Determinization Complexities of ω Automata[☆]

Yang Cai[a], Ting Zhang[b,]

[a] *MIT CSAIL*
The Stata Center, 32-G696
Cambridge, MA 02139 USA
[b] *Iowa State University*
226 Atanasoff Hall
Ames, IA 50011 USA

Abstract

Complementation and determinization are two fundamental notions in automata theory. The close relationship between the two has been well observed in the literature. In the case of nondeterministic finite automata on finite words (NFA), complementation and determinization have the same state complexity, namely $\Theta(2^n)$ where n is the state size. The same similarity between determinization and complementation was found for Büchi automata, where both operations were shown to have $2^{\Theta(n \lg n)}$ state complexity. An intriguing question is whether there exists a type of ω-automata whose determinization is considerably harder than its complementation. In this paper, we show that for all common types of ω-automata, the determinization problem has the same state complexity as the corresponding complementation problem at the granularity of $2^{\Theta(\cdot)}$.

Keywords: automata, complementation, determinization

1. Introduction

Complementation and determinization are two fundamental notions in automata theory. Complementation of an automaton $\mathcal{A}$ is the construction of another automaton $\mathcal{B}$ that recognizes the complement language of $\mathcal{A}$. Determinization of a nondeterministic automaton $\mathcal{A}$ is the construction of another automaton $\mathcal{B}$ that recognizes the same language as $\mathcal{A}$ does.

The close relationship between determinization and complementation has been well observed in the literature [1, 2]. A deterministic automaton can be trivially complemented by dualizing its acceptance condition. As a result, a lower bound on complementation applies to determinization while an upper bound on determinization applies to complementation.

For nondeterministic finite automata on finite words (NFA), complementation and determinization have the same state complexity $\Theta(2^n)$, where n is the state size.

Email addresses: `ycai@csail.mit.edu` (Ting Zhang), `tingz@cs.iastate.edu` (Ting Zhang)

For automata on infinite words (ω-automata), similar complexity results have been known. For example, the complementation and determinization of Büchi automata both require $2^{\Theta(n \lg n)}$ states. Different from NFA, there are a variety of ω-automata, differing in terms of acceptance conditions. An intriguing question is whether there exists a type of ω-automata whose determinization is considerably harder than its complementation. In this paper, we show that for all common types of ω-automata, the determinization problem has the same state complexity as the corresponding complementation problem at the granularity of $2^{\Theta(\cdot)}$.

Besides the theoretical inquriy, studying of the complementation and determinization complexity of ω-automata also have practical use, as ω-automata have wide applications in synthesis and verification of reactive concurrent systems and they are foundations of many decision procedures. For example, determinization of ω-automata is crucial in decision problems for tree temporal logics, logic games and system synthesis. Using game-theoretical semantics [3], complementation of ω-tree automata (ω-tree complementation) not only requires complementation of ω-automata, but also requires the complementary ω-automata be deterministic. Therefore, lowering the cost of ω-determinization also improves the performance of ω-tree complementation.

Related Work. Büchi started the theory of ω-automata. The original ω-automata used to establish the decidability of S1S [4] are now referred to as Büchi automata. Shortly after Büchi's work, McNaughton proved a fundamental theorem in the theory of ω-automata, that is, ω-regular languages (the languages that are recognized nondeterministic Büchi automata) are exactly those recognizable by the deterministic version of a type of ω-automata, now referred to as Rabin automata[1] [5].

The complexity of McNaughton's Büchi determinization is double exponential. In 1988, Safra proposed a type of tree structures (now referred to as Safra trees) to obtain a Büchi determinization that converts a nondeterministic Büchi automaton of state size n to an equivalent deterministic Rabin automaton of state size $2^{O(n \lg n)}$ and index size $O(n)$ [6]. Safra's construction is essentially optimal as the lower bound on state size for Büchi complementation is $2^{\Omega(n \lg n)}$ [7, 8]. Later, Safra generalized the Büchi determinization to a Streett determinization which, given a nondeterministic Streett automaton of state size n and index size k, produces an equivalent deterministic Rabin automaton of state size $2^{O(nk \lg nk)}$ and index size $O(nk)$ [9]. A variant Büchi determinization using a similar tree structure was proposed by Muller and Schupp [10]. Recently, Piterman improved both of Safra's determinization procedures with a node renaming scheme [11]. Piterman's constructions are more efficient than Safra's, though the asymptotical bounds in terms of $2^{O(\cdot)}$ are the same. A big advantage of Piterman's constructions, however, is to output deterministic parity automata, which is easier to manipulate than deterministic Rabin automata. For example, there exists no efficient procedure to complement deterministic Rabin automata to Büchi automata [12], while such complementation is straightforward and efficient for deterministic parity automata.

[1] [5] in fact used Muller condition which was converted from Rabin condition.

In [13, 14, 15] we established tight bounds for the complementation of ω-automata with rich acceptance conditions, namely Rabin, Streett and parity automata. The state complexities of the corresponding determinization problems, however, are yet to be settled. In particular, a large gap exists between the lower and upper bounds for Streett determinization. A Streett automaton can be viewed as a Büchi automaton $\langle Q, \Sigma, Q_0, \Delta, \mathcal{F}\rangle$ except that the acceptance condition $\mathcal{F} = \langle G, B\rangle_I$, where $I = [1..k]$ for some k and $G, B : I \to 2^Q$, comprises k pairs of *enabling sets* $G(i)$ and *fulfilling sets* $B(i)$. A run is accepting if for every $i \in I$, if the run visits $G(i)$ infinitely often, then so does it to $B(i)$. Therefore, Streett automata naturally express *strong fairness* conditions that characterize meaningful computations [16, 17]. Moreover, Streett automata can be exponentially more succinct than Büchi automata in encoding infinite behaviors of systems [12]. As a results, Streett automata have an advantage in modeling the behaviors of concurrent and reactive systems. For Streett determinization, the gap between current lower and upper bounds is huge: the lower bound is $2^{\Omega(n^2 \lg n)}$ [14] and the upper bound is $2^{O(nk \lg nk)}$ [9, 11] when k is large (say $k = \omega(n)$).

In this paper, we show a Streett determinization whose state complexity matches the lower bounds established in [14]. More precisely, our construction has state complexity $2^{O(n \lg n + nk \lg k)}$ for $k = O(n)$ and $2^{O(n^2 \lg n)}$ for $k = \omega(n)$. We note that this improvement is not only meant for large k. When $k = O(\log n)$, the difference between $2^{O(n \lg n + nk \lg k)}$ and $2^{O(nk \lg nk)}$ is already substantial. In fact, no matter how large k is, our state complexity is always bounded by $2^{12n}(0.37n)^{n^2+8n}$ while the current best bound for $k = n - 1$is $(12)^{n^2} n^{3n^2+n}$ [11].

The phenomenon that determinization and complementation have the same state complexity does not stop at Streett automata; we also show that this phenomenon holds for generalized Büchi automata, parity automata and Rabin automata. This raises a very interesting question: do determinization and complementation always "walk hand in hand"? Although the exact complexities of complementation and determinization for some or all types of ω-automata could be different[2], the "coincidence" at the granularity of $2^{\Theta(\cdot)}$ is already intriguing.

Our Approaches.. Our improved construction bases on two ideas. The first one is what we have exploited in obtaining tight upper bounds for Streett complementation [15], namely, the larger the size of Streett index size, the higher correlations in the runs of Streett automata. We used two tree structures: *ITS* (*Increasing Tree of Sets*) and *TOP* (*Tree of Ordered Partitions*) to characterize those correlations. We observed that there is a similarity between *TOP* and Safra trees for Büchi determinization [6]. As Safra trees for Streett determinization are generalization of those

[2]Recent work by Colcombet and Zdanowski, and by Schewe showed that the state complexity of determinization of Büchi automata on alphabets of *unbounded* size is between $\Omega((1.64n)^n)$ [18] and $O((1.65n)^n)$ [19], which is strictly higher than the state complexity of Büchi complementation which is between $\Omega(L(n))$ [20] and $O(n^2 L(n))$ [21] (where $L(n) \approx (0.76n)^n$)). However, Schewe's determinization construction produces Rabin automata with exponentially large index size, and it is not known whether Colcombet and Zdanowski's lower bound result can be generalized to Büchi automata on conventional alphabets of fixed size.

for Büchi determinization, we conjectured that *ITS* should have a role in improving Streett determinization. Our study confirmed this expectation; using *ITS* we can significantly reduce the size of Safra trees for Streett determinization.

The second idea is a new naming scheme. Bounding the size of Safra trees alone cannot bring down the state complexity when the Streett index is small (i.e., $k = O(n)$), because the naming cost becomes a dominating factor in this case. Naming is an integral part of Safra trees. Every node in a Safra tree is associated with a name, which is used to track changes of the node between the tree (state) and its successors. The current name allocation is a retail-style strategy; when a new node is created, a name from the pool of unused names is selected arbitrarily and assigned, and when a node is removed, its name is recycled to the pool. In contrast, our naming scheme is more like wholesale; the name space is divided into even blocks and every block is allocated at the same time. When a branch is created, an unused block is assigned to it, and when a branch is changed, the corresponding block is recycled.

Paper Organization. Section 2 presents basic notations and terminology in automata theory. Section 3 introduces Safra's determinization constructions for Büchi and Streett automata. Section 4 presents our improved determinization construction for Streett determinization. Section 5 establishes tight upper bounds for the determinization of Streett, generalized Büchi, parity, and Rabin automata. Section 6 concludes with some discussion on future work. All technical proofs are omitted from the main text, but they can be found in Appendix A.

2. Preliminaries

Basic Notations. Let Σ be a set. We use Σ^* (resp. Σ^ω) to denote the set of finite (resp. infinite) sequences of elements in Σ. Members in Σ^* (resp. Σ^ω) are called finite words (resp. infinite words or ω-words) over Σ.

Let $\mathbb{N}$ denote the set of natural numbers. We write $[i..j]$ for $\{k \in \mathbb{N} \mid i \leq k \leq j\}$. For an infinite sequence ϱ, we use $\varrho(i)$ $(i \in \mathbb{N})$ to denote the i-th component in ϱ. For a finite sequence α, we write $|\alpha|$ for the length of α, $\alpha(i)$ $(i \in [1..|\alpha|])$ for the i-th component of α, and $\alpha[i..j]$ for the subsequence of α from position i to position j.

Special Notations. We reserve n and k as parameters of a determinization instance (n for state size and k for index size). Define $I = [1..k]$.

ω-Automata. A finite automaton on infinite words (ω-automaton) is a tuple $\mathcal{A} = (\Sigma, Q, Q_0, \Delta, \mathcal{F})$ where Σ is an alphabet, Q is the finite set of states, $Q_0 \subseteq Q$ is the set of initial states, $\Delta \subseteq Q \times \Sigma \times Q$ is the transition relation, and $\mathcal{F}$ is the acceptance condition.

Run and Acceptance. A *finite run* of $\mathcal{A}$ from state q to state q' over a finite word w is a finite sequence of states ϱ of length $|w| + 1$ such that $\rho(1) = q, \rho(|w| + 1) = q'$ and $\langle \rho(i), w(i), \rho(i+1) \rangle \in \Delta$ for all $i \in [1..|w|]$. A *run* ϱ of $\mathcal{A}$ over an ω-word w is an infinite sequence of ϱ such that $\varrho(0) \in Q_0$ and, $\langle \varrho(i), w(i), \varrho(i+1) \rangle \in \Delta$ for $i \in \mathbb{N}$. We write $\mathit{Inf}(\varrho)$ for the set of states that occur infinitely often in ϱ. An ω-automaton accepts w if ϱ satisfies $\mathcal{F}$. The language of $\mathcal{A}$, written $\mathscr{L}(\mathcal{A})$, is the set of ω-words accepted by $\mathcal{A}$.

Acceptance Conditions and Types. ω-automata are classified according to their acceptance conditions, which usually is defined as a predicate on $\mathit{Inf}(\varrho)$. The following are common acceptance conditions. Let G and B be functions from I to 2^Q.

- *Generalized Büchi* condition $\langle B\rangle_I$: $\forall i \in I, \mathit{Inf}(\varrho) \cap B(i) \neq \emptyset$.
- *Büchi* condition $\langle B\rangle_I$ with $I = \{1\}$ (i.e., $k = 1$).
- *Streett* condition $\langle G, B\rangle_I$: $\forall i \in I$, $\mathit{Inf}(\varrho) \cap G(i) \neq \emptyset \to \mathit{Inf}(\varrho) \cap B(i) \neq \emptyset$.
- *Parity* condition $\langle G, B\rangle_I$ with $B(1) \subset G(1) \subset \cdots \subset B(k) \subset G(k)$.
- *Rabin* condition $[G, B]_I$: $\exists i \in I$, $\mathit{Inf}(\varrho) \cap G(i) \neq \emptyset \wedge \mathit{Inf}(\varrho) \cap B(i) = \emptyset$.

For simplicity, we denote a Büchi condition by F (i.e., $F = B(1)$) and call it the final set of $\mathcal{A}$. Note that Streett and Rabin conditions are dual to each other, and Büchi, generalized Büchi and parity automata are all subclasses of Streett automata. For a Streett condition $\langle G, B\rangle_I$, if there exist $i, i' \in I$, $B(i) = B(i')$, then we can simplify the condition by replacing both $\langle G(i), B(i)\rangle$ and $\langle G(i'), B(i')\rangle$ by $\langle G(i) \cup G(i'), B(i)\rangle$. For this reason, for a Streett condition $\langle G, B\rangle_I$, we assume that B is injective and hence $k = |I| \leq 2^n$.

Trees. A tree is a set $V \subseteq \mathbb{N}^*$ such that if $v \cdot i \in V$ then $v \in V$ and $v \cdot j \in V$ for $j \leq i$. Elements in V are called *nodes* and ϵ is called the *root*. Nodes $v \cdot i$ are *children* of v and they are *siblings* to one another. The set of v's children is denoted by $\mathit{ch}(v)$. A node is a leaf if it has no children. Given an alphabet Σ, a Σ-*labeled tree* is a pair $\langle V, L\rangle$, where V is a tree and $L : V \to \Sigma$ assigns a letter to each node in V. We refer to $v \in V$ as V-*value* (*structural value*) and $L(v)$ as L-*value* (*label value*) of v. In this paper we only consider finite trees.

3. Safra's Determinization

In this section we introduce Safra's constructions for Büchi and Streett determinization.

We start with Safra's Büchi determinization [6]. Let $\mathcal{A} = \langle Q, Q_0, \Sigma, \Delta, F\rangle$ be a nondeterministic Büchi automaton. By the standard subset construction [22], it is not hard to build a deterministic ω-automaton $\mathcal{B}$ such that if a run $\varrho = q_0 q_1 \cdots \in Q^\omega$ of $\mathcal{A}$ over an ω-word w is accepting, then so is the run $\tilde{\varrho} = \tilde{q}_0 \tilde{q}_1 \cdots \in (2^Q)^\omega$ of $\mathcal{B}$ over w, and $q_i \in \tilde{q}_i$ for every $i \geq 0$. The difficult part is to guarantee that an accepting run $\tilde{\varrho}$ of $\mathcal{B}$ over w induces an accepting run ϱ of $\mathcal{A}$ over w. Let $S \overset{F}{\hookrightarrow} S'$ mean that for every state $q' \in S'$, there exists a state $q \in S$ and a finite run ρ of $\mathcal{A}$ that goes from q to q' and visits F. The key idea in the determinization constructions [5, 6, 9, 10, 11] relies on the following lemma, which itself is a consequence of König's lemma.

Lemma 1 ([5, 6])**.** *Let $\tilde{\varrho} = \tilde{q}_0\tilde{q}_1 \cdots \in (2^Q)^\omega$ be an infinite sequence of subsets of Q and $(S_i)_\omega = S_0S_1\cdots$ an infinite subsequence of $\tilde{\varrho}$ such that for every $i \geq 0$, $S_i \overset{F}{\hookrightarrow} S_{i+1}$. Then there is an accepting run $\varrho = q_0q_1 \cdots \in Q^\omega$ of $\mathcal{A}$ such that $q_i \in \tilde{q}_i$ for every $i \geq 0$.*

The ingenuity of Safra construction is to efficiently organize subsets of states using a type of tree structures, now referred to as Safra trees. In a Safra tree, each node is equipped with a set label, which is a subset of Q. We simply say that a node *contains* the states in its set label. The standard subset construction is carried out on all nodes in parallel, and each node v in the tree gives birth to a new child which contains the states both in F and in v. We say that a node v turns *green* if every state in v appears in a child of v, and vice versa. We have $S \overset{F}{\hookrightarrow} S'$ where S and S' are the set labels of v at two consecutive moments of v being green. If v turns green infinitely often, then we have an desired $(S_i)_\omega$ as stated in Lemma 1.

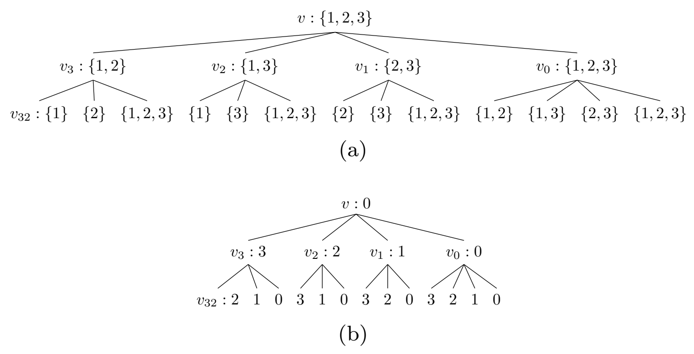
The idea was generalized to Streett determinization [9]. For a Streett automaton $\mathcal{A} = \langle Q, Q_0, \Sigma, \Delta, \langle G, B\rangle_I\rangle$, we say that a subset J of the index set I is a *witness set* for a run ϱ of $\mathcal{A}$, if for every j in J, ϱ visits $B(j)$ infinitely often while for every $j \in I \setminus J$, ϱ visits $G(j)$ only finitely many times. We call indices in $I \setminus J$ *negative obligations* and indices in J *positive obligations*. It is easily seen that a run ϱ is accepting if and only if ϱ admits a witness set $J \subseteq I$. We say that a finite run ρ *fulfills* J if for every j in J, ρ visits $B(j)$, but for every $j \in I \setminus J$, ρ does not visit $G(j)$. By $S \overset{J}{\hookrightarrow} S'$ we mean that every state in S' is reachable from a state in S via a finite run ρ that fulfills J. We have the following lemma analogous to Lemma 1.

Lemma 2 ([9])**.** *Let $\tilde{\varrho} = \tilde{q}_0\tilde{q}_1 \cdots \in (2^Q)^\omega$ be an infinite sequence of subsets of Q and $(S_i)_\omega = S_0S_1\cdots$ an infinite subsequence of $\tilde{\varrho}$ such that for every $i \geq 0$, $S_i \overset{J}{\hookrightarrow} S_{i+1}$. Then there is an accepting run $\varrho = q_0q_1 \cdots \in Q^\omega$ of $\mathcal{A}$ such that $q_i \in \tilde{q}_i$ for every $i \geq 0$.*

A naive implementation based on witness set detection would lead to double exponential blow-up in terms of n, for there are 2^k witness sets and k can be as large as 2^n. Safra's solution was to efficiently represent witness sets in the tree structure where each node is associated with a witness set and a state set. In fact, there are two other types of node labels: names and colors, which are important to the whole construction as well. But for now, we focus on witness sets and state sets, and the corresponding core process, what we call *sweeping and spawning*. We illustrate the process in a simple example for which $k = 3$ and $I = [1..3]$.

The initial state in the deterministic automaton $\mathcal{B}$ is a root v labeled with I and Q_0. The obligation of v is to detect runs that fulfills I. Once a run visits $B(3)$ (fulfilling a positive obligation), the run moves to a new child, waiting to visit $B(2)$, and once the run visits $B(2)$, the run moves to anther new child, waiting to visit $B(1)$, and so on. Technically, a finite run is represented by its frontier state. By saying that a run moves from node v to node v', we mean that the representative state of the run moves from v to v'.

This sweeping could be stalled at any node due to failure of fulfilling some positive obligations. For example, it could happen that from some point on, runs in v will



$v : \{1,2,3\}$
$v_3 : \{1,2\}$ $v_2 : \{1,3\}$ $v_1 : \{2,3\}$ $v_0 : \{1,2,3\}$
$v_{32} : \{1\}$ $\{2\}$ $\{1,2,3\}$ $\{1\}$ $\{3\}$ $\{1,2,3\}$ $\{2\}$ $\{3\}$ $\{1,2,3\}$ $\{1,2\}$ $\{1,3\}$ $\{2,3\}$ $\{1,2,3\}$
(a)
$v : 0$
$v_3 : 3$ $v_2 : 2$ $v_1 : 1$ $v_0 : 0$
$v_{32} : 2$ 1 0 3 1 0 3 2 0 3 2 1 0
(b)

Figure 1: Two equivalent representations of witness sets. In (a) nodes are explicitly labeled with witness sets while in (b) witness sets are implicitly represented by index labels.

never visit $B(3)$. But this does not necessarily mean that the run is not accepting, as it could fulfill a witness set that is a subset of I. The solution to prevent the stallment is to let v spawn a child v_3 with the witness set $I' = I \setminus \{3\} = \{1,2\}$ (see Figure 1a). Since the goal of v_3 is to fulfill I', it should not visit $G(3)$, the corresponding negative obligation. Any violation will result in the run being reset (the exact meaning of reset is shown in Step (1.3.2.3.2)). If a run fulfills I after a squence of movements, then the run finally resides in a special child that also has I as its witness set. If all children of v are special, then we say v turns *green*. It can be shown that if S and S' are the state sets of v at any two consecutive moments when v turns green, then we have $S \overset{J}{\hookrightarrow} S'$. But v may never turn green because not all runs fulfill I. That is exactly the reason that those special children of v should also have I as their witness sets, in that they spawn children, behaving just as v. Note that this *sweeping and spawning* process is applied on every node from the root to leaves. We refer the reader to [23, 11] for a detailed exposition of this process. Figure 1 shows part of a Safra tree for Streett determinization in two equivalent representations (with respect to witness sets). The representation shown in Figure 1b is used in Definition 1.

As mentioned before, additional two key ingredients in Safra's construction are names and colors, which track changes on nodes to identify those that turn green infinitely many times. In this paper, use three colors: *green*, *red* and *yellow*.

Definition 1 (Safra Trees for Streett Determinization (*STS*))**.** *A* Safra tree for Streett determinization *(STS) is a labeled tree* $\langle V, L\rangle$ *with* $L = \langle L_n, L_s, L_c, L_h\rangle$ *where*

1.1 $L_n : V \to [1..nk]$ *assigns each node a unique name.*

1.2 $L_s : V \to 2^Q$ *assigns each node a subset of* Q *such that for every node* v*,* $L_s(v) = \cup_{v' \in ch(v)} L_s(v')$ *and* $L_s(v') \cap L_s(v'') = \emptyset$ *for every two distinct* $v', v'' \in ch(v)$*.*

1.3 $L_c : V \to \{green, red, yellow\}$ *assigns each node a color.*

1.4 $L_h : V \to I \cup \{0\}$ *assigns each node an index in* $I \cup \{0\}$. *For a node* v, *let* $L_h^{\rightarrow}(v)$ *denote the sequence of* I*-elements from the root to* v *with* 0 *excluded and* $L_h^{set}(v)$ *the set of* I*-elements occurring in* $L_h^{\rightarrow}(v)$. *We require that for every node* v, *there is no repeated index in the sequence* $L_h^{\rightarrow}(v)$.

More precisely, an *STS* is a labeled and ordered tree; nodes are partially ordered by *older-than* relation. In the tree transformation (see Procedure 1), a newly added node is considered *younger* than its existing siblings. We choose not to define the ordering formally as the meaning is clear from the context. Also, for the sake of presentation clarity, we separate the following naming and coloring convention from the core construction (Step 1.3 of Procedure 1).

Rule 1 (Naming and Coloring on *STS*)**.**

1.1 Newly created nodes are marked red.

1.2 Nodes whose all descendants are removed are marked green.

1.3 Nodes are marked yellow unless they have been marked red or green.

1.4 Nodes with the empty set label are removed.

1.5 When a node is created, a name from the pool of unused name is selected and assigned to the node; when a node is removed, its name is recycled back to the pool.

Procedure 1 (Streett Determinization [9])**.** *Let* $\mathcal{A} = \langle Q, Q_0, \Sigma, \Delta, \langle G, B\rangle_I \rangle$ *(*$I = [1..k]$*) be a nondeterministic Streett automaton. The following procedure outputs a deterministic Rabin automaton* $\mathcal{B} = \langle \tilde{Q}, \tilde{q}_0, \Sigma, \tilde{\Delta}, [\tilde{G}, \tilde{B}]_{\tilde{I}} \rangle$ *(*$\tilde{I} = [1..nk]$*) such that*

1.1 $\tilde{Q}$ *is the set of STS.*

1.2 $\tilde{q}_0 \in \tilde{Q}$ *is the tree with just the root* v *such that* $L_n(v) = 1$, $L_s(v) = Q_0$, $L_c(v) = red$ *and* $L_h(v) = 0$.

1.3 $\tilde{\Delta} : \tilde{Q} \times \Sigma \to \tilde{Q}$ *is the transition function such that* $\tilde{q}' = \tilde{\Delta}(\tilde{q}, a)$ *is the STS obtained by applying the following transformation rule to* $\tilde{q}$.

1.3.1 Subset Construction*: for each node* v *in* $\tilde{Q}$, *update set label* $L_s(v)$ *to* $\Delta(L_s(v), a)$.

1.3.2 Expansion. *Apply the following transformations* downwards *from the root:*

1.3.2.1 If v *is a leaf with* $L_h^{set}(v) = I$, *stop.*

1.3.2.2 If v *is a leaf with* $L_h^{set}(v) \neq I$, *add a child* v' *to* v *such that* $L_s(v') = L_s(v)$ *and* $L_h(v') = \max(I \setminus L_h^{set}(v))$.

1.3.2.3 If v *is a node with* j *children* $v_1, \ldots, v_j$. *Let* $i_1, \ldots, i_j$ *be the corresponding index labels. Consider the following cases for each* $j' \in [1..j]$.

1.3.2.3.1 If $L_s(v_{j'}) \cap B(i_{j'}) \neq \emptyset$. *Add a child* v' *to* v *with* $L_s(v') = L_s(v_{j'}) \cap B(i_{j'})$ *and* $L_h(v') = \max([0..i_{j'}) \cap ((I \cup \{0\}) \setminus L_h^{set}(v)))$, *and remove the states in* $L_s(v_{j'}) \cap B(i_{j'})$ *from* $v_{j'}$ *and all the descendants of* $v_{j'}$.

1.3.2.3.2 *If $L_s(v_{j'}) \cap B(i_{j'}) = \emptyset$ and $L_s(v_{j'}) \cap G(i_{j'}) \neq \emptyset$. Add a child v' to v with $L_s(v') = L_s(v_{j'}) \cap G(i_{j'})$ and $L_h(v') = L_h(v)$, and remove the states in $L_s(v_{j'}) \cap G(i_{j'})$ from $v_{j'}$ and all the descendants of $v_{j'}$.*

1.3.3 Horizontal Merge. *For any state q and any two siblings v and v' such that $q \in L_s(v) \cap L_s(v')$, if $L_h(v) < L_h(v')$, or $L_h(v) = L_h(v')$ and v is* older than *v', then remove q from v' and all its descendants.*

1.3.4 Vertical Merge. *For each v, if all children of v have index label* 0*, then remove all descendants of v.*

1.4 *$[\tilde{G}, \tilde{B}]_{\tilde{I}}$ is such that for every $i \in \tilde{I}$*

$$\begin{aligned}\tilde{G}(i) &= \{\tilde{q} \in \tilde{Q} \mid \tilde{q} \textit{ contains a red node with name } i \\ &\qquad\qquad \textit{or does not contains a node with name } i\,\} \\ \tilde{B}(i) &= \{\tilde{q} \in \tilde{Q} \mid \tilde{q} \textit{ contains a green node with name } i\,\}\end{aligned}$$

Step (1.3.2.3.1) says that if a run in $v_{j'}$ fulfills the positive obligation $i_{j'}$ by visiting $B(i_{j'})$ ($L_s(v_{j'}) \cap B(i_{j'}) \neq \emptyset$), then the run moves into a new node v' ($L_s(v') = L_s(v_{j'}) \cap B(i_{j'})$), and v' continues to monitor if the run hits the next largest positive obligation in the witness set of its parent ($L_h(v') = \max([0..i_{j'}) \cap ((I \cup \{0\}) \setminus L_h^{set}(v)))$). Step (1.3.2.3.2) says that if this is not the case and the run in $v_{j'}$ also violates the negative obligation $i_{j'}$ by visiting $G(i_{j'})$ ($L_s(v_{j'}) \cap G(i_{j'}) \neq \emptyset$), then this run is *reset*, in the sense that the states in $L_s(v_{j'}) \cap G(i_{j'})$ moved into a new child of v.

4. Improved Streett Determinization

In this section, we show our improved construction for Streett determinization. Define $\mu = \min(n, k)$.

Improvement I.. The first idea is what we have applied to Streett complementation [15], namely, the larger the k, the more overlaps between $B(i)$'s and between $G(i)$'s ($i \in I$). Let us revisit the previous example, illustrated in Figure 1. Assume that $G(2) \subseteq G(3)$ (we say that $G(3)$ covers $G(2)$). If a run stays at v_3, then the run is not supposed to visit $G(3)$, or otherwise the run should have been reset by Step (1.3.2.3.2). Since the run cannot visit $G(2)$ either, there is no point to check if it is to visit $B(2)$, and hence v_3 does not need to have a child with index label 2 (in this case the node v_{32}). This simple idea already puts a cap on the size of *STS*. But it turns out that we can save the most if we exploit the redundancy on B instead of on G.

Reduction of Tree Size.. Step (1.3.2.3.1) at a non-leaf node v is to check, for every child v' of v, if a run visits $B(L_h(v'))$, and in the positive case, move the run into a new node. Let v' be a non-root node and v the parent of v'. Let $I_v = L_h^{set}(v)$. As Step (1.3.2) is executed recursively from top to bottom, it can be assured that at the moment of its arriving at v, for every $i \in I_v$, we have $L_s(v') \cap B(i) = \emptyset$. By an abuse of notation, we write $B(v)$ for $\cup_{j\in I_v} B(j)$, and hence we have $L_s(v') \cap B(v) = \emptyset$. Thus, there is no chance of missing a positive obligation even if we restrict L_h to be such that $B(L_h(v')) \not\subseteq B(v)$. It follows that each node "watches" at least one more state that has not been watched by its ancestors, and therefore along any path of an *STS*, there are at most μ nodes with non-zero index labels (recall that $\mu = \min(n,k)$ and note that the root is excluded as its index label is 0). Also, it can be shown by induction on tree height that an *STS* contains at most n nodes with index label 0, using the fact that set labels of sibling nodes are pairwise disjoint and the fact that if v is the parent of v' and $L_h(v') = 0$, then $L_s(v') \subset L_s(v)$. Therefore, the number of nodes in an *STS* is bounded by $n(\mu+1)$.

Reduction of Index Labels.. Let $I'_v = \{i \in I \mid B(i) \subseteq B(v)\}$. The above analysis tells us that $L_h(v') \in (I \setminus I'_v)$. However, we can further improve L_h such that there is no $j \in (I \setminus I'_v)$, $(B(j) \setminus B(v)) \subset (B(L_h(v')) \setminus B(v))$ and for any $j \in (I \setminus I'_v)$, $(B(j) \setminus B(v)) = (B(L_h(v')) \setminus B(v))$ implies $L_h(v') < j$. We say that $L_h(v')$ minimally extends $L_h^{\rightarrow}(v)$ if this condition holds.

To formalize the intuition of *minimal extension*, we introduce two functions $Cover : I^* \to 2^I$ and $Mini : I^* \to 2^I$ as in [15]. *Cover* maps finite sequences of I-elements to subsets of I such that

$$Cover(\alpha) = \{\, j \in I \;\mid\; B(j) \subseteq \bigcup_{i=1}^{|\alpha|} B(\alpha[i]) \,\}.$$

Note that $Cover(\epsilon) = \emptyset$. *Mini* also maps finite sequences of I-elements to subsets of I such that $j \in Mini(\alpha)$ if and only if $j \in I \setminus Cover(\alpha)$ and

$$\forall j' \in I \setminus Cover(\alpha) \left[j' \neq j \;\to\; \neg \left(B(j') \cup \bigcup_{i=1}^{|\alpha|} B(\alpha[i]) \;\subseteq\; B(j) \cup \bigcup_{i=1}^{|\alpha|} B(\alpha[i]) \right) \right], \tag{1}$$

$$\forall j' \in I \setminus Cover(\alpha) \left[j' < j \;\to\; \left(B(j') \cup \bigcup_{i=1}^{|\alpha|} B(\alpha[i]) \;\neq\; B(j) \cup \bigcup_{i=1}^{|\alpha|} B(\alpha[i]) \right) \right]. \tag{2}$$

$Mini(\alpha)$ consists of index candidates to *minimally* enlarge $Cover(\alpha)$; ties (with respect to set inclusion) are broken by numeric minimality (Condition (2)).

Mini plays a crucial role in reducing the combination of index labels. In the reduced Safra's trees (Definition 3), we identify a collection of paths such that each node appears on exactly one of those paths. The sequence of index labels on each of those paths corresponds to a path in a specific tree structure, which we refer to as *increasing tree of sets* (*ITS*) [15]. Counting the number of paths in *ITS* give us a better upper bound on the combination of index labels. Here we switch to an informal notation of labeled trees and we identify a node with the sequence of labels from the root to the node.

Definition 2 (Increasing Tree of Sets (*ITS*) [15]). *An ITS $\mathcal{T}(n,k,B)$ is an unordered I-labeled tree such that a node α exists in $\mathcal{T}(n,k,B)$ if and only if $\forall i \in [1..|\alpha|]$, $\alpha[i] \in Mini(\alpha[1..i))$.*

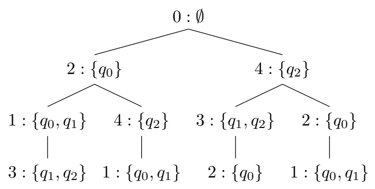


Figure 2: The *ITS* $\mathcal{T}(n,k,B)$ in Example 1. Note that $\{q_0,q_1\}$ and $\{q_1,q_2\}$ cannot appear in the first level because $\{q_0,q_1\}$ covers $\{q_0\}$ and $\{q_1,q_2\}$ covers $\{q_2\}$. The leftmost node at the bottom level is labeled by $\{q_1,q_2\}$ instead of by $\{q_2\}$ due to the index minimality requirement.

Several properties are easily seen from the definition. First, an *ITS* is uniquely determined by parameter n, k and B. Second, the length of the longest path in $\mathcal{T}(n,k,B)$ is bound by μ. Third, if β is a direct child of α, then β must contribute at least one new element that has not been seen from the root to α. Forth, the new contributions made by β cannot be covered by contributions made by another sibling β', with ties broken by selecting the one with smallest index. As $B : I \to 2^Q$ is one to one, we also view *ITS* as 2^Q-labeled trees.

Example 1 (*ITS*)**.** *Consider* $n = 3$*,* $k = 4$*,* $Q = \{q_0,q_1,q_2\}$*, and* $B : [1..4] \to 2^Q$ *such that*

$$B(1) = \{q_0,q_1\}, \qquad B(2) = \{q_0\}, \qquad B(3) = \{q_1,q_2\}, \qquad B(4) = \{q_2\}\,.$$

Figure 2 shows the corresponding $\mathcal{T}(n,k,B)$*.*

Improvement II.. The second idea is a batch-mode naming scheme to reduce name combinations. As shown before, an *STS* can have $n(\mu+1)$ nodes, which translates to $(n(\mu+1))!$ name combinations according to the current "first-come-first-serve" naming scheme, that is, picking an unused name when a new node is created and recycling the name when a node is removed. When $k = O(n)$, the naming cost is higher than all other complexity factors combined, as $(n(\mu+1))! = 2^{O(nk \lg nk)}$. However, this can be overcome by dividing the name space into even buckets and "wholesaling" buckets to specific paths in an *STS*.

Reduction of Names.. Let t be an *STS*. A *left spine* (left spine) is a maximal path $l = v_1 \cdots v_m$ such that v_m is a leaf, for any $i \in [2..m]$, v_i is the left-most child of v_{i-1}, and v_1 is not a left-most child of its parent. We call v_1 the head of l. Let $head : V \to V$ be such that $head(v) = v'$ if v' is the head of the left spine where v belongs to. We say that l is the i-th left spine of t if v_m is the i-th leaf of t, counting from left to right. It is clear that a tree with m leaves has m left spine, and every node is on exactly one left spine. Thus, an *STS* has at most n left spine. For this renaming scheme to work, we require that a newly created node be added to the right of all existing siblings of the same L_h-value. If a non-head node on a left spine has index label 0, then so should all of its siblings. But then all of them should have been removed. Therefore, only the head of a left spine can have index label 0, which means that a left spine can have at most $\mu + 1$ nodes.

We use $n(\mu + 1)$ names and divide them evenly into n buckets. Let b_i ($i \in [1..n]$) denote the i-th bucket, $b_{ij} = (\mu+1)(i-1)+j$ ($i \in [1..n], j \in [1..\mu+1]$) the j-th name in the i-th bucket. We say that $b_{i1} = (\mu + 1)(i-1) + 1$ is the *initial value* of b_i. Our naming strategy is as follows. Every left spine $l = v_1 \cdots v_m$ in an STS is associated with a name bucket b and $v_1, \cdots, v_m$ are assigned names continuously from the initial value of b. For example, if l is associated with bucket b_t, then $L_n(v_i) = (\mu+1)(t-1)+i$ for $i \in [1..m]$. The bucket association for each left spine in an STS t can be viewed as selection function $bucket : V \to [1..n]$ such that $bucket(v) = i$ if node v is assigned a name in the i-th bucket.

This naming strategy, however, comes with a complication; what if a leftmost sibling v is removed (due to Step (1.3.2.3)), and the second leftmost sibling v' (if exists) and all nodes belonging to the left spine of which v' is the head, "graft into" the left spine that v belongs to? The answer is that at the end of tree transformation, we need to rename those nodes that have moved into another left spine. If a tree transformation turns t into t', and during the process a node v joins another left spine l in t' (which is also in t before the transformation), then we rename v to a name in the bucket that l uses in t, and recycle the bucket with which v was associated in t.

Example 2 (New Naming Scheme)**.** *Figure 3 illustrates the changes of names in a sequence of tree transformations. We assume that there are* 10 *buckets* $b_1 - b_{10}$*, each of which is of size* 4*. Nodes in the graphs are denoted in the form* $v : L_n(v)$*; all other types of labels are omitted for simplicity.*

Definition 3 (Reduced Safra Trees for Streett Determinization (μSTS))**.** *A reduced Safra tree for Streett determinization (μSTS) is an STS* $\langle V, L\rangle$ *with* $L = \langle L_n, L_s, L_c, L_h\rangle$ *that satisfies the following additional conditions:*

3.1 Condition on L_h. *For each node v, if $Mini(L_h^{\to}(v)) \neq \emptyset$, then v is not a leaf node and for any child v' of v, $L_h(v') \in Mini(L_h^{\to}(v))$.*

3.2 Condition on L_n. *There exists a function $bucket : V \to [1..n]$ such that for every left spine $v_1 \cdots v_m$, we have $bucket(v_i) = bucket(v_j)$ for $i, j \in [1..m]$ and $L_n(v_i) = (\mu + 1)(bucket(v_i) - 1) + i$ for $i \in [1..m]$.*

As before, nodes in a μSTS are partially ordered by *older-than* relation. But we impose an additional *structural ordering* on nodes, that, for any two sibling v and v', v' is placed to the right of v if and only if $L_h(v) > L_h(v')$, or $L_h(v) = L_h(v')$ and v is *older than* v'. This structural ordering is needed for our renaming scheme (see the proof of Theorem 2).

Condition (3.2), together with the requirement that L_n is injective (Condition (1.1)), guarantees that no two distinct left spine in a tree share a bucket. Condition (3.1) says that every μSTS has fully grown left spines, that is, no leaf v can be further extended, as $Mini(L_h^{\to}(v)) = \emptyset$. To achieve this, we need the following procedure applied as the last step of each tree transformation.

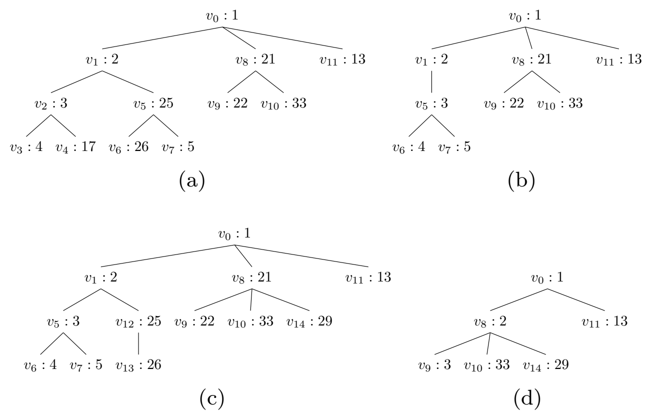


Figure 3: (a) shows an STS with 7 left spine: $l_1 : v_0v_1v_2v_3$, $l_2 : v_4$, $l_3 : v_5v_6$, $l_4 : v_7$, $l_5 : v_8v_9$, $l_6 : v_{10}$ and $l_7 : v_{11}$, associated with buckets b_1, b_5, b_7, b_2, b_6, b_9 and b_4, respectively. (b) shows the resulting STS after removing v_2 and its descendants. Nodes v_5 and v_6 migrate into l_1, and accordingly their name bucket b_7 is recycled. Also recycled is bucket b_5 due to the deletion of v_4. (c) shows the resulting STS after adding nodes v_{12}, v_{13} and v_{14} and forming two new left spine: $v_{12}v_{13}$ and v_{14}. Here v_{12} and v_{13} reuse the previously recycled bucket b_7, while node v_{14} takes an unused bucket b_8. (d) shows the resulting STS after removing v_1 and its descendants. Nodes v_8 and v_9 migrate into l_1 and take names 2 and 3, respectively. Buckets b_2, b_7 and b_6 are recycled accordingly.

Procedure 2 (Grow Left Spine)**.** *Repeat the following procedure until no new nodes can be added: if v is a leaf and $Mini(L_h^{\rightarrow}(v)) \neq \emptyset$, add a new child v' to v with $L_s(v') = L_s(v)$, $L_h(v') = \max(Mini(L_h^{\rightarrow}(v)))$, $L_c(v') = red$, and $L_n(v') = L_n(v)+1$.*

We note that the requirement that a μSTS has fully grown left spines is not essential; we can "grow" a μSTS "on-the-fly" as in [9, 11]. But this requirement simplifies the analysis on the number of combinations of index labels (see the proof of Theorem 2).

Rule 2 (Naming and Coloring on μSTS)**.** *Naming and Coloring convention for μSTS is the one for STS plus the following.*

2.1 Nodes in a left spine, from the head downwards, are assigned continuously increasing names, starting from the initial value of a bucket.

2.2 When a left spine is created, nodes in the left spine are assigned names from an unused name bucket; when a left spine is removed (which only happens when its head is removed), the name bucket of the left spine is recycled.

2.3 When a left spine l is grafted into another left spine l', the name bucket of l is recycled and nodes on l are renamed according to (1), as if they were on l' originally

2.4 Renamed nodes are marked red.

Procedure 3 (Improved Streett Determinization)**.** *Let $\mathcal{A} = \langle Q, Q_0, \Sigma, \Delta, \langle G, B\rangle_I\rangle$ be a nondeterministic Streett automaton. This procedure outputs a deterministic Rabin automaton $\mathcal{B} = \langle \tilde{Q}, \tilde{q}_0, \Sigma, \tilde{\Delta}, [\tilde{G}, \tilde{B}]_{\tilde{I}}\rangle$ ($\tilde{I} = [1..n(\mu+1)]$) such that*

3.1 $\tilde{Q}$ is the set of μSTS.

3.2 $\tilde{q}_0 \in \tilde{Q}$ is $\tilde{q}_0$ is a tree that is just a fully grown left spine obtained by growing the single root tree (which is defined in Procedure 1) according to Procedure (2).

3.3 $\tilde{\Delta} : \tilde{Q} \times \Sigma \to \tilde{Q}$ is the transition function such that $\tilde{q}' = \tilde{\Delta}(\tilde{q}, a)$ is the STS obtained by applying the following transformation rule to $\tilde{q}$.

3.3.1 Subset Construction. *For each node v in $\tilde{Q}$, update set label $L_s(v)$ to $\Delta(L_s(v), a)$.*

3.3.2 Expansion. *Apply the following transformations to non-leaf nodes recursively from the root. Let v be a node with j children $v_1, \ldots, v_j$ with $i_1, \ldots, i_j$ as the corresponding index labels. Consider the following cases for each $j' \in [1..j]$.*

3.3.2.1 If $L_s(v_{j'}) \cap B(i_{j'}) \neq \emptyset$. Add a child v' to v with $L_s(v') = L_s(v_{j'}) \cap B(i_{j'})$ and $L_h(v') = \max([0..i_{j'}) \cap (\{0\} \cup Mini(L_h^{\rightarrow}(v))))$, and remove the states in $L_s(v_{j'}) \cap B(i_{j'})$ from $v_{j'}$ and all the descendants of $v_{j'}$.

3.3.2.2 If $L_s(v_{j'}) \cap B(i_{j'}) = \emptyset$ and $L_s(v_{j'}) \cap G(i_{j'}) \neq \emptyset$. Add a child v' to v with $L_s(v') = L_s(v_{j'}) \cap G(i_{j'})$ and $L_h(v') = L_h(v)$, and remove the states in $L_s(v_{j'}) \cap G(i_{j'})$ from $v_{j'}$ and all the descendants of $v_{j'}$.

3.3.3 Horizontal Merge. *For any state q and any two siblings v and v' such that $q \in L_s(v) \cap L_s(v')$, if $L_h(v) < L_h(v')$, or $L_h(v) = L_h(v')$ and v is* older than *v', then remove q from v' and all its descendants.*

3.3.4 Vertical Merge. *For each v, if all children of v have index label* 0*, then remove all descendants of v.*

3.3.5 Grow the tree fully according to Procedure (2).

3.4 $[\tilde{G}, \tilde{B}]_{\tilde{I}}$ is such that for every $i \in \tilde{I}$

$$\tilde{G}(i) = \{\tilde{q} \in \tilde{Q} \mid \tilde{q} \textit{ contains a red node with name } i$$
$$\textit{or does not contains a node with name } i\,\}$$
$$\tilde{B}(i) = \{\tilde{q} \in \tilde{Q} \mid \tilde{q} \textit{ contains a green node with name } i\,\}$$

Note that besides naming and renaming, the only major difference between Procedures 3 and 1 is the way of selecting the next positive obligation. In Step (3.3.2.1), $Mini(L_h^{\rightarrow}(v))$ is used in the calculation, while in Step (1.3.2.3.1), $L_h^{set}(v)$ is used.

Theorem 1 (Streett Determinization: Correctness)**.** *Let $\mathcal{A} = \langle Q, Q_0, \Sigma, \Delta, \langle G, B\rangle_I\rangle$ be a Streett automaton with $|Q| = n$ and $I = [1..k]$, and $\mathcal{B} = \langle \tilde{Q}, \tilde{q}_0, \Sigma, \tilde{\Delta}, [\tilde{G}, \tilde{B}]_{\tilde{I}}\rangle$ the deterministic Rabin automaton obtained by Procedure 3. We have $\mathscr{L}(\mathcal{A}) = \mathscr{L}(\mathcal{B})$.*

Proof. ($\mathscr{L}(\mathcal{A}) \subseteq \mathscr{L}(\mathcal{B})$). This part of proof is almost identical to the one in [9]. We ought to show that if $\varrho = \tilde{q}_0\tilde{q}_1 \cdots$ is an run of $\mathcal{B}$ over an infinite word $w = w_0w_1 \ldots \in \mathscr{L}(\mathcal{A})$, then (1) a node v exists in every state in ϱ from some point on, (2) v turns green infinitely often, and (3) v has a fixed name $i \in \tilde{I}$. The argument in [9] guarantees the existence of such a node v with the first two properties. The only complication comes from renaming. We have the situation that v with name i exists in $\tilde{q}_j$, but it is renamed to i' in the following state $\tilde{q}_{j+1}$. This happens when v is on a left spine l whose head is a second left-most sibling in $\tilde{q}_j$ and the corresponding leftmost sibling is removed in $\tilde{q}_{j+1}$, resulting in nodes on l (including v) joining another left spine l' in $\tilde{q}_{j+1}$. However, such "grafting" can only happen to v finitely many times, as each time the height of the head of l' is strictly smaller than the height of the head of l. Therefore, v is eventually assigned a fixed name i, which gives us the third property.

($\mathscr{L}(\mathcal{B}) \subseteq \mathscr{L}(\mathcal{A})$). We ought to show that if $w = w_0w_1 \ldots \in \mathscr{L}(\mathcal{B})$, then the run $\varrho = \tilde{q}_0\tilde{q}_1 \cdots$ of $\mathcal{B}$ over w induces an accepting run of $\mathcal{A}$ over w. The assumption that ϱ is accepting means that there exists an $i \in \tilde{I}$ such that ϱ eventually never visits $\tilde{G}(i)$, but visits $\tilde{B}(i)$ infinitely often, or equivalently, i names a green or yellow node in every state in a suffix of ϱ and there are infinitely many occurrences when the nodes named by i are green. Since renamed nodes are marked red, all nodes named by i in the suffix have to be the same one. It follows that a node v eventually stays in every state in a suffix of ϱ and v turns green infinitely often. The rest of the proof is the same as the one in [9], with the help of Lemma 2 and the fact that using *Mini* to select the index labels of the children of v is sound and complete. $\square$

5. Complexity

In this section we state the complexity results for the determinization of Streett, generalized Büchi, parity and Rabin automata. The corresponding proofs are given in the appendix.

Theorem 2 (Streett Determinization: Complexity). *Let $\mathcal{A} = \langle Q, Q_0, \Sigma, \Delta, \langle G, B\rangle_I\rangle$ be a Streett automaton with $|Q| = n$ and $I = [1..k]$, and $\mathcal{B} = \langle \tilde{Q}, \tilde{q}_0, \Sigma, \tilde{\Delta}, [\tilde{G}, \tilde{B}]_{\tilde{I}}\rangle$ the deterministic Rabin automaton obtained by Procedure 3. For any k, $|\tilde{Q}| \leq 2^{12n}(0.37n)^{n^2+8n} = 2^{O(n^2 \lg n)}$ and $|\tilde{I}| = O(n^2)$. If $k = O(n)$, we have $|\tilde{Q}| = 2^{O(n \lg n + nk \lg k)}$ and $|\tilde{I}| = O(nk)$.*

Figure 4 breaks down the total cost into five categories and compares our construction with previous ones in each category. It turns out that the complexity analysis can be easily adapted for generalized Büchi and parity automata as they are subclasses of Streett automata.

†In Piterman's construction, the number of ordered trees times the number of name combinations is bounded by $(nk)^{nk}$. However, the second factor is still the dominating one, costing $2^{\Omega(nk \lg nk)}$. Also, there is no notion of color in Piterman's construction. Instead, each tree is associated with two special names both ranging from 1 to nk, resulting in the cost $O((nk)^2)$.

Cost	Safra's	Piterman's	Ours	
Ordered Tree	$2^{O(nk)}$	$2^{O(nk)}$	$2^{O(n\lg n)}$	
Name	$2^{O(nk\lg nk)}$	$2^{O(nk\lg nk)\dagger}$	$2^{O(n\lg n)}$	
Color	$2^{O(nk)}$	$O((nk)^2)^\dagger$	$2^{O(n\lg k)}$	$k = O(n)$
			$2^{O(n\lg n)}$	$k = \omega(n)$
Set Label	$2^{O(n\lg n)}$	$2^{O(n\lg n)}$	$2^{O(n\lg n)}$	
Index Label	$2^{O(nk\lg nk)}$	$2^{O(nk\lg nk)}$	$2^{O(n\lg n+nk\lg k)}$	$k = O(n)$
			$2^{O(n^2\lg n)}$	$k = \omega(n)$
Total	$2^{O(nk\lg nk)}$	$2^{O(nk\lg nk)}$	$2^{O(n\lg n+nk\lg k)}$	$k = O(n)$
			$2^{O(n^2\lg n)}$	$k = \omega(n)$

Figure 4: Cost breakdown of Streett determinization.

Corollary 1 (Generalized Büchi Determinization: Complexity)**.** *Let $\mathcal{A}$ be a generalized Büchi automaton with state size n and index size k. There is an equivalent deterministic Rabin automaton $\mathcal{B}$ with state size $2^{O(n\lg nk)}$. The index size is $O(nk)$ if $k = O(n)$ and $O(n^2)$ if $k = \omega(n)$.*

Corollary 2 (Parity Determinization: Complexity)**.** *Let $\mathcal{A}$ be a parity automaton with state size n and index size k. There is an equivalent deterministic Rabin automaton $\mathcal{B}$ with state size $2^{O(n\lg n)}$ and index size $O(nk)$.*

The determinization of a Rabin automaton with acceptance condition $[G, B]_I$ (the dual of $\langle G, B\rangle_I$ for $I = [1..k]$) can be straightforwardly obtained by running, in parallel, k modified Safra trees, each of which monitors runs for an individual Rabin condition $[G(i), B(i)]$ $(i \in I)$.

Theorem 3 (Rabin Determinization: Complexity)**.** *Let $\mathcal{A}$ be a Rabin automaton with state size n and index size k. There is an equivalent deterministic Rabin automaton $\mathcal{B}$ with state size $2^{O(nk\lg n)}$ and index size $O(nk)$.*

We note that it is unlikely that there exists a Safra-tree style determinization for Rabin automata, because an analogue of Lemma 2 fails due to the existential nature of Rabin acceptance conditions.

6. Concluding Remarks

In this paper we improved Safra's construction and obtained tight upper bounds on state complexity for the determinization of Streett, generalized Büchi, parity and Rabin automata. Combining these with the lower bound results in [14] and previous findings in the literature, we now have a complete characterization of determinization complexity of ω-automata of common types. Figure 5 summarizes the determinization complexities for ω-automata of common types..

Our results show an interesting phenomenon that in the asymptotic notation $2^{\Theta(\cdot)}$, complementation complexity is identical to determinization complexity. The

Type	Bound		Lower	Upper
Büchi	$2^{\Theta(n \lg n)}$		[7]	[6]
Generalized Büchi	$2^{\Theta(n \lg nk)}$		[20]	[24]
Streett	$2^{\Theta(n \lg n + nk \lg k)}$	$k = O(n)$	[14]	[15]
	$2^{\Theta(n^2 \lg n)}$	$k = \omega(n)$		
Rabin	$2^{\Theta(nk \lg n)}$		[13]	[24]
Parity	$2^{\Theta(n \lg n)}$		[7]	[15]

Figure 5: Complementation and determinization complexities for ω-automata of common types. The listed citations are meant for complementation only.

same phenomenon happens to finite automata on finite words. We believe it is worth investigating the reason behind this phenomenon.

As mentioned earlier, determinization procedures that output parity automata, like Piterman's constructions, are preferable to the classic ones that output Rabin automata. We plan to investigate how to combine Piterman's node renaming scheme with ours to obtain determinization procedures that output parity automata with optimal state complexity.

Appendix A. Complexity of Streett Determinization

Theorem 2 (Streett Determinization: Complexity). Let $\mathcal{A} = \langle Q, Q_0, \Sigma, \Delta, \langle G, B\rangle_I\rangle$ be a Streett automaton with $|Q| = n$ and $I = [1..k]$, and $\mathcal{B} = \langle \tilde{Q}, \tilde{q}_0, \Sigma, \tilde{\Delta}, [\tilde{G}, \tilde{B}]_{\tilde{I}}\rangle$ the deterministic Rabin automaton obtained by Procedure 3. For any k, $|\tilde{Q}| \leq 2^{12n}(0.37n)^{n^2+8n} = 2^{O(n^2 \lg n)}$ and $|\tilde{I}| = O(n^2)$. If $k = O(n)$, we have $|\tilde{Q}| = 2^{O(n \lg n + nk \lg k)}$ and $|\tilde{I}| = O(nk)$.

Proof. As mentioned before, a μSTS is a labeled and ordered tree with at most $n(\mu + 1)$ nodes. Also note that the *older-than* ordering can be subsumed into the structural ordering as we only compare ages between two siblings with the same L_h-value (Step (3.3.3)). Let $T(e, l)$ denote the number of ordered trees with e edges and l leaves. $T(e, l)$ are called *Narayana numbers*, which for $e, l \geq 1$, assume the following closed form [25, 26]:

$$T(e, l) = \frac{1}{e}\binom{e}{l}\binom{e}{l-1}.$$

Let $OT(n, \mu)$ be the set of ordered trees with at most n leaves and $n(\mu + 1)$ nodes. We have

$$|OT(n, \mu)| \leq \sum_{e=1}^{n\cdot(\mu+1)} \sum_{l=1}^{\min(n,e)} T(e, l) \leq n^3 \cdot T(n^2, n) \leq n\binom{n^2}{n}\binom{n^2}{n-1} \leq n^{4n}.$$

A μSTS is a tree in $OT(n, \mu)$ augmented with four components: name, color, set label, and index label. By Condition (1.2), the set labels of nodes are completely determined by the set labels of leaves. There are at most n^n ways to label leaves, each of which corresponds to a function from $[1..n] \to Q$.

Let us consider index labels on each left spine. Recall that an *ITS* $\mathcal{T}(n, k, B)$ is an unordered I-labeled tree such that a node α exists in $\mathcal{T}(n, k, B)$ if and only if $\forall i \in [1..|\alpha|]$, $\alpha[i] \in \mathit{Mini}(\alpha[1..i))$ (Definition 2). As stated before, only the head of a left spine can be labeled by 0. As each left spine is fully grown, the sequence of index labels from the leaf of a left spine to the head of the left spine corresponds to a path from a leaf going upwards in $\mathcal{T}(n, k, B)$, and hence in turn corresponds to a leaf in $\mathcal{T}(n, k, B)$ as the length of each left spine is fixed. Let $\mathbb{T}(n, k)$ denote the class of all *ITS* with fixed n and k. Let $l(n, k)$ be the maximum number of leaves an *ITS* in $\mathbb{T}(n, k)$ can have. Thus, the number of choices for each sequence of index labels on a left spine in an μSTS is bounded by $l(n, k)$, and the number of ways of index labeling an μSTS is bounded by $(l(n, k))^n$, which, by Lemma 3, is no more than $(n!)^n$, and is no more than $k^{n\mu}$ for $k = O(n)$.

The names along a left spine from the head downwards is continuously increasing from the initial value of a bucket. So they are completely determined once a bucket is chosen. There are at most n buckets, and therefore the number of ways to name a μSTS is bounded by $n!$.

The colors on a left spine also have some regularity. It is not hard to see that there is at most one green node in a left spine and all red nodes must be descendants of yellow nodes. So there are at most $(\mu+1)^2$ choices to color a left spine and hence the number of color combinations on an μSTS is bounded by $(\mu+1)^{2n}$.

Put all together, for any k, we have state complexity

$$\begin{aligned} n^{4n} \cdot n^n \cdot (n!)^n \cdot n! \cdot (\mu+1)^{2n} &\leq n^{4n} \cdot n^n \cdot (n!)^n \cdot n! \cdot (\mu+1)^{2n} \\ &\leq (n+1)^{7n} \cdot (0.37n)^{n^2+n} \\ &\leq 2^{12n} \cdot (0.37n)^{n^2+8n} = 2^{O(n^2 \lg n)}, \end{aligned}$$

and for $k = O(n)$, we have $\mu = O(k)$ and hence the state complexity

$$n^{4n} \cdot n^n \cdot (k)^{n\mu} \cdot n! \cdot (\mu+1)^{2n} = n^{8n} \cdot (k)^{n\mu} = 2^{O(n \lg n + nk \lg k)}.$$

Figure 4 compares the costs attributed by each component of STS between the previous constructions and ours. $\square$

Corollary 1 (Generalized Büchi Determinization: Complexity). Let $\mathcal{A}$ be a generalized Büchi automaton with state size n and index size k. There is an equivalent deterministic Rabin automaton $\mathcal{B}$ with state size $2^{O(n \lg nk)}$. The index size is $O(nk)$ if $k = O(n)$ and $O(n^2)$ if $k = \omega(n)$.

Proof. Let $\mathcal{A} = \langle Q, Q_0, \Sigma, \Delta, \langle B \rangle_I \rangle$. The acceptance condition $\langle B \rangle_I$ can be viewed as a Streett condition $\langle G, B \rangle_I$ where $G(i) = Q$ for every $i \in I$. In this case, the ITS is just a one-level tree (not counting the root). $l(n,k)$ is then bounded by k and the number of ways of putting index labels is bounded by $(l(n,k))^n = k^n$. Other complexity factors are all bounded by $n^{O(n)}$. So the total number of μSTS is bounded by $k^n \cdot n^{O(n)} = 2^{O(n \lg nk)}$. $\square$

Corollary 2 (Parity Determinization: Complexity). Let $\mathcal{A}$ be a parity automaton with state size n and index size k. There is an equivalent deterministic Rabin automaton $\mathcal{B}$ with state size $2^{O(n \lg n)}$ and index size $O(nk)$.

Proof. Let $\mathcal{A} = \langle Q, Q_0, \Sigma, \Delta, \langle G, B \rangle_I \rangle$. Because $B_1 \subset \cdots \subset B_k$, the ITS is just a directed line. As a result, there is no index combinations and other complexity factors are all bounded by $n^{O(n)}$. So we have $2^{O(n \lg n)}$. $\square$

Theorem 3 (Rabin Determinization: Complexity). Let $\mathcal{A}$ be a Rabin automaton with state size n and index size k. There is an equivalent deterministic Rabin automaton $\mathcal{B}$ with state size $2^{O(nk \lg n)}$ and index size $O(nk)$.

Proof. This theorem is a folklore, but here we sketch a proof just for the sake of completeness. Let $\mathcal{A} = \langle Q, Q_0, \Sigma, \Delta, [G, B]_I \rangle$. For each fixed i, $\mathcal{A}_i = \langle Q, Q_0, \Sigma, \Delta, [G(i), B(i)] \rangle$ is a Rabin automaton. $\mathcal{A}_i$ can be transformed into a Büchi automaton in $O(n)$ state, which is equivalent to a deterministic Rabin automaton $\mathcal{B}_i$ with $2^{O(n \lg n)}$ states. The desired $\mathcal{B}$ is just a product of $(\mathcal{B}_i)_{i \in I}$. Let $\mathcal{B}_i = \langle Q^{(i)}, q_0^{(i)}, \Sigma, \Delta_i, [G^{(i)}, B^{(i)}]_{I^{(i)}} \rangle$. We define $\mathcal{B} = \langle \tilde{Q}, \tilde{q}_0, \Sigma, \tilde{\Delta}, [\tilde{G}, \tilde{B}]_{\tilde{I}} \rangle$ where

$$\tilde{Q} = Q^{(1)} \times \cdots \times Q^{(k)},$$

$$\begin{aligned}
\tilde{q}_0 &= \langle q_0^{(1)}, \ldots, q_0^{(k)} \rangle, \\
\tilde{\Delta}(\langle q^{(1)}, \ldots, q^{(k)} \rangle, a) &= \langle \Delta_0(q^{(1)}, a), \ldots, \Delta_k(q^{(k)}, a) \rangle, \\
\tilde{I} &= I^{(1)} \times \{1\} \cup \cdots \cup I^{(k)} \times \{k\}, \\
\tilde{G}(\langle i_j, i \rangle) &= Q^{(1)} \times \cdots \times G^{(i)}(i_j) \times \cdots \times Q^{(k)}, \\
\tilde{B}(\langle i_j, i \rangle) &= Q^{(1)} \times \cdots \times B^{(i)}(i_j) \times \cdots \times Q^{(k)}.
\end{aligned}$$

It is easily to verify that $\mathcal{B}$ is deterministic, $\mathscr{L}(\mathcal{B}) = \mathscr{L}(\mathcal{A})$, the state size of $\mathcal{B}$ is $2^{O(nk \lg n)}$ and the index size of $\mathcal{B}$ is $O(nk)$. $\square$

Appendix B. Size of *ITS*

Let $\mathbb{T}(n, k)$ denote the class of all *ITS* with fixed n and k. Let $l(n, k)$ be the maximum number of leaves an *ITS* in $\mathbb{T}(n, k)$ can have.

Lemma 3 ([15])**.** *For any k, $l(n, k) \leq n!$ and $l(n, k) \leq k^\mu$ for $k = O(n)$.*

Proof. Let $l(n, k, B)$ denote the number of leaves in $\mathcal{T}(n, k, B)$. Let $B' : I' \to 2^Q$ where $I' = [1..k']$ be an extension of B such that $\mathit{range}(B')$ contains all singletons from Q, i.e.,

$$\begin{aligned}
&\forall i \in [1..k]\ B'(i) = B(i)\,, \\
&\{\{q\} \mid q \in Q\} \subseteq \mathit{range}(B')\,.
\end{aligned}$$

By Lemma 4, we assume without loss of generality that $[1..n]$ name the n singletons such that $B'(i) = q_{i-1}$ for $i \in [1..n]$. Because of existence of all singletons, the minimal extension at each node is always done by adding singletons. Formally, for any index sequence α, $\mathit{Mini}(\alpha) \subseteq [1..n]$. Therefore, each path from the root to a leaf in $\mathcal{T}(n, k', B')$ corresponds to a permutation of $[1..n]$ and vice versa. So we have $l(n, k', B') = n!$. By Lemma 5, $l(n, k, B) \leq l(n, k', B')$. As B is chosen arbitrarily, we have $l(n, k) \leq n!$.

For the special case where $k = O(n)$, we have $l(n, k, B) \leq k^\mu$ as the height of $\mathcal{T}(n, k, B)$ is bound by $\mu = \min(n, k)$ and the maximum branching of $\mathcal{T}(n, k, B)$ is bound by k. Since B is chosen arbitrarily, we have $l(n, k) \leq k^\mu$. $\square$

Lemma 4. *Let $B : I \to 2^Q$, $B' : I \to 2^Q$ be two injective functions such that $\mathit{range}(B) = \mathit{range}(B')$. Then $|\mathcal{T}(n, k, B)| = |\mathcal{T}(n, k, B')|$.*

Proof. The condition $\mathit{range}(B) = \mathit{range}(B')$ means that B and B' just name subsets of Q differently. We extend B, B' to functions from I^* to 2^Q such that for $\alpha \in I^*$,

$$B(\alpha) = \bigcup_{i=1}^{|\alpha|} B(\alpha[i])\,, \qquad\qquad B'(\alpha) = \bigcup_{i=1}^{|\alpha|} B'(\alpha[i])\,.$$

By Definition 2, children of a node α in an *ITS* is completely determined by $B(\alpha)$. So a node α in $\mathcal{T}(n, k, B)$ has the same number of children as a node α' in $\mathcal{T}(n, k, B')$

if $B(\alpha) = B'(\alpha')$. Moreover, if $\alpha_1, \ldots, \alpha_j$ are children of α and $\alpha'_1, \ldots, \alpha'_j$ are children of α', Then $B(\alpha_1), \ldots, B(\alpha_j)$ are just a permutation of $B'(\alpha'_1), \ldots, B'(\alpha'_j)$. By induction on tree height, there is a one-to-one correspondence between nodes in $\mathcal{T}(n,k,B)$ and nodes in $\mathcal{T}(n,k,B')$. Thus $|\mathcal{T}(n,k,B)| = |\mathcal{T}(n,k,B')|$. $\square$

Lemma 5. *Let $\mathcal{T}(n,k,B)$ and $\mathcal{T}(n,k',B')$ be two ITS such that B' extends B by naming a singleton that B does not name. Then $|\mathcal{T}(n,k,B)| \le |\mathcal{T}(n,k',B')|$.*

Proof. It suffices to show that $|\mathcal{T}(n,k,B)| \le |\mathcal{T}(n,k',B')|$ when $k' = k+1$, $I' = [1..k']$, $B'(i) = B(i)$ for $i \in I$ and $B'(k+1)$ names a new singleton. Without loss of generality we assume $B'(k+1) = \{q_0\}$. We show a tree transformation θ that turns $\mathcal{T}(n,k,B)$ into $\mathcal{T}(n,k+1,B')$.

Recall that we identify a node with the path from root to the node. By a node α, we mean $\alpha[1], \ldots, \alpha[|\alpha|]$ are labels on the path and $\alpha[|\alpha|]$ is the label of α. Let T_α denote the subtree rooted at α and $B(\alpha)$ be

$$\bigcup_{i=1}^{|\alpha|} B(\alpha[i])\,.$$

We say that a state $q \in Q$ is named by α if $q \in B(\alpha)$. We define θ on a subtree T_α as follows.

Case 1. The new state q_0 is named by α. In this case, $\theta(T_\alpha) = T_\alpha$.

Case 2. The new state q_0 has not been named by α. Let us assume that $\{q_0\}$ is named by α's children $\alpha_1, \ldots, \alpha_j$ ($j = 0$ means no children names $\{q_0\}$). Let $\alpha_{j+1}, \ldots, \alpha_{j'}$ be the rest children of α. Formally, we have

$$\begin{aligned}
&q_0 \notin B(\alpha)\,,\\
&q_0 \in B(\alpha_i)(i \in [1..j], j \ge 0)\,,\\
&q_0 \notin B(\alpha_i)(i \in [j+1..j'], j \ge 0)\,.
\end{aligned}$$

We have two sub-cases to consider.

Case 2.1. For some $i \in [1..j]$, $B(\alpha_i) \cup B(\alpha) = \{q_0\} \cup B(\alpha)$. In this case we must have $j = 1$. Let $\theta(T_\alpha)$ be a tree obtained from T_α by replacing the label of α_1 by $k+1$.

Case 2.2. For all $i \in [1..j]$, $B(\alpha_i) \cup B(\alpha) \supset \{q_0\} \cup B(\alpha)$. In this case let $\theta(T_\alpha)$ be a tree obtained from T_α by the following procedure.

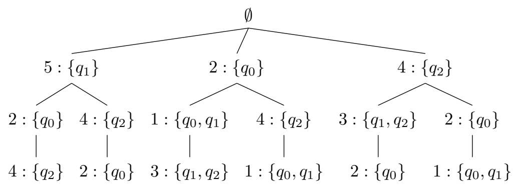
1. Add to α a new leaf α' labeled with $k+1$,
2. Remove subtrees $\alpha_1, \ldots, \alpha_j$ from α and make them children of α'
3. Add to α' $j'-j$ new leaves $\alpha'_1, \ldots, \alpha'_{j'-j}$, respectively, labeled with
$$\alpha_{j+1}[|\alpha_{j+1}|], \ldots, \alpha_{j'}[|\alpha_{j'}|].$$
4. Grow the new $j'-j$ leaves to full *ITS* according to *Mini* defined with respect to n, k' and B'.

In any case, $\theta(T_\alpha)$ is an *ITS* with root labeled with $\alpha[|\alpha|]$. It is clear from the above procedure that $|\theta(T_\alpha)| \geq |T_\alpha|$. Now let $\Theta(T)$ a tree obtained from T by applying θ on T level by level, from top to bottom. Example 3 shows such a transformation. It is not hard to verify that $\mathcal{T}(n, k', B') = \Theta(\mathcal{T}(n, k, B))$. Therefore, we have $|\mathcal{T}(n, k, B)| \leq |\mathcal{T}(n, k', B')|$. $\square$

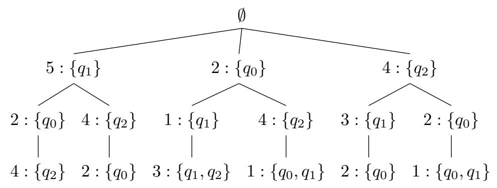
Example 3. *Consider the same setup:* $n = 3$, $k = 4$, $Q = \{q_0, q_1, q_2\}$, *and* $B : [1..4] \to 2^Q$ *such that*

$$B(1) = \{q_0, q_1\} \qquad B(2) = \{q_0\} \qquad B(3) = \{q_1, q_2\} \qquad B(4) = \{q_2\}$$

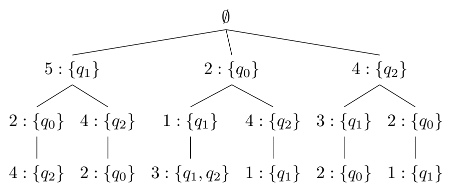
as in Example 1. Let $T_0 = \mathcal{T}(n, k, B)$. *Let* $k' = 5$, $B' = B \cup \{\langle 5, \{q_1\}\rangle$. *Applying* θ *to the root of* T_0 *we have* T_1*:*



Applying θ *to nodes* $\langle 2\rangle$ *and* $\langle 4\rangle$ *in* T_1, *we have* T_2*:*



Applying θ *to nodes* $\langle 2, 4\rangle$ *and* $\langle 4, 2\rangle$ *in* T_2, *we have* T_3*:*



And no more application of θ *is possible. It is not hard to verify that* $T_3 = \mathcal{T}(n, k + 1, B')$.

References

[1] Y. Choueka, Theories of automata on ω-types: a simplified appraoch, Journal of Computer and System Science 8 (1974) 117–141.

[2] M. Y. Vardi, The Büchi complementation saga, in: Proceedings of the 24th International Symposium on Theoretical Aspects of Computer Science (STACS 2007), Springer-Verlag, 2007, pp. 12–22.

[3] Y. Gurevich, L. Harrington, Trees, automata, and games, in: Proceedings of the 14th annual ACM symposium on Theory of computing (STOC'82), pp. 60–65.

[4] R. Büchi, Symposium on decision problems: On a decision method in restricted second order arithmetic, in: Logic, Methodology and Philosophy of Science, Proceeding of the 1960 International Congress, volume 44 of *Studies in Logic and the Foundations of Mathematics*, Elsevier, 1966, pp. 1–11.

[5] R. McNaughton, Testing and generating infinite sequences by a finite automaton, Information and Control 9 (1966) 521–530.

[6] S. Safra, On the complexity of omega -automata, in: Proceedings of the 29th Annual Symposium on Foundations of Computer Science, IEEE Computer Society, 1988, pp. 319–327.

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

[8] C. Löding, Optimal bounds for transformations of omega-automata, in: Proceedings of the 19th Conference on Foundations of Software Technology and Theoretical Computer Science, Springer-Verlag, 1999, pp. 97–109.

[9] S. Safra, Exponential determinization for omega-automata with strong-fairness acceptance condition (extended abstract), in: Proceedings of the 24th annual ACM symposium on Theory of computing (STOC'92), pp. 275–282.

[10] D. E. Muller, P. E. Schupp, Simulating alternating tree automata by nondeterministic automata: New results and new proofs of the theorems of rabin, mcnaughton and safra, Theoretical Computer Science 141 (1995) 69–107.

[11] N. Piterman, From nondeterministic buchi and streett automata to deterministic parity automata, in: 21st Annual IEEE Symposium on Logic in Computer Science, pp. 255–264.

[12] S. Safra, M. Y. Vardi, On ω-automata and temporal logic, in: Proceedings of the 21st annual ACM symposium on Theory of computing (STOC'89), ACM, 1989, pp. 127–137.

[13] Y. Cai, T. Zhang, H. Luo, An improved lower bound for the complementation of rabin automata, in: Proceedings of the 24th Annual IEEE Symposium on Logic In Computer Science, IEEE Computer Society, 2009, pp. 167–176.

[14] Y. Cai, T. Zhang, A tight lower bound for streett complementation, in: Proceedings of the IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2011), pp. 339–350.

[15] Y. Cai, T. Zhang, Tight upper bounds for streett and parity complementation, in: Proceedings of the 20th Conference on Computer Science Logic (CSL 2011), Dagstuhl Publishing, 2011, pp. 112–128.

[16] N. Francez, D. Kozen, Generalized fair termination, in: Proceedings of the 11th ACM SIGACT-SIGPLAN symposium on Principles of programming languages (POPL'84), ACM, 1984, pp. 46–53.

[17] N. Francez, Fairness, Springer-Verlag New York, Inc., 1986.

[18] T. Colcombet, K. Zdanowski, A tight lower bound for determinization of transition labeled Büchi automata, in: Proceedings of the 36th Internatilonal Collogquium on Automata, Languages and Programming (ICALP 2009), Springer-Verlag, 2009, pp. 151–162.

[19] S. Schewe, Tighter bounds for the determinization of büchi automata, in: Proceedings of the 12th International Conference on Foundations of Software Science and Computation Structures (FoSSaCS 2009), pp. 167–181.

[20] Q. Yan, Lower bounds for complementation of ω-automata via the full automata technique, in: Proceedings of the 33rd Internatilonal Collogquium on Automata, Languages and Programming (ICALP 2006), pp. 589–600.

[21] S. Schewe, Büchi complementation made tight, in: Proceedings of the 26th International Symposium on Theoretical Aspects of Computer Science (STACS 2009), pp. 661–672.

[22] M. O. Rabin, D. Scott, Finite automata and their decision problems, IBM Journal of Research and Development 3 (1959) 114–125.

[23] S. Schwoon, Determinization and complementation of streett automata, in: Automata Logics, and Infinite Games, volume 2500 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, 2002, pp. 257–264.

[24] O. Kupferman, M. Y. Vardi, Complementation constructions for nondeterministic automata on infinite words, in: Tools and Algorithms for the Construction and Analysis of Systems, volume 3440 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, 2005, pp. 206–221.

[25] T. V. Narayana, A partial order and its applications to probability theory, Sankhyā: The Indian Journal of Statistics (1933-1960) 21 (1959) 91–98.

[26] N. Dershowitz, S. Zaks, Enumerations of ordered trees, Discrete Mathematics 31 (1980) 9–28.