T}$ when reading a word $u \in {\Sigma_n^\pi}^+$ by $\rho(\mathcal{T}, u)$. If this finite part of the run ends in $\mathcal{T}'$, we also write $\rho(\mathcal{T}, u, \mathcal{T}')$. In particular, $\rho(\mathcal{T}, u, \mathcal{T}')$ implies that $\mathcal{T}'$ is reached from $\mathcal{T}$ when reading u. The accepting and rejecting nodes of $\rho(\mathcal{T}, u, \mathcal{T}')$ are the union of the accepting and rejecting nodes, respectively, of the individual transitions in this section of the run.

Definition 4.2.1 (Relevant change) *In a finite part $\rho(\mathcal{T}, u, \mathcal{T}')$, of our Rabin automaton $\mathcal{R}_n^\pi$ the* relevant change *is the minimal position v w.r.t. lexicographic order, where*

- *the node has been accepting or rejecting during the piece of the run, or*
- *where $\mathcal{T}(v) \neq \mathcal{T}'(v)$.*

We call the node v the relevant change, and we call it

- rejecting*, if $\rho(\mathcal{T}, u, \mathcal{T}')$ is rejecting at v,*
- accepting*, if $\rho(\mathcal{T}, u, \mathcal{T}')$ is accepting but not rejecting at v,*
- growing*, if it is not rejecting and $\mathcal{T}'_S(v) \supsetneq \mathcal{T}_S(v)$, and*
- shrinking*, if it is not rejecting, $\mathcal{T}'_S(v) = \mathcal{T}_S(v)$ and $\mathcal{T}'_P(v) \subsetneq \mathcal{T}_P(v)$.*

We use a set of language games, one for each subset $S \subseteq Q$ of the states Q of $\mathcal{P}_n^\pi$ with two or more states. The vertices of such a language game consist of the centre vertex, the initial vertex, and the working states W. These working states consist of the states of $\mathcal{R}_n^\pi$ with $\mathsf{reach}(\mathcal{T}) = S$. The target language is the language of all words *accepted* by $\mathcal{R}_n^\pi$, and we have the following edges:

- there is an edge (v_0, u, c) for all $u \in \Sigma_n^{\pi+}$ with $\rho(\mathcal{T}_0, u, \mathcal{T})$ and $\mathcal{T} \in W$,
- $(c, \varepsilon, \mathcal{T})$ for all $\mathcal{T} \in W$, and
- $(\mathcal{T}, u, c)$ if $\rho(\mathcal{T}, u, \mathcal{T}')$ is accepting, growing, or shrinking, and $\mathcal{T}' \in W$.

Lemma 4.2.2 *The verifier wins these language games.*

Proof. The verifier can simply use the strategy to monitor the state that the monitor DRA $\mathcal{R}_n^\pi$ from Corollary 3.3.4 would be in. He then has the winning strategy to play $(c, \varepsilon, \mathcal{T})$ when the automaton is in state $\mathcal{T}$.

To see that he wins the game with this strategy, we consider the run of $\mathcal{R}_n^\pi$ on the word defined by the play $(v_0, u_0, c)(c, \varepsilon, \mathcal{T}_1)(\mathcal{T}_1, u_1, c)(c, \varepsilon, \mathcal{T}_2)\ldots$, which refers to the word $u_0u_1u_2\ldots$.

The segments $\rho(\mathcal{T}_i, u_i, \mathcal{T}_{i+1})$ of the run have, for all $i \geq 1$, an accepting, growing, or shrinking relevant change.

Let us consider the relevant changes of these segments. There is a – with respect to lexicographic order – minimal one $v_{\min}$ that occurs infinitely often. Let us

choose a position i in the play such that no lexicographic smaller than v' is henceforth a relevant change. Then no node smaller than or equal to v (with respect to the lexicographic order) can henceforth be rejecting.

Clearly, if v is infinitely often accepting, then the verifier wins.

Let us assume for contradiction that there is a $j > i$ such that v is not accepting from position j onwards. Then, the set of states in $\mathcal{T}_l(v)$ must henceforth grow monotonously with l, and grow strictly every time v is growing. As this can only happen finitely many times, there is a $k > j$ such that v is henceforth neither accepting nor growing.

Then, the set of pure states in $\mathcal{T}_l(v)$ must henceforth shrink monotonously with l, and shrink strictly every time v is shrinking.

As this can only happen finitely often, this provides us with the required contradiction. □

To establish that a minimal Rabin automaton that recognises the language of $\mathcal{R}_n^\pi$ cannot be smaller than R_n^π, we show that the verifier needs all edges to win each of these games.

For this, we recall the structure of the strategy that the verifier applies: he would use $\mathcal{R}_n^\pi$ as a witness automaton, moving to the vertex that represents the state $\mathcal{T}$ that $\mathcal{R}_n^\pi$ would be in upon reading the finite word produced so far.

If one of his outgoing edges is removed, then there is one such state, say $\mathcal{T}$, he cannot respond to properly. Instead, he would have to go to a different state $\mathcal{T}'$. We show that, irrespective of the states $\mathcal{T}$ and $\mathcal{T}' \neq \mathcal{T}$ chosen, the spoiler can produce a word $u \in {\Sigma_n^\pi}^+$ such that $(\mathcal{T}, u, c)$ is an edge in G and $\rho(\mathcal{T}', u, \mathcal{T})$ is not accepting in any position.

If the spoiler has such an option, then *she* can use $\mathcal{R}_n^\pi$ as a witness automaton: whenever it is her move, she chooses an edge with the properties described above.

Lemma 4.2.3 *Let $\mathcal{T}$ and $\mathcal{T}'$ be two different states of $\mathcal{R}_n^\pi$ with $\mathsf{reach}(\mathcal{T}) = \mathsf{reach}(\mathcal{T}')$. Then there is a word $u \in {\Sigma_n^\pi}^+$ such that no node in $\rho(\mathcal{T}, u, \mathcal{T})$ is accepting and $\rho(\mathcal{T}', u, \mathcal{T})$ has an accepting, growing, or shrinking relevant change.*

Proof. We first identify the minimal position v in which $\mathcal{T}$ and $\mathcal{T}'$ are different, and the set P of all lexicographic smaller positions that are part of $\mathcal{T}$ (and thus of $\mathcal{T}'$).

We use a word $u = \sigma v$ that consists of an initial letter, in which all nodes but P are rejecting when staring in $\mathcal{T}$ and the nodes in P are neither accepting nor rejecting when staring in $\mathcal{T}$ or $\mathcal{T}'$. In the second phase, we re-build $\mathcal{T}$ without making any node in P accepting or rejecting.

Now let $\mathcal{T}(v) = (S_v, \pi_v, P_v)$, $\mathcal{T}'(v) = (S'_v, \pi_v, P'_v)$, and $\mathcal{T}(v) = (S_v, \pi_v, P_v)$ for all $v' \in P$. (Recall that the priority is defined by the position in the tree.)

We now distinguish four cases:

1. There is an $s \in S'_v \smallsetminus S_v$,
2. $S'_v = S_v$ and there is an $s \in P_v \smallsetminus P'_v$,
3. $S_v \supsetneq S'_v$, and
4. $S'_v = S_v$ and $P_v \subsetneq P'_v$.

We select the first letter of our word as follows.

1. If there is an $s \in S'_v \smallsetminus S_v$, then we can fix such an s and select the first letter of our word as follows:

   - We let $\pi_v \in \sigma(s, s')$ for all $s' \in S_v$.
   - For all $v' \in P$, all $s' \in P_{v'}$, and all $s'' \in S_{v'}$, we let $\pi_{v'} - 1 \in \sigma(s', s'')$.
   - If π is odd, we let $\pi \in \sigma(s', s'')$ for all $s', s'' \in \mathsf{reach}(\mathcal{T})$ (in order to maintain the set of reachable states).

   No further priority is included in any set $\sigma(s', s'')$ for $s', s'' \in Q$ and $s'' \in Q^\top$.

   This letter σ is chosen such that no node in P is accepting or rejecting in $\delta(\mathcal{T}, \sigma)$ or $\delta(\mathcal{T}', \sigma)$. While v is accepting in $\delta(\mathcal{T}', \sigma)$, it is rejecting in $\delta(\mathcal{T}, \sigma)$. All other positions are rejecting in these transitions.

2. If $S'_v = S_v$ and there is an $s \in P_v \smallsetminus P'_v$, then we can fix such an s and select the first letter of our word as follows:

- We let $\pi_v - 1 \in \sigma(s, s')$ for all $s' \in S_p$.
- For all $v' \in P$, all $s' \in P_{v'}$, and all $s'' \in S_{v'}$, we let $\pi_{v'} - 1 \in \sigma(s', s'')$.
- If π is odd, we let $\pi \in \sigma(s', s'')$ for all $s', s'' \in \mathsf{reach}(\mathcal{T})$ (in order to maintain the set of reachable states).

No further priority is included in any set $\sigma(s', s'')$ for $s', s'' \in Q$ and $s'' \in Q^\top$.

This letter σ is chosen such that no node in P is accepting or rejecting in $\delta(\mathcal{T}, \sigma)$ or $\delta(\mathcal{T}', \sigma)$. While v is accepting in $\delta(\mathcal{T}', \sigma)$, it is neither accepting nor rejecting in $\delta(\mathcal{T}, \sigma)$. All other positions are rejecting in these transitions.

3. If $S_v \supsetneq S'_v$, then we can fix an $s \in P_v$ and select the first letter of our word as follows:

   - We let $\pi_v - 1 \in \sigma(s, s')$ for all $s' \in S_v$.
   - For all $v' \in P$, all $s' \in P_{v'}$, and all $s'' \in S_{v'}$, we let $\pi_{v'} - 1 \in \sigma(s', s'')$.
   - If π is odd, we let $\pi \in \sigma(s', s'')$ for all $s', s'' \in \mathsf{reach}(\mathcal{T})$ (in order to maintain the set of reachable states).

   No further priority is included in any set $\sigma(s', s'')$ for $s', s'' \in Q$ and $s'' \in Q^\top$.

   This letter σ is chosen such that no node in P is accepting or rejecting in $\delta(\mathcal{T}, \sigma)$ or $\delta(\mathcal{T}', \sigma)$. While v is not rejecting (but may or may not be accepting) in $\delta(\mathcal{T}', \sigma)$, it is neither accepting nor rejecting in $\delta(\mathcal{T}, \sigma)$. All other positions are rejecting in these transitions.

4. If $S_v = S'_v$ and $P_v \subseteq P'_v$, then we can fix an $s \in P_v$ and select the first letter of our word as follows:

   - We let $\pi_v - 1 \in \sigma(s, s')$ for all $s' \in S_v$.
   - For all $v' \in P$, all $s' \in P_{v'}$, and all $s'' \in S_{v'}$, we let $\pi_{v'} - 1 \in \sigma(s', s'')$.
   - If π is odd, we let $\pi \in \sigma(s', s'')$ for all $s', s'' \in \mathsf{reach}(\mathcal{T})$ (in order to maintain the set of reachable states).

No further priority is included in any set $\sigma(s', s'')$ for $s', s'' \in Q$ and $s'' \in Q^\top$.

This letter σ is chosen such that neither v nor any node in P is accepting or rejecting in $\delta(\mathcal{T}, \sigma)$ or $\delta(\mathcal{T}', \sigma)$. All other positions are rejecting in these transitions.

Note that $\Delta(\mathcal{T}, \sigma) = \Delta(\mathcal{T}', \sigma)$ holds in all those cases. Starting with this letter, we can continue to build a word to reconstruct $\mathcal{T}$. Note that all we have to avoid during this construction is to make v or a node in P accepting.

The resulting fragments $\rho(\mathcal{T}', u, \mathcal{T})$ are accepting in cases (1) and (2), growing in case (3), and shrinking in case (4), such that $(\mathcal{T}', u, \mathcal{T})$ is a transition, while $\rho(\mathcal{T}, u, \mathcal{T})$ does not contain any accepting node. □

Corollary 4.2.4 *If any outgoing edge is removed from the verifier's centre vertex in any of these games, then the spoiler wins the language game.*

Together with Lemmata 4.1.3 and 4.2.2, Corollary 4.2.4 provides:

Theorem 4.2.5 *The full deterministic Rabin automaton $\mathcal{R}_n^\pi$ is the smallest deterministic Rabin automaton that recognises the language of the full parity automaton $\mathcal{P}_n^\pi$.*

4.2.2 To deterministic parity automata.

For our language game, we use a subset of the states of $\mathcal{D}_n^\pi$, which we call the *spiked states* and a fresh initial vertex as the spoiler vertices. $\mathcal{D}_n^\pi$ is obtained by determinising the full parity automaton $\mathcal{P}_n^\pi$ using the construction in SubSection 3.6.1. We call a state of $\mathcal{D}_n^\pi$ *spiked*, if its last position is a triple of the form $(\{q\}, 2, \{q\})$. This is a mild restriction and owed to the fourth case of the proof of Lemma 4.2.9. Most states are spiked.

Lemma 4.2.6 *$\mathcal{D}_n^\pi$ has more than twice as many spiked as unspiked states.*

Proof. Each unspiked state ends in a triple $(P, 2, P)$ with $|P| \geq 2$. We can simply replace it by $|P|$ pairs of triples, $(P, 2, P \setminus \{q\}), (\{q\}, 2, \{q\})$, for each $q \in P$. The

resulting state is spiked. Each non-spiked state produced at least two spiked states, each spiked state is produced by at most one state, and not every spiked state can be produced this way, e.g., states N with $|\mathsf{reach}(N)| = 1$ cannot. □

To define the edges leaving a spoiler vertex N, we refer to the finite part of a run of $\mathcal{D}_n^\pi$ that starts in N when reading a word $u \in {\Sigma_n^\pi}^+$ by $\rho(N, u)$. If this finite part of the run ends in N', we also write $\rho(N, u, N')$. In particular, $\rho(N, u, N')$ implies that N' is reached from N when reading u. The co-priority of $\rho(N, u, N')$ is the smallest co-priority that occurs in the respective sequence of a run.

Definition 4.2.7 (Relevant change) *In $\rho(N, u, N')$, the* relevant change *is the minimal position i, where*

- *the co-priority of $\rho(N, u, N')$ is $2i - 1$ or $2i$ i.e., position i was accepting or destroyed), or*
- *the i-th position of N and N' differ.*

For $N = \{(S_j, \pi_j, P_j)\}_{j \leq m}$ and $N' = \{(S'_j, \pi'_j, P'_j)\}_{j \leq m'}$, we call the relevant change i

- rejecting*, if the co-priority of $\rho(N, u, N')$ is $2i - 1$,*
- shrinking*, if $S'_i \subsetneq S_i$,*
- defying *if $S'_i = S_i$ and the co-priority of $\rho(N, u, N')$ is $2i + 1$, and*
- purifying*, if $S'_i{=}S_i$, $P'_i{\supsetneq}P_i$, and the co-priority is $>2i$.*

We use a language game, whose vertices consist of the centre vertex, the initial vertex, and and the working vertices W, which form a subset of the spiked states of $\mathcal{D}_n^\pi$. Following Lemma 4.1.3, we will, for each $S \subseteq Q$ with $|S| \geq 2$, use an individual game where W contains a spiked state N iff S is the set of states reachable in N ($\mathsf{reach}(N) = S$). The target language is the *complement* language of $\mathcal{D}_n^\pi$, and we have the following edges:

- there is an edge (v_0, u, c) for all $u \in {\Sigma_n^\pi}^+$ such that there is a spiked state $N \in W$ such that $\rho(N_0, u, N)$, where N_0 is the initial state of $\mathcal{D}_n^\pi$,

- (c, ε, N) for all spiked states N of $\mathcal{D}_n^\pi$ that are working states of the game, and
- (N, u, c) if $\rho(N, u, N')$ is rejecting, shrinking, defying, or purifying, and N' is spiked.

Lemma 4.2.8 *The verifier wins all of these language games.*

Proof. In the language game for each $S \subseteq Q$, the verifier can simply use the strategy to monitor the state that the monitor DPA $\mathcal{D}_n^\pi$ from Lemma 3.6.2 would be in. He then wins by playing (c, ε, N) when the automaton is in state N.

To see that he wins the game, we consider the run of $\mathcal{D}_n^\pi$ on the word defined by the play $(v_0, u_0, c)(c, \varepsilon, N_1)(N_1, u_1, c)(c, \varepsilon, N_2)\ldots$, which refers to the word $w = u_0u_1u_2\ldots$.

The run ρ of $\mathcal{D}_n^\pi$ on w can be decomposed into the finite segments $\rho(N_i, u_i, N_{i+1})$ for all $i \geq 0$. For all $i \geq 1$, their relevant changes are rejecting, shrinking, defying, or purifying.

Clearly, there is a minimal one $i_{\min}$ that occurs infinitely often. Consequently, no co-priority smaller than $2i_{\min} - 1$ can occur infinitely many times in ρ.

We can now distinguish four cases.

1. Assume that there are infinitely many rejecting relevant changes $i_{\min}$. Then the co-priority $2i_{\min} - 1$ occurs infinitely often in ρ, and the ω word w is rejected.
2. Assume that finitely many of the relevant changes with change priority $i_{\min}$ are rejecting, but infinitely many are shrinking. Then we can choose a position in the play where henceforth no relevant change with priority $< i_{\min}$, and no rejecting relevant change with priority $i_{\min}$ occurs. Consequently, the set of states at position $i_{\min}$ of N_i would henceforth shrink monotonously with growing i, and would infinitely often shrink strictly. But this is a contradiction.
3. Assume that finitely many of the relevant changes with change priority $i_{\min}$ are rejecting or shrinking, but infinitely many are defying. Then we can choose a position in the play where henceforth no relevant change with change priority

$< i_{\min}$, and no rejecting or shrinking relevant change with change priority $i_{\min}$ occurs. From this time onwards, no co-priority $\leq 2i_{\min}$ can occur on any segment of the run, while the co-priority $2i_{\min}+1$ occurs infinitely often.

4. Assume that finitely many of the relevant changes with change priority $i_{\min}$ are rejecting, shrinking, or defying. Then we can choose a position in the play where henceforth no relevant change j with $j < i_{\min}$, and no rejecting, shrinking, or defying relevant change with relevant change $i_{\min}$ occurs. Consequently, the set of pure states at position $i_{\min}$ of N_i would henceforth grow monotonously with growing i, and would infinitely often grow strictly. This is a contradiction, however.

□

To establish that the minimal size of a Rabin automaton that recognises the complement language of $\mathcal{D}_n^\pi$ cannot be significantly smaller than D_n^π, we will show that the verifier needs all edges to win this game.

For this, we recall the structure of the strategy that the verifier applies: he would use $\mathcal{D}_n^\pi$ as a witness automaton, moving to the vertex that represents the state $\mathcal{D}_n^\pi$ would be in upon reading the finite word produced so far. If one of his outgoing edges is removed, then there is one such state, say N, he cannot respond to properly. Instead, he would have to go to a different state N'.

We show that, irrespective of the state N that becomes unreachable and $N' \neq N$ chosen, the spoiler can produce a word $u \in {\Sigma_n^\pi}^+$ such that (N', u, c) is a transition in G and $\rho(N, u, N)$ has even co-priority.

If the spoiler has such an option, then *she* can use $\mathcal{D}_n^\pi$ as a witness automaton: initially, she selects an edge (v_0, u, c) such that $\rho(N_0, u, N)$ holds; henceforth she chooses, whenever she is in a vertex N, an edge (N', u, c) such that $\rho(N, u, N)$ holds, returning the run to N with dominating even co-priority. Thus, she can make sure that the constructed word is accepted.

Lemma 4.2.9 *Let N and N' be two different spiked states of $\mathcal{D}_n^\pi$ with $\mathsf{reach}(N) = \mathsf{reach}(N')$. Then there is a word $u \in \Sigma_n^{\pi+}$ such that the lowest co-priority occurring in $\rho(N, u, N)$ is even and $\rho(N', u, N)$ has a rejecting, shrinking, defying, or purifying relevant change.*

Proof. Let $N = \{(S_i, \pi_i, P_i)\}_{i \leq m}$ and $N' = \{(S'_i, \pi'_i, P'_i)\}_{i \leq m'}$. As $N \neq N'$, there is[2] a minimal $i_{\min} \leq \min\{m, m'\}$ such that $(S_{i_{\min}}, \pi_{i_{\min}}, P_{i_{\min}}) \neq (S'_{i_{\min}}, \pi'_{i_{\min}}, P'_{i_{\min}})$.

We will construct a word u such that

- $\rho(N, u, N)$ and $\rho(N', u, N)$ are fragments of runs,
- the minimal co-priority of $\rho(N, u, N)$, is even, and
- $i_{\min}$will be the relevant change in $\rho(N', u, N)$; it will be rejecting, shrinking, defying, or purifying.

We distinguish four cases.

1. Let us assume that there is an $s \in S_{i_{\min}} \smallsetminus S'_{i_{\min}}$. In this case, we choose such an s, and select the first letter $\sigma_{i_{\min}}$ of u such that
   - $\pi_{i_{\min}} \in \sigma_{i_{\min}}(s, s')$ for all $s' \in S_{i_{\min}}$,
   - $\pi_i - 1 \in \sigma_{i_{\min}}(s', s'')$ for all $i < i_{\min}$, $s' \in P_i$, and $s'' \in S_i$, and
   - if π is odd[3], $\pi \in \sigma_{i_{\min}}(s', s'')$ for all $s', s'' \in \mathsf{reach}(N)$.

   No further priority is included in any set $\sigma_{i_{\min}}(s', s'')$ for $s', s'' \in Q$ and $s'' \in Q^\top$.

   Starting with σ is the central step. The transition from N reading σ has co-priority $2i_{\min}$, the transition from N' reading σ has co-priority $2i_{\min} - 1$. Note that during the transition, all nodes in the history trees underlying N and N'

[2]Note that a spiked state N' cannot simply be longer than a spiked state N with $\mathsf{reach}(N) = \mathsf{reach}(N')$ (or vice versa): assuming that N is an initial sequence of N'. Then the rules (1), (2), (3), and (5) imply that S'_{m+1} must be disjoint with all S_i for $i \leq m$, which contradicts $\mathsf{reach}(N) = \mathsf{reach}(N')$.

[3]If the highest priority π of the defining NPA $\mathcal{P}_n^c$ is odd, then S_1 might be a strict subset of $\mathsf{reach}(N)$. This part is then an easy way to make sure that all states in $\mathsf{reach}(N)$ remain reachable. If π is even, then we have a tree on the lowest level, $S_1 = \mathsf{reach}(N)$.

that refer to a position $< i_{\min}$ are the same. They are also stable and non-accepting during this transition. However, while the node the position $i_{\min}$ of N refers to is accepting, the node position $i_{\min}$ of N' refers to is not stable. (Note that N and N' could refer to the same underlying tree.) The resulting state is the same for N and N'.

The next letters are to rebuild N. For all $i = i_{\min} + 1$ to m, we append a further letter σ_i to our partially constructed word u. We choose a state $s \in P_i$ and define σ_i as follows:

- $\pi_i \in \sigma_i(s, s')$ for all $s' \in S_i$,
- $\pi_j - 1 \in \sigma_i(s', s'')$ for all $j<i$, $s' \in P_j$ and $s'' \in S_j$, and
- if π is odd, $\pi \in \sigma_{i_{\min}}(s', s'')$ for all $s', s'' \in \mathsf{reach}(N)$.

No further priority is included in any set $\sigma_{i_{\min}}(s', s'')$ for $s', s'' \in Q$ and $s'' \in Q^{\top}$.

Clearly, reading i from a state that agrees with N on all positions $< i$ before reading σ_i, the resulting state will agree on all positions $\leq i$ with N after this transition.

The transition has a co-priority $\geq 2i - 1 > 2i_{\min}$. Thus, the word $u = \sigma_{i_{\min}} \sigma_{i_{\min}+1} \sigma_{i_{\min}+2} \dots \sigma_m$ has the required properties; in particular $\rho(N', u, N)$ has rejecting relevant change i.

In the remaining cases we have $S_{i_{\min}} \subseteq S'_{i_{\min}}$. Note that this implies $\pi_{i_{\min}} = \pi'_{i_{\min}}$.

2. The next case is $S_{i_{\min}} = S'_{i_{\min}}$ and there is a state $s \in P'_{i_{\min}} \setminus P_{i_{\min}}$. In this case, we fix such an s and start our word u with the letter $\sigma_{i_{\min}}$ that satisfies
   - $\pi_{i_{\min}} - 1 \in \sigma_{i_{\min}}(s, s')$ for all $s' \in S_{i_{\min}}$,
   - $\pi_i - 1 \in \sigma_{i_{\min}}(s', s'')$ for all $i < i_{\min}$, $s' \in P_i$, and $s'' \in S_i$, and,
   - if π is odd, $\pi \in \sigma_{i_{\min}}(s', s'')$ for all $s', s'' \in \mathsf{reach}(N)$.

No further priority is included in any set $\sigma_{i_{\min}}(s', s'')$ for $s', s'' \in Q$ and $s'' \in Q^\top$.

Starting u with this letter $\sigma_{i_{\min}}$ is again the central step. The transition from N has co-priority $2i_{\min}$, while the transition from N' has co-priority $2i_{\min} + 1$. We can now continue u in the same manner as above and use $u = \sigma_{i_{\min}}\sigma_{i_{\min}+1}\sigma_{i_{\min}+2}\dots\sigma_m$, and u will again satisfy the constraints; in particular $\rho(N', u, N)$ has a defying relevant change i.

3. In the next case, $S_{i_{\min}} \subsetneq S'_{i_{\min}}$, we fix an $s \in S_{i_{\min}}$ and start our word u with the letter $\sigma_{i_{\min}}$ that satisfies

   - $\pi_{i_{\min}} \in \sigma_{i_{\min}}(s, s')$ for all $s' \in S_{i_{\min}}$,
   - $\pi_i - 1 \in \sigma_{i_{\min}}(s', s'')$ for all $i < i_{\min}$, $s' \in P_i$, and $s'' \in S_i$, and,
   - if π is odd, $\pi \in \sigma_{i_{\min}}(s', s'')$ for all $s', s'' \in \mathsf{reach}(N)$.

   No further priority is included in any set $\sigma_{i_{\min}}(s', s'')$ for $s', s'' \in Q$ and $s'' \in Q^\top$.

   Starting u with this letter $\sigma_{i_{\min}}$ is again the central step. The transition from N or N' reading σ has co-priority $2i_{\min}$. We can again continue u in the same manner as above and use $u = \sigma_{i_{\min}}\sigma_{i_{\min}+1}\sigma_{i_{\min}+2}\dots\sigma_m$, and u will again satisfy the constraints; in particular $\rho(N', u, N)$ has shrinking relevant change i.

4. In the last case we have $S_{i_{\min}} = S'_{i_{\min}}$ and $P'_{i_{\min}} \subsetneq P_{i_{\min}}$. We first note that this implies $|P_{i_{\min}}| \geq 2$. The restriction to spiked states then provides $i_{\min} < m$. We can therefore refer to position $i_{\min} + 1$ of N.

   We choose an $s \in P_{i_{\min}+1}$ and start our word u with the letter $\sigma_{i_{\min}}$ that satisfies

   - $\pi_{i_{\min}+1} \in \sigma_{i_{\min}}(s, s')$ for all $s' \in S_{i_{\min}}$,
   - $\pi_i - 1 \in \sigma_{i_{\min}}(s', s'')$ for all $i \leq i_{\min}$, $s' \in P'_i$, and $s'' \in S_i$, and,
   - if π is odd, $\pi \in \sigma_{i_{\min}}(s', s'')$ for all $s', s'' \in \mathsf{reach}(N)$.

   No further priority is included in any set $\sigma_{i_{\min}}(s', s'')$ for $s', s'' \in Q$ and $s'' \in Q^\top$.

   Then the effect on N is obvious: the transition from N reading σ has co-priority $2i_{\min} + 2$. Starting from N', the same state is reached. The co-priority

is $2i_{\min} + 2$ if N' has a position $(S'_{i_{\min}+1}, \pi'_{i_{\min}+1}, P'_{i_{\min}+1})$ with $s \in S_{i_{\min}+1}$ and $\pi'_{i_{\min}+1} = \pi_{i_{\min}+1}$, and $2i_{\min} + 1$ otherwise. (Note that, for the case that $s \in P'_{i_{\min}}$, this would imply $\pi_{i_{\min}} = \pi_{i_{\min}+1} + 2$.)

We can again continue u in the same manner as above, although this results in the slightly shorter word $u = \sigma_{i_{\min}+1}\sigma_{i_{\min}+2}\sigma_{i_{\min}+3}\ldots\sigma_m$. The word u will again satisfy the constraints; in particular $\rho(N', u, N)$ has purifying relevant change i and $\rho(N, u, N)$ has co-priority $2i_{\min} + 2$.

□

Lemma 4.2.10 *If any outgoing edge is removed from the verifier's centre vertex in any of these games, then the spoiler wins the language game.*

Proof. If the spoiler has such an option, then *she* can use $\mathcal{D}_n^\pi$ as a witness automaton. Let N be the spiked state, to whom the outgoing edge from the centre is removed.

Initially, the spoiler plays a word u_0 with $\rho(N_0, u_0, N)$ by choosing the edge (v_0, u_0, c) from the initial vertex, such that N is reached from the initial state of $\mathcal{D}_n^\pi$. Henceforth she plays, whenever she is in a vertex N', the word u from the previous lemma by choosing the edge (N', u, c). This way, the two players construct a play $(v_0, u_0, c)(c, \varepsilon, N_1)(N_1, u_1, c)(c, \varepsilon, N_2)(N_2, u_2, c)(c, \varepsilon, N_3)$ $(N_3, u_3, c)(c, \varepsilon, N_4)(N_4, u_4, c)\ldots$.

For every $i \geq 1$, the segment $\rho(N_i, u_i, N_{i+1})$ of the run of $\mathcal{D}_n^\pi$ on the word $w = u_0u_1u_2u_3u_4\ldots$ has even minimal co-priority. Thus, the co-priority of the overall run is even. □

Together, the Lemmata 4.2.8, 4.2.10, and 4.1.3 provide:

Theorem 4.2.11 *Every Rabin automaton $\mathcal{R} = \big(S, \Sigma_n^\pi, s_0, \delta, R\big)$ that recognises the complement language of $\mathcal{P}_n^\pi$ must, for each non-empty subset*[4] *$S \subseteq Q$ of the states Q of $\mathcal{P}_n^\pi$, have at least as many states s with $\mathsf{reach}(s) = S$ as $\mathcal{D}_n^\pi$ has spiked states N with $\mathsf{reach}(N) = S$.*

[4]For each $q \in Q$ there is only a single state N of $\mathcal{D}_n^\pi$ with $\mathsf{reach}(N) = \{q\}$.

Corollary 4.2.12 *Every Rabin automaton that recognises the complement language of $\mathcal{P}_n^\pi$ must contain at least as many states as $\mathcal{D}_n^\pi$ has spiked states.*

By dualisation and the observation that parity automata are special Streett automata we simply get:

Corollary 4.2.13 *A deterministic Streett or parity automaton that recognises the language of $\mathcal{P}_n^\pi$ must have at least as many states as $\mathcal{D}_n^\pi$ has spiked states.*

The restriction to spiked states is minor – using the estimation of Lemma 4.2.6, we get:

Theorem 4.2.14 *$\mathcal{D}_n^\pi$ has less than 1.5 times as many states as the smallest deterministic Streett (or parity) automaton that recognises the language of $\mathcal{P}_n^\pi$.*

State sizes for two parameters are usually not crisp to represent. But for the simple base cases, Büchi and one pair Rabin automata, we get very nice results: it establishes that the known upper bound for determinising Büchi to parity automata [Sch09b] are tight and Piterman's algorithm for it [Pit07] is **optimal** modulo a factor of $3n$, where $2n$ stem from the fact that [Pit07] uses state based acceptance. With Lemmata 3.6.3 and 3.6.4 we get:

Corollary 4.2.15 *The determinisation of Büchi automata to Streett or parity automata leads to $\theta(n!(n-1)!)$ states, and the determinisation of one-pair Rabin automata to Streett or parity automata leads to $\theta(n!^2)$ states.*

4.3 Generalised Büchi lower bounds

In this section, we extend the same lower bound techniques to the determinisation of generalised Büchi automata.

4.3.1 To deterministic Rabin automata

We show that the function $\mathsf{ght}_k(n)$, that maps n to the number of generalised history trees for k accepting sets—and hence to the number of states of the resulting deterministic Rabin automaton obtained by our determinisation construction—is also a lower bound for the number of states needed for language equivalent deterministic Rabin automata.

On determinisation of the full generalised Büchi automaton $\mathcal{B}_n^k$, we get the deterministic Rabin automaton $\mathcal{R}_n^k$ whose states are the GHTs (triples $\mathcal{T}, l, h = d$ for all $d \in D$) that have previously been defined.

We use a set of language games, one for each subset $S \subseteq Q$ of the states Q of $\mathcal{B}_n^k$ with two or more states. The vertices of such a language game consist of the centre vertex, the initial vertex, and the working states W. These working states consist of the states of our deterministic Rabin automaton $\mathcal{R}_n^k$ with $\mathsf{reach}(d) = S$. To define the edges leaving a spoiler vertex d, we refer to the finite part of a run of $\mathcal{R}_n^k$ that starts in d when reading a word $u \in {\Sigma_n^k}^+$ by $\rho(d, u)$. If this finite part of the run ends in d', we also write $\rho(d, u, d')$. In particular, $\rho(\mathcal{T}, u, \mathcal{T}')$ implies that $\mathcal{T}'$ is reached from $\mathcal{T}$ when reading u. The accepting and rejecting nodes of $\rho(d, u, d')$ are the union of the accepting and rejecting nodes, respectively, of the individual transitions in this section of the run. The target language is the language of all words *accepted* by $\mathcal{R}_n^k$, and we have the following edges:

- there is an edge (v_0, u, c) for all $u \in {\Sigma_n^\pi}^+$ with $\rho(d_0, u, d)$ and $d \in W$,
- (c, ε, d) for all $d \in W$, and
- (d, u, c) if there is a node $v \in d$ that is accepting in $\rho(d, u, d')$, and $d' \in W$.

Lemma 4.3.1 *The verifier wins these language games.*

The verifier can simply use the strategy to monitor the state that the monitor DRA $\mathcal{R}_n^k$ from Corollary 3.4.4 would be in. He then has the winning strategy to play (c, ε, d) when the automaton is in state d.

To establish that the minimal Rabin automaton that recognises the language of $\mathcal{R}_n^k$ cannot be smaller than R_n^k, we show that the verifier needs all edges to win each of these games.

For this, we recall the structure of the strategy that the verifier applies: he would use $\mathcal{R}_n^k$ as a witness automaton, moving to the vertex that represents the state d that $\mathcal{R}_n^k$ would be in upon reading the finite word produced so far.

If one of his outgoing edges is removed, then there is one such state, say d, he cannot respond to properly. Instead, he would have to go to a different state d'. We show that, irrespective of the states d and $d' \neq d$ chosen, the spoiler can produce a word $u \in {\Sigma_n^k}^+$ such that (d, u, c) is an edge in G and $\rho(d', u, d)$ is not accepting in any position.

If the spoiler has such an option, then *she* can use $\mathcal{R}_n^k$ as a witness automaton: whenever it is her move, she chooses an edge with the properties described above.

Lemma 4.3.2 *Let d and d' be two different states of $\mathcal{R}_n^k$ with $\mathsf{reach}(d) = \mathsf{reach}(d')$. Then there is a word $u \in {\Sigma_n^k}^+$ such that no node in $\rho(d, u, d)$ is accepting.*

Proof. We distinguish two cases. First, we assume that $d = (\mathcal{T}, l, h)$ and $d' = (\mathcal{T}, l, h')$. This is the easy part: we can simply use a node $v \in \mathcal{T}$ such that $h(v) \neq h'(v)$, but this does not hold for any descendant of v. We then choose $Q' = \{q \in S \mid v = \mathsf{host}(q, d)\}$ to be the set of nodes hosted by v. (Note that these are the same for d and d'.)

In this case, we can simply play the one letter word $\alpha = \varepsilon \cup \{(q, h'(v), q) \mid q \in Q'\}$, which satisfies all the properties from above: clearly the transition is profitable for d' (as v is accepting in the respective transition $\rho(d', \alpha, d'')$ whereas $\rho(d, u, d)$ holds while none of the nodes of $\mathcal{T}$ is accepting.

Now we assume that $d = (\mathcal{T}, l, h)$ and $d' = (\mathcal{T}', l', h')$ with $(\mathcal{T}', l') \neq (\mathcal{T}, l)$. But for this case, we can almost use the same strategy for choosing a finite word u as for ordinary history trees [CZ09]. The extra challenge is that, when reconstructing d, it is not enough to spawn a new child, we also have to update h, which can be done

using a sequence of letters like the letter α from above after reconstructing a node v.

□

With the help of this lemma, we can have the following corollary.

Corollary 4.3.3 *If any outgoing edge is removed from the verifier's centre vertex in any of these games, then the spoiler wins the language game.*

A winning strategy for the spoiler is as follows. From the initial vertex, the spoiler plays a word u corresponding to $\rho(d_0, u, d)$, where d_0 is the initial state of the deterministic Rabin automaton. A good response from the verifier would be to move to d. But d has been removed and the verifier has to move to a different game state, say d' for some $d' \neq d$. The spoiler responds according to Lemma 4.3.2. Using this strategy, the spoiler ensures that, when the play gets infinite, there is no node that is always eventually accepting and the deterministic Rabin automaton $\mathcal{D}_\mathcal{G}$ does not accept this word and the spoiler wins.

Together with Lemmata 4.1.3 and 4.3.1, Corollary 4.3.3 provides:

Theorem 4.3.4 *The full deterministic Rabin automaton $\mathcal{R}_n^k$ is the smallest deterministic Rabin automaton that recognises the language of the full generalised Büchi automaton $\mathcal{B}_n^k$. $\mathcal{R}_n^k$ has size at least $\mathsf{ght}_k(n)$.*

4.3.2 To deterministic parity automata

We find a lower bound for determinising generalised Büchi to parity automata in a similar way to the case of parity determinisation. We use the complement language of $\mathcal{B}_n^k$ as our target language.

Lemma 4.3.5 *We can reuse the language games from Subsection 4.2.2 to prove a lower bound for the determinisation of generalised Büchi automata to parity automata.*

Proof. Let $\mathcal{T}_v$ be a state of a deterministic parity automaton produced by determinising a nondeterministic parity automaton. Let d_v be a state of the deterministic parity

automaton produced as a result of our construction for generalised Büchi automata. For a letter α and a state $\mathcal{T}_v$ (or d_v), $\rho(\mathcal{T}_v, \alpha)$ (or $\rho(d_v, \alpha)$) is a transition that produces the state $\mathcal{T}_{v+1}$ (or respectively d_{v+1}).

By our determinisation mechanism in Sections 3.6.1 and 3.6.2, there is a minimum position p in the underlying tree where there is a difference in the LIRs of the new state T_{v+1} (or respectively d_{v+1}) produced by the transition from the previous state T_v (or d_v). It is easy to to find a mapping between this position p in the NHT $\mathcal{T}_v$ and a position p in the GHT d_v. This describes an injection to the triples (S_k, π_k, P_k) from LIR-GHTs d_k. We can thus reuse the language games from Subsection 4.2.2 to prove a lower bound for the determinisation of generalised Büchi automata to parity automata. The target language is the complement language of $\mathcal{B}_n^k$ over Σ_n^k analogous to the complement language of $\mathcal{D}_n^\pi$ in Section 4.2.2. □

The following lemmata are stated without proof in order to avoid repeating the same arguments from Subsection 4.2.2.

Lemma 4.3.6 *The verifier wins all of these language games.*

Lemma 4.3.7 *Let N and N' be two different spiked states of $\mathcal{B}_n^k$ with $\mathsf{reach}(N) = \mathsf{reach}(N')$. Then there is a word $u \in {\Sigma_n^k}^+$ such that the lowest co-priority occurring in $\rho(N, u, N)$ is even and $\rho(N', u, N)$ has a rejecting, shrinking, defying, or purifying relevant change.*

Lemma 4.3.8 *If any outgoing edge is removed from the verifier's centre vertex in any of these games, then the spoiler wins the language game.*

Together, the Lemmata 4.3.6, 4.3.8, and 4.1.3 provide:

Theorem 4.3.9 *Every Rabin automaton $\mathcal{R} = \left(S, \Sigma_n^k, s_0, \delta, R\right)$ that recognises the complement language of $\mathcal{B}_n^k$ must, for each non-empty subset[5] $S \subseteq Q$ of the states Q of $\mathcal{B}_n^k$, have at least as many states s with $\mathsf{reach}(s) = S$ as $\mathcal{D}_n^k$ has spiked states N with $\mathsf{reach}(N) = S$.*

[5]For each $q \in Q$ there is only a single state N of $\mathcal{B}_n^k$ with $\mathsf{reach}(N) = \{q\}$.

Corollary 4.3.10 *Every Rabin automaton that recognises the complement language of $\mathcal{B}_n^k$ must contain at least as many states as $\mathcal{D}_n^k$ has spiked states.*

By dualisation and the observation that parity automata are special Streett automata we simply get:

Corollary 4.3.11 *A deterministic Streett or parity automaton that recognises the language of $\mathcal{B}_n^k$ must have at least as many states as $\mathcal{D}_n^k$ has spiked states.*

The restriction to spiked states is minor – using the estimation of Lemma 4.2.6, we get:

Theorem 4.3.12 *$\mathcal{D}_n^k$ has less than 1.5 times as many states as the smallest deterministic Streett (or parity) automaton that recognises the language of $\mathcal{B}_n^k$.*

4.4 Summary

In this chapter, we have established the following results.

1. We showed that our determinisation construction from nondeterministic parity automata (and consequently, 1-pair Rabin) to deterministic Rabin automata was optimal.

2. We showed that our determinisation construction from nondeterministic parity automata to deterministic parity automata was tight upto a small constant factor of 1.5. In the case of determinising 1-pair Rabin automata to deterministic parity automata, we have matching upper and lower bounds.

3. We showed that Piterman's determinisation construction from nondeterministic Büchi automata with state-based acceptance condition to deterministic parity automata was optimal modulo $3n$.

4. We showed that our determinisation construction from nondeterministic generalised Büchi automata to deterministic Rabin automata was optimal.

5. We showed that our determinisation construction from nondeterministic generalised Büchi automata to deterministic parity automata was tight upto a small constant factor of 1.5.

In the next chapter, we will introduce complementation constructions for parity and generalised Büchi automata that use the data structures arising from determinisation and show that they are tight.

Chapter 5

Complementation

The complementation problem for any automaton $\mathcal{A}$ recognising a language $\mathcal{L}(\mathcal{A})$ is a procedure performed on $\mathcal{A}$ that returns an automaton $\mathcal{C}$ recognising the complementary language $\mathcal{L}(\overline{\mathcal{A}})$. In the case of automata on finite words, complementation is easy. In fact, complementing a nondeterministic finite automaton is as easy as determinising the given automaton and dualising it. For ω-automata, while this procedure still works, it turns out that there are more efficient ways[Var07] to complement ω-automata. For example, the problem of Büchi complementation is in $o\big((0.76\,n)^n\big)$[Sch09a], while Büchi determinisation is in $\theta(1.65\,n)^n$[Sch09b] which is a significant gap.

In this chapter, we consider the problem of complementation for nondeterministic parity and generalised Büchi automata. The usual way of complementing these automata is to convert them first to Büchi automata and then apply the tight complementation procedure[Sch09a] based on tight level rankings.

Instead of going through tight level rankings, we try to reconnect determinisation and complementation as in the case of automata on finite words. We devise succinct structures for complementation inspired by the structures we use for our tight determinisation constructions. In other words, we connect determinisation and complementation and still avoid the full complexity of determinisation that would otherwise be the case.

5.1 Complementing nondeterministic Generalised Büchi automata and Büchi automata

In order to construct a concise data structure for complementation, we first show that we can cut acceptance into two phases: a finite phase where we track only the reachable states, and an infinite phase where we also track acceptance. We then use this simple observation to devise an abstract complementation procedure, and then suggest a succinct data structure for it.

Let $\mathcal{A}$ be a nondeterministic generalised Büchi automaton recognising the language $L(\mathcal{A})$. Let $\mathcal{D}$ be a deterministic Rabin automaton that also recognises $L(\mathcal{A})$.

We first argue that acceptance of a word $\alpha \cdot \alpha'$ with $\alpha \in \Sigma^*$ and $\alpha' \in \Sigma^\omega$ depends only on α' and the states reachable through α.

Lemma 5.1.1 *If $I \rightarrow^\alpha Q \Leftrightarrow I \rightarrow^\beta Q$ then $\alpha \cdot \alpha' \in L(\mathcal{A}) \Leftrightarrow \beta \cdot \alpha' \in L(\mathcal{A})$.*

Proof. It is easy to see how an accepting run of $\mathcal{A}$ on $\alpha \cdot \alpha'$ can be turned into an accepting run on $\beta \cdot \alpha'$, and vice versa. □

This provides us with the following abstract description of a nondeterministic acceptance mechanism for the complement language of $\mathcal{A}$.

1. When reading an ω-word α, we first keep track of the reachable states R for a finite amount of time. (subset construction)

2. Eventually, we swap to a tree that consists only of nodes that are henceforth stable, and that are the only nodes that are henceforth stable, such that none of these nodes is henceforth accepting.

3. We verify the property described in (2).

Lemma 5.1.2 *The abstract decision procedure accepts an input word iff it is rejected by the deterministic Rabin automaton $\mathcal{D}$.*

Proof. The 'only if' direction follows directly from the previous lemma.

For the 'if' direction, we can guess a point i in the run of $\mathcal{D}$ on α where all eventually stable nodes are introduced and stable, and none of them is henceforth accepting. We claim that we can simply guess this point of time, but instead of going to the respective generalised tree $d_i = (\mathcal{T}, l, h)$, we go to $d'_i = (\mathcal{T}', l', h')$, where $\mathcal{T}'$ is the restriction of $\mathcal{T}$ to the henceforth stable states, and l' and h' are the restrictions of l and h to $\mathcal{T}'$. (Note that the subtree of henceforth stable nodes is always ordered.)

Clearly, all nodes in $\mathcal{T}'$ are stable, and none of them are accepting in the future. It remains to show that none of their descendants is stable. Assume one of the children a node $v \in \mathcal{T}'$ spawns eventually is stable. We now consider a part of the 'run' of our mechanism starting at i, $d'_i d'_{i+1} d'_{i+2} \ldots$. Invoking König's Lemma, we get a run $\rho = q_0 q_1 \ldots$ such that, for some $j > i$ and for all $m > j$, some $v_j = \mathsf{host}(q_j, d'_j)$, which is a true descendant of v, is the host of q_j. Using a simple inductive argument that exploits that v is henceforth stable but not accepting, this implies for the run $\rho = d_0 d_1 \ldots$ that, for the same $j > i$ and for all $m > j$, some $v'_j = \mathsf{host}(q_j, d'_j)$, which is a true descendant of v, is a host of q_j. This implies in turn that some descendant of v is eventually stable and thus leads to a contradiction. □

A similar argument holds for the deterministic Rabin automaton that is the result of determinising a nondeterministic Büchi automaton.

We call an ordered tree *flat* if it contains only nodes of length ≤ 1.

Lemma 5.1.3 *We can restrict the choice in (2) to flat trees.*

Proof. If we rearrange the nodes in $\mathcal{T}$ following the "stealing and hosting order", that is, mapping a node v with length ≥ 1 to a smaller node v' with length ≥ 1 if either v' is an ancestor of v or an initial sequence of v is an older sibling of an initial sequence of v', then this describes a unique bijection $b : \mathcal{T} \to \mathcal{F}$, where $\mathcal{F}$ is the flat tree with $|\mathcal{T}| = |\mathcal{F}|$, and we choose $d'_i = (\mathcal{F}, l' : b(v) \mapsto l(\mathsf{pred}(b(v))) \cup l(v) \setminus l(\mathsf{pred}(v)), h' : b(v) \mapsto h(v))$ instead of $d_i = (\mathcal{T}, l, h)$. (The complicated looking $l' : b(v) \mapsto l(\mathsf{pred}(b(v))) \cup l(v) \setminus l(\mathsf{pred}(v))$ just means $v = \mathsf{host}(q, d_i) \Leftrightarrow b(v) = \mathsf{host}(q, d'_i)$, that is, the hosts are moved, not the full label.)

It is easy to see that, if we compare two runs starting in d_i and d'_i on any word, they keep this relation. □

To obtain a succinct data structure for the second phase of the run, we do not follow the precise development of the individual new children of the henceforth stable nodes, but rather follow simple subset constructions. One subset that is kept for all stable nodes is the *union* of the nodes of its children. Note that, to keep track of this union, it is not necessary to keep track of the distribution of these sets to the different children (let alone their descendants).

To check that all children spawned at a particular point j in a run will eventually be deleted, one can keep track of an additional subset: the union of all labels of nodes of children that already existed in j. If this subset runs empty, then all of these children have been removed. Vice versa, if all of these children are removed, then this subset runs empty.

Note that these subsets are, in contrast to the nodes in the flat generalised history tree (or history tree), not numbered. For efficiency, note that it suffices to use the second subset for only one of the nodes in the flat trees at a time, changing this node in a round robin fashion.

Theorem 5.1.4 *The algorithm outlined above describes a nondeterministic Büchi automaton that recognises the complement of the language of $\mathcal{A}$.*

The trees and sets we need can be encoded using the following data structure. We first enrich the set of states by a fresh marker m, used to mark the extra subset for new children in the stable node under consideration, to $Q_m = Q \cup \{m\}$. We then add the normal subsets that capture the label of all children as a child of each stable state as a single child of this state. For a single stable state that we currently track, we add a (possibly second and then younger) child for new children. The labelling is as described above, except that m is added to the label for new children (which otherwise might be empty) and its ancestors.

When we are complementing ordinary Büchi automata, the above data structure

is clearly a flat tree. Additionally, when we are complementing generalised Büchi automata, we can now choose $h : v \to k+1$ for all non-stable nodes v in this tree. As this naming convention clearly identifies these nodes, we can represent the tree as a flat tree.

5.1.1 Complexity of complementing generalised Büchi automata

In this section, we establish lower bounds for the complementation of generalised Büchi automata and show that the construction we outlined tightly meets these lower bounds. Our lower bound proof builds on *full* automata, defined in Section 4.1.2. A generalised Büchi automaton $\mathcal{B}_n^k = (\Sigma, Q, I, T, \{F_i \mid i \in [k]\})$ is called *full* if

- $\Sigma_n^k = 2^{Q \times [k+1] \times Q}$, $|Q| = n$, and $I = Q$,
- $T = \{(q, \sigma, q') \mid \exists i \in [k+1].\ (q, i, q') \in \sigma\}$, and
- $F_i = \{(q, \sigma, q') \mid (q, i, q') \in \sigma\}$.

As each generalised Büchi automaton with n states and k accepting sets can be viewed as a language restriction (by alphabet projection) of a full automaton, full automata are useful tools in establishing lower bounds. We show that, for each $\mathcal{B}_n^k$, there is a family of $L_n^k \subseteq \Sigma_n^k$ such that a^ω is not in the language of $\mathcal{B}_n^k$ for any $a \in L_n^k$, and each nondeterministic generalised Büchi automaton that recognises the complement language of $\mathcal{B}_n^k$ must have at least $|L_n^k|$ states. The size of this alphabet is such that the size of $\mathcal{B}_n^k$ is between $|L_n^k|$ and $|L_{n+1}^{k+1}|$, which provides us with tight bounds for the complementation of generalised Büchi automata.

Let us first define the letters in L_n^k. We call a function $f : Q \to \mathbb{N}$ *full* if its domain is $[n]$ for some n. Let f be a full function with domain $[n]$ then we call a function $f_\# : [n] \to [k]$ a *k-numbering* of f. We denote by $\mathsf{enc}(f, f_\#)$ the letter encoding a function f with k-numbering $f_\#$ as the letter that satisfies

- $(p, \mathsf{enc}(f, f_\#), q) \in T$ iff $f(q) \leq f(p)$, and
- $(p, \mathsf{enc}(f, f_\#), q) \in F_b$ iff *either* $f(q) < f(p)$ *or* ($f(p) = f(q)$ and $f_\#\big(f(p)\big) \neq b$).

Obviously, if two full functions f, g with respective k-numberings $f_\#, g_\#$ encode the same letter $\mathsf{enc}(f, f_\#) = \mathsf{enc}(g, g_\#)$, then they are equal. First, we note that the word a^ω is not in the language of $\mathcal{B}_n^k$.

Lemma 5.1.5 *Let f be a full function, $f_\#$ be a k-numbering of f, and let a be the letter encoded by f and $f_\#$. Then a^ω is rejected by $\mathcal{B}_n^k$.*

Proof. Assume, for contradiction, that there is an accepting run ρ of $\mathcal{B}_n^k$ on a^ω. By the definition of an encoding, the sequence $f_i = f(\rho_i)$ is monotonously decreasing. It will therefore stabilise eventually, say at j. (I.e., $\forall l \geq j.\ f(\rho_l) = f(\rho_j)$.) By the definition of accepting transitions for encoded letters there will henceforth be no more transition from the final transitions $F_{f_\#(f_j)}$. □

We now define $L_n^k = \{\mathsf{enc}(f, f_\#) \mid f \text{ is full and } f_\# \text{ is a } k\text{-numbering of } f\}$.

Theorem 5.1.6 *A generalised Büchi automaton $\mathcal{C}_n^k$ that recognises the complement language of $\mathcal{B}_n^k$ has at least $|L_n^k|$ states.*

Proof. The previous lemma establishes that a^ω is accepted by $\mathcal{C}_n^k$. We choose accepting runs ρ_a with infinity set I_a for each letter $a \in L_n^k$, and show by contradiction that I_a and I_b are disjoint for two different letters $a, b \in L_n^k$.

Assume that this is not the case for two different letters a and b. It is then simple to infer from their accepting runs ρ_a and ρ_b natural numbers $l, m, n, a \in \mathbb{N}$ such that $\rho = \rho_a[0, l](\rho_b[a, a + m]\rho_a[l, l + n])^\omega$ is accepting. Then $w = a^l(b^m a^n)^\omega$ is accepted by $\mathcal{C}_n^k$, as ρ is a run of w. We lead this to a contradiction by showing that w is in the language of $\mathcal{B}_n^k$.

We have $a = \mathsf{enc}(f, f_\#) \neq b = \mathsf{enc}(g, g_\#)$. Let us first assume $f = g$. Then $f_\# \neq g_\#$, and we can first choose an i with $f_\#(i) \neq g_\#(i)$ and then a q with $f(q) = i$. It is now simple to construct an accepting run with trace q^ω for $\mathcal{B}_n^k$ for w.

Let us now assume $f \neq g$. We then set i to the minimal number such that f and g differ in i ($f^{-1}(i) \neq g^{-1}(d)$, where $^{-1}$ denotes the preimage of i. W.l.o.g., we assume $f^{-1}(i) \setminus g^{-1}(i) \neq \emptyset$. We choose a $q \in f^{-1}(i) \setminus g^{-1}(i)$. It is now again simple to construct an accepting run with trace q^ω for $\mathcal{B}_n^k$ for w. ↯

This closes the case distinction and provides the main contradiction. □

The only thing that remains to be shown is tightness. But there is obviously an injection from the flat trees described at the end of the complementation (plus the subsets) of an automaton with n states and k accepting pairs into L_{n+1}^{k+1}. This provides:

Proposition 5.1.7 *The size of the language $|L_{n+1}^{k+1}|$ is bigger than the size of the generalised Büchi automaton $|\mathcal{B}_n^k|$.*

This provides bounds which are tight in k and n with a negligible margin of 1. For large k, the size $|L_n^k|$ can be approximated by $\left(\frac{kn}{e}\right)^n$: It is not hard to show that the size of L_n^k is dominated by encodings that refer to functions from $[n]$ onto $[n]$, and the number of these encodings is $n!k^n$. (E.g., $|L_n^n| < (e-1)n!n^n$.)

Our conjecture is that the construction is tight at least in n. The reason for this assumption is that the increment in n stems from the round robin construction that keeps track of the stable node under consideration, while the alphabet L_n^k refers to the far more restricted case that stable states never spawn new children, rather than merely requiring that none of the children spawned is henceforth stable.

As Büchi automata are a special case of generalised Büchi automata, this tightness up to a factor of n carries over to Büchi complementation. It is not difficult to show that our tight complementation construction and the optimal rank-based construction of [Sch09a] operate similarly and that these methods converge.

5.2 Complementing parity automata

The construction described in this section draws from the introduction of efficient techniques for the determinisation of parity automata in Section 3.3. The nested history trees used there have been our inspiration for the *flattened nested history trees* that form the core data structure in the complementation from Subsection 5.2.2 and are the backbone of the lower bound proof from Subsection 5.2.4.

The intuition for the complementation is to use the nondeterministic power of a Büchi automaton to reduce the size of the data stored for determinisation. As usual, this nondeterministic power is intuitively used to guess a point in time, where all nodes of the nested history trees from parity determinisation from Section 3.3, which are eventually always stable, are henceforth stable. Alongside, the set of stable nodes can be guessed.

Like in the construction for generalised Büchi automata, the structure can then be flattened, preserving the 'nicking order', the order in which older nodes and descendants take preference in taking states of the nondeterministic parity automaton that is determinised. The complement automaton runs in two phases: a first phase before this guessed point in time, and a second phase after this point, where the run starts in such a flattened tree.

We will first introduce *flattened nested history trees* as our main data structure. While we take inspiration from the nested history trees of Section 3.3, the construction is self-contained. Secondly, we will show that Büchi automata recognising the complement language of the full nondeterministic parity automaton $\mathcal{P}_n^{\Pi}$ need to be large by showing disjointness properties of accepting runs for a large class of words, one for each full flattened nested history tree introduced in Subsection 5.2.1. The definition of this language is also instructive in how the data structure is exploited.

We extend our data structure by markers, resulting in *marked flattened trees*, which are then used as the main part of the state space of the natural complementation construction introduced in Subsection 5.2.2. We show correctness of our complementation construction in Subsection 5.2.3 and tightness up to an $O(n)$ factor in Subsection 5.2.4.

Note that all our constructions assume $\max \Pi \geq 2$, and therefore do not cover the less expressive coBüchi automata.

5.2.1 Flattened nested history trees & marked flattened trees

Flattened nested history trees (FNHTs) are the main data structure used in our complementation algorithm. For a given parity automaton $\mathcal{P} = (Q, \Sigma, I, T, \mathsf{pri} : T \to \Pi)$, an FNHT over the set Q of states, maximal priority $\pi_m = \max \Pi$ and maximal even priority $\pi_e = \mathsf{opt}\Pi$, is a tuple $(\mathcal{T}, l_s : \mathcal{T} \to 2^Q, l_l : \mathcal{T} \to 2\mathbb{N}, l_p : \mathcal{T} \to 2^Q, l_r : \mathcal{T} \to 2^Q)$, where $\mathcal{T}$(an ordered, labelled tree) is a non-empty, finite, and prefix closed subset of finite sequences of natural numbers and a special symbol $\mathfrak{s}$ (for *stepchild*), $\omega \cup \{\mathfrak{s}\}$, that satisfies the constraints given below. We call a node $v\mathfrak{s} \in \mathcal{T}$ a *stepchild* of v, and refer to all other nodes vc with $c \in \omega$ as the *natural children* of v. $[\pi](v) = \{vc \mid c \in \omega$ and $vc \in \mathcal{T}\}$ is the set of natural children of v. The root is a stepchild.

The **constraints** an FNHT quintuple has to satisfy are as follows:

- Stepchildren have only natural children, and natural children only stepchildren.
- Only natural children and, when the highest priority π is odd, the root may be leafs.
- $\mathcal{T}$ is order closed: for all $c, c' \in \omega$ with $c < c'$, $vc' \in \mathcal{T}$ implies $vc \in \mathcal{T}$.
- For all $v \in \mathcal{T}$, $l_s(v) \neq \emptyset$.
- If v is a stepchild, then $l_p(v) = \emptyset$.
- If v is a stepchild, then $l_s(v) = l_r(v) \cup \bigcup_{v' \in [\pi](v)} l_s(v')$.

  The sets $l_s(v')$ and $l_s(v'')$ are disjoint for all $v', v'' \in [\pi](v)$ with $v' \neq v''$, and $l_r(v)$ is disjoint with $\bigcup_{v' \in [\pi](v)} l_s(v')$.
- If v is a natural child, then $l_p(v) \neq \emptyset$, $l_s(v) = l_p(v) \cup l_r(v)$, and $l_p(v) \cap l_r(v) = \emptyset$.
- If a natural child v is not a leaf, then $l_s(v\mathfrak{s}) = l_p(v)$.
- $l_l(\varepsilon) = \pi_e$ and, for all $v \in \mathcal{T}$, $l_l(v) \geq 2$.
- If $v\mathfrak{s} \in \mathcal{T}$, then $l_l(v\mathfrak{s}) = l_l(v) - 2$, and if $vc \in \mathcal{T}$ for $c \in \omega$, then $l_l(vc) = l_l(v)$.

The elements in $l_s(v)$ are called the states, $l_p(v)$ the pure states, and $l_r(v)$ the recurrent states of a node v, and $l_l(v)$ is called its level. Note that the level follows a simple pattern: the root is labelled with the maximal even priority, $l_l(\varepsilon) = \pi_e$, the level of natural children is the same as the level of their parents, and the level of a stepchild $v\mathfrak{s}$ of a node v is two less than the level of v. For a given maximal even priority π_e, the level is therefore redundant information that can be reconstructed from the node and π_i. For a given set Q and maximal priority π, $\mathsf{fnht}(Q, \pi)$ denotes the flattened nested history trees over Q. An FNHT is called *full* if the states $l_s(\varepsilon) = Q$ of the root is the full set Q.

To include an acceptance mechanism, we enrich FNHTs to *marked flattened tress* (MFTs), which additionally contain a *marker* v_m and a marking set Q_m, such that

- either $v_m = (\overline{v}, r)$ with $\overline{v} \in \mathcal{T}$ is used to mark that we follow a breakpoint construction on the recurrent states, in this case $l_r(\overline{v}) \supseteq Q_m \neq \emptyset$,
- or $v_m = (\overline{v}, p)$ such that $\overline{v}$ is a leaf in $\mathcal{T}$ is used to mark that we follow a breakpoint construction on the pure states of a leaf $\overline{v}$, in this case $l_p(\overline{v}) \supseteq Q_m \neq \emptyset$.

The marker is used to mark a property to be checked. For markers $v_m = (\overline{v}, r)$, the property is that a particular node would not spawn stable children in a nested history tree. As usual in Safra like constructions, this is checked with a breakpoint, where a breakpoint is reached when all children of a node spawned prior to the last breakpoint die. For markers $v_m = (\overline{v}, p)$, the property is that all runs that are henceforth trapped in the pure nodes of v must eventually encounter a priority $l_l(v) - 1$. This priority is then dominating, and implies rejection as an odd priority. We check these properties round robin for all nodes in $\mathcal{T}$, skipping over nodes, where the respective sets $l_r(\overline{v})$ or $l_p(\overline{v})$ are empty, as the breakpoint there is trivially reached immediately.

For a given FNHT $(\mathcal{T}, l_s, l_l, l_p, l_r)$, $\mathsf{next}(v_m)$ is a mapping from a marker v_m to a marker/marking set pair $(\overline{v}, r), l_r(\overline{v})$ or $(\overline{v}, p), l_p(\overline{v})$. The new marker is the first

marker after v_m in some round robin order such that the set $l_r(\overline{v})$ or $l_p(\overline{v})$, resp., is non-empty.

If $(\mathcal{T}, l_s, l_l, l_p, l_r)$ is an FNHT and v_m and Q_m satisfy the constraints for markers and marking sets from above, then $(\mathcal{T}, l_s, l_l, l_p, l_r; v_m, Q_m)$ is a marked flattened tree. For a given set Q and priorities Π with maximal priority $\pi = \max \Pi$, $\mathsf{mft}(Q, \pi)$ denotes the marked flattened trees over Q. A marking is called *full* if either $v_m = (\overline{v}, r)$ and $Q_m = l_r(\overline{v})$, or $v_m = (\overline{v}, p)$ and $Q_m = l_p(\overline{v})$.

5.2.2 Construction

For a given nondeterministic parity automaton $\mathcal{P} = (Q, \Sigma, I, T, \mathsf{pri} : T \to \Pi)$ with maximal even priority $\pi_e > 1$, we construct a nondeterministic Büchi automaton $\mathcal{C} = (Q', \Sigma, \{I\}, T', F)$ that recognises the complement language of $\mathcal{P}$ as follows. First we set $Q' = Q_1 \cup Q_2$ with $Q_1 = 2^Q$ and $Q_2 = \mathsf{mft}(Q, \pi)$, and $T' = T_1 \cup T_t \cup T_2$, where

- $T_1 \subseteq Q_1 \times \Sigma \times Q_1$ are transitions in an initial part Q_1 of the states of $\mathcal{C}$,
- $T_t \subseteq Q_1 \times \Sigma \times Q_2$ are transfer transitions that can be taken only once in a run, and
- $T_2 \subseteq Q_2 \times \Sigma \times Q_2$, are transitions in a final part Q_2 of the states of $\mathcal{C}$,

where T_1 and T_2 are deterministic. We first define a transition function δ for the subset construction and functions δ_i for all priorities $i \in \Pi$, and then the sets T_1, T_t, and T_2.

- $\delta : (S, \sigma) \mapsto \{q \in Q \mid \exists s \in S.\ (s, \sigma, q) \in T\}$,
- $\delta_i : (S, \sigma) \mapsto \big\{q \in Q \mid \exists s \in S.\ (s, \sigma, q) \in T \text{ and } \mathsf{pri}\big((s, \sigma, q)\big) \succcurlyeq i\big\}$,
- $T_1 = \big\{(S, \sigma, S') \in Q_1 \times \Sigma \times Q_1 \mid S' = \delta(S, \sigma)\big\}$,

  where only transitions $(\emptyset, \sigma, \emptyset)$ are accepting.

- $T_t = \{(S, \sigma, (\mathcal{T}, l_s, l_l, l_p, l_r; v_m, Q_m)) \in Q_1 \times \Sigma \times Q_2 \mid l_s(\varepsilon) = \delta(S, \sigma)\}$ and we have that$(\mathcal{T}, l_s, l_l, l_p, l_r; v_m, Q_m)$ is a marked flattened tree.
- $T_2 = \{((\mathcal{T}, l_s, l_l, l_p, l_r; v_m, Q_m), \sigma, s) \in Q_2 \times \Sigma \times Q_2 \mid$
  - if v is a stepchild, then $l''_s(v) = \delta_{l_l(v)+1}(l_s(v), \sigma)$
  - if v is a natural child, then $l''_s(v) = \delta_{l_l(v)-1}(l_s(v), \sigma)$
  - if v is a natural child, then $l''_r(v) = \delta_{l_l(v)-1}(l_r(v), \sigma) \cup \delta_{l_l(v)}(l_s(v), \sigma)$,
  - starting at the root, we then define inductively:
    - $l'_s(\varepsilon) = l''_s(\varepsilon)$,
    - if vc is a natural child, then $l'_s(vc) = (l''_s(vc) \cap l'_s(v)) \setminus \bigcup_{c'<c} l''_s(vc')$, $l'_r(vc) = l''_r(vc) \cap l'_s(vc)$, and $l'_p(vc) = l'_s(vc) \setminus l'_r(vc)$, and
    - if $v\mathfrak{s}$ is a stepchild, then $l'_s(v\mathfrak{s}) = l'_p(v)$.
  - if one exists, we extend the functions to obtain the unique FNHT $(\mathcal{T}, l'_s, l_l, l'_p, l'_r)$ (otherwise $\mathcal{C}$ blocks)
  - if $v_m = (\overline{v}, r)$ then $Q'_m = \delta_{l_l(\overline{v})-1}(Q_m, \sigma) \cap l'_r(\overline{v})$, and if $v_m = (\overline{v}, p)$ then $Q'_m = \delta_{l_l(\overline{v})-3}(Q_m, \sigma) \cap l'_p(\overline{v})$,
  - if $Q'_m = \emptyset$, then the transition is accepting and we have $s = (\mathcal{T}, l'_s, l_l, l'_p, l'_r; \mathsf{next}(v_m))$,
  - if $Q'_m \neq \emptyset$, then the transition is not accepting and we have $s = (\mathcal{T}, l'_s, l_l, l'_p, l'_r; v_m, Q'_m)$.

5.2.3 Correctness

To show that $\mathcal{L}(\mathcal{C})$ is the complement of $\mathcal{L}(\mathcal{P})$, we first show that a word accepted by $\mathcal{C}$ is rejected by $\mathcal{P}$ and then, vice versa, that a word accepted by $\mathcal{P}$ is rejected by $\mathcal{C}$.

Lemma 5.2.1 *If $\mathcal{C}$ has an accepting run on α, then $\mathcal{P}$ rejects α.*

Proof. Let $\rho = S_0 S_1 \ldots$ be an accepting run of $\mathcal{C}$ on α that stays in Q_1. Thus, there is an $i \in \omega$ such that, for all $j \geq i$, $S_j = \emptyset$. But if we consider any run $\rho' = q_0 q_1 q_2 \ldots$

of $\mathcal{P}$ on α, then it is easy to show by induction that $q_k \in S_k$ holds for all $k \in \omega$, which contradicts $S_i = \emptyset$; that is, in this case $\mathcal{P}$ has no run on α.

Let us now assume that $\rho = S_0 S_1 \dots S_i s_{i+1} s_{i+2} \dots$ is an accepting run of $\mathcal{C}$ on α, where $(S_i, \alpha_i, s_{i+1}) \in T_t$ is the transfer transition taken. (Recall that runs of $\mathcal{C}$ must either stay in Q_1 or contain exactly one transfer transition.)

Let us assume for contradiction that $\mathcal{P}$ has an accepting run $\rho' = q_0 q_1 q_2 \dots$ with even dominating priority $e = \limsup_{j\to\infty} \mathsf{pri}\big((q_j, \alpha_j, q_{j+1})\big)$. Let, for all $j > i$, $s_j = \big(\mathcal{T}, l_s^j, l_l, l_p^j, l_r^j; v_m^j, Q_m^j\big)$ and $S_j = l_s^j(\varepsilon)$. It is again easy to show by induction that $q_j \in S_j$ for all $j \in \omega$. Let now $v_j \in \mathcal{T}$ be the longest node with $l_l^j(v_j) \geq e$ and $q_j \in l_s^j(v_j)$. Note that such a node exists, as $q_j \in S_j = l_s^j(\varepsilon)$ holds. We now distinguish the two cases that the v_j do and do not stabilise eventually and show contradictions in both cases.

1. Assume that there are an $i' > i$ and a $v \in \mathcal{T}$ such that, for all $j \geq i'$, $v_j = v$. We choose i' big enough that $\mathsf{pri}(q_{j-1}, \alpha_{j-1}, q_j) \succ e + 1$ holds for all $j \geq i'$.

   If v is a stepchild, then $q_j \in l_r^j(v)$ for all $j \geq i'$. Using the assumption that ρ is accepting, there is an $i'' > i'$ such that $(s_{i''-1}, \alpha_{i''-1}, s_{i''})$ is accepting, and $v_m^{i''} = (v, r)$. (Note that $q_{i''} \in l_r^{i''}(v)$ implies $l_r^{i''}(v) \neq \emptyset$.) But then we have $q_{i''} \in Q_m^{i''} = l_r^{i''}(v)$, and an inductive argument provides $(s_j, \alpha_j, s_{j+1}) \notin F$ and $q_j \in Q_m^j$ for all $j \geq i''$. This contradicts that ρ is accepting.

   If v is a natural child, then we distinguish three cases. The first one is that there is a $j' \geq i'$ such that $q_{j'} \in l_r^{j'}(v)$. Then we can show by induction that $q_j \in l_r^j(v)$ for all $j \geq j'$ and follow the same argument as for stepchildren, using $i'' > j'$.

   The second is that $q_j \in l_p^j(v)$ holds for all $j \geq i'$. There are now again a few sub-cases that each lead to contradiction. The first is that $l_l(v) = e$. But in this case, we can choose a $j > i'$ with $\mathsf{pri}\big((q_j, \alpha_j, q_{j+1})\big) = e$ and get $q_{j+1} \in l_r^{j+1}(v)$ (contradiction). The second is that $l_l(v) > e$ and v is not a leaf. But in that case, $l_l(v\mathfrak{s}) \geq e$ holds and $q_j \in l_p^j(v)$ implies $q_j \in l_p^j(v\mathfrak{s})$, which contradicts

the maximality of v. Finally, if $l_l(v) > e$ and v is a leaf of $\mathcal{T}$, we get a similar argument as for stepchildren: Using the assumption that ρ is accepting, there is an $i'' > i'$ such that $(s_{i''-1}, \alpha_{i''-1}, s_{i''})$ is accepting, and $v_m^{i''} = (v, p)$. (Note that $q_{i''} \in l_p^{i''}(v)$ implies $l_p^{i''}(v) \neq \emptyset$.) But then we have $q_{i''} \in Q_m^{i''} = l_p^{i''}(v)$, and an inductive argument provides $(s_j, \alpha_j, s_{j+1}) \notin F$ and $q_j \in Q_m^j$ for all $j \geq i''$. This contradicts that ρ is accepting.

2. Assume that the v_j do not stabilise. Let v be the longest sequence such that v is an initial sequence of almost all v_j, and let $i' > i$ be an index such that v is an initial sequence of v_j for all $j \geq i'$. Note that q_j is in $l_s(v'_j)$ for all ancestors v'_j of v_j.

   First, we assume for contradiction that there is a $j > i'$ with $\mathsf{pri}\big((q_j, \alpha_j, q_{j+1})\big) = e' \succ l_l(v)$ (note that the 'better than' relation implies that $e' > l_l(v)$ is even). Then we select a maximal ancestor v' of v with $l_l(v') = e'$; note that such an ancestor is a natural child, as a stepchild has only natural children, and all of them have the same level.

   As v' is an ancestor of v_j and v_{j+1}, $q_j \in l_s^j(v')$ and $q_{j+1} \in l_s^{j+1}(v')$ hold, and by the transition rules thus imply $q_{j+1} \in l_r^{j+1}(v')$, which contradicts $q_{j+1} \in l_s^{j+1}(v_{j+1})$. (Note that $l_l(v') > l_l(v) \geq l_l(v_{j+1})$ holds.)

   Second, we show that $\mathsf{pri}\big((q_j, \alpha_j, q_{j+1})\big) \preccurlyeq l_l(v) + 1$ holds infinitely many times. For this, we first note that the non-stability of the sequence of v_j-s implies that at least one of the following three events happen for infinitely many $j > i'$.

   (a) v is a stepchild, $q_j \in l_s^j(vc)$ for some child vc of v, but, for all children vc' of v, $q_{j+1} \notin l_s^{j+1}(vc')$,

   (b) v is a stepchild, $q_j \in l_s^j(vc)$ for some child vc of v, and $q_{j+1} \in l_s^{j+1}(vc')$ for some older sibling vc' of vc, that is, for $c' > c$, or

   (c) v is a natural child, $q_j \notin l_s^j(v\mathfrak{s})$, but $q_{j+1} \in l_s^{j+1}(v\mathfrak{s})$.

Note that this is just the counter position to "v_j stabilises or v is not maximal". In all three cases, the definition of T_2 requires that $\mathsf{pri}\big((q_j, \alpha_j, q_{j+1})\big) \preccurlyeq l_l(v)+1$.

As the first observation implies that there may only be finitely many transitions with even priority $> l_l(v)$ and the second observation implies that there are infinitely many transitions in ρ' with odd priority $> l_l(v)$, they together imply that $\limsup_{j\to\infty} \mathsf{pri}\big((q_j, \alpha_j, q_{j+1})\big)$ is odd, which leads to the final contradiction.

□

Lemma 5.2.2 *If $\mathcal{P}$ has an accepting run on α, then $\mathcal{C}$ rejects α.*

Proof. Let $\rho = q_0 q_1 q_2 \ldots$ be an accepting run of $\mathcal{P}$ on α with even dominating priority $e = \limsup_{j\to\infty} \mathsf{pri}\big((q_j, \alpha_j, q_{j+1})\big)$.

Let us first assume for contradiction that $\mathcal{C}$ has an accepting run $\rho' = S_0 S_1 \ldots$ which is entirely in Q_1. It is then easy to show by induction that $q_i \in S_i$ holds for all $i \in \omega$, such that no transition of (S_i, α_i, S_{i+1}) is accepting.

Let us now assume for contradiction that $\mathcal{C}$ has an accepting run $\rho' = S_0 S_1 \ldots S_i s_{i+1} s_{i+2} \ldots$, where $(S_i, \alpha_i, s_{i+1}) \in T_t$ is the transfer transition taken. (Recall that runs of $\mathcal{C}$ must either stay in Q_1 or contain exactly one transfer transition.) Let further $s_j = (\mathcal{T}, l_s^j, l_l, l_p^j, l_r^j; v_m^j, Q_m^j)$ and $S_j = l_s^j(\varepsilon)$ for all $j > i$.

It is easy to show by induction that, for all $j \in \omega$, $q_j \in S_j$ holds. We choose an $i_\varepsilon > i$ such that, for all $k \geq i_\varepsilon$, $\mathsf{pri}\big((q_{k-1}, \alpha_{k-1}, q_k)\big) \leq e$ holds.

Let us now look at the nodes $v \in \mathcal{T}$, such that $q_j \in l_s^j(v)$, where $j \geq i_\varepsilon$.

Construction basis.

We have already shown $q_j \in S_j = l_s^j(\varepsilon)$ for all $j > i$, and thus in particular for all $j \geq i_\varepsilon$.

Construction step.

If, for some **stepchild** $v \in \mathcal{T}$ with $l_l(v) \geq e$ and some $i_v \geq i_\varepsilon$, it holds for all $j \geq i_v$ that $q_j \in l_s^j(v)$, then the following holds for all $j \geq i_v$: if $v' \in [\pi](v)$ is a natural child

of v and $q_j \in l_s^j(v')$, then either $q_{j+1} \in l_s^{j+1}(v')$, or there is a younger sibling v'' of v' in $\mathcal{T}$ such that $q_{j+1} \in l_s^{j+1}(v'')$.

As transitions to younger siblings can only occur finitely often without intermediate transitions to older siblings, we have one of the following two cases:

1. for all $j \geq i_v$, $q_j \in l_s^j(v)$, but for every natural child v' of v, $q_j \notin l_s^j(v')$, or
2. there is a natural child v' of v and an index $i_{v'} \geq i_v$ such that, for all $j \geq i_{v'}$, $q_j \in l_s^j(v')$.

As v is a stepchild, the first case implies that $q_j \in l_r^j(v)$ for all $j \geq i_v$. However, using the assumption that ρ' is accepting, there is an $i'_v > i_v$ such that $(s_{i'_v-1}, \alpha_{i'_v-1}, s_{i'_v})$ is accepting, and $v_m^{i'_v} = (v, r)$, as the marker is circulating in a round robin fashion. (Note that $q_{i'_v} \in l_r^{i'_v}(v)$ implies $l_r^{i'_v}(v) \neq \emptyset$.) But then we have $q_{i'_v} \in Q_m^{i'_v} = l_r^{i'_v}(v)$, and an inductive argument provides $(s_j, \alpha_j, s_{j+1}) \notin F$ and $q_j \in Q_m^j$ for all $j \geq i'_v$.

In the second case, we continue with v' and the index $i_{v'}$.

If, for some **natural child** $v \in \mathcal{T}$ with $l_l(v) > e$ and some $i_v \geq i_\varepsilon$, it holds for all $j \geq i_v$ that $q_j \in l_s^j(v)$, then one of the following holds.

1. There is an $i'_v \geq i_v$ such that $q_{i'_v} \in l_r^{i'_v}(v)$.
2. For all $j \geq i_v$, $q_j \in l_p^j(v)$.

In the first case, it is easy to show by induction that $q_j \in l_r^j(v)$ holds for all $j \geq i_{v'}$. We can then again use the assumption that ρ' is accepting. Consequently, there is an $i''_v > i'_v$ such that $(s_{i''_v-1}, \alpha_{i''_v-1}, s_{i''_v})$ is accepting, and $v_m^{i''_v} = (v, r)$, as the marker is circulating in a round robin fashion. (Note that $q_{i''_v} \in l_r^{i''_v}(v)$ implies $l_r^{i''_v}(v) \neq \emptyset$.) But then we have again $q_{i''_v} \in Q_m^{i''_v} = l_r^{i''_v}(v)$, and an inductive argument provides $(s_j, \alpha_j, s_{j+1}) \notin F$ and $q_j \in Q_m^j$ for all $j \geq i''_v$.

In the second case, if v is not a leaf, then it holds for all $j \geq i_{v\mathfrak{s}} = i_v$ that $q_j \in l_s^j(v\mathfrak{s})$, and we can continue with $v\mathfrak{s}$. If v is a leaf, we again use the assumption that ρ' is accepting. Consequently, there is an $i'_v > i_v$ such that $(s_{i'_v-1}, \alpha_{i'_v-1}, s_{i'_v})$ is accepting, and $v_m^{i'_v} = (v, p)$, as the marker is circulating in a round robin fashion.

(Note that v is a leaf and that $q_{i'_v} \in l_p^{i'_v}(v)$ implies $l_p^{i'_v}(v) \neq \emptyset$.) But then we have $q_{i'_v} \in Q_m^{i'_v} = l_p^{i'_v}(v)$, and an inductive argument provides $(s_j, \alpha_j, s_{j+1}) \notin F$ and $q_j \in Q_m^j$ for all $j \geq i'_v$.

If, for some **natural child** $v \in \mathcal{T}$ with $l_l(v) = e$ and some $i_v \geq i_\varepsilon$, it holds for all $j \geq i_v$ that $q_j \in l_s^j(v)$, then there is, by the definition of e, a $j > i_v$ with $\mathsf{pri}(q_{j-1}, \alpha_{j_1}, q_j) = e$. But then $q_{j-1} \in l_s^{j-1}(v)$ and $q_j \in l_s^j(v)$ imply $q_j \in l_r^j(v)$. It is then easy to establish by induction that $q_{j'} \in l_r^{j'}(v)$ for all $j' \geq j$. We can then again use the assumption that ρ' is accepting. Consequently, there is a $j' > j$ such that $(s_{j'-1}, \alpha_{j'-1}, s_{j'})$ is accepting, and $v_m^{j'} = (v, r)$, as the marker is circulating in a round robin fashion. (Note that $q_{j'} \in l_r^{j'}(v)$ implies $l_r^{j'}(v) \neq \emptyset$.) But then we have again $q_{j'} \in Q_m^{j'} = l_r^{j'}(v)$, and an inductive argument provides $(s_k, \alpha_k, s_{k+1}) \notin F$ and $q_k \in Q_m^k$ for all $k \geq j'$.

Contradiction. As the level is reduced by two every second step, one of the arguments that contradict the assumption that ρ' is accepting is reached in at most π_e steps. □

Corollary 5.2.3 *$\mathcal{C}$ recognises the complement language of $\mathcal{P}$.* □

5.2.4 Lower bound and tightness

In order to establish a lower bound, we use a sub-language of the full automaton $\mathcal{P}_n^\Pi$ from Section 4.1.1. Note that partial functions from $Q \times Q$ to Π would work as well as the alphabet for the full automaton. The larger alphabet is chosen for technical convenience in the proofs. Any other language recognised by a nondeterministic parity automaton $\mathcal{P}$ with n states and priorities Π can essentially be obtained by a language restriction via alphabet restriction from $\mathcal{P}_n^\Pi$.

We show that an automaton that recognises this sub-language must have at least as many states as there are full FNHTs in $\mathsf{fnht}(Q, \pi)$ for $n = |Q|$ and $\pi = \max \Pi$.

To show this, we define two letters for each full FNHT $t = (\mathcal{T}, l_s, l_l, l_p, l_r) \in \mathsf{fnht}(Q, \pi)$. $\beta_t : Q \times Q \to 2^\Pi$ is the letter where:

- if v is a stepchild and $p, q \in l_s(v)$, then $l_l(v)+1 \in \beta_t(p,q)$ (provided $l_l(v)+1 \in \Pi$),
- if v is a stepchild, $p \in l_r(v)$, and $q \in l_s(vc)$ for some $c \in \omega$, then $l_l(v) \in \beta_t(p,q)$,
- if v is a stepchild, $c, c' \in \omega$, $c < c'$, $vc' \in \mathcal{T}$, $p \in l_s(vc')$, and $q \in l_s(vc)$, then $l_l(v) \in \beta_t(p,q)$,
- if v is a natural child, $p \in l_p(v)$, and $q \in l_r(v)$ then $l_l(v) \in \beta_t(p,q)$.
- if v is a natural child and $p, q \in l_r(v)$, then $l_l(v) - 1 \in \beta_t(p,q)$, and
- if v is a natural child and $p, q \in l_p(v)$, then $l_l(v) - 1 \in \beta_t(p,q)$.

$\gamma_t : Q \times Q \to 2^\Pi$ is the letter where $i \in \gamma_t(p,q)$ if $i \in \beta_t(p,q)$ and additionally:

- if v is a natural child, $l_l(v) - 2 \in \Pi$, and $p, q \in l_r(v)$, then $l_l(v) - 2 \in \gamma_t(p,q)$,
- if v is a stepchild and $p, q \in l_r(v)$, then $l_l(v) \in \gamma_t(p,q)$, and
- if v is a natural child, $l_l(v) - 2 \in \Pi$, and $p, q \in l_p(v)$, then $l_l(v) - 2 \in \gamma_t(p,q)$.

For a high integer $h > |\mathsf{fnht}(Q,\pi)|$, we now define the ω-word $\alpha^t = (\beta_t {\gamma_t}^{h-1})^\omega$, which consists of infinitely many sequences of length h that start with a letter β_t and continue with $h-1$ repetitions of the letter γ_t, for each full FNHT $t \in \mathsf{fnht}(Q,\pi)$.

We first observe that α^t is rejected by $\mathcal{P}_n^\Pi$.

Lemma 5.2.4 $\alpha^t \notin \mathcal{L}(\mathcal{P}_n^\Pi)$.

Proof. By Lemma 5.2.3, it suffices to show that the complement automaton $\mathcal{C}$ of $\mathcal{P}_n^\Pi$, as defined in Section 5.2.2 accepts α^t. The language is constructed such that $\mathcal{C}$ has a run $\rho = Q(t; v_m^1, Q_m^1)(t; v_m^2, Q_m^2)(t; v_m^3, Q_m^3)\ldots$, in which the transition $\big((t; v_m^i, Q_m^i), \alpha_i^t, (t; v_m^{i+1}, Q_m^{i+1})\big)$ is accepting for $i > 0$ if $i \mod h = 0$. □

Let $\mathcal{B}$ be some automaton with states S that recognises the complement language of $\mathcal{P}_n^\Pi$. We now fix an accepting run $\rho_t = s_0 s_1 s_2 \ldots$ for each word α^t and define the set A_t of states in an 'accepting cycle' as $A_t = \big\{ s \in S \mid \exists i, j, k \in \omega \text{ with } 1 \leq j <$

$k \leq h$ such that $s = s_{ih+j} = s_{ih+k}\}$ holds, and define the interesting states $I_t = A_t \cap \mathsf{infin}(\rho_t)$.

Lemma 5.2.5 *For $t \neq t'$, I_t and $I_{t'}$ are disjoint ($I_t \cap I_{t'} = \emptyset$).*

Proof idea. The proof idea is to assume that a state $s \in I_t \cap I_{t'}$, and use it to construct a word from α_t and $\alpha_{t'}$ and an accepting run of $\mathcal{B}$ on the resulting word from ρ_t and $\rho_{t'}$, and then show that it is also accepted by $\mathcal{P}_n^{\Pi}$.

Proof. Let us assume for contradiction that $s \in I_t \cap I_{t'}$ for $t = (\mathcal{T}, l_s, l_l, l_p, l_r) \neq t' = (\mathcal{T}', l'_s, l'_l, l'_p, l'_r)$.

Noting that we can change the role of t and t', we fix two positions i and i' in the run ρ_t of α_t such that $s = s_i = s_{i'}$, and there is a $j \in \omega$ such that $jh < i < i' \leq j(h+1)$, and two positions j and j' in $\rho_{t'} = s'_0 s'_1 s'_2 \ldots$ such that $j < j'$, $s = s'_j = s'_{j'}$ and there is a $k \in \omega$ with $j \leq k < j'$ such that $(s'_k, \alpha^{t'}_k, s'_{k+1})$ is an accepting transition of $\mathcal{B}$. Note that the definition of I_t provides the first and the definition of $I_{t'}$ the latter.

For the finite words $\beta_1 = \alpha^t_0 \alpha^t_1 \ldots \alpha^t_{i-1}$, $\gamma_1 = s_0 s_1 \ldots s_{i-1}$, $\beta_2 = {\gamma_t}^{i'-1}$, $\gamma_2 = s_i s_{i+1} \ldots s_{i'-1}$, $\beta_3 = \alpha^{t'}_j \alpha^{t'}_{j+1} \ldots \alpha^{t'}_{j'-1}$, and $\gamma_3 = s_j s_{j+1} \ldots s_{j'-1}$, $\rho^{t'}_t = \gamma_1(\gamma_2\gamma_3)^\omega$ is an accepting run of the input word $\alpha^{t'}_t = \beta_1(\beta_2\beta_3)^\omega = \alpha_0\alpha_1\alpha_2\ldots$.

We now show that $\alpha^{t'}_t$ or $\alpha^t_{t'}$ is accepted by $\mathcal{P}_n^{\Pi}$.

We start with the degenerated case that $\mathcal{T} = \{\varepsilon\}$ is the FNHT where the root is a leaf, and thus $\pi = \max \Pi$ odd. (The case $\mathcal{T}' = \{\varepsilon\}$ is similar.) We select a $q \in l'_s(0)$, and consider the run $\rho = q^\omega$ of $\mathcal{P}_n^{\Pi}$ on $\alpha^{t'}_t$. By construction, $\mathsf{pri}(q, \alpha_k, q) \leq \mathsf{opt}\Pi = l_l(\varepsilon)$ holds for all $k \geq i$. Moreover, $\alpha_k = \gamma_t$ holds for infinitely many $k \in \omega$. (In particular, it holds if $k \geq i$ and $(k-i) \mod (i'-i+j'-j) < i'-i$.) For all of these transitions, $\mathsf{pri}(q, \alpha_k, q) = \mathsf{opt}\Pi = l_l(\varepsilon)$ holds, such that $\limsup_{n\to\infty} \big(\overline{\rho}(i)\big) = \mathsf{opt}\Pi$ is even.

Starting with the level $\lambda = \mathsf{opt}\Pi$ of the root and the whole trees $\mathcal{T}$ and $\mathcal{T}'$, we now run through the following construction.

1. We look at the case that there is **some difference in the label of some natural child** $v \in \mathcal{T} \cap \mathcal{T}'$ on the level λ. If there is an oldest child $v \in \mathcal{T} \cap \mathcal{T}'$ with $l_s(v) \neq l'_s(v)$, we assume w.l.o.g. that there is a $q \in l_s(v) \smallsetminus l'_s(v)$. Then there

are two sub-cases, first that there is a $q' \in l_s(v) \cap l'_s(v)$, and second that $l_s(v) \cap l'_s(v) = \emptyset$. In the latter case we choose a $q' \in l'_s(v)$. In both sub-cases, the run $\rho = q^{i'}(q'^{j'-j}q^{i'-i})^\omega = q_0q_1q_2\ldots$ of $\mathcal{P}_n^\Pi$ on $\alpha_t^{t'}$ satisfies $\mathsf{pri}(q_k, \alpha_k, q_{k+1}) \succcurlyeq \lambda - 1$ for all $k \in \omega$, and $\mathsf{pri}(q_k, \alpha_k, q_{k+1}) \succcurlyeq \lambda$ when $q_k = q$ and $q_{k+1} = q'$. (Note that in this case $\alpha_k \in \{\beta_{t'}, \gamma_{t'}\}$ holds.)

Otherwise $l_s(v) = l'_s(v)$ holds for all natural children $v \in \mathcal{T} \cap \mathcal{T}'$ on level λ, and there is a $v \in \mathcal{T} \cap \mathcal{T}'$ on level λ such that $l_r(v) \neq l'_r(v)$. We assume w.l.o.g. that there is a $q \in l_r(v) \smallsetminus l'_r(v)$. We choose a $q' \in l_p(v)$. (Note that $q \neq q' \in l_s(v) = l'_s(v)$.) Then the run $\rho = q^{i'}(q'^{j'-j}q^{i'-i})^\omega = q_0q_1q_2\ldots$ of $\mathcal{P}_n^\Pi$ on $\alpha_t^{t'}$ satisfies $\mathsf{pri}(q_k, \alpha_k, q_{k+1}) \succcurlyeq \lambda - 1$ for all $k \in \omega$, and $\mathsf{pri}(q_k, \alpha_k, q_{k+1}) \succcurlyeq \lambda$ when $q_k = q$ and $q_{k+1} = q'$. (Note that in this case $\alpha_k = \gamma_t$ holds.)

2. We next look at the case where $l_s(v) = l'_s(v)$ and $l_r(v) = l'_r(v)$ holds for all natural children $v \in \mathcal{T} \cap \mathcal{T}$ on level π_e, but there is a natural **child** v on level λ **in the symmetrical difference** of $\mathcal{T}$ and $\mathcal{T}'$. Let us assume w.l.o.g. that $v \in \mathcal{T}'$. Let $q \in l'_s(v)$ and let v be the child of v'. This immediately implies that $q \in l_r(v)$. Thus, the run $\rho = q^\omega$ of $\mathcal{P}_n^\Pi$ on $\alpha_t^{t'}$ satisfies $\mathsf{pri}(q, \alpha_k, q) \succcurlyeq \lambda - 1$ for all $k > i$, and $\mathsf{pri}(q, \alpha_k, q) \succcurlyeq \lambda$ whenever $\alpha_k = \gamma_t$, which happens infinitely often.

3. We finally look at the case where the nodes of $\mathcal{T}$ and $\mathcal{T}'$ on level λ are the same, and where $l_s(v) = l'_s(v)$ and $l_r(v) = l'_r(v)$ hold for all nodes v of $\mathcal{T}$ on level λ, but there is a node v on level λ which is a **leaf** in $\mathcal{T}$ but not in $\mathcal{T}'$. (The case "leaf in $\mathcal{T}'$ but not in $\mathcal{T}$" is entirely symmetric.) Thus, $v\mathfrak{s}0$ is a node in $\mathcal{T}'$, and we select a $q \in l_s(v\mathfrak{s}0)$. If we now consider the run $\rho = q^\omega$ of $\mathcal{P}_n^\Pi$ on $\alpha_t^{t'}$, then $\mathsf{pri}(q, \alpha_k, q) \succcurlyeq \lambda - 3$ holds for all $k > i$. At the same time $\mathsf{pri}(q, \alpha_k, q) \succcurlyeq \lambda - 2$ holds whenever $\alpha_k = \gamma_t$, which happens infinitely often.

If neither of these cases holds, then there must be a natural child v on level λ such that $v\mathfrak{s} \in \mathcal{T} \cap \mathcal{T}'$ and $l_s(v\mathfrak{s}) = l_p(v) = l'_p(v) = l'_s(v\mathfrak{s})$, such that t and t' differ on the descendants of v. We then continue the construction by reducing λ to $\lambda - 2$ and intersecting $\mathcal{T}$ and $\mathcal{T}'$ with the descendants of v in t and t', respectively, and restrict

the co-domain of the labelling functions of t and t' accordingly. This construction will lead to a difference in at most $0.5 \cdot \mathsf{opt}\Pi$ steps. □

Theorem 5.2.6 *An automaton $\mathcal{B}$ that recognises the complement language of $\mathcal{P}_n^\Pi$ has at least as many states as $\mathsf{fnht}(Q, \max \Pi)$ contains full FNHTs, where $\mathsf{fnht}(Q, \max \Pi)$ denotes the flattened nested history trees over a given state set Q for a maximal priority Π.*

Proof. We prove the claim with a case distinction. The first case is that $I_t \neq \emptyset$ holds for all full FNHT $t \in \mathsf{fnht}(Q, \max \Pi)$. Lemma 5.2.5 shows that the sets of interesting states are pairwise disjoint for different trees $t \neq t'$, such that, as none of them is empty, $\mathcal{B}$ has at least as many states as $\mathsf{fnht}(Q, \max \Pi)$ contains full FNHTs.

The second case is there is a full FNHT $t \in \mathsf{fnht}(Q, \max \Pi)$ such that $I_t = \emptyset$. By Lemma 5.2.4, each $\rho_t = s_0 s_1 s_2 \ldots$ is an accepting run. Let now $i \in \omega$ be an index, such that, for all $j \geq i$, $s_j \in \mathsf{infin}(\rho_t)$, and $k \geq i$ an integer with $k \mod h = 0$. $I_t = \emptyset$ implies that $s_{k+j} \neq s_{k+j'}$ for all j, j' with $1 \leq j < j' \leq h$. Then $\mathcal{B}$, and even $\mathsf{infin}(\rho_t)$, has at least $h - 1$ different states, and the claim follows with $h > |\mathsf{fnht}(Q, \max \Pi)|$. □

To show tightness, we proceed in three steps. In a first step, we provide an injection from MFTs with non-full marking to MFTs with full marking.

Next, we argue that the majority of FNHTs is full. Taking into account that there are at most $|Q|$ different markers makes it simple to infer that the states of our complementation construction divided by the lower bound from Theorem 5.2.6 is in $O(n)$.

Lemma 5.2.7 *There is an injection from MFTs with non-full marking to MFTs with full marking in $\mathsf{mft}(Q, \pi)$.*

Proof. For non-trivial trees $\mathcal{T} \neq \{\emptyset\}$, we can simply map an MFT $(\mathcal{T}, l_s, l_l, l_p, l_r; v_m, Q_m)$

- for $v_m = (\overline{v}, p)$ to the MFT $\left(\mathcal{T}', l'_s, l'_l, l'_p, l'_r; v_m, l'_p(\overline{v})\right)$ and

- for $v_m = (\overline{v}, r)$ to the MFT $\big(\mathcal{T}', l'_s, l'_l, l'_p, l'_r; v_m, l'_r(\overline{v})\big)$, where

$\mathcal{T}'$ differs from $\mathcal{T}$ only in that it has a fresh node v, which is the youngest sibling of v_m. l'_s, l'_p, l'_r differ from l_s, l_p, l_r only in $\overline{v}$ and v (where v is only in the pre-image of l'_s, l'_l, l'_p, l'_r). We set $l'_s(v) = l'_p(v) = Q_m$ and, consequently, $l'_r(v) = \emptyset$. We also set $l'_s(v) = l_s(v) \setminus Q_m$.

For $v_m = (\overline{v}, p)$, we set $l'_r(\overline{v}) = l_r(\overline{v})$ and $l'_p(\overline{v}) = l_p(\overline{v}) \setminus Q_m$. Note that, by the definition of markers, $\overline{v}$ is a leaf, and $l'_p(\overline{v})$ is non-empty because the marking in $(\mathcal{T}, l_s, l_l, l_p, l_r; v_m, Q_m)$ is not full.

For $v_m = (\overline{v}, r)$, we set $l'_r(\overline{v}) = l_r(\overline{v}) \setminus Q_m$ and $l'_p(\overline{v}) = l_p(\overline{v})$. Note that $l'_r(\overline{v})$ is non-empty because the marking in $(\mathcal{T}, l_s, l_l, l_p, l_r; v_m, Q_m)$ is not full.

It is easy to see that the resulting MFT is well formed in both cases. What remains is the corner case of $\mathcal{T} = \{\varepsilon\}$.

$(\mathcal{T}, l_s, l_l, l_p, l_r; (\varepsilon, r), Q_m)$ and map it to $(\mathcal{T}', l'_s, l'_l, l'_p, l'_r; (\varepsilon, r), Q_m)$ for $\mathcal{T}' = \{\varepsilon, 0\}$ and $l'_s(\varepsilon) = l_s(\varepsilon)$, $l'_s(\varepsilon) = Q_m$, $l'_p(\varepsilon) = l'_p(0) = \emptyset$, and consequently $l'_s(0) = l'_r(0) = l_s(\varepsilon) \setminus Q_m$. (Note that the latter is non-empty because the marking in $(\mathcal{T}, l_s, l_l, l_p, l_r; (\varepsilon, r), Q_m)$ is not full.) This is again a well formed MFT with full marking.

It is easy to see that the resulting function is injective. □

In Lemma 5.2.7, we have shown that the majority of MFTs have a full marking. Next we will see that the majority of FNHTs is full. (Note that neither mapping is surjective.)

Lemma 5.2.8 *There is an injection from non-full to full FNHTs in* $\mathsf{fnht}(Q, \pi)$.

Proof. To obtain such an injection, it suffices to map a non-full FNHT $(\mathcal{T}, l'_s, l'_l, l'_p, l'_r)$ to the FNHT $(\mathcal{T}', l'_s, l'_l, l'_p, l'_r)$ where $\mathcal{T}'$ differs from $\mathcal{T}$ only in that it has a fresh youngest child v of the root.

l'_s agrees with l_s on every node of $\mathcal{T}$ except for the root ε, and l'_p, l'_r agree with l_p, l_r on every node of $\mathcal{T}$. We set $l'_s(\varepsilon) = Q$, $l'_s(v) = l'_p(v) = Q \setminus l_s(\varepsilon)$, and $l'_r(v) = \emptyset$.

It is obvious that the resulting FNHT $(\mathcal{T}', l'_s, l'_l, l'_p, l'_r)$ is full and well formed, and it is also obvious that the mapping is injective. □

Theorem 5.2.9 *There is a complementation construction for parity automata that is tight up to a factor of $4n+1$, where $n = |Q|$ is the number of states of the complemented parity automaton.*

Proof. For the number of MFTs, Lemma 5.2.7 shows that they are at most twice the number of MFTs with full marking. Note that the marker (v_m, p) can only refer to leafs where $l_p(v_m)$ is non-empty and markers (v_m, r) can only refer to nodes where $l_r(v_m)$ is non-empty. It is easy to see that all sets described in this way are pairwise disjoint. This implies that there are at most $|Q|$ such markers. Thus, the number of MFTs with full marking is at most n times the number of FNHTs.

By Lemma 5.2.8, the number of FNHTs is in turn at most twice as high as the number of all full FNHTs. Thus we have bounded the number of MFTs by $4n$ times the number of full FNHTs used to estimate the lower bound in Theorem 5.2.6, irrespective of the priorities.

What remains is the trivial observation that the second part of the state-space, the subset construction, is dwarfed by the number of MFTs. Consequently, we can estimate the state-space of the complement automaton divided by the lower bound from Theorem 5.2.6 by $4n + 1$. □

Chapter 6

Summary and discussion

In this chapter, we recap our results and discuss some issues.

6.1 Summary of results

We have comprehensively studied the problems of determinisation and complementation of generalised Büchi automata. These are two classes of ω-automata that occur in practical applications.

We started with Schewe's tight construction[Sch09b] for the determinisation of Büchi automata in our quest to arrive at tight determinisation of general and more succinct acceptance conditions. In order to efficiently handle parity automata, we started with a special case : 1-pair Rabin automata (analogous to a parity automaton with 3 colours, or an automaton expressing the intersection of a Büchi and coBüchi condition). We discovered that we can easily nest different levels of states of deterministic automata to handle parity conditions. The extension to the case of generalised Büchi automata has proven to be even simpler.

The popularity of the parity condition in formal verification is reflected by the research that has gone into establishing faster solutions for emptiness games of parity automata. Determinising to parity automata is therefore quite attractive and we have shown that we can produce deterministic parity automata with our constructions.

We have used the rich history of interplay between automata and game theory to establish lower bounds for these constructions. By composing a deterministic automaton with a game that requires only positional strategies, we were able to prove that we cannot determinise to Rabin automata with even a single state lesser than the upper bound. That these games can handle such complicated structures beyond ordinary history trees is quite surprising. A careful and intricate treatment of these games served to establish this lower bound and is more involved than for the case of proving a lower bound for ordinary Büchi automata.

We invoked a rather intuitive argument to prove a lower bound for determinising to parity automata. By changing our target language to the complement, we obtained tight bounds for Streett conditions — the dual of Rabin conditions are in fact, also parity conditions —, means that the same tighter lower bound applies to deterministic parity automata. We summarise our results for determinisation in the table below.

The first column relates to a particular construction on ω-automata. We refer to the type of automata by using letters in $\{N, D\} \times \{B, G, R, P, R_1\}$ where N and D relate to the branching mode— nondeterministic or deterministic— and the rest refer to Büchi, generalised Büchi Rabin, parity or 1-pair Rabin automata. The second column states where a precise upper bound was given. The third column states where the precise lower bound was established. The fourth column indicates the degree of tightness of the results. 'Thesis' refers to results shown in this thesis.

We have shown these results were on automata where the accepting condition is on transitions. While this has provided clean results, transference of these results to the case of state-based acceptance is conjectured to be straightforward but with a small gap between the bounds for transition-based acceptance.

We moved on to the problem of complementing ω-automata. The known optimal complementation constructions do not start with determinisation. We have shown that tight determinisation can really be used as a starting point for tight complementation. We have devised succinct data structures inspired by those that arise from

Transformation	Upper Bound	Lower Bound	Comment
$NB \to DR$	[Sch09b]	[CZ09]	Matching bounds
$NB \to DP$	[Pit07] [Sch09b]	Thesis	Optimal modulo $3n$ for the case of automata recognising word languages optimal for the case of trace automata
$NG \to DR$	Thesis	Thesis	Matching bounds
$NG \to DP$	[KPV06]	Thesis	Tight up to a constant of $\leq$1.5
$NR_1 \to DR$	Thesis	Thesis	Matching bounds
$NR_1 \to DP$	Thesis	Thesis	Matching bounds
$NP \to DR$	Thesis	Thesis	Matching bounds
$NP \to DP$	Thesis	Thesis	Tight up to a constant of $\leq$1.5

Table 6.1: Complexities for determinisation

determinisation and end up with a tight (in n) complementation procedure for both generalised Büchi automata and parity automata.

6.2 Discussion

Although the rough complexities of determinisation and complementation of ω-automata have been known in the field of algorithmic verification for more than two decades, a study of the precise complexity took until a few years ago to finally arrive at tight bounds even for the simple Büchi case. It is surprising that an attempt to determinise generalised Büchi automata did not immediately follow the techniques that became standard for translating LTL to automata. All of this points towards the intricate nature of Safra's determinisation construction.

The inherent difficulty of determinisation and complementation has posed challenges to the automata-theoretic approach to formal verification. [Var07] talks about some of these issues. The lack of a simple complementation construction (the GOAL toolkit[TTH13b], for example performs complementation via full determinisation and then employs optimising heuristics) has prevented the use of automata-theoretic toolsets that accept specifications as nondeterministic Büchi or parity automata.

Attempting to circumvent this perceived difficulty, researchers have looked for 'Safraless' procedures[KV05] in the automata-theoretic approach to verification.

While this approach certainly has its merits, it is not prudent to immediately discount 'Safraful' methods. A first edition of the web based probabilistic model-checker IscasMC[HLS$^+$14] has employed our generalised Büchi determinisation algorithm and was found to be competitive at evaluating probabilities of LTL.

Nevertheless, a case can be made for the statement that determinisation constructions will not always result in faster practical solutions. There are indeed many properties that do not need to be specified by nondeterministic automata and there is considerable leeway for speed-up in such situations. A prudent approach would be to combine both 'Safraful' and 'Safraless' methods and optimise according to the situation. A nice example is the current iteration of the aforementioned probabilistic model-checker IscasMC which now uses a lazy method[HLSZ13]: it first checks if a subset construction can be performed; if that result is inconclusive, a breakpoint construction is performed, keeping the full determinisation construction as the last fall-back option because of the high complexity of determinising ω-automata.

It is also worthy to note that the update rule for the LIR construction is less strict than the update rule for other tree-focused constructions. Exploiting this update rule while taking transitions can result in reduction of the state-space.

Safra's construction is not the only determinisation construction for ω-automata. There are a couple of other approaches, for example, the construction of Muller and Schupp[MS95]. These constructions seem to converge and it is easy to use Muller and Schupp's construction to also produce deterministic automata with the same structure as those produced by Safra's construction. Muller and Schupp's construction could also be an alternative starting point to achieve these results.

There are several different approaches to complementing ω-automata. The theoretically optimal constructions are not based on determinisation. Yet a comprehensive study of all known complementation methods in [TFVT10] shows that a determinisation based approach with heuristics performs more efficiently than the theoretically optimal constructions. We thus have on the one side, determinisation based constructions that are efficient, but have no guarantees, and on the other side,

simpler constructions that are equipped with theoretical guarantees. Our complementation constructions take the best of both worlds, unifying beauty and efficiency and it is hoped that this will lead to cleaner implementations of automata-theoretic toolsets for formal verification.

Bibliography

[Büc62] J. Richard Büchi. On a decision method in restricted second order arithmetic. In *International Congress on Logic, Methodology, and Philosophy of Science 1960*, pages 1–11, 1962.

[Chu62] Alonzo Church. Logic, arithmetic and automata. *Proceedings of the international congress of mathematicians*, pages 23–35, 1962.

[CZ09] Thomas Colcombet and Konrad Zdanowski. A tight lower bound for determinization of transition labeled büchi automata. In *Automata, Languages and Programming, 36th International Colloquium, ICALP 2009, Rhodes, greece, July 5-12, 2009, Proceedings, Part II*, pages 151–162, 2009.

[CZ11a] Yang Cai and Ting Zhang. A tight lower bound for streett complementation. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2011, December 12-14, 2011, Mumbai, India*, pages 339–350, 2011.

[CZ11b] Yang Cai and Ting Zhang. Tight upper bounds for streett and parity complementation. In *Computer Science Logic, 25th International Workshop / 20th Annual Conference of the EACSL, CSL 2011, September 12-15, 2011, Bergen, Norway, Proceedings*, pages 112–128, 2011.

[CZL09] Yang Cai, Ting Zhang, and Haifeng Luo. An improved lower bound for the complementation of rabin automata. In *Proceedings of the 24th Annual IEEE Symposium on Logic in Computer Science, LICS 2009, 11-14 August 2009, Los Angeles, CA, USA*, pages 167–176, 2009.

[Dur14] Alexandre Duret-Lutz. LTL translation improvements in spot 1.0. *IJCCBS*, 5(1/2):31–54, 2014.

[FKV06] Ehud Friedgut, Orna Kupferman, and Moshe Y. Vardi. Büchi complementation made tighter. *Int. J. Found. Comput. Sci.*, 17(4):851–868, 2006.

[FS13] Bernd Finkbeiner and Sven Schewe. Bounded synthesis. *International Journal on Software Tools for Technology Transfer*, 15(5-6):519–539, 2013.

[GKSV03] Sankar Gurumurthy, Orna Kupferman, Fabio Somenzi, and M. Y. Vardi. On complementing nondeterministic büchi automata. In *Correct Hardware Design and Verification Methods, 12th IFIP WG 10.5 Advanced Research Working Conference, CHARME 2003, L'Aquila, Italy, October 21-24, 2003, Proceedings*, pages 96–110, 2003.

[GPVW95] Rob Gerth, Doron Peled, Moshe Y. Vardi, and Pierre Wolper. Simple on-the-fly automatic verification of linear temporal logic. In *Protocol Specification, Testing and Verification XV, Proceedings of the Fifteenth IFIP WG6.1 International Symposium on Protocol Specification, Testing and Verification, Warsaw, Poland, June 1995*, pages 3–18, 1995.

[HLS+14] Ernst Moritz Hahn, Yi Li, Sven Schewe, Andrea Turrini, and Lijun Zhang. IscasMC: A web-based probabilistic model checker. In *Nineteenth international symposium of the Formal Methods Europe association (FM)*, volume 8442 of *Lecture Notes in Computer Science*, pages 312–317. Springer, 2014.

[HLSZ13] Ernst Moritz Hahn, Guangyuan Li, Sven Schewe, and Lijun Zhang. Lazy determinisation for quantitative model checking. *CoRR*, abs/1311.2928, 2013.

[JPZ08] Marcin Jurdzinski, Mike Paterson, and Uri Zwick. A deterministic subexponential algorithm for solving parity games. *SIAM Journal on Computing*, 38(4):1519–1532, 2008.

[Kla94] Nils Klarlund. Progress measures, immediate determinacy, and a subset construction for tree automata. *Ann. Pure Appl. Logic*, 69(2-3):243–268, 1994.

[KPV06] Orna Kupferman, Nir Piterman, and Moshe Y Vardi. Safraless compositional synthesis. In *Computer Aided Verification*, pages 31–44. Springer, 2006.

[Kur94] Robert P. Kurshan. *Computer-aided Verification of Coordinating Processes: The Automata-theoretic Approach*. Princeton University Press, Princeton, NJ, USA, 1994.

[KV01] Orna Kupferman and Moshe Y. Vardi. Weak alternating automata are not that weak. *ACM Trans. Comput. Log.*, 2(3):408–429, 2001.

[KV05] Orna Kupferman and Moshe Y. Vardi. Safraless decision procedures. In *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2005), 23-25 October 2005, Pittsburgh, PA, USA, Proceedings*, pages 531–542, 2005.

[KW08] Detlef Kähler and Thomas Wilke. Complementation, disambiguation, and determinization of büchi automata unified. In *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik,*

Iceland, July 7-11, 2008, Proceedings, Part I: Tack A: Algorithms, Automata, Complexity, and Games, volume 5125 of *Lecture Notes in Computer Science*, pages 724–735. Springer, 2008.

[Löd99] Christof Löding. Optimal bounds for transformations of omega-automata. In *Foundations of Software Technology and Theoretical Computer Science, 19th Conference, Chennai, India, December 13-15, 1999, Proceedings*, pages 97–109, 1999.

[LW09] Wanwei Liu and Ji Wang. A tighter analysis of piterman's büchi determinization. *Inf. Process. Lett.*, 109(16):941–945, 2009.

[McN66] Robert McNaughton. Testing and generating infinite sequences by a finite automaton. *Information and Control*, 9(5):521–530, 1966.

[Mic88] Max Michel. Complementation is more difficult with automata on infinite words. *CNET, Paris*, 15, 1988.

[Mos84] Andrzej Wlodzimierz Mostowski. Regular expressions for infinite trees and a standard form of automata. In *Computation Theory - Fifth Symposium, Zaborów, Poland, December 3-8, 1984, Proceedings*, volume 208 of *Lecture Notes in Computer Science*, pages 157–168. Springer, 1984.

[MS95] David E. Muller and Paul E. Schupp. Simulating alternating tree automata by nondeterministic automata: New results and new proofs of the theorems of rabin, mcnaughton and safra. *Theor. Comput. Sci.*, 141(1&2):69–107, 1995.

[Péc86] Jean-Pierre Pécuchet. On the complementation of büchi automata. *Theoretical Computer Science*, 47:95–98, 1986.

[Pit07] Nir Piterman. From nondeterministic büchi and streett automata to deterministic parity automata. *Logical Methods in Computer Science*, 3(3), 2007.

[Pnu77] Amir Pnueli. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science, Providence, Rhode Island, USA, 31 October - 1 November 1977*, pages 46–57, 1977.

[PP06] Nir Piterman and Amir Pnueli. Faster solutions of rabin and streett games. In *21th IEEE Symposium on Logic in Computer Science (LICS 2006), 12-15 August 2006, Seattle, WA, USA, Proceedings*, pages 275–284, 2006.

[PSVW87] A Prasad Sistla, Moshe Y Vardi, and Pierre Wolper. The complementation problem for büchi automata with applications to temporal logic. *Theoretical Computer Science*, 49(2):217–237, 1987.

[Rab69] Michael O Rabin. Decidability of second-order theories and automata on infinite trees. *Transactions of the American Mathematical Society*, pages 1–35, 1969.

[RS59] Michael O Rabin and Dana Scott. Finite automata and their decision problems. *IBM journal of research and development*, 3(2):114–125, 1959.

[Saf88] Shmuel Safra. On the complexity of ω-automata. In *29th Annual Symposium on Foundations of Computer Science, White Plains, New York, USA, 24-26 October 1988*, pages 319–327, 1988.

[Saf92] Shmuel Safra. Exponential determinization for omega-automata with strong-fairness acceptance condition (extended abstract). In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing, May 4-6, 1992, Victoria, British Columbia, Canada*, pages 275–282, 1992.

[Sch07] Sven Schewe. Solving parity games in big steps. In *Proceedings of the 27th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2007), 12–14 December, New Delhi, India*, volume 4805 of *Lecture Notes in Computer Science*, pages 449–460. Springer-Verlag, 2007.

[Sch09a] Sven Schewe. Büchi complementation made tight. In *Proceedings of the 26th International Symposium on Theoretical Aspects of Computer Science (STACS 2009), 26–28 February, Freiburg, Germany*, volume 3 of *Leibniz International Proceedings in Informatics*, pages 661–672. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany, 2009.

[Sch09b] Sven Schewe. Tighter bounds for the determinisation of Büchi automata. In *Proceedings of the Twelfth International Conference on Foundations of Software Science and Computation Structures (FoSSaCS 2009), 22–29 March, York, England, UK*, volume 5504 of *Lecture Notes in Computer Science*, pages 167–181. Springer-Verlag, 2009.

[SF06a] Sven Schewe and Bernd Finkbeiner. Satisfiability and finite model property for the alternating-time μ-calculus. In *Proceedings of the 15th Annual Conference of the European Association for Computer Science Logic (CSL 2006), 25–29 September, Szeged, Hungary*, volume 4207 of *Lecture Notes in Computer Science*, pages 591–605. Springer-Verlag, 2006.

[SF06b] Sven Schewe and Bernd Finkbeiner. Synthesis of asynchronous systems. In *Proceedings of the 16th International Symposium on Logic-Based Program Synthesis and Transformation (LOPSTR 2006), 12–14 July, Venice, Italy*, volume 4407 of *Lecture Notes in Computer Science*, pages 127–142. Springer-Verlag, 2006.

[SS78] William J. Sakoda and Michael Sipser. Nondeterminism and the size of two way finite automata. In *Proceedings of the 10th Annual ACM Symposium on Theory of Computing, May 1-3, 1978, San Diego, California, USA*, pages 275–286, 1978.

[Str82] Robert S Streett. Propositional dynamic logic of looping and converse is elementarily decidable. *Information and control*, 54(1):121–141, 1982.

[SV12] Sven Schewe and Thomas Varghese. Tight bounds for the determinisation and complementation of generalised Büchi automata. In *Proceedings of the 10th International Symposium on Automated Technology for Verification and Analysis (ATVA 2012), 3–6 October, Thiruvananthapuram, Kerala, India*, volume 7561 of *Lecture Notes in Computer Science*, pages 42–56. Springer-Verlag, 2012.

[SV14a] Sven Schewe and Thomas Varghese. Determinising parity automata. In *Proceedings of the 39th International Symposium on Mathematical Foundations of Computer Science (MFCS 2014), 25–29 August, Budapest, Hungary*, 2014.

[SV14b] Sven Schewe and Thomas Varghese. Tight bounds for complementing parity automata. In *Proceedings of the 39th International Symposium on Mathematical Foundations of Computer Science (MFCS 2014), 25–29 August, Budapest, Hungary*, 2014.

[TFVT10] Ming-Hsien Tsai, Seth Fogarty, Moshe Y. Vardi, and Yih-Kuen Tsay. State of büchi complementation. In *Implementation and Application of Automata - 15th International Conference, CIAA 2010, Winnipeg, MB, Canada, August 12-15, 2010. Revised Selected Papers*, pages 261–271, 2010.

[Tho99] Wolfgang Thomas. Complementation of Büchi automata revisited. In *Jewels are Forever*, pages 109–120. Springer, 1999.

[TTH13a] Ming-Hsien Tsai, Yih-Kuen Tsay, and Yu-Shiang Hwang. GOAL for games, omega-automata, and logics. In *Computer Aided Verification - 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings*, pages 883–889, 2013.

[TTH13b] Ming-Hsien Tsai, Yih-Kuen Tsay, and Yu-Shiang Hwang. Goal for games, omega-automata, and logics. In Natasha Sharygina and Helmut Veith, editors, *Computer Aided Verification*, volume 8044 of *Lecture Notes in Computer Science*, pages 883–889. Springer Berlin Heidelberg, 2013.

[Var07] Moshe Y Vardi. The Büchi complementation saga. In *STACS 2007*, pages 12–22. Springer, 2007.

[Wil01] Thomas Wilke. Alternating tree automata, parity games, and modal m-calculus. *Bulletin of the Belgian Mathematical Society Simon Stevin*, 8(2):359, 2001.

[Yan08] Qiqi Yan. Lower bounds for complementation of omega-automata via the full automata technique. *Logical Methods in Computer Science*, 4(1), 2008.

[Zie98] Wieslaw Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theor. Comput. Sci.*, 200(1-2):135–183, 1998.