Determinising Parity Automata*

Sven Schewe and Thomas Varghese

University of Liverpool

Abstract. Parity word automata and their determinisation play an important role in automata and game theory. We discuss a determinisation procedure for nondeterministic parity automata through deterministic Rabin to deterministic parity automata. We prove that the intermediate determinisation to Rabin automata is optimal. We show that the resulting determinisation to parity automata is optimal up to a small constant. Moreover, the lower bound refers to the more liberal Streett acceptance. We thus show that determinisation to Streett would not lead to better bounds than determinisation to parity. As a side-result, this optimality extends to the determinisation of Büchi automata.

1 Introduction

The quest for optimal complementation [19,21,14] and determinisation [11,12,9,15,3] of nondeterministic automata has been long and fruitful. The quest for optimal Büchi complementation techniques seems to have been settled with matching upper [14] and lower [21] bounds. A similar observation might, on first glance, be made for Büchi determinisation, as matching upper [15] and lower [3] bounds were established shortly after those for complementation. However, while these bounds are tight to the state, they refer to deterministic Rabin automata only, with exponentially many Rabin pairs in the states of the initial automaton.

Choosing Rabin automata as targets is not the only natural choice. The dual Streett acceptance condition is a similarly natural goal, and determinising to parity automata seems to be an even more attractive target, as emptiness games for parity automata have a lower computational complexity compared to emptiness games for Streett or Rabin automata. For parity and Streett automata, however, no similarly tight result is known. Indeed, the best known algorithm [9] provides an $O(n!^2)$ bound on the states [15] (for state-based acceptance; the bound can be improved to $O(n!(n-1)!)$ when transition based acceptance is used) of a deterministic parity automaton obtained from a nondeterministic Büchi automaton with n states, as compared to the approximately $(1.65n)^n$ states of deterministic Rabin automaton [15,3].

Another argument for using parity or Streett conditions is that determinisation constructions are often nested. E.g., in distributed synthesis [10,7,5], several co-determinisation (determinisation of the complement language) steps are used. Using Rabin automata as a target in one step, one has to use a determinisation

* A full version with proofs is available at `http://arxiv.org/abs/1401.5394`.

E. Csuhaj-Varjú et al. (Eds.): MFCS 2014, Part I, LNCS 8634, pp. 486–498, 2014.

technique for Streett automata in the next. Streett determinisation, however, is significantly more involved and expensive [13,9].

In this paper, we introduce determinisation procedures for nondeterministic parity automata (NPA) to deterministic Rabin and parity automata (DRA and DPA). Using an algorithmic representation that extends the determinisation procedures from [15], we show that the number of states used in the determinisation of nondeterministic Büchi automata cannot be reduced by a single state, while we establish the tightness of our parity determinisation procedure to below an approximation factor of 1.5, even if we allow for Streett acceptance. This also shows that determinising parity automata to Rabin automata leads to a smaller blow-up than the determinisation to parity or Streett. As a special case, this holds in particular for Büchi automata.

Transition-Based Acceptance. We use a transition based acceptance mechanism for various reasons, but most importantly, cleaner results. Transition-based acceptance mechanisms have proven to be a more natural target of automata transformations. Indeed, all determinisation procedures quoted above have a natural representation with an acceptance condition on transitions, and 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 [20,16] and the treatment of asynchronous systems [17], where the statespace grows by multiplication with the acceptance information (e.g., maximal priority on a finite sequence of transitions), while it can only shrink in case of transition based acceptance. Similarly, tools like SPOT [4] translate LTL formulas to more concise automata with a transition-based acceptance mechanism.
Related Work. Besides the work on complementing [19,21,14] and determinising [11,12,9,15,3] Büchi automata, tight bounds have been obtained for generalised Büchi automata [18], and specialised algorithms for complementing [2] and determinising Streett [13,9] automata have been studied. The construction of deterministic CoBüchi automata with a one-sided error, which is correct for CoBüchi recognisable languages [1], and decision procedures that use emptiness equivalent Büchi [8] or safety [6] automata have also been studied.

2 Preliminaries

We denote the set of non-negative integers by ω, i.e., $\omega = \{0, 1, 2, 3, ...\}$. For a finite alphabet Σ, we use Σ^*, Σ^+, and Σ^ω to denote the set of finite, non-empty finite, and infinite sequences over Σ, respectively. An infinite *word* $\alpha : \omega \to \Sigma$ is an infinite sequence of letters $\alpha_0\alpha_1\alpha_2 \cdots$ from Σ. We use $[k]$ to represent $\{1, 2, \ldots, k\}$.

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 with a non-empty subset $I \subseteq Q$ of initial states, Σ is a finite alphabet, $T \subseteq Q \times \Sigma \times Q$ is a transition relation that maps states and input letters to sets of successor states,

and $\mathcal{F}$ is an acceptance condition. 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 $\mathsf{infin}(\rho) = \{q \in Q \mid \forall i \in \omega\ \exists j > i \text{ such that } \rho(j) = q\}$ denote the set of all states that occur infinitely often during the run ρ. Likewise, let $\mathsf{infin}(\overline{\rho}) = \{t \in T \mid \forall i \in \omega\ \exists j > i \text{ such that } \overline{\rho}(j) = t\}$ denote the set of all transitions that are taken infinitely many times in ρ.

In this paper, we use acceptance conditions over transitions. Acceptance mechanisms over states can be defined accordingly. *Rabin* automata are ω-automata, whose acceptance is defined by 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 $\mathsf{infin}(\overline{\rho}) \cap A_i \neq \emptyset = \mathsf{infin}(\overline{\rho}) \cap R_i$. *Streett* automata are ω-automata, whose 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.

Parity automata are ω-automata, whose acceptance is defined by a priority function $\mathsf{pri} : T \to [c]$ for some $c \in \mathbb{N}$. 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 is even. Parity automata can be viewed as special Rabin or Streett automata. In older works, the parity condition was referred to as Rabin chain: one can represent it by choosing A_i and R_i as the set of transitions with priority $\leq 2i$ and $\leq 2i-1$, respectively. This results in a chain $A_i \subseteq R_i \subseteq A_{i+1} \subseteq \ldots$.

One-pair Rabin automata $\mathcal{R}_1 = \big(Q, \Sigma, I, T, (A, R)\big)$ are of special technical interest in this paper. They are Rabin automata with a singleton index set, such that we directly refer to the only pair (A, R). *Büchi* automata can be viewed as one-pair Rabin automata with an empty set of rejecting states $R = \emptyset$.

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

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. 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 $\delta : (q, \alpha) \mapsto r \Leftrightarrow (q, \alpha, r) \in T$.

3 Determinisation

We will tackle the determinisation of parity automata in three steps. Firstly, we will recall history trees, the data structure for determinising Büchi automata. Secondly, we will adjust this data structure and adapt the Büchi determinisation

procedure to determinise one-pair Rabin automata. Finally, we will show that this data structure can be nested for the determinisation of parity automata.

In [15,18], we use ordered labelled trees to depict the states of the deterministic automaton. The ordered labelled trees are called *history trees* in [15,18].

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

Determinising One-Pair Rabin Automata. For one-pair Rabin automata, it suffices to adjust this data structure slightly. 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, $\forall v \in \mathcal{T} \setminus \{\varepsilon\}\ \exists q \in l(v)\ \forall v' \in \mathcal{T}.\ v = \mathsf{pred}(v') \Rightarrow q \notin l(v')$, and the label of the root ε *equals* the union of its children's labels, $l(\varepsilon) = \bigcup \{l(v) \mid v \in \mathcal{T} \cap \omega\}$.

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 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 (defined later) refer to this node.

Transition Mechanism. We determine $\Delta{:}\big((\mathcal{T}, l), \sigma\big) \mapsto (\mathcal{T}', l')$ 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.* 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 $d' = \big(\mathcal{T}', l'\big)$ 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.) The correctness proof of this construction follows the same lines as the proof of the Büchi construction.

Lemma 1. $L(\mathcal{R}_1) \subseteq L(\mathcal{D}_1)$

Notation. For a state q of $\mathcal{R}_1$ and an RHT $d = (\mathcal{T}, l)$, 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.

The *proof idea* is the same as for Büchi determinisation [15]: 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 $d_0, d_1 \dots$ be the run of $\mathcal{D}_1$ on α and $q_0, q_1, \dots$ an accepting run of $\mathcal{R}_1$ on α. We then define the sequence $v_0, v_1, \dots$ with $v_i = \mathsf{host}(q_i, d_i)$, which contains a longest eventually stable prefix v.

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 v is eventually 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 contradicts the assumption that v is the *longest* stable prefix.

Lemma 2. $L(\mathcal{D}_1) \subseteq L(\mathcal{R}_1)$

For the run $d_0 d_1 \dots$ of $\mathcal{D}_1$ on α, we fix an ascending chain $i_0 < i_1 < i_2 \dots$ of indices, such that v is not rejecting in any transition $(d_{j-1}, \alpha(j-1), d_j)$ for $j \geq i_0$ and such that $(d_{i_j - 1}, \alpha(i_j - 1), d_{i_j}) \in A_v$ for all $j \geq 0$. The proof idea is the usual way of building a tree of initial sequences of runs: we build a tree of initial sequences of runs of $\mathcal{R}_1$ that contains a sequence $q_0 q_1 q_2 \dots q_{i_j}$ 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 $i \in [i_k, i_{k+1}[$ s.t. $(q_i, \alpha(i), q_{i+1}) \in A$ is accepting.

This infinite tree has an infinite branch by König's Lemma. By construction, this branch is an accepting run of $\mathcal{R}_1$ on α.

Corollary 1. $L(\mathcal{R}_1) = L(\mathcal{D}_1)$.

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 subtree 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 [15], we get:

Theorem 1. $\inf\{c \mid \#\mathsf{rht}(n) \in O((cn)^n)\} = \inf\{c \mid \#\mathsf{ht}(n) \in O((cn)^n)\} \approx 1.65$.

The full version shows that $\#\mathsf{rht}(n)$ is only a constant factor bigger than $\#\mathsf{ht}(n)$.

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 $\mathcal{P} = (Q, \Sigma, I, T, \mathsf{pri} : T \to [c])$, and we select $e = 2\lfloor 0.5c \rfloor$.

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*, iff it ends in $\mathfrak{s}$. The root ε is called a Rabin root iff $c > e$. A node $v \in \mathcal{T}$ is called a *base node* iff 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 iff 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: $\forall v{\in}\mathcal{T} \setminus \mathsf{base}(\mathcal{T})$
  $l(v) = \{q \in l(v') \mid v' \in \mathcal{T} \text{ and } v = \mathsf{pred}(v')\}$ 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 c) a nesting of RHTs. Indeed, for $c = 3$, we simply get the RHTs, and λ is the constant function with domain $\{2\}$. For odd $c > 3$, 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 $c - 3$.

The transition mechanism from the previous subsection 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 $R_a = \{t \in T \mid \mathsf{pri}(t) > a \text{ and } \mathsf{pri}(t) \text{ is odd}\}$; 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, $N_a = T \setminus R_a$.

Construction. Let $\mathcal{P} = \big(P, \Sigma, I, T, \{\mathsf{pri} : P \to [c]\big)$ be a nondeterministic parity automaton with $|P| = n$ states. We construct a language equivalent deterministic Rabin automaton $\mathcal{DR} = (D, \Sigma, d_0, \Delta, \{(A_i, R_i) \mid i \in J\})$ where,

- D is the set of NHTs over P (i.e., with $l(\varepsilon) \subseteq P$) whose root has level e, where $e = c$ if c is even, and $e = c - 1$ if c 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 we obtain an NHT.
- 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. 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 c, 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 *no 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.* We spawn new children for every node in the NHT. For nodes $v \in \mathcal{T}$ that are no 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 no 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 $\mathcal{DR}$ will be in A_v for exactly the $v \in \mathcal{T}_v \cap \mathcal{T}_e$. Note that Rabin roots 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 $\mathcal{T}_o = \mathsf{rename}(\mathcal{T}_r)$, we update a pair $(\mathcal{T}_r, l_2)$ from Step 5 to $d' = \big(\mathcal{T}_o, l'\big)$ 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 $d' = (\mathcal{T}', l', \lambda')$.

Lemma 3. $L(\mathcal{P}) \subseteq L(\mathcal{DR})$

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

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.

Lemma 4. $L(\mathcal{DR}) \subseteq L(\mathcal{P})$

The proof of this lemma is essentially the proof of Lemma 2 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.

Corollary 2. $L(\mathcal{P}) = L(\mathcal{DR})$.

Determinising to a Deterministic Parity Automata $\mathcal{D}$. Deterministic parity automata seem to be a nice target when determinising ω-automata [9,15,18] given that algorithms that solve parity games (e.g., for acceptance games of alternating and emptiness games of nondeterministic tree automata) have a lower complexity when compared to solving Rabin games. For applications that involve co-determinisation, the parity condition also avoids the Streett condition.

Safra's determinisation construction (and younger variants) intuitively enforces a parity-like order on the nodes of history trees. By storing the order in which nodes are introduced during the construction, we can capture the Index Appearance Records construction that is used to convert Rabin or Streett automata to parity automata. To achieve this, we augment the states of the deterministic automaton (RHTs or NHTs) with a *later introduction record* (LIR), an abstraction of the order in which the non-Rabin roots of the ordered trees are introduced. (Rabin roots provide redundant information and are omitted.)

For an ordered tree $\mathcal{T}$ with m nodes that are no Rabin roots, an LIR is a sequence $v_1, v_2, \dots v_m$ that contains the nodes of $\mathcal{T}$ that are no Rabin roots nodes, such that, each node appears after its ancestors and older siblings. 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, c_v, P_v) where $S_v = l(v)$, is the label of v, $c_v = \lambda(v)$ the level of v, and $P_v = \{q \in Q \mid v = \mathsf{host}_{c_v}(q, d)\}$ is the set of states c_v hosted by v. The v can be reconstructed by the order and level. We call the possible sequences of these triples *LIR-NHTs*. Obviously, each LIR-NHT defines an NHT, but not the other way round.

A finite sequence $(S_1, c_1, P_1)(S_2, c_2, P_2)(S_3, c_3, P_3) \dots (S_k, c_k, P_k)$ of triples is a LIR-NHT if it satisfies the following requirements for all $i \in [k]$.

1. $P_i \subseteq S_i$,
2. $\{P_i\} \cup \{S_j \mid j{>}i,\ c_i{=}c_j$, and $S_j \cap S_i \neq \emptyset\}$ partitions S_i.
3. $\{S_j \mid j > i,\ c_i = c_j + 2$, and $S_j \cap P_i \neq \emptyset\}$ partition P_i.
4. If the highest priority of $\mathcal{P}$ is even, then $c_i = e$ implies $S_i \subseteq S_1$. (Then, the lowest level construction is Büchi and the first triple refers to the root.)
5. For $c_i < e$, 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 $N = (S_1, c_1, P_1)(S_2, c_2, P_2)(S_3, c_3, P_3) \dots$ 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 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 $ne + 1$.

Lemma 5. *Given an NPA $\mathcal{P}$ with n states and maximal priority c, we can construct a language equivalent deterministic parity automaton $\mathcal{D}$ with $ne + 1$ priorities for $e = 2\lfloor 0.5c \rfloor$, whose states are the LIR-NHTs described above.*

Lemma 6. *The DPA resulting from determinising a one-pair Rabin automaton $\mathcal{R}_1$ has $O(n!^2)$ states, and $O\big(n!(n-1!)\big)$ if $\mathcal{R}_1$ is Büchi.*

Lower Bound. We finally show that our determinisation to Rabin automata is optimal, and 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 to establish the

lower bound for determinisation to Rabin is similar to [3,18], in that we use the states of the Rabin automaton in a language game. Such 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 simple multi-graph (V, E). 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 from the edge to 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 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 2. *[3] If the verifier wins a language game for a language recognised by a DRA $\mathcal{R}$ with r states, then he wins it using a strategy with memory r.*

To prove optimality of determinising to Rabin, we expand the hardness proofs [3,18] based on *full automata* to the determinisation of parity automata.

The words from each vertex is the language for the Rabin automaton where we have some well defined progress, either some index which occurs accepting but not rejecting, or an index which occurs not rejecting with the target NHT either growing or shrinking.

It is easy to establish that the verifier wins these games. To prove that the minimal DRA that recognises the language of a full parity automaton cannot be smaller than a DRA obtained with our construction, we show that the verifier needs all edges to win these games. The techniques to establish this are similar to those for previous hardness proofs [3,18].

The surprising result is that this technique can be extended to establish optimality for determinisation *to* parity automata. This can be achieved by aiming for Streett automata that accept the complement language. This technique is successful, because the parity construction proves to be optimal (up to a factor of 1.5) for determinisation to Streett automata.

Theorem 3. *The DPA that we construct has less than* 1.5 *times as many states as the smallest deterministic Streett (or parity) automaton that recognises the language of an NPA recognising the hardest possible language.*

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:

the known upper bound for determinising Büchi to parity automata [15] are tight and Piterman's algorithm for it [9] is **optimal** modulo a factor of $3n$, where $2n$ stem from the fact that [9] uses state based acceptance. With Lemma 6 we get:

Corollary 3. *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.*

References

1. Boker, U., Kupferman, O.: Co-ing Büchi made tight and useful. In: Proc. of LICS, pp. 245–254 (2009)
2. Cai, Y., Zhang, T.: Tight upper bounds for Streett and parity complementation. In: Proc. of CSL, pp. 112–128 (2011)
3. Colcombet, T., Zdanowski, K.: A tight lower bound for determinization of Büchi automata. In: Albers, S., Marchetti-Spaccamela, A., Matias, Y., Nikoletseas, S., Thomas, W. (eds.) ICALP 2009, Part II. LNCS, vol. 5556, pp. 151–162. Springer, Heidelberg (2009)
4. Duret-Lutz, A.: LTL translation improvements in SPOT. In: VECoS, pp. 72–83. BCS (2011)
5. Finkbeiner, B., Schewe, S.: Uniform distributed synthesis. In: Proc. of LICS 2005, pp. 321–330 (2005)
6. Finkbeiner, B., Schewe, S.: Bounded synthesis. International Journal on Software Tools for Technology Transfer, online-first: 1–12 (2012)
7. Kupferman, O., Vardi, M.Y.: Synthesizing distributed systems. In: Proc. of LICS 2001, pp. 389–398 (2001)
8. Kupferman, O., Vardi, M.Y.: Safraless decision procedures. In: Proc. of FOCS 2005, pp. 531–540 (2005)
9. Piterman, N.: From nondeterministic Büchi and Streett automata to deterministic parity automata. Journal of LMCS 3(3:5) (2007)
10. Pnueli, A., Rosner, R.: Distributed reactive systems are hard to synthesize. In: Proc. of FOCS 1990, pp. 746–757 (1990)
11. Rabin, M.O.: Decidability of second order theories and automata on infinite trees. Transaction of the AMS 141, 1–35 (1969)
12. Safra, S.: On the complexity of ω-automata. In: Proc. of FOCS 1988, pp. 319–327 (1988)
13. Safra, S.: Exponential determinization for omega-automata with strong-fairness acceptance condition. In: Proc. of STOC 1992, pp. 275–282 (1992)
14. Schewe, S.: Büchi complementation made tight. In: Proc. of STACS 2009, pp. 661–672 (2009)
15. Schewe, S.: Tighter bounds for the determinisation of büchi automata. In: de Alfaro, L. (ed.) FOSSACS 2009. LNCS, vol. 5504, pp. 167–181. Springer, Heidelberg (2009)
16. Schewe, S., Finkbeiner, B.: Satisfiability and finite model property for the alternating-time μ-calculus. In: Ésik, Z. (ed.) CSL 2006. LNCS, vol. 4207, pp. 591–605. Springer, Heidelberg (2006)
17. Schewe, S., Finkbeiner, B.: Synthesis of asynchronous systems. In: Puebla, G. (ed.) LOPSTR 2006. LNCS, vol. 4407, pp. 127–142. Springer, Heidelberg (2007)
18. Schewe, S., Varghese, T.: Tight bounds for the determinisation and complementation of generalised büchi automata. In: Chakraborty, S., Mukund, M. (eds.) ATVA 2012. LNCS, vol. 7561, pp. 42–56. Springer, Heidelberg (2012)

19. Vardi, M.Y.: The Büchi complementation saga. In: Thomas, W., Weil, P. (eds.) STACS 2007. LNCS, vol. 4393, pp. 12–22. Springer, Heidelberg (2007)
20. Wilke, T.: Alternating tree automata, parity games, and modal μ-calculus. Bull. Soc. Math. Belg. 8(2) (May 2001)
21. Yan, Q.: Lower bounds for complementation of omega-automata via the full automata technique. Journal of LMCS, 4(1:5) (2008)