Logical Methods in Computer Science
Vol. 3 (3:5) 2007, pp. 1–21
www.lmcs-online.org

Submitted Nov. 9, 2006
Published Aug. 14, 2007

# FROM NONDETERMINISTIC BÜCHI AND STREETT AUTOMATA TO DETERMINISTIC PARITY AUTOMATA *

NIR PITERMAN

Ecole Polytechnique Fédéral de Lausanne (EPFL)
*e-mail address*: nir.piterman@epfl.ch

Abstract. In this paper we revisit Safra's determinization constructions for automata on infinite words. We show how to construct deterministic automata with fewer states and, most importantly, parity acceptance conditions. Determinization is used in numerous applications, such as reasoning about tree automata, satisfiability of CTL*, and realizability and synthesis of logical specifications. The upper bounds for all these applications are reduced by using the smaller deterministic automata produced by our construction. In addition, the parity acceptance conditions allows to use more efficient algorithms (when compared to handling Rabin or Streett acceptance conditions).

## 1. Introduction

One of the fundamental questions in the theory of automata is determinism vs. nondeterminism. Another related 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 determinization and complementation are strongly related. Indeed, if the machine is deterministic we just have to dualize its answer. If the machine is nondeterministic we do not have a simple solution.

In the theory of finite automata on finite words the relation between nondeterministic and deterministic automata is well understood. We know that there exists an efficient procedure that gets a nondeterministic automaton with $n$ states and constructs a deterministic automaton with $2^n$ states accepting the same language [RS59]. This construction is tight [HMU00]. By dualizing the acceptance condition of the deterministic automaton we get an automaton for the complement language, which is again tight [HMU00].

In his proof that satisfiability of S1S is decidable, Büchi introduces nondeterministic automata on infinite words [Büc62]. Büchi takes a 'normal' finite automaton and runs it on infinite words. A run of such an automaton is an infinite sequence of states, instead of a finite sequence. The set of states *recurring* infinitely often is used to define the acceptance condition. A run is accepting according to the *Büchi condition* if the set of recurring states intersects the set of accepting states.

*2000 ACM Subject Classification:* F.1.1 Models of Computation, F.4.3 Formal Languages.
*Key words and phrases:* determinization, finite automata, Büchi, Streett, parity.

* This is an extended version of [Pit06].

LOGICAL METHODS IN COMPUTER SCIENCE
DOI:10.2168/LMCS-3 (3:5) 2007

In the case of finite automata on infinite words determinization and complementation are much more involved. Given a deterministic Büchi automaton one can easily construct a nondeterministic Büchi automaton for the complement language [Kur87]. However, deterministic Büchi automata are not closed under complementation [Lan69]. This forced the introduction of more complex acceptance conditions such as Muller, Rabin, Streett, and parity. A Rabin acceptance condition is a set of pairs of subsets of the states. A run is accepting according to a Rabin condition if there exists a pair $\langle E, F\rangle$ such that the set of recurring states does not intersect $E$ but does intersect $F$. The Streett condition is the dual of Rabin. A run is accepting according to a Streett condition if for every pair $\langle R, G\rangle$ we have that if $G$ intersects the set of recurring states so must $R$. A parity condition gives an integer priority to every state and a run is accepting if the minimal recurring priority is even. The number of priorities is the *index* of the parity condition. Rabin and Streett conditions are more general than parity in the following sense. A parity condition of index $2k$ can be written as a Rabin (or Streett) condition with $k$ pairs (without modifying the structure of the automaton). We can translate a Rabin or Streett condition with $k$ pairs to a parity condition of index $2k+1$ using a gadget with $k^2k!$ states, thus we multiply the number of states of the automaton by $k^2k!$. All three conditions are strong enough to allow determinization [Tho90].

In the case of automata on infinite words determinization and complementation are no longer so strongly coupled. Determinization can be used for complementation by dualizing the acceptance condition of the deterministic automaton. However, there are complementation constructions that are much simpler than determinization. Specifically, Büchi showed that the class of languages recognized by nondeterministic Büchi automata is closed under complement without determinization [Büc62]. Sistla, Vardi, and Wolper suggested a singly exponential complementation construction [SVW85], however with a quadratic exponent. This was followed by a complementation construction by Klarlund [Kla91] and a very elegant complementation via alternating automata by Kupferman and Vardi [KV01]. The latter construction was recently improved to give a complement automaton with at most $(0.96n)^n$ states [FKV04], which is currently the best complementation construction. See also [Tho90].

Determinization constructions for automata on infinite words followed a similar path[1]. McNaughton showed a determinization construction that is doubly exponential and results in an automaton with the Muller acceptance condition [McN66]. Safra gives a determinization construction which takes a nondeterministic Büchi automaton with $n$ states and returns a deterministic Rabin automaton with at most $(12)^n n^{2n}$ states and $n$ pairs [Saf88]. An alternative determinization with a similar upper bound that also results in a deterministic Rabin automaton was given by Muller and Schupp [MS95]. Michel showed that this is asymptotically optimal and that the best possible upper bound for determinization and complementation is $n!$ [Mic88, Löd98].

Safra's idea is to use multiple subsets for one state of the deterministic automaton and organize them in the form of a tree. The root of the tree is the classical subset construction for automata on finite words. In every transition, a node with set of states $S$ spawns a new son that includes all the accepting states in $S$. Thus, all the states in a leaf are the endpoints of runs that agree (more or less) on the number of times they have visited the acceptance set. In order to keep the tree finite, we ensure that every state is followed in at most one

[1]Incidentally, both determinization constructions provided the best upper bound for complementation at the time of their introduction.

branch of the tree. Whenever a state is followed in more than one branch we keep only the copy in the oldest branch. Furthermore, whenever all the states followed by some node have visited the acceptance set, the node is marked as accepting and all its descendants are removed. The Rabin acceptance condition associates a pair with every node in the tree. There should be some node that is erased from the tree at most finitely often and marked accepting infinitely often for a run to be accepting.

The fact that stronger acceptance conditions are introduced raises the question of determinization of automata using these conditions. Rabin and parity automata can be easily converted to Büchi automata. Given a Rabin automaton with $n$ states and $k$ pairs there exists an equivalent nondeterministic Büchi automaton with $n(k+1)$ states. Applying Safra's determinization on top of this automaton produces a deterministic Rabin automaton with $(12)^{n(k+1)}(n(k+1))^{2n(k+1)}$ states and $n(k+1)$ pairs. For Streett automata, going through nondeterministic Büchi automata is far from optimal. A nondeterministic Streett automaton with $n$ states and $k$ pairs can be converted to a nondeterministic Büchi automaton with $n2^k$ states [Cho74], which is optimal [SV89]. Combining this conversion with the determinization results in a doubly exponential deterministic automaton. In order to handle Streett automata, Safra generalized his determinization construction [Saf92]. Given a Streett automaton with $n$ states and $k$ pairs he constructs a Rabin automaton with $(nk)^{O(nk)}$ states and $O(nk)$ pairs.

We mentioned that the Rabin and Streett conditions are duals; the dual of the parity condition is parity again. Sometimes, given a nondeterministic automaton, we need to generate a deterministic automaton for the complementary language, a process called *co-determinization* (e.g., for converting alternating tree automata to nondeterministic tree automata). While complementing a deterministic automaton can be easily done by dualizing the acceptance condition, such a dualization for a Rabin or Streett automaton results in an automaton of the second type. Thus, co-determinization of a Büchi (or Streett) automaton results in a deterministic Streett automaton. Translating from Streett to Rabin or parity is exponential, we add a gadget with $k^2k!$ states where $k$ is the number of pairs of the Streett condition [Saf92]. Thus, we multiply the number of states of the automaton by $k^2k!$. The translation of Rabin to Streett or parity is dual and has exactly the same complexity.

Determinization has many uses other than complementation. For example, Rabin uses McNaughton's determinization of Büchi automata to complement nondeterministic Rabin tree automata [Rab72].[2] A node in an infinite tree belongs to infinitely many branches. A tree automaton has to choose states that handle all branches in a single run. In many cases, we want all branches of the tree to belong to some word language. If we have a deterministic automaton for this word language, we run it in all directions simultaneously. This kind of reasoning enables conversion of alternating tree automata to nondeterministic tree automata and complementation of nondeterministic tree automata (cf. [Rab72, Tho90, Var98]).

Deterministic automata are used also for solving games and synthesizing strategies. In the context of games, the opponent may be able to choose between different options. Using a deterministic automaton we can follow the game step by step and monitor the goal of the game. For example, 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. [KV98, dAHM01]). Using Safra's determinization, reasoning about tree

[2]Rabin uses this complementation in order to prove that satisfiability of S2S is decidable [Rab72]. 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.

automata reduces to reasoning about nondeterministic Rabin tree automata and reasoning about general games reduces to reasoning about Rabin games. Some of these applications use co-determinization, the deterministic automaton for the complementary language.

In this paper we revisit Safra's determinization constructions. We show that we can further compact the tree structure used by Safra to get a smaller representation of the deterministic automata. By using dynamic node names instead of the static names used by Safra we can construct directly a deterministic parity automaton. Specifically, starting from a nondeterministic Büchi automaton with $n$ states, we end up with a deterministic parity automaton with $2n^n n!$ states and index $2n$ (instead of Rabin automaton with $(12)^n n^{2n}$ states and $n$ pairs). Starting from a Streett automaton with $n$ states and index $k$, we end up with a deterministic parity automaton with $2n^n(k+1)^{n(k+1)}(n(k+1))!$ states and index $2n(k+1)$ (instead of Rabin automaton with $(12)^{n(k+1)} n^n (k+1)^{n(k+1)} (n(k+1))^{n(k+1)}$ states and $n(k+1)$ pairs). For both constructions, complementation is done by considering the same automaton with a dual parity condition.

Though dividing the number of states by $12^n$ is not negligible, the main importance of our result is in the fact that the resulting automaton is a parity automaton instead of Rabin. Solving Rabin games (equivalently, emptiness of nondeterministic Rabin tree automata) is NP-complete in the number of pairs [EJ88]. Solution of parity games is in NP∩co-NP. The current best upper bound for solving Rabin games is $mn^{k+1}k!$ where $m$ is the number of transitions, $n$ the number of states, and $k$ the number of pairs [PP06]. Using our determinization construction instead of reasoning about Rabin conditions we can consider parity conditions. The best upper bound for solving parity games is $mn^{k/2}$ [Jur00] (cf. [BSV03, JPZ06] for other solutions). That is, we save a multiplier of at least $kk!$.

The gain by using our determinization is even greater when we consider applications that use co-determinization. As Streett is the dual of Rabin it follows that solving Streett games is co-NP-complete. Even if we ignore the computational difficulty, the Rabin acceptance condition at least 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. In order to solve Streett games we require exponential memory [DJW97, Hor05]. Applications like nondeterminization of alternating tree automata use co-determinization but require the result to be a Rabin or parity automaton. Hence, the resulting deterministic Streett automaton has to be converted to a parity automaton. Again, the price tag of this conversion is a blowup of $k^2k!$ where $k$ is the number of pairs. As the complexity of reasoning about parity games is $mn^{k/2}$, the extra multiplier grows to $(k^2k!)^k$.

Recently, Kupferman and Vardi showed that they can check the emptiness of an alternating parity tree automaton without directly using Safra's determinization [KV05]. Their construction can be used for many game / tree automata applications that require determinization. However, Kupferman and Vardi use Safra's determinization to get a bound on the size of the minimal model of the alternating tree automaton. Given such a bound, they can check emptiness by restricting the search to small models. Our improved construction implies that the complexity of their algorithm reduces from $(12)^{n^2} n^{4n^2+2n} (n!)^{2n}$ to $(2n^n n!)^{2n}$.

## 2. Nondeterministic Automata

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

A *nondeterministic automaton* is $N = \langle \Sigma, S, \delta, s_0, \alpha\rangle$, where $\Sigma$ 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 $\alpha$ 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 $s_0s_1\cdots \in S^\omega$ such that $s_0$ is the initial state and for all $j \geq 0$ we have $s_{j+1} \in \delta(s_j, w_j)$. For a run $r = s_0s_1\cdots$, 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 $\alpha$ is a set of pairs $\{\langle E_1, F_1\rangle, \ldots, \langle E_k, F_k\rangle\}$ where for all $i$ 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 $\alpha$ 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 $\alpha$ is also a set of pairs $\{\langle R_1, G_1\rangle, \ldots, \langle R_k, G_k\rangle\}$ where for all $i$ 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 $\alpha$ 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 $\alpha$ 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 $\alpha$ if for some even $i$ we have $inf(r) \cap F_i \neq \emptyset$ and for all $i' < i$ we have $inf(r) \cap F_{i'} = \emptyset$. A *Büchi* condition $\alpha$ is a subset of $S$. A run is *accepting* according to the Büchi condition $\alpha$ if $inf(r) \cap \alpha \neq \emptyset$. That is, the run visits infinitely often states from $\alpha$. 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). *Determinization* 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 determinization constructions take nondeterministic Büchi or Streett automata and convert them to deterministic Rabin automata. We describe these two constructions below.

We use acronyms in $\{N, D\} \times \{R, S, P, B\} \times \{W\}$ to denote automata. The first symbol stands for the branching mode of the automaton: $N$ for nondeterministic 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, in our case $W$ for words. For example, a DRW is a deterministic Rabin word automaton and an NBW is a nondeterministic Büchi word automaton.

## 3. Determinization of Büchi Automata

In this section we give a short exposition of Safra's determinization [Saf88] and show how to improve it. We replace the constant node names with dynamic names, which allow us to simulate the *index appearance record*[3] construction within the deterministic automaton. We get a deterministic automaton with fewer states and in addition a parity automaton instead of Rabin.

3.1. **Safra's Construction.** Here we describe Safra's determinization construction [Saf88, Saf89]. The construction takes an NBW and constructs an equivalent DRW. Safra constructs a tree of subset constructions. Every node in the tree is labeled by the states it follows. The labels of siblings are disjoint and the label of a node is a strict superset of the union of the labels of its descendants. The sons are ordered according to their age. The transition of a tree replaces the label of every node by the set of possible successors. If the label now includes some accepting states, we add a new son to the node with all these accepting states. Intuitively, the states that label the sons of a node have already visited an accepting state. Thus, the states in the label of a node that are not in the labels of its descendants are states that still owe a visit to the acceptance set. If a state occurs in two sibling nodes (or more), we remove it from the younger sibling and keep it only in the older sibling. If the label of a node becomes equal to the union of labels of its descendants then we mark this node as accepting and remove all its descendants. If some node remains eventually always in the tree and is marked accepting infinitely often, the run is accepting. Formally, we have the following.

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

**Definition 3.1.** 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 $p(v)$.
- $\psi$ 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 two disjoint subsets of $V$. They are used to define the Rabin acceptance condition.

The following claim is proved in [Saf88, Saf89, Jut97, KV05].

**Claim 3.2.** *The number of Safra trees over $S$ is not more than* $(12)^n n^{2n}$.

[3]The index appearance record is the gadget that allows to translate Rabin and Streett conditions to parity conditions [Saf92]. It is a permutation over the pairs in the Rabin / Streett condition with two pointers into the permutation.

*Proof.* The number of ordered trees on $n$ nodes is the $(n-1)$th Catalan number. We know that $Cat(n) = \frac{(2n)!}{n!(n+1)!}$ and $Cat(n-1) \leq 4^n$. We represent the naming of nodes by $f : [n] \to [n]$ that associates the $i$th node with its name $f(i)$. There are at most $n^n$ such functions. The labeling function is $l : S \to [n]$ where $l(s) = i$ means that $s$ belongs to the $i$th node and all its ancestors. Finally, we represent $E$ and $F$ by a function $a : V \to \{\emptyset, E, F\}$ such that $a(i) = \emptyset$ means that $i \notin E \cup F$, $a(i) = E$ means that $i \in E$, and $a(i) = F$ means that $i \in F$. There are at most $3^n$ such functions.

To summarize, the number of trees is at most $4^n \cdot 3^n \cdot n^n \cdot n^n = (12)^n n^{2n}$. □

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

- $D$ is the set of Safra trees over $S$. For a state $d \in D$ we denote by a $d$ subscript the components of $d$. For example, $N_d$ is the set of nodes of $d$ and $l_d$ is the labeling of $d$.
- $d_0$ is the tree with a single node 1 labeled by $\{s_0\}$ where $E$ is $V - \{1\}$ and $F$ is the empty set.
- Let $\alpha' = \{\langle E_1, F_1 \rangle, \ldots, \langle E_n, F_n \rangle\}$ be the Rabin acceptance condition 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 $d$. We use temporarily the set of names $V'$ disjoint from $V$.
  (1) For every node $v$ with label $S'$ replace $S'$ by $\delta(S', \sigma)$ and set $E$ and $F$ to the empty set.
  (2) For every node $v$ with label $S'$ such that $S' \cap \alpha \neq \emptyset$, create a new node $v' \in V'$ which becomes the youngest child of $v$. Set its label to be $S' \cap \alpha$.
  (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 nodes to $E$.
  (7) Change the nodes in $V'$ to nodes in $V$.

**Claim 3.3.** *The transition is well defined.*

*Proof.* It is simple to see that the label of every node is a proper superset of the union of the labels of its children and that the labels of two siblings are disjoint. We have to show that the $n$ nodes in $V$ are sufficient to complete step 7.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 $n$ node names are sufficient. □

**Theorem 3.4.** [Saf88] $L(\mathcal{D}) = L(\mathcal{N})$. □

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

3.2. **From NBW to DPW.** We now present our construction. Intuitively, we take Safra's construction and replace the constant node name with a dynamic one that decreases as nodes below it get erased from the tree. Using the new names we can give up the "older than" 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 $\mathcal{N} = \langle \Sigma, S, \delta, s_0, \alpha \rangle$ be an NBW with $|S| = n$. For the sake of the proof we would like to treat the nodes as entities. Hence, we distinguish between the set of nodes $V = [2n]$ of a tree and their names that may change and range over $[n]$. All important information (tree structure, label) can be associated with the names and in practice the distinction between nodes and their names is not needed.

**Definition 3.5.** 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 naming 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. The number $e$ is used to memorize the minimal node that changed its name and $f$ the minimal node that is equivalent to its descendants.

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

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

**Claim 3.6.** *The number of compact Safra trees over $S$ is not more than $2n^n n!$.*

*Proof.* Just like Safra trees there are at most $n$ nodes. We use only the names of the nodes. The parent of the node has a smaller name. Thus, the parenthood relation can be represented by a sequence of (at most) $n-1$ pointers, where the $i$th pointer is pointing to a value in $1, \ldots, i-1$. It follows that there are at most $(n-1)!$ such trees. 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. There are $n$ options for $e$ and $f$ each. In order to define the acceptance condition (see below) we need to know the value of $e$ in case $e \leq f$ and the value of $f$ in case $f < e$. Thus, we need $2n$ possible values. It follows that there are at most $2n \cdot (n-1)! \cdot n^n = 2n^n n!$ different compact Safra trees. □

We construct the DPW $\mathcal{D}$ equivalent to $\mathcal{N}$. Let $\mathcal{D} = \langle \Sigma, D, \rho, d_0, \alpha' \rangle$ 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 named 1 where $e = 2$ and $f = 1$.
- The parity acceptance condition $\alpha' = \langle F_0, \ldots, F_{2n-1} \rangle$ is defined as follows.
  - $F_0 = \{d \in D \mid f = 1 \text{ and } e > 1\}$

- $F_{2i+1} = \{d \in D \mid e = i+2 \text{ and } f \geq e\}$
- $F_{2i+2} = \{d \in D \mid f = i+2 \text{ and } e > f\}$

Note that the case $e = 1$ is not considered above. 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) For every node $v$ with label $S'$ such that $S' \cap \alpha \neq \emptyset$, create a new son $v' \notin N$ of $v$. Set its label to $S' \cap \alpha$. Set its name to the minimal value greater than all used names. We may have to use temporarily names 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 names of green nodes. Notice that no node in $[(n+1)..(2n)]$ can be green.
  (5) Remove all nodes with empty labels. Set $e$ to the minimum of $n+1$ and the names of nodes removed during all stages of the transformation. Notice that the priority of a state is even only when $f < e$. Thus, green nodes that are removed cannot make a state of even priority.
  (6) Let $Z$ denote the set of nodes removed during all previous stages of the transformation. For every node $v$ let $rem(v)$ be $|\{v' \in Z \mid M(v') < M(v)\}|$. That is, we count how many nodes are removed during the transformation and have smaller name than the name of $v$. For every node $v$ such that $l(v) \neq \emptyset$ we change the name of $v$ to $M(v) - rem(v)$. It is simple to see that the resulting names 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 [Saf88].

**Theorem 3.7.** $L(\mathcal{D}) = L(\mathcal{N})$.

*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 for all $i \geq 0$ we have $s_i \in l_i(1)$ and $e_i > 1$. 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 $i_0$ be this point. There exists a point $i_1 > i_0$ such that $s_{i_1} \in \alpha$. It follows that for all $i > i_1$ we have $s_i$ 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 name. However, this can happen finitely often and hence we treat $v_1$ as constant. The name $M(v_1)$ may decrease finitely often until it is constant. Let $i_2$ be such that for all $i > i_2$ we have $a_1 = M_i(v_1)$. As $M_i(v_1) = a_1$ for all $i > i_2$ it follows that $e_i > a_1$ for all $i > i_2$.

Suppose that step 4 is applied to $v_1$ infinitely often (equivalently, $f \leq a_1$ infinitely often). It follows that for every odd $a' < 2a_1 - 2$ we have $F_{a'}$ is visited finitely often and either $F_{2a_1-2}$ is visited infinitely often or there exists some even $a' < 2a_1 - 2$ such that $F_{a'}$ 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 names 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 $F_{2a}$ be the minimal set to be visited infinitely often. It follows that eventually always $e_i > a+1$ and infinitely often $f_i = a+1$. We first prove two claims.

**Claim 3.8.** *For every $i \in \mathbb{N}$, $v \in N_i$, and every state $s \in l_i(v)$ we have $s$ is reachable from $s_0$ reading $w[0, i-1]$.*

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

**Claim 3.9.** *Consider $i, i' \in \mathbb{N}$ such that $i < i'$, $d_i, d_{i'} \in F_{2a}$ for some $a$, and for all $a' \leq 2a$ and for all $i < j < i'$ we have $d_j \notin F_{a'}$. Then there exists a node $v$ such that $M_j(v) = a+1$ for all $i \leq j \leq i'$ 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 $\alpha$.*

*Proof.* There exists some node $v$ such that $M_i(v) = a+1$ (as $d_i \in F_{2a}$). By assumption, for every $a' < 2a$ the set $F_{a'}$ is not visited between $i$ and $i'$. Hence, for every node $v'$ such that $M_i(v) \leq a+1$ we have that $M_j(v') = M_i(v')$ for all $i \leq j \leq i'$. That is, between $i$ and $i'$ all nodes whose name is at most $a+1$ do not change their names. In particular, for all $i \leq j \leq i'$ we have $M_j(v) = a+1$. We show that for every $i \leq j < i'$ and every descendant $v'$ of $v$, every state in $l_j(v')$ is reachable from some state in $l_i(v)$ along a run visiting $\alpha$. As $v$ is a leaf in $d_i$ for $j = i$ this is obviously true. Suppose it is true for $j$ and prove for $j+1$. We know that for every descendant $v'$ of $v$ either $l_{j+1}(v') \subseteq \delta(l_j(v), w_j) \cap \alpha$ or for some descendant $v''$ of $v$ we have $l_{j+1}(v') \subseteq \delta(l_j(v''), w_j)$ ($v''$ 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. In particular, if $i' = i+1$ then all states in $l_{i'}(v)$ are in $\alpha$ and we are done. □

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 $\alpha$. 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 $w$.

Let $(s_0, 0)$ label the root of $T$. Let $i_0$ be the maximal location such that for all $a' < 2a$ the set $F_a$ is not visited after $i_0$. Let $v$ be the node such that for all $i > i_0$ we have $M_i(v) = a+1$. Let $i_1$ be the minimal location such that $i_1 > i_0$ and $f_{i_1} = a+1$ (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 $l_{i_j}(v)$ stamped by time $i_j$, and $f_{i_j} = a+1$ (step 4 was applied to $v$). That is, for every state $s$ in $l_{i_j}(v)$ there exists a leaf labeled $(s, i_j)$. We know that $F_{2a}$ is visited infinitely often. Hence, there exists $i_{j+1} > i_j$ such that $f_{i_{j+1}} = a+1$ (step 4 is applied to $v$). For every state $s'$ in $l_{i_{j+1}}(v)$ we add a node to $T$ and label it $(s', i_{j+1})$. From Claim 3.9 there exists a state $s$ in $l_{i_j}(v)$ such that $s'$ is reachable from $s$ reading $w[i_j, i_{j+1}-1]$ with a run that visits $\alpha$. We connect $(s', i_{j+1})$ to $(s, i_j)$.

From Claim 3.8 it follows that every edge $(s_0, 0), (s', i_1)$ corresponds to some run starting in $s_0$, ending in $s'$, and reading $w[0, i_1 - 1]$. From Claim 3.9, every other edge in the tree $(s, i_j), (s', i_{j+1})$ corresponds to some run starting in $s$, ending in $s'$, reading $w[i_j, i_{j+1} - 1]$, and visiting $\alpha$. 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 3.10.** *For every NBW $\mathcal{N}$ with $n$ states there exists a DPW $\mathcal{D}$ with $2n^n n!$ states and index $2n$ such that $L(\mathcal{D}) = L(\mathcal{N})$.* □

We note that this improves Safra's construction in two ways. First, we reduce the number of states from $(12)^n n^{2n}$ to $2n^n n!$. Second, our automaton is a parity automaton which is amenable to simpler algorithms. For example, given a DPW it is possible to check in polynomial time what is the minimal parity index that enables recognition of the same language, and to find such an optimal parity index on the same automaton structure [CM99]. On the other hand, finding the minimal Rabin index of a DRW is NP-hard [KPBV95] and it may be the case that an optimal condition cannot be found on the same structure [KMM04]. Many times we are interested in a deterministic automaton for the complement language, a process called co-determinization. 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 $k^2 k!$ where $k$ is the number of Rabin pairs. A similar effect occurs when using deterministic automata in the context of games. Solution of Rabin games incurs an additional multiplier of $k^2 k!$. With our construction this penalty is avoided.

## 4. Determinization of Streett Automata

In this section we give a short exposition of Safra's determinization of Streett automata [Saf92] and show how to improve it. Again, we replace the constant node names with dynamic names. We get a deterministic automaton with fewer states and in addition a parity automaton instead of Rabin. The intuition is similar to the construction in Section 3.

### 4.1. Safra's Construction.

Here we describe Safra's determinization for Streett Automata [Saf92]. The construction takes an NSW and constructs an equivalent DRW.

As mentioned, in the case of Streett automata, determinization via conversion to Büchi automata is less than optimal. Safra generalizes his construction to work for Streett automata. The idea is still to use a set of subset constructions. Let $\mathcal{S} = \langle \Sigma, S, \delta, s_0, \alpha \rangle$ be an NSW where $\alpha = \{\langle R_1, G_1 \rangle, \ldots, \langle R_k, G_k \rangle\}$. We say that a run $r$ of $\mathcal{S}$ is accepting according to the *witness set* $J \subseteq [k]$ if for every $j \in J$ we have $inf(r) \cap R_j \neq \emptyset$ and for every $j \notin J$ we have $inf(r) \cap G_j = \emptyset$. It is easy to construct an NBW whose language consists of all words accepted according to witness set $J$. The NBW has two parts. In the first part it waits until all visits to $G_j$ for $j \notin J$ have occurred. Then it moves nondeterministically to the second part where it waits for visits to $R_j$ for each $j \in J$ according to their order and disallows visits to $G_j$ for every $j \notin J$. If the automaton loops through all $j \in J$ infinitely often the run is accepting. Unfortunately, the number of possible witness sets is exponential.

Safra's construction arranges all possible runs of the NSW and all relevant witness sets in a tree structure. A state is again a tree of subset constructions. Every node in a tree

represents a process that is monitoring some witness set and checking this witness set. The node for witness set $J$ follows some set of states. It waits for visits to $R_j$ for every $j \in J$ (in descending order), if this happens without visiting $G_j$ for $j \notin J$ then the node succeeds and starts all over again.

A *Streett Safra tree* is a tree whose nodes are labeled by subsets of the states in $S$. The labels of siblings are disjoint and the labels of sons form a partition of the label of the parent. In addition every node is annotated by a subset $J \subseteq [k]$. The annotation of a son misses at most one element from the annotation of the parent. Every node that is not a leaf has at least one son whose annotation is a strict subset. In addition, children are ordered according to their age.

The root node monitors the set $[k]$ as a possible witness set. If some node is annotated with $J$ and has a child annotated $J - \{j\}$ this means that the child has given up on the hope that $R_j$ will occur. If a node has given up on $R_j$ but visits $G_j$ then the states visiting $G_j$ have no place in this node and they are moved to a new sibling. Similarly, if a node has given up on $R_j$ and visits $R_j$ then the states visiting $R_j$ have no place in this node and they are moved to a new sibling. Whenever the label of a node gets empty it is removed from the tree. If all the states followed by a node completed a cycle through its witness set, all the descendants of this node are removed and it is marked accepting. The Rabin condition associates a pair with every node. A run is accepting if some node is erased finitely often and marked accepting infinitely often.

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

**Definition 4.1.** 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 $p(v)$.
- $\psi$ 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 two disjoint subsets of $V$. They are used to define the Rabin acceptance condition.

The following claim is proved in [Saf92, Sch01].

**Claim 4.2.** *The number of Streett Safra trees over $S$ is at most*

$$(12)^{n(k+1)} n^n (k+1)^{n(k+1)} (n(k+1))^{n(k+1)}.$$

*Proof.* The number of ordered trees on $m$ nodes is at most $4^m$. We represent the naming of the nodes by $f : [m] \to [m]$. There are at most $m^m$ such functions. The labeling of a node is determined by the labels of the leaves in the subtree below it and labels of leaves are disjoint. The labeling function $S \to [n]$ associates a state $s$ with the leaf it belongs to. There are at most $n$ leaves and $n^n$ such functions. 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. The edge annotation function is $h : [m] \to [0..k]$ associating an index to the target node of the edge. Finally, $E$ and $F$ are represented by a function $a : V \to \{\emptyset, E, F\}$. The number of trees is at most $4^m \cdot 3^m \cdot m^m \cdot n^n \cdot (k+1)^m = (12)^{n(k+1)} n^n (k+1)^{n(k+1)} (n(k+1))^{n(k+1)}$. □

We construct the DRW $\mathcal{D}$ equivalent to $\mathcal{S}$. Let $\mathcal{D} = \langle \Sigma, D, \rho, d_0, \alpha' \rangle$ 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.
- Let $\alpha' = \{\langle E_1, F_1 \rangle, \ldots, \langle E_m, F_m \rangle\}$ be the Rabin acceptance condition 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 $\delta(l(v), \sigma)$. We use temporarily the set of names $V'$ disjoint from $V$.
  (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 $v' \in V'$. 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 $j_i = 0$). Call the entire procedure recursively on $v_1, \ldots, v_l$ (call recursively also for sons created in step 2 above).
  For every son $v_i$ and every state $s \in l(v_i)$ do the following.
    (a) If $s \in R_{j_i}$, remove $s$ from the label of $v_i$ and all its descendants. Add a new youngest son $v' \in V'$ to $v$. Set $l(v') = \{s\}$ and $h(v') = h(v) - \{max(\{0\} \cup (h(v) \cap \{1, \ldots, j_i - 1\}))\}$.
    (b) If $s \in G_{j_i}$, remove $s$ from the label of $v_i$ and all its descendants. Add a new youngest son $v' \in V'$ to $v$. Set $l(v') = \{s\}$ and $h(v') = h(v) - \{j_i\}$.[4]
  (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 $E$, remove unused names from $F$, and change the nodes in $V'$ to nodes in $V$.

**Claim 4.3.** *The transition is well defined.*

[4]We note that in Safra's original construction [Saf92, Sch01] the rank of the new node is set to $h(v') = h(v) - \{max(h(v))\}$. In case that both $G_{j_i}$ and $R_{j_i}$ are visited infinitely often this may lead to the following situation. Suppose that the node $v$ has a son $v'$ that is waiting for a visit to $R_{j_i}$ where $j_i$ is not the maximum in $h(v)$. In the case that $G_{j_i}$ is visited, the runs are moved to new siblings that await $max(h(v))$ again. This way, the run may cycle infinitely often between $max(h(v))$ and $j_i$, leading to incompleteness of the construction.

*Proof.* It is simple to see that the label of every node is equal to the union of the labels of its children and that the labels of two siblings are disjoint. We have to show that the $m$ nodes in $V$ are sufficient to change the nodes in $V'$ to nodes in $V$.

There exists a path from the root to a leaf where no edge is annotated by 0. For every edge annotated 0, there is a path from the target of this edge to a leaf where no edge is annotated by 0. Hence, there are at most $n-1$ edges annotated by 0. Every other edge is either annotated by some index $i \in [k]$ or connects a parent $v$ to a son $v'$ such that there is some state $s \in S$ such that $s \in l(v)$ and $s \notin l(v')$. Thus, there can be at most $nk$ such edges. Totally, $n(k+1)$ node names are sufficient. □

**Theorem 4.4.** [Saf92] $L(\mathcal{D}) = L(\mathcal{S})$.

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

4.2. **From NSW to DPW.** We now present our construction. Let $\mathcal{S} = \langle \Sigma, S, \delta, s_0, \alpha \rangle$ be an NSW where $\alpha = \{\langle R_1, G_1 \rangle, \ldots, \langle R_k, G_k \rangle\}$ and $|S| = n$. Denote $m = n(k+1)$. For the sake of the proof, we distinguish between the set of nodes $V = [2m]$ of a tree and their names that range over $[m]$. All important information (tree structure, label) can be associated with the names and in practice the distinction between nodes and their names is not needed.

**Definition 4.5.** 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 naming 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 "older than" relation and replace the sets $E$ and $F$ by numbers $e$ and $f$. The naming $M$ is a bijection from $N$ to $[|N|]$. That is, the names of nodes in $N$ are consecutive starting from the root, which is named 1.

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

**Claim 4.6.** *The number of compact Streett Safra trees over $S$ is not more than*

$$2n^n(k+1)^{n(k+1)}(n(k+1))!.$$

*Proof.* Just like Streett Safra trees there are at most $m$ nodes. We use only the names of the nodes. The parent of the node has a smaller name. Thus, the parenthood relation can be represented by a sequence of (at most) $m-1$ pointers, where the $i$th pointer is pointing to a value in $1, \ldots, i-1$. It follows that there are at most $(m-1)!$ such trees. 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 $i$th leaf. 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. The edge annotation is represented by a function $h : [m] \to [0, \ldots, k]$. In order to define the acceptance condition (see below) we need to know the value of $e$ in case $e \leq f$ and the value of $f$ in case $f < e$. Thus, we need $2m$ possible values.

It follows that the number of compact Streett Safra tress is at most $2m \cdot (m-1)! \cdot n^n \cdot (k+1)^m = 2m! \cdot n^n \cdot (k+1)^m = 2n^n(k+1)^{n(k+1)}(n(k+1))!$. □

We construct the DPW $\mathcal{D}$ equivalent to $\mathcal{S}$. Let $\mathcal{D} = \langle \Sigma, D, \rho, d_0, \alpha' \rangle$ 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 $\{s_0\}$, named 1, and annotated $[k]$. We set $e = 2$ and $f = 1$.
- The parity acceptance condition $\alpha' = \langle F_0, \ldots, F_{2m-1} \rangle$ is defined as follows.
  - $F_0 = \{d \in D \mid f = 1 \text{ and } e > 1\}$
  - $F_{2i+1} = \{d \in D \mid e = i + 2 \text{ and } f \geq e\}$
  - $F_{2i+2} = \{d \in D \mid f = i + 2 \text{ and } e > f\}$

  As before, the case where $e = 1$ 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 (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 names. We may use temporarily names out of the range $[m]$.
  (3) Let $v_1, \ldots, v_l$ be the sons of $v$ (ordered according to their names) 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 $j_i = 0$). Call recursively the entire procedure 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 $s \in R_{j_i}$, remove $s$ from the label of $v_i$ and all its descendants. Add a new son $v'$ to $v$. Set $l(v') = \{s\}$, $h(v') = h(v) - \{max(\{0\} \cup (h(v) \cap \{1, \ldots, j_i - 1\}))\}$, and set $M(v')$ to the minimal value larger than all used names.
    (b) 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 $l(v') = \{s\}$, $h(v') = h(v) - \{j_i\}$, and set $M(v')$ to the minimal value larger than all used names.
  (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 $M(v_i) < M(v_{i'})$ then remove $s$ from the label of $v_{i'}$ and all its descendants.
  (6) Remove sons with empty label. Set $e$ to the minimum of its previous value and the minimal name of a 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 name of a removed descendant. Set $f$ to the minimum of its previous value and the name of $v$.

Let $Z$ denote the set of nodes removed during this recursive procedure. For every node $v$ let $rem(v)$ be $|\{v' \in Z \mid M(v') < M(v)\}|$. That is, we count how many nodes got removed during the recursive transformation and their name is smaller than the name of $v$. For every node $v$ such that $l(v) \neq \emptyset$ we change the name of $v$ to $M(v) - rem(v)$. The resulting names 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 4.7.** $L(\mathcal{D}) = L(\mathcal{S})$.

*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 witness set for $r$. 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 for all $i \geq 0$ we have $s_i \in l_i(1)$ and $e_i > 1$. Let $i_1$ be the location such that for all $i > i_1$ we have $s_i \in inf(r)$. That is, all states appearing after $i_1$ appear infinitely often in the run. In particular, for all $i > i_1$ we have $s_i \notin G_j$ for all $j \notin J$.

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 for all $i > i_2$ node 1 is not a leaf. Then for all $i > i_2$ 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 the son containing $r$ 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 $G_j$ again 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 name. 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 for all $i > i_3$ we have $s_i$ appears in $l_i(v_1)$ and $v_1$ is a son of 1. We know that for all $i > i_3$ we have $J \subseteq h_i(v_1)$. The name $M(v_1)$ may decrease finitely often until it is constant. Let $i_4 > i_3$ be such that for all $i > i_4$ we have $a_1 = M_i(v_1)$. As $M_i(v_1) = a_1$ for all $i > i_4$ it follows that $e_i > a_1$ for all $i > i_4$.

If step 7 is applied to node $v_1$ infinitely often then we are done. Otherwise, we construct by induction a sequence $1, v_1, \ldots, v_o$ such that eventually $v_1, \ldots, v_o$ do not change their names and $r$ appears in the label of all of them. Furthermore, we have $J \subseteq h(v_o)$ (which implies that $J \subseteq h(v_{o'})$ for all $o \in [1..o]$). As the number of 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 $F_{2a}$ be the minimal set to be visited infinitely often. It follows that eventually always $e_i > a + 1$ and infinitely often $f_i = a + 1$.

We write in short *avoids* $G_{\overline{J}}$ instead of avoids $G_j$ for every $j \notin J$ and *visits* $R_J$ instead of visits $R_j$ for every $j \in J$. We first prove two claims.

**Claim 4.8.** *For every $i \in \mathbb{N}$, $v \in N_i$, and every state $s \in l_i(v)$ we have $s$ is reachable from $s_0$ reading $w[0, i-1]$.*

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

The following claim shows that if some node $v$ is colored green twice, and it is not removed and does not change its name between the two greens, then all the runs followed by $v$ visit $R_J$, where $J$ is the annotation of $v$. We essentially prove that all states that are followed by a son $v'$ of $v$ annotated by $j \in J$ are endpoints for runs that already visited $R_{j'}$ for all $j' \in J$ such that $j' > j$. When $v$ is colored green the second time, all states followed by $v$ are in sons annotated 0. This means that all $R_J$ is visited.

**Claim 4.9.** *Consider $i, i' \in \mathbb{N}$ such that $i < i'$, $d_i, d_{i'} \in F_{2a}$ for some $a$, and for all $a' \leq 2a$ and for every $o \in [i..i']$ we have $d_o \notin F_{a'}$. Then there exists a node $v$ such that $M_o(v) = a+1$ for every $o \in [i..i']$ 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 $G_{\overline{J}}$ and visits $R_J$.*

*Proof.* As $d_i \in F_{2a}$, there exists some node $v$ such that $M_i(v) = a+1$. By assumption, for every $a' < 2a$ the set $F_{a'}$ is not visited between $i$ and $i'$. Hence, for every node $v'$ such that $M_i(v') \leq a+1$ and for every $o \in [i..i']$ we have that $M_o(v') = M_i(v')$. That is, between $i$ and $i'$ all nodes whose name is at most $a+1$ do not change their names. In particular, for every $o \in [i..i']$ we have $M_o(v) = a+1$. In addition, there exists $J \subseteq [k]$ such that for every $o \in [i..i']$ we have $h_o(v) = J$.

We find a run followed by node $v$ between $i$ and $i'$ that avoids $G_{\overline{J}}$ and visits $R_J$.

We first show that all runs followed by $v$ avoid $G_{\overline{J}}$. One of the invariants maintained by the transition is that if a node $v$ is annotated by a set $J$ then it cannot be labeled by states in $G_j$ for $j \notin J$. Formally, suppose that for some $o \in [i..i']$ there exists $s \in l_o(v)$ 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 $j \in h_o(v')$ 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 3b on the transformation from $d_{o-1}$ to $d_o$ the state $s$ would have been moved from $v''$ to some other son of $v'$.

We show now that for every $o \in [i..i']$ and every $s \in l_o(v)$ 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 $R_{[(j+1)..k] \cap J}$, reading $w[i, o-1]$, and ending in $s$. We prove this by induction on $o$. 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 $o$ and prove for $o+1$. Consider a state $s$ appearing in $l_{o+1}(v)$ 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 $d_o$ 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 name). Otherwise, $s$ appears in a son created by step 3a. It follows that there is some predecessor $s'$ of $s$ in a son $v''$ of $v$ in $d_o$ 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 $R_{[(j'+1)..k] \cap J}$. In addition $s$ is in $R_{j'}$. 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 $R_J$. □

We find a witness set $J \subseteq [k]$ 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 $(t, i')$ corresponds to a run starting in $s$, ending in $t$, reading $w[i, i'-1]$, avoiding $G_{\overline{J}}$, and visiting $R_J$. 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 $w$ according to witness set $J$.

Let $(s_0, 0)$ label the root of $T$. Let $i_0$ be the minimal location such that for all $a' < 2a$ the set $F_{a'}$ is not visited after $i_0$. Let $v$ be the node such that for all $i > i_0$ we have $M_i(v) = a + 1$. Let $J \subseteq [k]$ be such that for all $i > i_0$ we have $h_i(v) = J$. Let $i_1$ be the minimal location such that $i_1 > i_0$ and $f_{i_1} = a + 1$ (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 $l_{i_o}(v)$ stamped by time $i_o$, and $f_{i_o} = a + 1$ (step 7 was applied to $v$). That is, for every state $s$ in $l_{i_o}(v)$ there exists a leaf labeled $(s, i_o)$. We know that $F_{2a}$ is visited infinitely often. Hence, there exists a minimal $i_{o+1} > i_o$ such that $f_{i_{o+1}} = a + 1$ (step 7 is applied to $v$). For every state $s'$ in $l_{i_{o+1}}(v)$ we add a node to the tree and label it $(s', i_{o+1})$. From Claim 4.9 there exists a state $s$ in $l_{i_o}(v)$ such that $s'$ is reachable from $s$ reading $w[i_o, i_{o+1} - 1]$ with a run that avoids $G_{\overline{J}}$ and visits $R_J$. We connect $(s', i_{o+1})$ to $(s, i_o)$.

From Claim 4.8 it follows that every edge $(s_0, 0), (s', i_1)$ corresponds to some run starting in $s_0$, ending in $s'$, and reading $w[0, i_1 - 1]$. From Claim 4.9, every other edge in the tree $(s, i_o), (s', i_{o+1})$ corresponds to some run starting in $s$, ending in $s'$, reading $w[i_o, i_{o+1} - 1]$, avoiding $G_{\overline{J}}$, and visiting $R_J$. 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 4.10.** *For every NSW $\mathcal{S}$ with $n$ states and index $k$ there exists a DPW $\mathcal{D}$ with $2n^n(k+1)^{n(k+1)}(n(k+1))!$ states and index $2n(k+1)$ such that $L(\mathcal{D}) = L(\mathcal{S})$.* □

As before, when compared to Safra's construction, we reduce the number of states 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 and more important construct directly a parity automaton. In the case of NBW we reduce the maximal number of states from $(12)^n n^{2n}$ to $2n^n n!$. In the case of NSW we reduce the maximal number of states from $(12)^{n(k+1)} n^n (k+1)^{n(k+1)} (n(k+1))^{n(k+1)}$ to $2n^n (k+1)^{n(k+1)} (n(k+1))!$. 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 contrast, 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 $k^2 k!$, where $k$ is the number of Rabin pairs of the automaton. Our upper bound improves the best known upper bound in numerous applications, such as solving games, complementation of tree automata, emptiness of alternating tree automata, satisfiability of $\mu$-calculus with backward modalities and CTL$^*$. In particular, in the recent emptiness algorithm for alternating parity tree automata [KV05] the upper bound is reduced from $(12)^{n^2} n^{4n^2+2n} (n!)^n$ to $(2n^n n!)^{2n}$.

There are lower bounds for both determinization constructions. For an NBW with $n$ states the best possible DPW has at least $n!$ states [Mic88]. For an NSW with $n$ states and $k$ Streett pairs the best possible DPW has at least $(\Omega(nk))^n$ states [Yan06]. 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 $\omega$-regular languages we cannot hope for a determinization construction with constant index. The language $L_k = \{w \in [1..k]^\omega \mid min(inf(w)) \text{ is even}\}$ is in DPW[k] but not in DPW[k-1]. It is simple to construct an NBW with $k$ states recognizing $L_k$. This suggests that a determinization of NBW with $k$ states may result in DPW with $k$ priorities. It is an interesting question whether the $2k$ priorities produced by our construction are indeed necessary. A similar question arises for NSW.

## Acknowledgment

I thank T.A. Henzinger for fruitful discussions, O. Kupferman and M.Y. Vardi for discussions on Safra's construction and comments on an earlier version, Y. Lustig for comments on an earlier version and for tightening the analysis of the number of states, and the referees for comments and suggesting the lower bound on the index of NBW.

## References

[BSV03] H. Björklund, S. Sandberg, and S. Vorobyov. A discrete subexponential algorithm for parity games. In *20th Annual Symposium on Theoretical Aspects of Computer Science*, volume 2607 of *Lecture Notes in Computer Science*, pages 663–674. Springer-Verlag, 2003.

[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 $\omega$-tapes: A simplified approach. *Journal of Computer and System Sciences*, 8:117–141, 1974.

[CM99] O. Carton and R. Maceiras. Computing the rabin index of a parity automaton. *Theoretical Informatics and Applications*, 33(6):495–506, 1999.

[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. Jurdziński, 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.

[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.

[JPZ06] M. Jurdziński, M. Paterson, and U. Zwick. A deterministic subexponential algorithm for solving parity games. In *Proceedings of ACM-SIAM Symposium on Discrete Algorithms*, pages 117–123. ACM/SIAM, 2006.

[Jur00] M. Jurdziński. 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.

[Kla91] N. Klarlund. Progress measures for complementation of $\omega$-automata with applications to temporal logic. In *Proc. 32nd IEEE Symp. on Foundations of Computer Science*, pages 358–367, San Juan, October 1991.

[KMM04] O. Kupferman, G. Morgenstern, and A. Murano. Typeness for $\omega$-regular automata. In *2nd International Symposium on Automated Technology for Verification and Analysis*, volume 3299 of *Lecture Notes in Computer Science*, pages 324–338. Springer-Verlag, 2004.

[KPBV95] S.C. Krishnan, A. Puri, R.K. Brayton, and P.P. Varaiya. The Rabin index and chain automata, with applications to automata and games. In *Computer Aided Verification, Proc. 7th International Conference*, pages 253–266, Liege, July 1995.

[Kur87] R.P. Kurshan. Complementing deterministic Büchi automata in polynomial time. *Journal of Computer and System Science*, 35:59–71, 1987.

[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.

[KV01] O. Kupferman and M.Y. Vardi. Weak alternating automata are not that weak. *ACM Trans. on Computational Logic*, 2(2):408–429, July 2001.

[KV05] O. Kupferman and M.Y. Vardi. Safraless decision procedures. In *Proc. 46th IEEE Symp. on Foundations of Computer Science*, pages 531–540, Pittsburgh, October 2005.

[Lan69] L.H. Landweber. Decision problems for $\omega$–automata. *Mathematical Systems Theory*, 3:376–384, 1969.

[Löd98] C. Löding. Methods for the transformation of $\omega$-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.

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

[MS95] D.E. Muller and P.E. Schupp. Simulating alternating tree automata by nondeterministic automata: New results and new proofs of theorems of Rabin, McNaughton and Safra. *Theoretical Computer Science*, 141:69–107, 1995.

[Pit06] N. Piterman. From nondeterministic Büchi and Streett automata to deterministic parity automata. In *Proc. 25th Symposium on Logic in Computer Science*, pages 255–264. IEEE press, 2006.

[PP06] N. Piterman and A. Pnueli. Faster solution of Rabin and Streett games. In *Proc. 21st Symposium on Logic in Computer Science*, pages 275–284. IEEE, IEEE press, 2006.

[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.

[Saf88] S. Safra. On the complexity of $\omega$-automata. In *Proc. 29th IEEE Symp. on Foundations of Computer Science*, pages 319–327, White Plains, October 1988.

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

[Saf92] S. Safra. Exponential determinization for $\omega$-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.

[SV89] S. Safra and M.Y. Vardi. On $\omega$-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 of *Lecture Notes in Computer Science*, pages 465–474, Nafplion, July 1985. Springer-Verlag.

[Tho90] W. Thomas. Automata on infinite objects. *Handbook of Theoretical Computer Science*, pages 133–191, 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, July 1998.

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

[Yan06] Q. Yan. Lower bounds for complementation of $\omega$-automata via the full automata technique. In *Proc. 33rd Intl. Colloq. on Automata, Languages and Pr ogramming*, volume 4052 of *Lecture Notes in Computer Science*, pages 589–600. Springer-Verlag, 2006.