From Nondeterministic Büchi and Streett Automata to Deterministic Parity Automata

Nir Piterman
Ecole Polytechnique Fédéral de Lausanne (EPFL)

Abstract

Determinization and complementation are fundamental notions in computer science. When considering finite automata on finite words determinization gives also a solution to complementation. Given a nondeterministic finite automaton there exists an exponential construction that gives a deterministic automaton for the same language. Dualizing the set of accepting states gives an automaton for the complement language. In the theory of automata on infinite words, determinization and complementation are much more involved. Safra provides determinization constructions for Büchi and Streett automata that result in deterministic Rabin automata. For a Büchi automaton with n states, Safra constructs a deterministic Rabin automaton with $n^{O(n)}$ states and n pairs. For a Streett automaton with n states and k pairs, Safra constructs a deterministic Rabin automaton with $(nk)^{O(nk)}$ states and $n(k+1)$ pairs.

Here, we reconsider Safra's determinization constructions. We show how to construct automata with fewer states and, most importantly, parity acceptance condition. Specifically, starting from a nondeterministic Büchi automaton with n states our construction yields a deterministic parity automaton with n^{2n+2} states and index $2n$ (instead of a Rabin automaton with $(12)^n n^{2n}$ states and n pairs). Starting from a nondeterministic Streett automaton with n states and k pairs our construction yields a deterministic parity automaton with $n^{n(k+2)+2}(k+1)^{2n(k+1)}$ states and index $2n(k+1)$ (instead of a Rabin automaton with $(12)^{n(k+1)} n^{n(k+2)}(k+1)^{2n(k+1)}$ states and $n(k+1)$ pairs). The parity condition is much simpler than the Rabin condition. In applications such as solving games and emptiness of tree automata handling the Rabin condition involves an additional multiplier of $n^2 n!$ (or $(n(k+1))^2 (n(k+1))!$ in the case of Streett) which is saved using our construction.

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 [24]. This construction is also tight (cf. [8]). By dualizing the acceptance condition of the deterministic automaton we get an automaton for the complement language, which is again tight (cf. [8]).

In his proof that satisfiability of S1S is decidable, Büchi introduces nondeterministic automata on infinite words [2]. 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.

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 [16]. However, deterministic Büchi automata are not closed under complementation [17]. This forced the introduction of more complex acceptance conditions such as 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 $\langle E, F \rangle$ we have that if F intersects the set of recurring states so must E. A parity condition, is syntactically both Rabin and Streett. A parity condition gives a 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. A Rabin or Streett condition with n pairs is equivalent to a parity condition of index $2n + 1$ using a gadget with $n^2 n!$ states. All three conditions are strong enough to allow determinization [33].

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 [2]. Sistla, Vardi, and Wolper suggested a single exponential complementation construction [32], however with a quadratic exponent. This was followed by a very elegant complementation construction of Kupferman and Vardi [14]. This construction was recently improved to give a complement automaton with at most $(0.96n)^n$ states [7], which is currently the best complementation construction. See also [33].

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 Müller acceptance condition [19]. 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 [26]. Michel showed that this is essentially optimal and that the best possible upper bound for determinization and complementation is $n!$ [20, 18].

Safra's idea is to use a tree of subset constructions. 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 branch of the tree, we keep the copy that visited the acceptance set the most. 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 finitely often and marked accepting

infinitely often for a run to be accepting.

The introduction of stronger acceptance conditions 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 [3], which is optimal [29]. Combining this conversion with the determinization results in a doubly exponential deterministic automaton. In order to handle Streett automata, Safra generalized his determinization construction [28]. Given a Streett automaton with n states and k pairs he constructs a Rabin automaton with $(nk)^{O(nk)}$ states and $O(nk)$ pairs. As Streett automata are more general than Büchi automata, the lower bound shows that this is essentially optimal.

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 (this is called *co-determinization*). While complementing a deterministic automaton can be easily done by dualizing the acceptance condition, such a dualization results in an automaton of a different 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 $k^2 k!$ states where k is the number of pairs of the Streett condition [28]. The translation of Rabin to Streett or parity is dual and has exactly the same complexity.

Determinization has many uses other than complementation. Indeed, Rabin uses McNaughton's determinization of Büchi automata to complement nondeterministic Rabin tree automata [23].[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. [23, 33, 34]). Deterministic automata are used also for solv-

---

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

[2]Rabin uses this complementation in order to prove that satisfiability of S2S is decidable [23]. 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.

ing 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 [22] (cf. [13, 4]). Using Safra's determinization, reasoning about tree 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 n^{2n+2} 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 $n^{n(k+2)+2}(k+1)^{2n(k+1)}$ states and index $2n(k+1)$ (instead of Rabin automaton with $(12)^{n(k+1)}n^{n(k+2)}(k+1)^{2n(k+1)}$ states and $n(k+1)$ pairs). For both constructions, complementation is 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 [6]. 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 [21]. Using our determinization construction instead of reasoning about Rabin conditions we can consider parity conditions. The best upper bound for solving parity games is $mn^{k/2}$ [10] (cf. [1, 11] 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 [5, 9]. 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 $k^2 k!$ where k is the number of pairs. As the complexity of reasoning about parity games is $mn^{k/2}$, the extra multiplier grows to $(k^2 k!)^k$.

Recently, Kupferman and Vardi showed that they can check the emptiness of an alternating parity tree automaton without directly using Safra's determinization [15]. 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 n^{4n^2+2n}.

2 Nondeterministic Automata

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 $\sigma_i \cdots \sigma_j$.

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 $w = w_0 w_1 \cdots$ is an infinite sequence of states $s_0 s_1 \ldots \in S^\omega$ 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 $r = s_0 s_1 \ldots$, let $inf(r) = \{s \in S \mid s = s_i$ for infinitely many i'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 denote automata by acronyms in $\{N, D\} \times \{R, S, P, B\} \times \{T, W\}$. 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: 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.

3 Determinization of Büchi Automata

In this section we give a short exposition of Safra's determinization [26] and show how to improve it. We replace the constant node names with dynamic names, which allow us to simulate the *index appearance record* 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 [26, 27]. 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 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. We move states occurring in more than one node to older siblings. 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 $\mathcal{N} = \langle \Sigma, S, \delta, s_0, \mathcal{F} \rangle$ be an NBW with $|S| = n$. Let $V = [n]$. 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 two disjoint subsets of V. They are used to define the Rabin acceptance condition.

The following claim is proven in [26, 27, 12, 15].

Claim 3.1 *The number of nodes in a Safra tree is at most n. The number of Safra trees over $\mathcal{N}$ is not more than $(12)^n n^{2n}$.*

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 $Cat(n) = \frac{(2n)!}{n!(n+1)!} \leq 4^n$. We represent the naming of nodes by $f : [n] \to [n]$ that associates the ith 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 ith node and all its ancestors. Finally, we represent E and F by a function $a : V \to \{0, 1, 2\}$ such that $a(i) = 0$ means that $i \notin E \cup F$, $a(i) = 1$ means that $i \in E$, and $a(i) = 2$ means that $i \in F$. There are at most 3^n such functions.

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

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 $\{\langle E_1, F_1\rangle, \ldots, \langle E_n, F_n\rangle\}$ 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. 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 $S' \cap \mathcal{F} \neq \emptyset$, create a new node $v' \in V'$ 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.
  7. Change the nodes in V' to nodes in V.

Theorem 3.2 [26] $L(\mathcal{D}) = L(\mathcal{N})$.

For other expositions of this determinization we refer the reader to [12, 18, 25].

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 (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 $|S| = n$. For the sake of the proof we would like to treat the node as an entity. Hence, we distinguish between the set of nodes $V = [2n]$ of a tree and their numbers that may change and range over $[n]$. All important information (tree structure,

label) can be associated with the numbers and in practice the names are not needed.

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. 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 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 3.3 *The number of compact Safra trees over S 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.[3] $\quad\square$

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 $\mathcal{F}' = \langle F_0, \ldots, F_{2n-1}\rangle$ is defined as follows.
  - $F_0 = \{d \in D \mid f = 1\}$
  - $F_{2i+1} = \{d \in D \mid e = i+2 \text{ and } f \geq e\}$
  - $F_{2i+2} = \{d \in D \mid f = i+2 \text{ and } e > f\}$

---

[3]We note that there is much order in the numbering of the nodes which we have not used to reduce the number of states. 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.

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.[4]

- 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 $S' \cap \mathcal{F} \neq \emptyset$, 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 [26].

Theorem 3.4 $L(\mathcal{D}) = L(\mathcal{N})$.

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 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 3.5 *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 $w[0, i]$. $\square$

Claim 3.6 *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. $\qquad\square$

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.

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 3.6 there exists a state s in $l_a(v)$ such that s' is reachable from s reading $w[a, a' - 1]$ with a run that visits $\mathcal{F}$. We connect (s', a') to (s, a).

From Claim 3.5 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 3.6, 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. $\qquad\square$

Theorem 3.7 *For every NBW $\mathcal{N}$ with n states there exists a DPW with n^{2n+2} states and index $2n$ that is equal to $\mathcal{N}$.*

We note that this improves Safra's construction in two ways. First, we reduce the number of states from $(12)^n n^{2n}$ to n^{2n+2}. Second, 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-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 [28]. 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 $k^2 k!$. Obviously, 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 [28] 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. Due to space restrictions, all proofs in this section are given in Appendix A.

4.1 Safra's Construction

Here we describe Safra's determinization for Streett Automata [28]. 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 $\mathcal{S} = \langle \Sigma, S, \delta, s_0, \mathcal{F} \rangle$ be an NSW where $\mathcal{F} = \{\langle R_1, G_1 \rangle, \ldots, \langle R_k, G_k \rangle\}$. 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 $inf(r) \cap R_k \neq \emptyset$ and for every $j \notin J$ we have $inf(r) \cap G_k = \emptyset$. It is easy to construct an NBW whose language is 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 foreach $j \in J$ 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. Formally, we have the following.

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 $V = [m]$. 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 two disjoint subsets of V. They are used

to define the Rabin acceptance condition. The following claim is proven in [28, 31].

Claim 4.1 *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 at most $(12)^{n(k+1)} n^{n(k+2)} (k+1)^{2n(k+1)}$.*

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 $\{\langle E_1, F_1 \rangle \ldots, \langle E_m, F_m \rangle\}$ 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 $\delta(l(v), \sigma)$. 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 youngest 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 $j_i = 0$). 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 $s \in R_{j_i}$, 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 $h(v') = h(v) - \{max(h(v) \cap \{1, \ldots, j_i - 1\})\}$.
     (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 $h(v') = h(v) - \{j_i\}$.[5]
  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.

---

[5]We note that in Safra's original construction [28, 30] 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 maximal 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.

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.

Theorem 4.2 [28] $L(\mathcal{D}) = L(\mathcal{N})$.

For other expositions of this determinization we refer the reader to [12, 30].

4.2 From NSW to DPW

We now present our construction. 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 $m = n(k+1)$. For the sake of the proof, we distinguish between the set of nodes $V = [2m]$ of a tree and their numbers that range over $[m]$. All important information (tree structure, label) can be associated with the numbers and in practice names are not needed.

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

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 $s \in R_{j_i}$, 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.

     (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 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.

9

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 [28].

Theorem 4.4 $L(\mathcal{D}) = L(\mathcal{S})$.

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 $2n(k+1)$ that is equal to $\mathcal{S}$.*

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 Acknowledgments

References

[1] H. Björklund, S. Sandberg, and S. Vorobyov. A discrete subexponential algorithm for parity games. In *20th STACS*, LNCS 2607, pp 663–674. Springer-Verlag, 2003.

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

[3] Y. Choueka. Theories of automata on ω-tapes: A simplified approach. *JCSS*, 8:117–141, 1974.

[4] L. de Alfaro, T. Henzinger, and R. Majumdar. From verification to control: dynamic programs for omega-regular objectives. In *16th LICS*, pp 279–290, 2001.

[5] S. Dziembowski, M. Jurdzinski, and I. Walukiewicz. How much memory is needed to win infinite games. In *12th LICS*, pp 99–110, 1997.

[6] E. Emerson and C. Jutla. The complexity of tree automata and logics of programs. In *29th FOCS*, pp 328–337, 1988.

[7] E. Friedgut, O. Kupferman, and M. Vardi. Büchi complementation made tighter. In *2nd ATVA*, LNCS 3299, pp 64–78. Springer-Verlag, 2004.

[8] J. Hopcroft, R. Motwani, and J. Ullman. *Introduction to Automata Theory, Languages, and Computation*. 2000.

[9] F. Horn. Streett games on finite graphs. In *2nd GDV*, 2005.

[10] M. Jurdzinski. Small progress measures for solving parity games. In *17th STACS*, LNCS 1770, pp 290–301. Springer-Verlag, 2000.

[11] M. Jurdziński, M. Paterson, and U. Zwick. A deterministic subexponential algorithm for solving parity games. In *SODA*, 2006.

[12] C. Jutla. Determinization and memoryless winning strategies. *IC*, 133(2):117–134, 1997.

[13] O. Kupferman and M. Vardi. Freedom, weakness, and determinism: from linear-time to branching-time. In *13th LICS*, pp 81–92, 1998.

[14] O. Kupferman and M. Vardi. Weak alternating automata are not that weak. *ACM TCL*, 2(2):408–429, 2001.

[15] O. Kupferman and M. Vardi. Safraless decision procedures. In *46th FOCS*, 2005.

[16] R. Kurshan. Complementing deterministic Büchi automata in polynomial time. *JCSS*, 35:59–71, 1987.

[17] L. Landweber. Decision problems for ω–automata. *MST*, 3:376–384, 1969.

[18] C. Löding. Methods for the transformation of ω-automata: Complexity and connection to second-order logic. MSc thesis, Kiel, 1998.

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

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

[21] N. Piterman and A. Pnueli. Jurdzinski-ing Rabin and Streett. 2006. submitted.

[22] A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *16th POPL*, pp 179–190, 1989.

[23] M. Rabin. Automata on infinite objects and Church's problem. *AMS*, 1972.

[24] M. Rabin and D. Scott. Finite automata and their decision problems. *IBM JRD*, 3:115–125, 1959.

[25] M. Roggenbach. Determinization of Büchi-automata. In *Automata, Logics, and Infinite Games: A Guide to Current Research*, LNCS 2500, pp 43–60. Springer-Verlag, 2001.

[26] S. Safra. On the complexity of ω-automata. In *29th FOCS*, pp 319–327, 1988.

[27] S. Safra. *Complexity of automata on infinite objects*. PhD thesis, Weizmann Institute, 1989.

[28] S. Safra. Exponential determinization for ω-automata with strong-fairness acceptance condition. In *24th STOC*, 1992.

[29] S. Safra and M. Vardi. On ω-automata and temporal logic. In *21st STOC*, pp 127–137, 1989.

[30] S. Schwoon. Determinization and complementation of streett automata. In *Automata, Logics, and Infinite Games: A Guide to Current Research*, LNCS 2500, pp 79–91, 2001.

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

[32] A. Sistla, M. Vardi, and P. Wolper. The complementation problem for Büchi automata with applications to temporal logic. In *10th ICALP*, LNCS 194, pp 465–474, 1985.

[33] W. Thomas. Automata on infinite objects. *Handbook of TCS*, pp 165–191, 1990.

[34] M. Vardi. Reasoning about the past with two-way automata. In *25th ICALP*, LNCS 1443, pp 628–641, 1998.

[35] K. Wagner. On ω-regular sets. *IC*, 43:123–177, 1979.

A Proofs from Section 4

We prove Claim 4.1.

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 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 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 and n^n such functions. 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 $a : V \to \{0, 1, 2\}$.

To summarize, the number of trees is at most $2^{2m} \cdot 3^m \cdot m^m \cdot n^n \cdot (k+1)^m = (12)^{n(k+1)} n^{n(k+2)} (k+1)^{2n(k+1)}$. $\square$

We prove Claim 4.3.

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 prove Theorem 4.4.

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 A.1 *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 $w[0, i]$. $\square$

Claim A.2 *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 3b 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 3a. 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 $j \in J$. $\square$

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 A.1 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 A.2, 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. $\square$