From Nondeterministic Büchi and Streett Automata to Deterministic Parity Automata

N. Piterman*

October 19, 2005

Abstract

Determinisation and complementation are foundational notions in computer science. When considering finite automata on finite words nondeterminisation and complementation are one and the same. Given a nondeterministic finite automaton there exists an exponential construction that gives a deterministic automaton for the same language. Inverting the set of accepting states gives us the automaton for the complement language. In the theory of automata on infinite words determinisation and complementation are much more involved. Safra provides determinisation constructions for Büchi and for Streett automata that result in deterministic Rabin automata. The construction starts with an automaton with n states (and index k for the Streett case) and results in a Rabin automaton with $n^{O(n)}$ states ($(nk)^{O(nk)}$ states and $O(nk)$ index for the Streett case).

Here, we reconsider Safra's determinisation constructions. We show how to construct automata with less states, smaller index, and most importantly, parity acceptance condition. Specifically, starting from a nondeterministic Büchi automaton with n states our construction yields an automaton with n^{2n+2} states (instead of $2^{5n}n^{2n}$), index $2n-1$ (instead of $4n+1$), and a parity automaton (instead of Rabin). Starting from a nondeterministic Streett automaton with n states and index k our construction yields an automaton with $n^{n(k+2)+2}(k+1)^{2n(k+1)}$ states (instead of $2^{5n(k+1)}n^{n(k+2)}(k+1)^{2n(k+1)}$), index $2n(k+1)$ (instead of $4n(k+1)+1$), and a parity automaton (instead of Rabin). For every practical application, the fact that our automaton is parity saves an additional multiplier of $n^2(2n)!$ in the case of Büchi and $(2n(k+1))^2(2n(k+1))!$ in the case of Streett.

*EPFL - I&C - MTC, Lausanne, Switzerland. Email:firstname.lastname@epfl.ch

1 Introduction

One of the fundamental questions in the theory of automata is the difference between determinism and nondeterminism. Another interesting question is the question of complementation. That is, given some machine (in some complexity class) can we produce a machine (in the same class) that accepts the complement language. The problems of determinisation and complementation are strongly related. Indeed, if the machine is deterministic we just have to invert its answer. If the machine is nondeterministic we do not have a simple solution.

In the theory of finite automata on finite words, however, these two questions are resolved. We know that there exists an efficient procedure that gets a nondeterministic automaton and constructs a deterministic automaton accepting the same language [RS59]. This construction takes a nondeterministic automaton with n states and produces a deterministic automaton with 2^n states. The construction is also tight, i.e., there is a family of languages $\{L_n\}_{n\in\mathcal{N}}$ such that L_n can be recognized by a nondeterministic automaton with n states and the minimal deterministic automaton recognizing L_n has 2^n states (cf. [HMU00]). This also solves the problem of complementation. As mentioned, given a deterministic automaton one can obtain a deterministic automaton for the complement language by complementing the set of accepting states. In order to complement a nondeterministic automaton one first constructs a deterministic automaton for the same language and then complements the deterministic automaton. This construction is again tight (i.e., cannot be improved from 2^n).

In the case of finite automata on infinite words determinisation and complementation are more involved. In his proof that satisfiability of S1S is decidable, Büchi uses nondeterministic Büchi automata [Büc62]. As part of the satisfiability procedure he gives a nonelementary complementation construction for nondeterministic Büchi automata. Thus proving that the class of ω-regular languages is closed under complementation. When considering determinisation of automata on infinite words, the first problem is that the Büchi condition is no longer strong enough. That is, there are languages recognized by nondeterministic Büchi automata which cannot be recognized by deterministic Büchi automata. In order to consider determinisation for automata on infinite words we have to consider more general acceptance conditions. McNaughton showed a determinisation construction that is doubly exponential and results in an automaton with the Müller acceptance condition [McN66]. Sistla, Vardi, and Wolper suggested a single exponential complementation construction [SVW85], however with a quadratic exponent. Safra gives a determinisation construction which takes a nondeterministic Büchi automaton with n states and returns a deterministic Rabin automaton with at most $(2n)^{3n}$ states and $2n$ pairs [Saf89]. Michel showed that this is essentially optimal and that the best upper bound for determinisation and complementation is $n!$ [Mic88, Löd98]. This was followed by a very elegant construction of Kupferman and Vardi that complements a nondeterministic Büchi automaton without first determinizing it [KV98]. This construction was recently improved to give a complement automaton with at most $(1.06n)^n$ states [FKV04], which is currently the best available complementation construction.

As mentioned, in order to determinize automata on infinite words we have to consider stronger acceptance conditions. Moreover, complementing the acceptance condition (similar to complementing the set of accepting states for automata on finite words) may result in a different acceptance condition. For example, the complement of a Rabin condition is Streett, the complement of a Büchi condition is co-Büchi. On the other hand, the complement of a parity condition (resp. Müller) condition is again parity (resp. Müller). Translations among different acceptance conditions involve exponential constructions. For example, in order to translate a Müller condition to parity condition, one has to add a component with $n!$ states where n is the number of states of the original automaton. In a similar way, in order to translate Streett or Rabin conditions to parity conditions, one has to add a component with $k!$ states k is the number of pairs of the Streett or Rabin condition.

Safra determinisation handles Büchi automata. It can be used also for determinisation of Rabin and parity automata. Given a Rabin automaton with n states and k pairs there exists a nondeterministic Büchi automaton with nk states. The resulting deterministic Rabin automaton has $2^{5nk}(nk)^{2nk}$ states and $2nk$ pairs. Similarly, starting with a parity automaton we construct a nondeterministic Büchi automaton with $\frac{nk}{2}$ states. This solution does not work for Streett automata. The best conversion of a nondeterministic Streett automaton with n states and index k results in a nondeterministic Büchi automaton with $n2^k$ states [Cho74], which is optimal [SV89]. Thus, using Safra's determinisation for Streett automata results in a doubly exponential deterministic automaton. In order to handle Streett automata, Safra generalized his determinisation construction [Saf92]. Given a Streett automaton with n states and index k he constructs a Rabin automaton with $nk^{O(nk)}$ states and $O(nk)$ pairs. As Streett automata are more general that Büchi automata, the lower bound shows that this is essentially optimal.

For automata on words, determinisation is more general than complementation. Indeed, Rabin uses McNaughton's determinisation of word automata to complement nondeterministic Rabin tree automata [Rab72]. He uses this complementation in order to prove that satisfiability of S2S is decidable [Rab72].[1] As exemplified by Rabin's result, deterministic automata are used to reason about tree automata [Rab72, Tho90, Var98]. Conversion of alternating tree automata to nondeterministic tree automata uses determinisation as a subroutine and solves complementation of nondeterministic tree automata. Deterministic automata are used also for solving games. In order to solve a game in which the goal is an LTL formula, one first converts the LTL formula to a deterministic automaton and then solves the resulting Rabin game [PR89] (cf. also [dAHM01, ATM03]). We hint that some of these constructions use co-determinisation, that is construct a deterministic automaton for the complement language (see below).

[1] This is essentially the same use that Büchi had for the complementation of Büchi automata. In the context of tree automata one has to use a more general acceptance condition.

One problem with the Safra construction is that it results in a Rabin automaton. Rabin acceptance condition are more expressive than Büchi but this expressiveness comes at a price. Solving emptiness of nondeterministic Rabin tree automata (equivalently, Rabin games) is NP-complete [EJ88]. In order to complement a Rabin acceptance condition we have to use Streett acceptance conditions, which are co-NP-complete. Thus, the complement of a deterministic Rabin automaton is a deterministic Streett automaton. Even if we ignore the computational difficulty, the Rabin acceptance condition allows using memoryless strategies. That is, when reasoning about Rabin games (or Rabin tree automata) the way to resolve nondeterminism relies solely on the current location. This is not the case for Streett. Indeed, in order to solve Streett games we require exponential memory [DJW97, Hor05]. This means, that when we use co-determinisation we end up with a deterministic Streett automaton, which is less convenient in practice. As mentioned, converting a nondeterministic Streett word automaton to a nondeterministic Büchi word automaton involves a blow up of 2^k states where k is the Streett index. In order to convert the Streett acceptance condition to Rabin or parity in the context of tree automata or games one has to endure a blow up of $k^2k!$ [Saf92].

In this paper we consider again Safra's determinisation constructions. We show that we can further compact the tree structure used by Safra to get a smaller representation of the deterministic automata. Furthermore, we can reduce the index of the resulting automaton, and most important, we can construct directly a deterministic parity automaton. Specifically, when starting from a nondeterministic Büchi automaton with n states, we end up with a deterministic automaton with n^{2n+2} states (instead of $2^{5n}n^{2n}$), parity index $2n-1$ (instead of $4n+1$), and most important, our automaton is a parity automaton. This means that the complement is exactly the same automaton with a dual parity acceptance condition. In particular, for the case of co-determinisation we get n^{2n+2} states instead of $2^{5n}n^{2n+2}(2n)!$ states. When starting from a Streett automaton with n states and index k, we end up with a deterministic automaton with $n^{n(k+2)+2}(k+1)^{2n(k+1)}$ states (instead of $2^{5n(k+1)}n^{n(k+2)}(k+1)^{2n(k+1)}$), parity index $2n(k+1)-1$ (instead of $4n(k+1)+1$, and most important, our automaton is a parity automaton. Again, in the case of co-determinisation our number of states does not change while the classical construction has to be multiplied by $(2n(k+1))^2(2n(k+1))!$.

Recently, Kupferman and Vardi showed that they can check the emptiness of an alternating parity tree automaton without directly using Safra's determinisation [KV05]. Notice that they do not convert the alternating automaton to a nondeterministic one, they only solve the emptiness problem. Their construction can be used for most game / tree automata applications that require determinisation. However, Kupferman and Vardi use Safra's determinisation to get a bound on the size of the minimal model of the alternating tree automaton. Given such a bound, they can search for such models by constructing simpler nondeterministic tree automata. Using our improved bound the algorithm of Kupferman and Vardi is improved from $2^{10n^2}n^{4n^2}(2n!)$ to n^{4n^2}.

2 Preliminaries

Given a finite set Σ, a *word* over Σ is a finite or infinite sequence of symbols from Σ. We denote by Σ^* the set of finite sequences over Σ and by Σ^ω the set of infinite sequences over Σ. Given a word $w = \sigma_0\sigma_1\sigma_2 \cdots \in \Sigma^* \cup \Sigma^\omega$, we denote by $w[i, j]$ the word $\sigma_i \cdots \sigma_j$.

2.1 Nondeterministic Automata

A *nondeterministic automaton* is $N = \langle \Sigma, S, \delta, s_0 F\rangle$, where Σ is a finite alphabet, S is a finite set of states, $\delta : S \times \Sigma \to 2^S$ is a transition function, $s_0 \in S$ is an initial state, and F is an acceptance condition to be defined below. A *run* of N on a word $w = w_0w_1 \cdots$ is an infinite sequence of states $s_0s_1 \ldots \in S^\omega$ such that $s_0 \in S_0$ and forall $j \geq 0$ we have $s_{j+1} \in \delta(s_j, w_j)$. For a run $r = s_0s_1 \ldots$, let $inf(r) = \{s \in S \mid s = s_i \text{ for infinitely many } i\text{'s}\}$ be the set of all states occurring infinitely often in the run. We consider four acceptance conditions. A *Rabin* condition F is a set of pairs $\{\langle E_1, F_1\rangle, \ldots, \langle E_k, F_k\rangle\}$ where forall i we have $E_i \subseteq S$ and $F_i \subseteq S$. We call k the *index* of the Rabin condition. A run is *accepting* according to the Rabin condition F if there exists some i such that $inf(r) \cap E_i = \emptyset$ and $inf(r) \cap F_i \neq \emptyset$. That is, the run visits finitely often states from E_i and infinitely often states from F_i. The *Streett* condition is the dual of the Rabin condition. Formally, a *Streett* condition F is also a set of pairs $\{\langle R_1, G_1\rangle, \ldots, \langle R_k, G_k\rangle\}$ where forall i we have $R_i \subseteq S$ and $G_i \subseteq S$. We call k the *index* of the Streett condition. A run is *accepting* according to the Streett condition F if for every i either $inf(r) \cap G_i = \emptyset$ or $inf(r) \cap R_i \neq \emptyset$. That is, the run either visits G_i finitely often or visits R_i infinitely often. As a convention for pairs in a Rabin condition we use E and F and for pairs in a Streett condition we use R and G. A *parity* condition F is a partition $\{F_0, \ldots, F_k\}$ of S. We call k the *index* of the parity condition. A run is *accepting* according to the parity condition F if for some even i we have $inf(r) \cap F_i \neq \emptyset$ and forall $i' < i$ we have $inf(r) \cap F_{i'} = \emptyset$. A *Büchi* condition F is a subset of S. A run is *accepting* according to the Büchi condition F if $inf(r) \cap F \neq \emptyset$. That is, the run visits infinitely often states from F. A word w is *accepted* by N if there exists some accepting run of N over w. The *language* of N is the set of words accepted by N. Formally, $L(N) = \{w \mid w \text{ is accepted by } N\}$. Two automata are *equivalent* if they accept the same language.

Given a set of states $S' \subseteq S$ and a letter $\sigma \in \Sigma$, we denote by $\delta(S', \sigma)$ the set $\bigcup_{s \in S'} \delta(s, \sigma)$. Similarly, for a word $w \in \Sigma^*$ we define $\delta(S', w)$ in the natural way: $\delta(S', \epsilon) = S'$ and $\delta(S', w\sigma) = \delta(\delta(S', w), \sigma)$. For two states s and t and $w \in \Sigma^*$, we say that t is *reachable from s reading w* if $t \in \delta(\{s\}, w)$.

An automaton is *deterministic* if for every state $s \in S$ and letter $\sigma \in \Sigma$ we have $|\delta(s, \sigma)| = 1$. In that case we write $\delta : S \times \Sigma \to S$. We use deterministic automata to *complement* a word automaton, i.e., construct an automaton that accepts the complement language. We can use deterministic automata also as *monitors* in games (see below). *Determinisation* for automata on finite words is relatively simple [RS59]. For automata on infinite words this is not the case.

Deterministic Büchi automata are strictly weaker than nondeterministic Büchi automata. However, for every nondeterministic Büchi automaton there exists an equivalent deterministic automaton with one of the stronger acceptance conditions. The best determinisation constructions take nondeterministic Büchi or Streett automata and converts them to deterministic Rabin automata. We describe these two constructions below.

We denote automata by acronyms in $\{N, U, D\} \times \{R, S, P, B\} \times \{T, W\}$. The first symbol stands for the branching mode of the automaton: N for nondeterministic, U for universal, and D for deterministic. The second symbol stands for the acceptance condition of the automaton: R for Rabin, S for Streett, P for parity, and B for Büchi. The last symbol stands for the object the automaton is reading: T for trees and W for words. For example, a DRW is a deterministic Rabin word automaton and a NBT is a nondeterministic Büchi tree automaton.

2.2 Safra's Determinisation Construction

Here we describe Safra's determinisation construction [Saf89]. The construction takes an NBW and constructs an equivalent DRW. This section is largely copied from [Jut97, Löd98].

Let $\mathcal{N} = \langle \Sigma, S, \delta, s_0, \mathcal{F} \rangle$ be an NBW with $|S| = n$. Let $V = [2n]$. We first define Safra trees.

A *Safra tree* t over S is $\langle N, r, p, \psi, l, E, F \rangle$ where the components of t are as follows.

- $N \subseteq V$ is a set of nodes.
- $r \in N$ is the root node.
- $p : N \to N$ is the parent function defined over $N - \{r\}$, defining for every $v \in N - \{r\}$ its parent $p(v)$.
- ψ is a partial order defining "older than" on siblings (i.e., children of the same node).
- $l : N \to 2^S$ is a labeling of the nodes with subsets of S. The label of every node is a proper superset of the union of the labels of its sons. The labels of two siblings are disjoint.
- $E, F \subseteq V$ are used to define the Rabin acceptance condition.

The following claim is proven in [Saf89, Jut97].

Claim 2.1 *The number of nodes in a Safra tree is at most n. The number of Safra trees over $\mathcal{N}$ is not more than $2^{5n}n^{2n}$.*

Proof: As the labels of siblings are disjoint and the union of labels of children is a proper subset of the label of the parent it follows that every node is the minimal (according to the subset order on the labels) to contain (at least) some state $s \in S$. It follows that there are at most n nodes.

The number of ordered trees on n nodes is the nth Catalan number. That is $Cat(n) = \frac{(2n)!}{n!(n+1)!} \leq 4^n$. In addition we have to represent the names of the nodes. We represent the set of active names in the tree by a subset of V. There are at most 2^{2n} such subsets. We represent the naming of nodes by $f : [n] \rightarrow [n]$ that associates the ith node in the tree with the $f(i)$th used name. There are at most n^n such functions. The labeling function is $l : S \rightarrow [n]$ where $l(s) = i$ means that s belongs to the ith active node and all its ancestors. Finally, $E = V - N$ is the set of inactive names (see below) and F can be maintained as a subset of the active nodes, i.e., at most 2^n options.

To summarize, the number of trees is at most $2^{2n} \cdot 2^{2n} \cdot 2^n \cdot n^n \cdot n^n = 2^{5n} n^{2n}$.
□

We construct the DRW $\mathcal{D}$ equivalent to $\mathcal{N}$. Let $\mathcal{D} = \langle \Sigma, D, \rho, d_0, \mathcal{F}' \rangle$ where the components of $\mathcal{D}$ are as follows.

- D is the set of Safra trees over S.
- d_0 is the tree with a single node 1 labeled by $\{s_0\}$ where E is $V - \{1\}$ and F is the empty set.
- The Rabin acceptance condition $\mathcal{F}'$ is $\{\langle E_1, F_1 \rangle, \ldots, \langle E_{2n}, F_{2n} \rangle\}$ where $E_i = \{d \in D \mid i \in E_d\}$ and $F_i = \{d \in D \mid i \in F_d\}$.
- For every tree $d \in D$ and letter $\sigma \in \Sigma$ the transition $d' = \rho(d, \sigma)$ is the result of the following transformations on d.

  1. For every node v with label S' replace S' by $\rho(S', \sigma)$ and set E and F to the empty set.
  2. For every node v with label S' such that $S' \cap \mathcal{F} \neq \emptyset$, create a new node $v' \notin N$ which becomes the youngest child of v. Set its label to be $S' \cap \mathcal{F}$.
  3. For every node v with label S' and state $s \in S'$ such that s also belongs to the label of an older sibling v' of v, remove s from the label of v and all its descendants.
  4. Remove all nodes with empty labels.
  5. For every node v whose label is equal to the union of the labels of its children, remove all descendants of v. Add v to F.
  6. Add all unused names to E

Theorem 2.2 [Saf89] $L(\mathcal{D}) = L(\mathcal{N})$

For other expositions of this determinisation we refer the reader to [Jut97, Löd98, Rog01].

2.3 Safra's Determinisation of Streett Automata

Here we describe Safra's determinisation construction for Streett Automata [Saf92]. The construction takes an NSW and constructs an equivalent DRW. This section is largely copied from [Saf92, Jut97, Sch01].

Let $\mathcal{S} = \langle \Sigma, S, \delta, s_0, \mathcal{F} \rangle$ be an NSW where $\mathcal{F} = \{\langle R_1, G_1 \rangle, \ldots, \langle R_k, G_k \rangle\}$ and $|S| = n$. Let $m = n(k+1)$ and $V = [2m]$. We first define Streett Safra trees.

A *Streett Safra tree* t over S is $\langle N, r, p, \psi, l, h, E, F \rangle$ where the components of t are as follows.

- $N \subseteq V$ is the set of nodes.
- $r \in N$ is the root node.
- $p : N \to N$ is the parent function defined over $N - \{r\}$, defining for every $v \in N - \{r\}$ its parent $p(v)$.
- ψ is a partial order defining "older than" on siblings (i.e., children of the same node).
- $l : N \to 2^S$ is a labeling of nodes with subsets of S. The label of every node is equal to the union of the labels of its sons. The labels of two siblings are disjoint.
- $h : N \to 2^{[k]}$ annotates every node with a set of indices from $[k]$. The root is annotated by $[k]$. The annotation of every node is contained in that of its parent and it misses at most one element from the annotation of the parent. Every node that is not a leaf has at least one son with strictly smaller annotation.
- $E, F \subseteq V$ are used to define the Rabin acceptance condition.

The following claim is proven in [Saf92, Sch02].

Claim 2.3 *The number of nodes in a Streett Safra tree is at most $n(k+1)$. The number of Streett Safra trees over $\mathcal{S}$ is not more than $2^{5n(k+1)} n^{n(k+2)} (k+1)^{2n(k+1)}$.*

Proof: The labeling is determined by the labels of the leaves. As the labels of all leaves are disjoint there are at most n leaves. We can represent the annotation h by annotating every edge by the value $j \in [k]$ such that j is in the annotation of the parent and not in the annotation of the son. If no such j exists then we annotate the edge by 0. There exists a path from the root to a leaf all whose edges are not annotated by 0. For every edge annotated 0, there is a path from the target of this edge to a leaf all whose edges are not annotated by 0. Hence, there are at most $n-1$ edges annotated by 0. Every other edge is corresponds to some index $i \in [k]$ or some state $s \in S$ so there can be at most nk such edges. Totally, there are at most $n(k+1)$ nodes.

The number of ordered trees on m nodes is at most 4^m. We represent the set of active names in the tree by a subset of V. There are at most 2^{2m} such subsets. We represent the naming of the nodes by $f : [m] \to [m]$. There are at most m^m such functions. The labeling function $S \to [n]$ associates a state s with the leaf it belongs to (the label of every node is determined by the labels of the leaves in the subtree below it). There are at most n leaves. The edge annotation function is $h : [m] \to [0..k]$ associating an index to the target node of the edge. Finally, $E = V - N$ is the set of inactive nodes and F can be maintained as a subset of the active nodes, i.e., at most 2^m options.

To summarize, the number of trees is at most $2^{2m} \cdot 2^{2m} \cdot 2^m \cdot m^m \cdot n^n \cdot (k+1)^m = 2^{5n(k+1)} n^{n(k+2)} (k+1)^{2n(k+1)}$. □

We construct the DRW $\mathcal{D}$ equivalent to $\mathcal{S}$. Let $\mathcal{D} = \langle \Sigma, D, \rho, d_0, \mathcal{F}' \rangle$ where the components of $\mathcal{D}$ are as follows.

- D is the set of Streett Safra trees over S.
- d_0 is the tree with a single node 1 labeled by $\{s_0\}$ where E is $V - \{1\}$ and F is the empty set.
- The Rabin acceptance condition $\mathcal{F}'$ is $\{\langle E_1, F_1 \rangle \ldots, \langle E_{2m}, F_{2m} \rangle\}$ where $E_i = \{d \in D \mid i \in E_d\}$ and $F_i = \{d \in D \mid i \in F_d\}$.
- For every tree $d \in D$ and letter $\sigma \in \Sigma$ the transition $d' = \rho(d, \sigma)$ is the result of the following (recursive) transformation applied on d starting from the root. Before we start, we set E and F to the empty set and replace the label of every node v by $\delta(l(v), \sigma)$.
  1. If v is a leaf such that $h(v) = \emptyset$ stop.
  2. If v is a leaf such that $h(v) \neq \emptyset$, add to v a new youngest son v'. Set $l(v') = l(v)$ and $h(v') = h(v) - \{max(h(v))\}$.
  3. Let $v_1, \ldots, v_l$ be the sons of v (ordered from oldest to youngest) and let $j_1, \ldots j_l$ be the indices such that $j_i \in h(v) - h(v_i)$ (note that $|h(v) - h(v_i)| \leq 1$; in case that $h(v) = h(v_i)$ we have $j_i = 0$). Apply the procedure recursively on $v_1, \ldots, v_l$ (including sons created in step 2 above).

     For every son v_i and every state $s \in l(v_i)$ do the following.
     (a) If $s \in G_{j_i}$, remove s from the label of v_i and all its descendants. Add a new youngest son v' to v. Set $l(v') = \{s\}$ and $h(v') = h(v) - \{max(h(v))\}$.
     (b) If $s \in R_{j_i}$, remove s from the label of v_i and all its descendants. Add a new youngest son v' to v. Set $l(v') = \{s\}$ and $h(v') = h(v) - \{max(h(v) \cap \{1, \ldots, j_i - 1\})\}$.
  4. If a state s appears in $l(v_i)$ and $l(v_{i'})$ and $j_i < j_{i'}$ then remove s from the label of $v_{i'}$ and all its descendants.
  5. If a state s appears in $l(v_i)$ and $l(v_{i'})$ and $j_i = j_{i'}$ then remove s from the label of the younger sibling and all its descendants.

6. Remove sons with empty label.
7. If all sons are annotated by $h(v)$ remove all the sons and all their descendants. Add v to F.

Finally, we add all unused names to E and remove unused names from F.

Theorem 2.4 [Saf92] $L(\mathcal{D}) = L(\mathcal{N})$

For other expositions of this determinisation we refer the reader to [Jut97, Sch01].

3 From NBW to DPW

In this section we show how to improve Safra's determinisation construction. We simply close the gaps between the names of nodes in Safra's construction and get a deterministic automaton with less states. In addition, the resulting automaton is a parity automaton instead of Rabin and its index is half the index of the original.

Intuitively, we replace the constant node name with a dynamic one that decreases as nodes below it get erased from the tree (called number below). Using the new names we can give up the sibling relation. The smaller the name of a node the older it is. Furthermore, the names give a natural parity order on good and bad events. Erasing a node is a bad event (which forces all nodes with greater name to change their name). Finding that the label of some name is equal to the union of labels of its descendants is a good event. The key observation is that a node can change its name at most a finite number of times without being erased. It follows, that the names of all nodes that stay eventually in the tree get constant. Thus, bad events happen eventually only to nodes that get erased from the tree. Then we can monitor good events that happen to the nodes with constant names and insist that they happen infinitely often. Formally, we have the following.

Let $\mathcal{N} = \langle \Sigma, S, \delta, s_0, \mathcal{F} \rangle$ be an NBW with $|S| = n$. Here we distinguish between the set of nodes $V = [2n]$ of a tree and their numbers that range over $[n]$. This is important for the proofs however for practical applications we need only the numbers.

A *compact Safra tree* t over S is $\langle N, M, 1, p, l, e, f \rangle$ where the components of t are as follows.

- $N \subseteq V$ is a set of nodes.
- $M : N \to [n]$ is the numbering function.
- $1 \in N$ such that $M(1) = 1$ is the root node.
- $p : N \to N$ is the parent function.

- $l : N \to 2^S$ is a labeling of the nodes with subsets of S. The label of every node is a proper superset of the union of the labels of its sons. The labels of two siblings are disjoint.
- $e, f \in [n+1]$ are used to define the parity acceptance condition.

Notice that we give up the sibling relation and replace the sets E and F by numbers e and f. We require that the numbering M is a bijection from N to $[|N|]$. That is, the numbers of the nodes in N are consecutive starting from the root, which is numbered 1.

The following claim is proven much like the similar proof for Safra trees.

Claim 3.1 *The number of compact Safra trees over $\mathcal{N}$ is not more than n^{2n+2}.*

Proof: Just like Safra trees there are at most n nodes. We use only the numbers of the nodes. The parenthood relation is represented by a function $p : [n] \to [n]$. As in Safra trees, every node has at least one unique state in S that belongs to it. We add the function $l : S \to [n]$ that associates a state with the minimal node (according to the descendant order in the tree) to which it belongs. Finally, there are n options for e and f each. It follows that there are at most $n \cdot n \cdot n^n \cdot n^n = n^{2n+2}$ different compact Safra trees.[2] □

We construct the DPW $\mathcal{D}$ equivalent to $\mathcal{N}$. Let $\mathcal{D} = \langle \Sigma, D, \rho, d_0, \mathcal{F}' \rangle$ where the components of $\mathcal{D}$ are as follows.

- D is the set of compact Safra trees over S.
- d_0 is the tree with a single node 1 labeled $\{s_0\}$ and numbered 1 where $e = 2$ and $f = 1$.
- The parity acceptance condition $\mathcal{F}' = \langle F_0 \cdots, F_{2n-1} \rangle$ is defined as follows.
  - $F_0 = \{d \in D \mid f = 1\}$
  - $F_{2i+1} = \{d \in D \mid e = i + 2 \text{ and } f \geq e\}$
  - $F_{2i+2} = \{d \in D \mid f = i + 2 \text{ and } e > f\}$

  Note that we do not consider the case $e = 1$. In this case the label of the root is empty. This is a rejecting sink state.
- For every tree $d \in D$ and letter $\sigma \in \Sigma$ the transition $d' = \rho(d, \sigma)$ is the result of the following transformations on d.

  1. For every node v with label S' replace S' by $\delta(S', \sigma)$.

[2] We note that there is much ordering in the numbering of the nodes which we have not used. We know that the numbers respect the parenthood relation. If we add order to the sons of a node (which practically comes for free: the number of ordered trees on n nodes is 4^n and the number of unordered trees is 3^n) then the numbers respect this order as well.

2. For every node v with label S' such that $S' \cap \mathcal{F} \neq \emptyset$, create a new son $v' \notin N$ of v. Set its label to $S' \cap \mathcal{F}$. Set its number to the minimal value greater than all used numbers. We may have to use temporarily numbers in the range $[n+1..2n]$.
3. For every node v with label S' and state $s \in S'$ such that s belongs also to some sibling v' of v such that $M(v') < M(v)$, remove s from the label of v and all its descendants.
4. For every node v whose label is equal to the union of the labels of its children, remove all descendants of v. Call such nodes *green*. Set f to the minimum of $n+1$ and the numbers of green nodes. Notice that all nodes in $[n+1..2n]$ cannot be green.
5. Remove all nodes with empty labels. Set e to the minimum of $n+1$ and the numbers of removed nodes.
6. Let Z denote the set of nodes removed during the transformation. For every node v let $empty(v)$ be $|\{v' \in Z \mid M(v') < M(v)\}|$. That is, we count how many nodes have empty label and smaller number than the number of v. For every node v such that $l(v) \neq \emptyset$ we change the number of v to $M(v) - empty(v)$. The resulting numbers are consecutive again and in the range $[n]$.

We show that the two automata are equivalent. The proof is an adaptation of Safra's proof [Saf89].

Theorem 3.2 $L(\mathcal{D}) = L(\mathcal{N})$

Proof: Consider $w \in L(\mathcal{N})$. We have to show $w \in L(\mathcal{D})$. Let $r = s_0 s_1 \cdots$ be an accepting run of $\mathcal{N}$ on w. Let $r' = d_0 d_1 \cdots$ be the run of $\mathcal{D}$ on w and let $d_i = \langle N_i, M_i, 1, p_i, l_i, e_i, f_i \rangle$. It is simple to see that forall $i \geq 0$ we have $s_i \in l_i(1)$. If step 4 is applied infinitely often to node 1 (equivalently, $f = 1$ infinitely often, or during the transformation of the trees the label of 1 equals the labels of its sons) then r' visits F_0 infinitely often.

Otherwise, from some point onwards in r' we have step 4 is not applied to node 1. Let j_1 be this point. There exists a point $j' > j_1$ such that $s_{j'} \in \mathcal{F}$. It follows that forall $j > j'$ we have s_j belongs to some son v_1 of 1. Notice, that just like in Safra's case, the run r may start in some son of 1 and move to a son with a smaller number. However, this can happen finitely often and hence we treat v_1 as constant. The number $M(v_1)$ may decrease finitely often until it is constant. Let o_1 be such that forall $o > o_1$ we have $a_1 = M_o(v_1)$. As $M_o(v_1) = a_1$ forall $o > o_1$ it follows that $e_o > a_1$ forall $o > o_1$.

Suppose that step 4 is applied to v_1 infinitely often (equivalently, $f \leq a_1$ infinitely often). It follows that for every odd $p < 2a_1 - 2$ we have F_p is visited finitely often and either F_{2a_1-2} is visited infinitely often or there exists some even $p' < 2a_1 - 2$ such that F_p is visited infinitely often. In this case $\mathcal{D}$ accepts w.

Otherwise, step 4 is applied to v_1 finitely often. We construct by induction a sequence $v_1, \ldots, v_k$ such that eventually $v_1, \ldots, v_k$ do not change their numbers and r belongs to all of them. As the number of active nodes in a tree (nodes v such that $l(v) \neq \emptyset$) is bounded by n we can repeat the process only finitely often. Hence, w is accepted by $\mathcal{D}$.

In the other direction, consider $w \in L(\mathcal{D})$. Let $r' = d_0 d_1 \cdots$ be the accepting run of $\mathcal{D}$ on w where $d_i = \langle N_i, M_i, 1, p_i, l_i, f_i, e_i \rangle$. Let F_{2b} be the minimal set to be visited infinitely often. It follows that eventually always $e_i > b+1$ and infinitely often $f_i = b+1$.

We construct an infinite tree with finite branching degree. The root of the tree corresponds to the initial state of $\mathcal{N}$. Every node in the tree is labeled by some state of $\mathcal{N}$ and a time stamp i. An edge between the nodes labeled (s,i) and (t,j) corresponds to a run starting in s, ending in t, reading $w[i,j-1]$, and visiting $\mathcal{F}$. From König's lemma this tree contains an infinite branch. The composition of all the run segments in this infinite branch is an infinite accepting run of $\mathcal{N}$ on w.

We first prove two claims. The first, showing that all the states of $\mathcal{N}$ that appear in labels of nodes of a state of $\mathcal{D}$ are reachable from the initial state of $\mathcal{N}$. The second, proves that if for some $2j$ the set F_{2j} is visited in d_i and in $d_{i'}$ and no visit to $F_{j'}$ for $j' < 2j$ occurs between i and i', then there exists a node v such that $M_a(v) = j+1$ forall $i \leq a \leq i'$ and for every state s in $l_{i'}(v)$ we find a run segment of $\mathcal{N}$ that starts from some state of $l_i(v)$, visits $\mathcal{F}$, and ends in s.

Claim 3.3 *For every i, j, and every state $s \in l_i(j)$ we have s is reachable from s_0 reading $w[0,i-1]$.*

Proof: We prove the claim for all $j \geq 1$ by induction on i. Clearly, it holds for $i = 0$. Suppose that it holds for i. As $l_{i+1}(j) \subseteq \delta(l_i(j'), w_i)$ for some j' it follows that every state in $l_{i+1}(j)$ is reachable from s_0 reading $w[0,i]$. □

Claim 3.4 *Let $i < i'$ be two locations such that $d_i, d_{i'} \in F_{2j}$ for some j and forall $j' \leq 2j$ and forall $i < a < i'$ we have $d_a \notin F_{j'}$. Then there exists a node v such that $M_a(v) = j+1$ forall $i \leq a \leq i'$ and every state s in $l_{i'}(v)$ is reachable from some state in $l_i(v)$ reading $w[i,i'-1]$ with a run that visits $\mathcal{F}$.*

Proof: There exists some node v such that $M_i(v) = j+1$ (as $d_i \in F_{2j}$). By assumption, for every $j' < 2j$ the set $F_{j'}$ is not visited between i and i'. Hence, for every node v' such that $M_i(v) \leq j+1$ we have that $M_a(v') = M_i(v')$ forall $i \leq a \leq i'$. That is, between i and i' all nodes whose number is at most $j+1$ do not change their numbers. In particular, forall $i \leq a \leq i'$ we have $M_a(v) = j+1$. We show that for every $i < a < i'$ and every descendant v' of v, every state in $l_a(v')$ is reachable from some state in $l_i(v)$ along a run visiting $\mathcal{F}$. Consider some descendant v' of v appearing in d_{i+1} (there is at most one, it must exist as F_{2j} is not visited between i and i'). As $l_{i+1}(v') \subseteq \delta(l_i(v), w_i) \cap \mathcal{F}$ this is obviously true for $i+1$. Suppose it is true for a and prove for $a+1$. We know that for every descendant v' of v either $l_{a+1}(v') \subseteq \delta(l_a(v), w_a) \cap \mathcal{F}$ or for some descendant v'' of v we have $l_{a+1}(v') \subseteq \delta(l_a(v''), w_a)$ (v'' may be v'). As during the transformation from $d_{i'-1}$ to $d_{i'}$ the label $l_{i'}(v)$ equals the union of labels of sons of v the claim follows. □

We are now ready to build the tree T. Let $(s_0, 0)$ label the root of T. Let i_0 be the maximal location such that forall $j < 2b$ the set F_j is not visited after i_0. Let v be the node such that forall $i > i_0$ we have $M_i(v) = b + 1$. Let i_1 be the minimal location such that $i_1 > i_0$ and $f_{i_1} = b + 1$ (that is step 4 was applied to v). For every state s in $l_{i_1}(v)$ we add a node to T, label it by (s, i_1) and connect it to the root. We extend the tree by induction. We have a tree with leafs labeled by the states in $l_a(v)$ stamped by time a, and $f_a = b + 1$ (step 4 was applied to v). That is, for every state s in $l_a(v)$ there exists a leaf labeled (s, a). We know that F_{2b} is visited infinitely often. Hence, there exists $a' > a$ such that $f_{a'} = k + 1$ (step 4 is applied to v). For every state s' in $l_{a'}(v)$ we add a node to T and label it (s', a'). From Claim 3.3 it follows that every edge $(s_0, 0), (s', i')$ corresponds to some run starting in s_0, ending in s', and reading $w[0, i' - 1]$. From Claim 3.4, every other edge in the tree $(s, a), (s', a')$ corresponds to some run starting in s, ending in s', reading $w[a, a' - 1]$, and visiting $\mathcal{F}$. From König's lemma there exists an infinite branch in the tree. This infinite branch corresponds to an accepting run of $\mathcal{N}$ on w. □

Theorem 3.5 *For every NBW $\mathcal{N}$ with n states there exists a DPW with n^{2n+2} states and index $2n - 1$ that is equal to $\mathcal{N}$.*

We note that this improves Safra's construction in three ways. First, we reduce the number of states from $2^{5n}n^{2n}$ to n^{2n+2}. Second the index of our automaton is $2n - 1$ instead of $4n + 1$ (which results from a Rabin condition with $2n$ pairs). Finally, our automaton is a parity automaton which is amenable to simpler algorithms. Many times we are interested in a deterministic automaton for the complement language, a process called co-determinisation. The natural complement of a DRW is a DSW. However,the Streett acceptance condition is less convenient in many applications (due to the fact that Streett acceptance conditions require memory). Thus, the complement automaton is usually converted to a DPW using the IAR construction [Saf92]. In such a case, one would have to multiply the number of states by $(2n^2)(2n)!$. A similar effect occurs when using deterministic automata in the context of games. Solution of Rabin games incurs an additional multiplier of $(2n)^2(2n)!$. Obviously, with our construction this penalty is avoided.

4 From NSW to DPW

In this section we show how to improve Safra's determinisation for Streett automata. We simply close the gaps between the names of nodes in Safra's construction and get a deterministic automaton with less states. In addition, the resulting automaton is a parity automaton instead of Rabin and its index is half the index of the original. The intuition is similar to the construction in Section 3.

Let $\mathcal{S} = \langle \Sigma, S, \delta, s_0, \mathcal{F} \rangle$ be an NSW where $\mathcal{F} = \{\langle R_1, G_1 \rangle, \ldots, \langle R_k, G_k \rangle\}$ and $|S| = n$. Denote $m = n(k + 1)$. Here we distinguish between the set of nodes

$V = [2m]$ of a tree and their numbers that range over $[m]$. This is important for the proofs however for practical applications we need only the numbers.

A *compact Streett Safra tree* t over S is $\langle N, M, 1, p, l, h, e, f\rangle$ where the components of t are as follows.

- $N \subseteq V$ is a set of nodes.
- $M : N \to [m]$ is the numbering function.
- $1 \in N$ such that $M(1) = 1$ is the root node.
- $p : N \to N$ is the parent function.
- $l : N \to 2^S$ is a labeling of the nodes with subsets of S. The label of every node is equal to the union of the labels of its sons. The labels of two siblings are disjoint.
- $h : N \to 2^{[k]}$ annotates every node with a set of indices from $[k]$. The root is annotated by $[k]$. The annotation of every node is contained in that of its parent and it misses at most one element from the annotation of the parent. Every node that is not a leaf has at least one son with strictly smaller annotation.
- $e, f \in [m + 1]$ are used to define the parity acceptance condition.

Notice that we give up the sibling relation and replace the sets E and F by numbers e and f. The numbering M is a bijection from N to $[|N|]$. That is, the numbers of nodes in N are consecutive starting from the root, which is numbered 1.

The following claim is proven much like the similar proof for Streett Safra trees.

Claim 4.1 *The number of compact Streett Safra trees over $\mathcal{S}$ is not more than* $n^{n(k+2)+2}(k + 1)^{2n(k+1)}$.

Proof: Just like Streett Safra trees there are at most m nodes. We use only the numbers of the nodes. The parenthood relation is represented by a function $p : [m] \to [m]$. As the labels of the leaves form a partition of the set of states S there are at most n leaves. We add the function $l : S \to [n]$ that associates a state with the unique leaf to which it belongs. Setting $l(s) = i$ means that s belongs to the ith leaf. We can represent the annotation h by annotation every edge by the value $j \in [k]$ such that j is in the annotation of the parent and not in the annotation of the son. If no such j exists then we annotate the edge by 0. The edge annotation is represented by a function $h : [m] \to [0, \ldots, k]$. Finally, there are m options for e and f each.

It follows that there are at most $m \cdot m \cdot m^m \cdot n^n \cdot (k+1)^m = (n(k+1))^2(n(k+1))^{n(k+1)}n^n(k + 1)^{n(k+1)} = n^{n(k+2)+2}(k + 1)^{2n(k+1)}$ different compact Streett Safra trees. $\square$

We construct the DPW $\mathcal{D}$ equivalent to $\mathcal{S}$. Let $\mathcal{D} = \langle \Sigma, D, \rho, d_0, \mathcal{F}' \rangle$ where the components of $\mathcal{D}$ are as follows.

- D is the set of compact Streett Safra trees over $\mathcal{S}$.
- d_0 is the tree with a single node 1 labeled $\{s_0\}$, numbered 1, and annotated $[k]$. We set $e = 2$ and $f = 1$.
- The parity acceptance condition $\mathcal{F}' = \langle F_0, \ldots, F_{2m-1} \rangle$ is defined as follows.
  - $F_0 = \{d \in D \mid f = 1\}$
  - $F_{2i+1} = \{d \in D \mid e = i + 2 \text{ and } f \geq e\}$
  - $F_{2i+2} = \{d \in D \mid f = i + 2 \text{ and } e > f\}$

  As before, we do not handle the case where $e = 1$.
- For every tree $d \in D$ and letter $\sigma \in \Sigma$ the transition $d' = \rho(d, \sigma)$ is the result of the following (recursive) transformation applied on d starting from the root. Before we start, we set e and f to $m + 1$ and replace the label of every node v by $\delta(l(v), \sigma)$.
  1. If v is a leaf such that $h(v) = \emptyset$ stop.
  2. If v is a leaf such that $h(v) \neq \emptyset$, add to v a new son v'. Set $l(v') = l(v)$, $h(v') = h(v) - \{max(h(v))\}$, and set $M(v')$ to the minimal value greater than all used numbers. We may use temporarily numbers out of the range $[m]$.
  3. Let $v_1, \ldots, v_l$ be the sons of v (ordered according to their numbers) and let $j_1, \ldots, j_l$ be the indices such that $j_i = max((h(v) \cup \{0\}) - h(v_i))$ (note that $|h(v) - h(v_i)| \leq 1$; in case that $h(v) = h(v_i)$ we have $j_i = 0$). Apply the procedure recursively on $v_1, \ldots, v_l$ (including sons created in step 2 above).

     For every son v_i and every state $s \in l(v_i)$ do the following.
     (a) If $s \in G_{j_i}$, remove s from the label of v_i and all its descendants. Add a new son v' to v. Set $l(v') = \{s\}$, $h(v') = h(v) - \{max(h(v))\}$, and set $M(v')$ to the minimal value larger than all used numbers.
     (b) If $s \in R_{j_i}$, remove s from the label of v_i and all its descendants. Add a new son v' to v. Set $l(v') = \{s\}$, $h(v') = h(v) - \{max((J \cup \{0\}) \cap \{1, \ldots, j_i - 1\})\}$, and set $M(v')$ to the minimal value larger than all used numbers.
  4. If a state s appears in $l(v_i)$ and $l(v_{i'})$ and $j_i < j_{i'}$ then remove s from the label of $v_{i'}$ and all its descendants.
  5. If a state s appears in $l(v_i)$ and $l(v_{i'})$, $j_i = j_{i'}$, and $M(v_i) < M(v')$ then remove s from the label of v' and all its descendants.

6. Remove sons with empty label. Set e to the minimum of its previous value and the minimal number of removed descendant.
7. If all sons are annotated by $h(v)$ remove all sons and all their descendants. Set e to the minimum of its previous value and the minimal number of removed descendant. Set f to the minimum of its previous value and the number of v.

Let Z denote the set of nodes removed during this recursive procedure. For every node v let $empty(v)$ be $|\{v' \in Z \mid M(v') < M(v)\}|$. That is, we count how many nodes got their label empty during the recursive transformation and and their number is smaller than the number of v. For every node v such that $l(v) \neq \emptyset$ we change the number of v to $M(v) - empty(v)$. The resulting numbers are consecutive again and in the range $[m]$.

We show that the two automata are equivalent. The proof is an adaptation of Safra's proof [Saf92].

Theorem 4.2 $L(\mathcal{D}) = L(\mathcal{S})$

Proof: Consider $w \in L(\mathcal{S})$. We have to show $w \in L(\mathcal{D})$. Let $r = s_0 s_1 \cdots$ be an accepting run of $\mathcal{S}$ on w. Let $J \subseteq [k]$ be the maximal set such that for every $j \notin J$ we have $inf(r) \cap G_j = \emptyset$ and for every $j \in J$ we have $inf(r) \cap R_j \neq \emptyset$. Let $r' = d_0 d_1 \cdots$ be the run of $\mathcal{D}$ on w and let $d_i = \langle N_i, M_i, 1, p_i, l_i, h_i, e_i, f_i \rangle$. It is simple to see that forall $i \geq 0$ we have $s_i \in l_i(1)$. Let i_1 be the location such that forall $i > i_1$ we have $s_i \in inf(f)$. That is, all states appearing after i_1 appear infinitely often in the run. In particular, forall $i > i_1$ we have $s_i \notin G_j$ forall $j \notin J$.

If step 7 is applied infinitely often to node 1 (equivalently, $f = 1$ infinitely often, or during the application of transitions the descendants of 1 are all annotated by $[k]$) then r' visits F_0 infinitely often. Otherwise, from some point onwards in r' we have step 7 is not applied to node 1. Let $i_2 > i_1$ be this point. It follows that forall $i > i_2$ we have 1 is not a leaf. Then forall $i > i_2$ we have s_i appears in the label of some son of 1. This son can be changed a finite number of times. The annotation of the edge to this son can only decrease. If the edge is annotated by some $j \in J$ then r eventually visits again R_j and r is migrated to some son annotated by $j' < j$. If the edge is annotated by some $j \notin J$ then r never visits again G_j and the only way to migrate to a different son is if r somehow appears again in a different son with smaller annotation, or if r appears again in a different son with smaller number. Obviously, this can happen a finite number of times and eventually r stays in the same son of 1. The edge to this son is either annotated by 0 or by some $j_1 \notin J$. Formally, let $i_3 > i_2$ be such that forall $i > i_3$ we have s_i appears in $l_i(v_1)$ and v_1 is a son of 1. We know that forall $i > i_3$ we have $J \subseteq h_i(v_1)$. The number $M(v_1)$ may decrease finitely often until it is constant. Let $i_4 > i_3$ be such that forall $i > i_4$ we have $a_1 = M_i(v_1)$. As $M_i(v_1) = a_1$ forall $i > i_4$ it follows that $e_i > a_1$ forall $i > i_4$.

If step 7 is applied to node v_1 infinitely often then we are done. Otherwise, we construct by induction a sequence $1, v_1, \ldots, v_k$ such that v_{a+1} is a son of v_a, eventually $v_1, \ldots, v_k$ do not change their numbers and r appears in the label of all of them. Furthermore, we have $J \subseteq h(v_k)$ (which implies that $J \subseteq h(v_a)$ forall $1 \leq a \leq k$). As the number of the active nodes in a tree is bounded by m we can repeat the process only finitely often. Hence, w is accepted by $\mathcal{D}$.

In the other direction, consider $w \in L(\mathcal{D})$. Let $r' = d_0 d_1 \cdots$ be the accepting run of $\mathcal{D}$ on w where $d_i = \langle N_i, M_i, 1, p_i, l_i, h_i, e_i, f_i \rangle$. Let F_{2b} be the minimal set to be visited infinitely often. It follows that eventually always $e_i > b + 1$ and infinitely often $f_i = b + 1$.

We find a subset $J \subseteq [k]$ and construct an infinite tree with finite branching degree. The root of the tree corresponds to the initial state of $\mathcal{S}$. Every node in the tree is labeled by some state of $\mathcal{S}$ and a time stamp i. An edge between the nodes labeled (s, i) and (t, j) corresponds to a run starting in s, ending in t, reading $w[i, j-1]$, avoiding all sets G_j for $j \notin J$, and visiting all R_j for $j \in J$. From König's lemma this tree contains an infinite branch. The composition of all the run segments in this infinite branch is an infinite accepting run of $\mathcal{S}$ on w.

We first prove two claims. The first, showing that all the states of $\mathcal{S}$ that appear in labels of nodes of a state of $\mathcal{D}$ are reachable from the initial state of $\mathcal{S}$. The second, proves that if for some $2j$ the set F_{2j} is visited in d_i and $d_{i'}$ and no visit to $F_{j'}$ for $j' < 2j$ occurs between i and i', then there exists a node v such that $M_a(v) = j + 1$ forall $i \leq a \leq i'$ and for every state s in $l_{i'}(v)$ we find a run segment of $\mathcal{S}$ that starts in some state in $l_i(v)$, avoids G_j forall $j \notin h_i(v)$, visits R_j forall $j \in h_i(v)$, and ends in s.

Claim 4.3 *For every i, j, and every state $s \in l_i(j)$ we have s is reachable from s_0 reading $w[0, i-1]$.*

Proof: We prove the claim for all $j \geq 1$ by induction on i. Clearly, it holds for $i = 0$. Suppose that it holds for i. As $l_{i+1}(j) \subseteq \delta(l_i(j'), w_i)$ for some j' it follows that every state in $l_{i+1}(j)$ is reachable from s_0 reading $w[0, i]$. ☐

Claim 4.4 *Let $i < i'$ be two locations such that $q_i, q_{i'} \in F_{2j}$ for some j and forall $j' \leq 2j$ and forall $i < a < i'$ we have $q_a \notin F_{j'}$. Then there exists a node v such that $M_a(v) = j + 1$ forall $i \leq a \leq i'$ and every state s in $l_{i'}(v)$ is reachable from some state in $l_i(v)$ reading $w[i, i'-1]$ with a run that avoids G_j forall $j \notin h_i(v)$ and visits R_j forall $j \in h_i(v)$.*

Proof: There exists some node v such that $M_i(v) = j + 1$ (as $d_i \in F_{2j}$). By assumption, for every $j' < 2j$ the set $F_{j'}$ is not visited between i and i'. Hence, for every node v' such that $M_i(v) \leq j + 1$ we have that $M_a(v') = M_i(v')$ forall $i \leq a \leq i'$. That is, between i and i' all nodes whose number is at most $j+1$ do not change their numbers. In particular, forall $i \leq a \leq i'$ we have $M_a(v) = j+1$. In addition, there exists $J \subseteq [k]$ such that $h_a(v) = J$ forall $i \leq a \leq i'$.

We find a run followed by node v between i and i' that avoids G_j forall $j \in J$ and visits R_j forall $j \in J$. We first show that all runs followed by v do not visit

G_j for $j \notin J$. Suppose that for some $i \leq a \leq i'$ there exists $s \in l_a(v)$ such that $s \in G_j$ for some $j \notin J$. Let v' be the youngest (according to the parenthood relation) ancestor of v such that $j \in h_a(v')$ and let v'' be the son of v' that is an ancestor of v (it may be v itself). It follows that the edge from v' to v'' is labeled by j. Then, when applying step 3a on the transformation from d_{a-1} to d_a the state s would have been moved from v'' to some other son of v'.

We show now that for every $i < a < i'$ and every $s \in l_a(v)$ such that s appears in a son of v whose edge is annotated $j \in J$ there exists a run starting in some state in $l_i(v)$, visiting R_j forall $j' \in J$ such that $j' > j$, reading $w[i, a-1]$, and ending in s. We prove this by induction on a. The first thing in the transformation from d_i to d_{i+1} is to put all the elements in $l_{i+1}(v)$ in a son labeled by $max(J)$. Clearly, this satisfies our requirement. Suppose that it is true for a and prove for $a+1$. Consider a state s appearing in $l_{a+1}(v)$ in a son v' such that the edge (v, v') is annotated by j. If there is a predecessor of s in the same son in d_a then the claim follows (this covers the case where the same state appears in a node with smaller annotation or in a node with same annotation but smaller number). Otherwise, s appears in a son created by step 3b. It follows that there is some predecessor s' of s in a son v' of v in d_a such that (v, v') is annotated by the minimal $j' > j$ such that $j' \in J$. Then, by induction there exists a run that ends in s' and visits R_j forall $j'' > j'$. In addition s is in $G_{j'}$. The claim follows.

As during the transformation from $d_{i'-1}$ to $d_{i'}$ all the states $s \in l_{i'}(v)$ are found in sons whose edge is annotated by 0 we conclude that every state $s \in l_{i'}(v)$ is reachable along a run that visits R_j forall $j \in J$. □

We are now ready to build the tree t. Let $(s_0, 0)$ label the root of t. Let i be the minimal location such that forall $j < 2b$ the set F_j is not visited after i. Let v be the node such that forall $i' > i$ we have $M_{i'}(v) = b+1$. Let $J \subseteq [k]$ be such that forall $i' > i$ we have $h_{i'}(v) = J$. Let i_1 be the minimal location such that $i_1 > i$ and $f_{i_1} = b+1$ (that is step 7 was applied to v). For every state s in $l_{i_1}(v)$ we add a node to t, label it by (s, i_1) and connect it to the root. We extend the tree by induction. We have a tree with leaves labeled by the states in $l_a(v)$ stamped by time a, and $f_a = b+1$ (step 7 was applied to v). That is, for every state s in $l_a(v)$ there exists a leaf labeled (s, a). We know that F_{2b} is visited infinitely often. Hence, there exists $a' > a$ such that $f_{a'} = b+1$ (step 7 is applied to v). For every state s' in $l_{a'}(v)$ we add a node to the tree and label it (s', a'). From Claim 4.3 it follows that every edge $(s_0, 0), (s', i')$ corresponds to some run starting in s_0, ending in s', and reading $w[0, i'-1]$. From Claim 4.4, every other edge in the tree $(s, a), (s', a')$ corresponds to some run starting in s, ending in s', reading $w[a, a'-1]$, avoiding G_j forall $j \notin J$, and visiting R_j forall $j \in J$. From König's lemma there exists an infinite branch in the tree. This infinite branch corresponds to an accepting run of $\mathcal{S}$ on w. □

Theorem 4.5 *For every NSW $\mathcal{S}$ with n states and index k there exists a DPW with $n^{n(k+2)+2}(k+1)^{2n(k+1)}$ states and index $2m(k+1)-1$ that is equal to $\mathcal{S}$.*

As before, when compared to Safra's construction, we reduce the number of states, reduce the index, and get a parity automaton. The advantages are similar to those described in Section 3.

5 Conclusions and Future Work

We improved both of Safra's determinization constructions. In both cases, we reduce the number of states, cut the index of the resulting automaton by half, and most important construct directly a parity automaton. In the case of NBW we reduce the maximal number of states from $2^{5n}n^{2n}$ to n^{2n+2}. The index is reduced from $4n + 1$ to $2n - 1$. In the case of NSW we reduce the maximal number of states from $2^{5n(k+1)}n^{n(k+2)}(k+1)^{2n(k+1)}$ to $n^{n(k+2)+2}(k+1)^{2n(k+1)}$. The index is reduced from $4n(k+1)$ to $2n(k+1)-1$. The fact that our automata are parity automata makes them easier to use 'down the line'. The algorithms for solving parity games are much simpler than those that solve Rabin games. In particular, Rabin games are NP-complete in the Rabin index while parity games are known to be in NP∩co-NP. The complement of a DPW is again a DPW. In contract, the complement of a DRW is a DSW. In order to get back to Rabin (or parity) one has to multiply the number of states by $k^2k!$, where k is the Rabin index of the automaton. Our upper bound improves the best known upper bound in numerous applications, such as solving games, emptiness of alternating tree automata, satisfiability of μ-calculus with backward modalities and other applications. In particular, in the recent emptiness algorithm for alteranting parity tree automata [KV05] the upper bound is reduced from $2^{10n^2}n^{4n^2}(2n)!$ to n^{4n^2}.

Both determinization constructions have matching lower bounds. For an NBW witn n states the best possible DPW has at least $n!$ states [Mic88]. For an NSW with n states and index h the best possible DPW has at least $max(h!, n!)$ states (cf. [Löd98]). We have gotten closer to this lower bound however there is still a large gap between the lower bound and the upper bound. We are not aware on similar lower bounds on the index of the resulting automata. As DPW[k+1] recognize more languages than DPW[k] [Wag79] and NBW recognize all ω-regular langauges we cannot hope for a determinization construction with constant index. We are not aware of a lower bound on the parity index of an NBW. That is, we are searching for a family of NBW with n states that require parity index $f(n)$. A similar question arises for NSW.

References

[ATM03] R. Alur, S. La Torre, and P. Madhusudan. Modular strategies for infinite games on recursive game graphs. In *Proc. 15th International Conference on Computer-Aided Verification*, volume 2725 of *Lecture Notes in Computer Science*, pages 67–79. Springer-Verlag, 2003.

[ATW05] C. S. Althoff, W. Thomas, and N. Wallmeier. Observations on determinization of büchi automata. In *10th International Conference on the*

Implementation and Application of Automata, Lecture Notes in Computer Science. Springer-Verlag, 2005.

[Büc62] J.R. Büchi. On a decision method in restricted second order arithmetic. In *Proc. International Congress on Logic, Method, and Philosophy of Science. 1960*, pages 1–12, Stanford, 1962. Stanford University Press.

[Cho74] Y. Choueka. Theories of automata on ω-tapes: A simplified approach. *Journal of Computer and System Sciences*, 8:117–141, 1974.

[dAHM01] L. de Alfaro, T.A. Henzinger, and R. Majumdar. From verification to control: dynamic programs for omega-regular objectives. In *Proceedings of the 16th Annual Symposium on Logic in Computer Science*, pages 279–290. IEEE Computer Society Press, 2001.

[DJW97] S. Dziembowski, M. Jurdzinski, and I. Walukiewicz. How much memory is needed to win infinite games. In *Proc. 12th IEEE Symp. on Logic in Computer Science*, pages 99–110, 1997.

[EJ88] E.A. Emerson and C. Jutla. The complexity of tree automata and logics of programs. In *Proc. 29th IEEE Symp. on Foundations of Computer Science*, pages 328–337, White Plains, October 1988.

[FKV04] E. Friedgut, O. Kupferman, and M.Y. Vardi. Büchi complementation made tighter. In *2nd International Symposium on Automated Technology for Verification and Analysis*, volume 3299 of *Lecture Notes in Computer Science*, pages 64–78. Springer-Verlag, 2004.

[GH82] Y. Gurevich and L. Harrington. Trees, automata, and games. In *Proc. 14th ACm Symp. on Theory of Comouting*, pages 60–65. ACM Press, 1982.

[HKR97] T.A. Henzinger, O. Kupferman, and S. Rajamani. Fair simulation. In *Proc. 8th Conference on Concurrency Theory*, volume 1243 of *Lecture Notes in Computer Science*, pages 273–287, Warsaw, July 1997. Springer-Verlag.

[HMU00] J.E. Hopcroft, R. Motwani, and J.D. Ullman. *Introduction to Automata Theory, Languages, and Computation (2nd Edition).* Addison-Wesley, 2000.

[Hor05] F. Horn. Streett games on finite graphs. In *Proc. 2nd Workshop on Games in Design and Verification*, 2005.

[Jur00] M. Jurdzinski. Small progress measures for solving parity games. In *17th Annual Symposium on Theoretical Aspects of Computer Science*, volume 1770 of *Lecture Notes in Computer Science*, pages 290–301. Springer-Verlag, 2000.

[Jut97] C.S. Jutla. Determinization and memoryless winning strategies. *Information and Computation*, 133(2):117–134, 1997.

[KB05] J. Klein and C. Baier. Experiments with deterministic ω-automata for formulas of linear temporal logic. In *10th International Conference on Implementation and Application of Automata*, Lecture Notes in Computer Science. Springer-Verlag, 2005.

[KV98] O. Kupferman and M.Y. Vardi. Freedom, weakness, and determinism: from linear-time to branching-time. In *Proc. 13th IEEE Symp. on Logic in Computer Science*, pages 81–92, June 1998.

[KV05] O. Kupferman and M.Y. Vardi. Safraless decision procedures. In *Proc. 46th IEEE Symp. on Foundations of Computer Science*, 2005.

[Löd98] C. Löding. Methods for the transformation of ω-automata: Complexity and connection to second-order logic. Master's thesis, Christian-Albrechts-University of Kiel, 1998.

[McN66] R. McNaughton. Testing and generating infinite sequences by a finite automaton. *Information and Control*, 9:521–530, 1966.

[MH84] S. Miyano and T. Hayashi. Alternating finite automata on ω-words. *Theoretical Computer Science*, 32:321–330, 1984.

[Mic88] M. Michel. Complementation is more difficult with automata on infinite words. CNET, Paris, 1988.

[PP06] N. Piterman and A. Pnueli. Jurdzinski-ing rabin and streett. 2006. submitted.

[PR89] A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *Proc. 16th ACM Symp. on Principles of Programming Languages*, pages 179–190, Austin, January 1989.

[Rab72] M.O. Rabin. Automata on infinite objects and Church's problem. *Amer. Mathematical Society*, 1972.

[Rog01] M. Roggenbach. Determinization of büchi-automata. In *Automata, Logics, and Infinite Games: A Guide to Current Research*, volume 2500 of *Lecture Notes in Computer Science*, pages 43–60. Springer-Verlag, 2001.

[RS59] M.O. Rabin and D. Scott. Finite automata and their decision problems. *IBM Journal of Research and Development*, 3:115–125, 1959.

[Saf89] S. Safra. *Complexity of automata on infinite objects*. PhD thesis, Weizmann Institute of Science, Rehovot, Israel, 1989.

[Saf92] S. Safra. Exponential determinization for ω-automata with strong-fairness acceptance condition. In *Proc. 24th ACM Symp. on Theory of Computing*, Victoria, May 1992.

[Sch01] S. Schwoon. Determinization and complementation of streett automata. In *Automata, Logics, and Infinite Games: A Guide to Current Research*, volume 2500 of *Lecture Notes in Computer Science*, pages 79–91. Springer-Verlag, 2001.

[Sch02] S. Schwoon. *Model-checking pushdown systems*. PhD thesis, Technische Universität München, 2002.

[SV89] S. Safra and M.Y. Vardi. On ω-automata and temporal logic. In *Proc. 21st ACM Symp. on Theory of Computing*, pages 127–137, Seattle, May 1989.

[SVW85] A.P. Sistla, M.Y. Vardi, and P. Wolper. The complementation problem for Büchi automata with applications to temporal logic. In *Proc. 10th International Colloquium on Automata, Languages and Programming*, volume 194, pages 465–474, Nafplion, July 1985. Lecture Notes in Computer Science, Springer-Verlag.

[THB95] S. Tasiran, R. Hojati, and R.K. Brayton. Language containment using non-deterministic omega-automata. In *Proc. of 8th CHARME: Advanced*

Research Working Conference on Correct Hardware Design and Verification Methods, volume 987 of *Lecture Notes in Computer Science*, pages 261–277, Frankfurt, October 1995. Springer-Verlag.

[Tho90] W. Thomas. Automata on infinite objects. *Handbook of Theoretical Computer Science*, pages 165–191, 1990.

[Var98] M.Y. Vardi. Reasoning about the past with two-way automata. In *Proc. 25th International Coll. on Automata, Languages, and Programming*, volume 1443 of *Lecture Notes in Computer Science*, pages 628–641. Springer-Verlag, Berlin, July 1998.

[VW86] M.Y. Vardi and P. Wolper. Automata-theoretic techniques for modal logics of programs. *Journal of Computer and System Science*, 32(2):182–221, April 1986.

[Wag79] K. Wagner. On ω-regular sets. *Information and Control*, 43:123–177, 1979.