A Tight Lower Bound for Determinization of Transition Labeled Büchi Automata

Thomas Colcombet and Konrad Zdanowski[⋆]

Liafa/Cnrs/Université Paris 7, Denis Diderot, Paris, France

Abstract. In this paper we establish a lower bound hist(n) for the problem of translating a Büchi word automaton of size n into a deterministic Rabin word automaton when both the Büchi and the Rabin condition label transitions rather than states. This lower bound exactly matches the known upper bound to this problem. The function hist(n) is in $\Omega((1.64n)^n)$ and in $o((1.65n)^n)$.

Our result entails a lower bound of hist($n-1$) when the input Büchi automaton has its Büchi acceptance condition labeling states (as it is usual). Those lower bounds remain when the output deterministic Rabin automaton has its Rabin acceptance condition labeling states.

1 Introduction

Since the seminal work of Büchi [Büc62], automata running over infinite words (of length ω) have become a fundamental object in automata theory. The automata introduced by Büchi are non-deterministic and use a so called *Büchi acceptance condition*: a run of the automaton is accepting if some set of states is visited infinitely often. Büchi established that it is possible to complement them, though those automata cannot be made deterministic in general. The next fundamental step in this theory is the result of McNaughton [McN66] stating that Büchi automata can be effectively transformed into deterministic automata (of doubly exponential size) using a *Muller acceptance condition*; i.e., a run is accepting if it satisfies a boolean combination of atomic properties of the form ‘the state q is visited infinitely often’[1]. Those two operations of complementation and determinization of automata running on infinite words are the two key results in this theory, and a long line of research is dedicated to the search of the exact complexity of those two operations (in terms of state blow-up).

The complementation saga. The quest for the exact complexity for the complementation of Büchi automata has been the subject of a long list of works (see e.g., [Sch09a]). Now both lower bounds and upper bounds are well known and tightly related. Indeed, there is a function *tight* which is in $O((0.76n)^n)$ such that (lower bound) for all n, there exists a Büchi automaton of size n such

[⋆] Both authors are supported by the Anr project Jade (“Jeux et Automates, Décidabilité et Extensions”).

[1] The original statement is that every Büchi automaton is equivalent to a boolean combination of deterministic Büchi automata.

that every Büchi automaton for the complement language has $\Omega(\mathit{tight}(n-1))$ states [Yan08], and (upper bound) for all Büchi automaton of size n, there exists a Büchi automaton for its complement of size $O(\mathit{tight}(n+1))$ [Sch09a]. Since $O(n^2\mathit{tight}(n-1)) = O(\mathit{tight}(n+1))$, lower and upper bound to the complementation problem only differ by a quadratic factor. This is the end of the quest.

The determinization problem: The origins. In the theory of finite automata, determinization is easy to describe [RS64]. Given a non-deterministic (finite word) automaton of size n, one constructs a deterministic automaton that maintains the set of states that the original automaton could have reached at the current position. The resulting automaton has size 2^n (2^n-1 if one does not require completeness). This construction is known to be optimal (it is even optimal for complementation). For automata running over infinite words, the information maintained by the above construction is not sufficient. Safra was the first to provide a solution oriented toward efficiency [Saf88]. In his construction the reachable states are maintained, and furthermore organized in a tree-structure (called a Safra tree) which carries information on the history of the possible runs reaching each state. The resulting construction takes as input a non-deterministic Büchi automaton of size n, and produces a deterministic automaton of size at most $(12)^n n^{2n}$ states using *a Rabin acceptance condition*; i.e., a run is accepting if it satisfies a disjunction of properties of the form 'the set of state E is visited infinitely often, and the set of states F finitely often' (each such pair (E,F) is named a *Rabin pair*). The automaton constructed by Safra uses $2n$ Rabin pairs: this is achieved by furthermore maintaining in each Safra tree a name attached to each node. This naming mechanism allows a dynamic reuse of the Rabin pairs. The ideas developed by Safra in his seminal work are underlying most of the other solutions to the determinization problem.

The determinization problem: Further developments. Piterman [Pit07] proposes a modification to Safra's constructions for producing an automaton using a *parity acceptance condition*; i.e., every state has a priority in ω, and a run is accepting if the least priority seen infinitely often is even. This construction, not only produces a parity condition (which is a further restriction on the Rabin condition), but also improves on the original upper bound of Safra, reaching a deterministic automaton of size at most $2nn^n n!$. By a finer analysis of Piterman's solution, Liu and Wang reach an upper bound of $2n(n!)^2$ [LW]. Independently, Schewe in [Sch09b] gives a solution in $o((2.66n)^n)$ for the size of a Rabin deterministic automaton using 2^{n-1} Rabin pairs[2]. In the same paper, Schewe provides the best known upper bound for producing a deterministic parity automaton in $O((n!)^2)$ (recall that $n! \approx (0.36n)^n$).

Extensions and Variants. Safra has generalized his construction: he shows how to directly determinize automata that use *Streett acceptance condition*; i.e., the dual of Rabin: a run is accepting if it satisfies a conjunction of properties of the

[2] In fact one can argue on the respective advantages of this solution with respect to others since it is obtained by completely removing the naming system in Safra's original solution, and the price to pay for that is the 2^{n-1} number of Rabin pairs.

form 'if the set of states E is visited infinitely often, then the set of states F is visited infinitely often'. A construction with similar characteristics has been described by Muller and Schupp using different techniques [MS95]. The latter has been revisited by Kähler and Wilke for giving a unified view on complementation, determinization, and disambiguisation[3] of Büchi automata[KW08]. Finally, Piterman in [Pit07] describes the best known determinization procedure that takes a Streett automaton as input, and produces a deterministic parity automaton as output.

Lower bounds. The first non-trivial lower bound to the problem of determinization was $n!$ [Löd99], derived from an early lower bound for complementation [Mic88]. The best known lower bound is in $\Omega((0.76n)^n)$ [Yan06], and holds even if the resulting automaton uses a Muller acceptance condition.

Decomposition. Schewe [Sch09b] develops the idea that the correct way of describing determinization constructions consists in first isolating what he calls the *principle acceptance mechanism* that captures the core of the construction, and then waving on this construction for producing deterministic automata of various characteristics. The *principle acceptance mechanism* takes the form of a deterministic Rabin-automaton which has the non-standard characteristic that the Rabin condition labels transitions rather than states. It has 2^{n-1} Rabin pairs and $hist(n) = o((1.65n)^n)$ states, where $hist(x)$ is the function giving the number of states of an output automaton in Schewe's construction.

Contributions. In this paper, we establish that the principle acceptance mechanism isolated by Schewe is optimal. More precisely, we prove that there exists a transition-labeled Büchi automaton of size n accepting a language L_n such that every transition-labeled deterministic Rabin automaton accepting the language L_n has at least $hist(n)$ states. If the input Büchi automaton has its Büchi condition labeling states, then we have a lower bound of $hist(n-1)$. Since transition-labeled automata are at least as compact as their state-labeled counterparts, those lower bounds can be directly transferred to state-labeled Rabin-automata as output. Since both $hist(n)$ and $hist(n-1)$ are in $\Omega((1.64n)^n)$, this improves the previous lower bound of $\Omega((0.76n)^n)$ to $\Omega((1.64n)^n)$.

2 Basic Definitions and Facts

2.1 Basic Notions and Notations

For a sequence u, the i-th element of u is denoted by $u(i)$ with the convention that elements in a sequence are labeled from 0. The empty sequence is denoted by ε and the length of a sequence u is denoted by $|u|$. If we assume an ordering between elements of sequences then the *lexicographic ordering* of sequences is defined as $u <_{\text{lex}} v$ if there exists i such that $u(i) < v(i)$ and for all $j < i$, $u(j) = v(j)$ or $|u| < |v|$ and for all $i < |u|$, $u(i) = v(i)$. Then $u \leq_{\text{lex}} v$ if $u <_{\text{lex}} v$ or $u = v$.

[3] An automaton is non-ambiguous if there is at most one accepting run per input. This is weaker than determinism.

A *tree* T is a subset of $\omega^{<\omega}$ such that if $xi \in T$ then $x \in T$ and $xj \in T$ for all $j < i$. The elements of T are called *nodes.* The *size* of T is the number of nodes in T. A node of the form xi is called a *child* of the node x. If x is a prefix of y we write $x \preceq y$, and we write $x \prec y$ if $x \preceq y$ and $x \neq y$. When $x \preceq y$, x is called an *ancestor of* y, and y a *descendant of* x. When $x \prec y$, x is called a *strict ancestor of* y, and y a *strict descendant of* x.

2.2 Automata

We define here automata running on infinite inputs. Le us emphasize the fact that, in this work, the accepting condition refers to transitions rather than states.

A *(finite) automaton* is a tuple $\mathcal{A} = (Q, \Sigma, I, \Gamma, \Delta)$, where Q is a set of *states*, Σ is an *input alphabet*, I is a set of *initial states*, Γ is an *output alphabet* and $\Delta \subseteq Q \times \Sigma \times \Gamma \times Q$ is the *transition relation.*

We see a finite automaton $\mathcal{A}$ as a non-deterministic transducer from Σ^ω to Γ^ω. For a word $u \in \Sigma^\omega$, $\rho = (p_0, b_0, p_1)(p_1, b_1, p_2)(p_2, b_3, p_3)\dots$ is a *run of* $\mathcal{A}$ *over* u if $p_0 \in I$ and $\forall i\, (p_i, u(i), b_i, p_{i+1}) \in \Delta$. The *output of* ρ, $\mathrm{Out}(\rho)$ is the word $b_0 b_1 b_2 \dots$.

The automaton $\mathcal{A}$ is called *deterministic* if $\mathcal{A}$ has exactly one initial state and Δ is a functional relation, that is for each $q \in Q$ and $a \in \Sigma$ there is at most one transition of the form (q, a, b, p) in Δ. We will take the convention that when an automaton is deterministic, we denote its transition relation by δ, and we use it also as a mapping from $Q \times \Sigma$ to Q. This mapping is naturally extended into a mapping from $Q \times \Sigma^*$ to Q in the usual manner. Thus, $\delta(q, u)$ is the unique state p such that after reading u when starting in q the automaton $\mathcal{A}$ is in the state p.

An automaton $\mathcal{B}$ is a *Büchi automaton* if $\Gamma = \{0, 1\}$. The language L_B is the set of words v in $\{0, 1\}^\omega$ such that v contains infinitely many zeros. A run ρ of of $\mathcal{B}$ is *accepting* if $\mathrm{Out}(\rho) \in L_B$. An automaton $\mathcal{B}$ *accepts* a word $u \in \Sigma^\omega$ if there is an accepting run of $\mathcal{B}$ on u.

An automaton $\mathcal{R}$ is a *Rabin automaton* with h conditions (i.e, h Rabin pairs) for $h \in \omega$, if $\Gamma = \mathcal{P}(\{r_1, s_1, r_2, s_2, \dots, r_h, s_h\})$. The Rabin language L_R is the set of words v in Γ^ω such that for some i, $r_i \in v(n)$ for finitely many n, and $s_i \in v(n)$ for infinitely many n. As above, a run ρ of a Rabin automaton $\mathcal{R}$ is *accepting* if $\mathrm{Out}(\rho) \in L_R$ and $\mathcal{R}$ *accepts* u if there is an accepting run of $\mathcal{R}$ on u.

We are interested in providing a lower bound to the number of states needed for the determinization of a Büchi automaton. Hence, we define the *size* of an automaton as the number of its states. For a Büchi automaton $\mathcal{B}$, let $D(\mathcal{B})$ be the size of a smallest deterministic Rabin automaton which recognizes the same language. Let $D(n)= \max\{D(\mathcal{B}) : \mathcal{B}$ is a Büchi automaton of size $n\}$.

An important remark made by Yan in [Yan08] is that for each n there exists a canonical Büchi automaton with n states which can simulate every Büchi automaton of this size: the full automaton. In our context, it gives the following definition.

Definition 1 (Full automaton). *The full Büchi automaton of size n is the automaton $\mathcal{B}_n = (Q, \Sigma, Q, \{0, 1\}, \Delta)$ such that $Q = \{1, \dots, n\}$ and $\Sigma = \mathcal{P}(Q \times \{0, 1\} \times Q)$ and $\Delta = \{(q, a, b, p) \colon (q, b, p) \in a\}$.*

Thus, the language of $\mathcal{B}_n$ is a set of words u such that there is a path $\rho = (p_0, b_0, p_1)(p_1, b_1, p_2)(p_2, b_2, p_3)\dots$ such that for each i, $(p_i, b_i, p_{i+1}) \in u(i)$ and for infinitely many i, $b_i = 0$. We denote this language L_n.

Since full automata can simulate every automaton of the same size, we naturally get the following lemma. It follows from it that to prove lower or upper bounds on $D(n)$ it is enough to consider only full automata $\mathcal{B}_n$.

Lemma 1 ([Yan06]). *$D(n) = D(\mathcal{B}_n)$.*

2.3 Games

A *game* is a tuple $\mathcal{G} = (V, V_E, V_A, p_I, \Gamma, \mathrm{Move}, L)$, where V is a set of *positions* which is partitioned into the *positions for Eva V_E* and the *positions for Adam V_A*, $p_I \in V$ is the *initial position* of $\mathcal{G}$, Γ is the *labeling alphabet*, $\mathrm{Move} \subseteq V \times \Gamma \times V$ is the set of possible *moves*, and $L \subseteq \Gamma^\omega$ is the *winning condition.* A tuple $(v, a, w) \in \mathrm{Move}$ indicates that there is a move from v to w which produces letter a. A game using the winning condition L is called an *L-game.*

During a play of the game $\mathcal{G}$, the two players, Adam and Eva, make moves according to Move, the player to whom belong the current positions choosing the next move. Formally, a *play* is a maximal sequence $\pi = (p_0, a_0, p_1, a_1, p_2, a_2, \dots)$ such that $p_0 = p_I$ and for each i, $(p_i, a_i, p_{i+1}) \in \mathrm{Move}$. Let $\pi_\Gamma = (a_0, a_1, a_2, \dots)$. Eva *wins the play* π if $\pi_\Gamma \in L$. Otherwise, Adam wins the play.

A strategy for the player X is a function which tells the player what moves he should choose depending on the finite history of moves played so far. Formally, a *strategy* (for Eva or for Adam) is a total mapping from finite sequences $(p_0, a_0, \dots, a_{n-1}, p_n)$ into Move. A play π is compatible with a strategy σ for Eva (resp. Adam) if for all prefixes $(p_0, \dots, p_{n-1}, a_{n-1}, p_n)$ of π, if $p_{n-1} \in V_E$ (resp. in V_A), then $\sigma(p_0, \dots, p_{n-1}) = (p_{n-1}, a_{n-1}, p_n)$. A strategy σ for Eva (resp. Adam) *is winning* if Eva (resp. Adam) wins every play compatible with σ.

A *strategy with memory m for Eva* is described as $(M, \mathrm{update}, \mathrm{choice}, \mathrm{init})$ in which M is a set of size m called the *memory*, update is a mapping from $M \times \mathrm{Move}$ to M, choice is a mapping from $V_E \times M$ to Move, and $\mathrm{init} \in M$. The mapping update is defined for moves, but can naturally be extended to paths in the game. The strategy described by $(M, \mathrm{update}, \mathrm{choice}, \mathrm{init})$ is the one which to each path $\pi = (p_0, a_0, \dots, a_{n-1}, p_n)$ with $p_n \in V_E$ associates $\mathrm{choice}(p_n, \mathrm{update}(\mathrm{init}, \pi))$. A player X wins a game with memory n if it has a winning strategy with memory m. A *positional strategy* corresponds to the case $m = 1$.

We call a game $\mathcal{G}$ a *Rabin-game* if L is the Rabin language L_R over some alphabet Γ. Eva has positional winning strategies in Rabin games:

Theorem 1 ([Kla94],[Zie98]). *For every Rabin-game, if Eva wins, she can win using a positional strategy.*

2.4 Reduction

A standard way to use a deterministic automaton is for game reduction. Indeed, given a deterministic automaton for a language L with (e.g. Rabin) acceptance condition F with n states, and an L-game, one can perform the product of the game with the automaton, yielding an F-game. From the determinism of the automaton, one can derive that the winner of the two games is the same. Pushing further, if the F-game admits positional strategies for Eva (and it is the case for Rabin-games according to Theorem 1), one can see the states of the automaton as maintaining the memory for a strategy in the L-game: Eva needs memory n in the original game. We obtain:

Lemma 2. *If Eva wins an L-game, and there exists a deterministic Rabin-automaton for L with n states, then Eva wins using a strategy with memory n.*

It is standard to use this result for proving upper bound results on the memory needed for the winner of a game. In this work, we take the opposite point of view and read the above lemma as follows: *"If Eva wins an L-game, and requires memory n for that, then every deterministic Rabin-automaton for L has size at least n"*. This provides an argument for proving lower bounds on determinization problems.

3 The Determinization Construction

Now, we will describe briefly the construction of Schewe from [Sch09b]. This construction takes as input a Büchi automaton $\mathcal{B}$, and produces as output a deterministic Rabin automaton $\mathcal{R}$ such that $L(\mathcal{B}) = L(\mathcal{R})$. This construction captures the core mechanism of the famous construction of Safra [Saf88] (it removes from it the node naming system that was used for reducing the number of Rabin pairs). For more explanation of the construction we use here, we refer the reader to [Sch09b].[4] For a more extensive description of the original Safra construction, we suggest [Tho97].

Let us fix a Büchi automaton $\mathcal{B} = (Q, I, \Sigma, \{0,1\}, \Delta)$ with n states. For a set $S \subseteq Q$ and a letter $a \in \Sigma$, let $\Delta(S,a) = \{q : \exists p \in S \exists b\, (p,a,b,q) \in \Delta\}$ and $\Delta_0(S,a) = \{q : \exists p \in S\, (p,a,0,q) \in \Delta\}$.

In the construction we work with trees with nodes labeled by subsets of states of a Büchi automaton. A *labeling* of T is a function from nodes of T to the set of labels (in our case: subsets of Q). A label of a node $x \in T$ is denoted by $T(x)$. We use a convention that for $x \notin T$, $T(x) = \emptyset$. It does not cause any ambiguity because the labels of nodes of T will be always nonempty. For a node x we order its children by the "*older than*" relation and we say that xi is older than xj for $i < j$.

Recall that siblings in a tree have, by definition, consecutive numbers starting from 0. For this reason, the process of deleting a node requires some explanations. *Deleting* a node x in a tree T consists in a) removing the node x and all its descendants, and b) shifting all the younger siblings of x to the left, i.e., the direct right sibling of x takes the place of x, etc...

[4] Schewe's construction takes a state-labeled Büchi automaton as input rather than a transition labeled automaton. Nevertheless, his approach easily adapts to this case.

Now, we define the deterministic Rabin automaton $\mathcal{R}$ accepting $L(\mathcal{B})$. The set of states of $\mathcal{R}$ is the set of trees T of size at most $|Q|$ labeled with subsets of Q and such that

1. each node in T is labeled by a nonempty set of states,
2. the label of a node x is a strict superset of the union of its children labels,
3. labels of siblings are disjoint.

We call such trees, after [Sch09b], *history trees.* According to our notation for $x \in T$, $T(x)$ is the label of the node x in T. In particular, by the second point above, $T(\varepsilon)$ denotes the set of states which occur in some label of T. We denote by $\mathcal{H}$ the set of history trees.

The initial state of $\mathcal{R}$ is a one node tree labeled by the set of initial states of $\mathcal{B}$. We always call this initial tree T_0 (in general T_0 depends on the set of initial states of $\mathcal{B}$ but in the cases we consider it will be just a one node tree labeled with Q). Let T be the current state of $\mathcal{R}$, and a be the currently read letter. The transition function is defined in multiple steps as follows.

1. For each node $x \in T$, replace the label of x with $\Delta(T(x), a)$.
2. For each node $x \in T$ originally labeled by S, if $\Delta_0(S, a)$ is nonempty then form a new youngest child of x and label it with $\Delta_0(S, a)$.
3. For each node x and for each state $q \in T(x)$, if q belongs to an older sibling of x, then delete q from labels of x and all its descendants.
4. Now, we contract the tree obtained so far:
   4.1 for each node x such that its label is nonempty and is equal to the sum of labels of its descendants, delete all strict descendants of x, and call x *green*,
   4.2 for each node xi that has an empty label, mark xi and all the nodes in trees rooted at younger siblings of xi as red, and delete the subtree rooted in xi.
5. The output of the transition is a set $\mathcal{E}$ of what we call *events.* For each red node x we put $(x, A) \in \mathcal{E}$ and for each green x we put (x, E) in $\mathcal{E}$.

The events (x, A) play the role of r_i in the definition of a Rabin language L_R and events (x, E) correspond to s_i. Thus, the Rabin condition simply states that there exists x such that (x, A) is seen finitely many times and (x, E) infinitely many times. In other words, there will be a node x such that from some time on it will never be deleted or moved to the left during point 4.2 of the transition and it will be green infinitely many times. We will denote by Λ the set of possible *events*, i.e., pairs of the form either (x, A) or (x, E) in which x is a possible node in an history tree (there are 2^{n-1}-many such x).

Since the Rabin automaton defined above is deterministic, its transition relation can be seen as a partial function from $\mathcal{H} \times \Sigma$ to $\mathcal{P}(\Lambda) \times \mathcal{H}$. Given a history tree T, and a letter a, $\delta(T, a)$ denotes the tree obtained by the above procedure, and $\mathcal{E}(T, a)$ the set of produced events. The mapping δ is extended into a mapping from $\mathcal{H} \times \Sigma^*$ to $\mathcal{H}$ by $\delta(T, \varepsilon) = T$, and $\delta(T, ua) = \delta(\delta(T, u), a)$. The mapping $\mathcal{E}$ is extended into a mapping from $\mathcal{H} \times \Sigma^*$ to $\mathcal{P}(\Lambda)$ by $\mathcal{E}(T, \varepsilon) = \emptyset$ and $\mathcal{E}(T, ua) = \mathcal{E}(T, u) \cup \mathcal{E}(\delta(T, u), a)$.

Theorem 2 ([Sch09b], variant of [Saf88]). *Let $\mathcal{B}$ be a nondeterministic Büchi automaton and let $\mathcal{R}$ be a deterministic Rabin automaton constructed as above for $\mathcal{B}$. Then, $L(\mathcal{B}) = L(\mathcal{R})$.*

3.1 Complexity of the Construction

The size complexity of the construction described above is measured as the number of states of the constructed Rabin automaton. Then, let us define the function $\mathit{hist}(n)$ as the number of history trees for Q of size n. The subject of the paper is to prove that this value also provides a lower bound to the determinization construction. This function is analyzed in [Sch09b] where it is proved to be in $o((1.65n)^n)$. On the other hand, the following lower bound on $\mathit{hist}(n)$ can be proved.

Lemma 3 (M. Bouvel, D. Rossin). *The function $\mathit{hist}(n)$ is in $\Omega((1.64n)^n)$.*

4 Optimality of Determinization

During this section we fix a full automaton of size n, $\mathcal{B}_n = (Q, \Sigma, Q, \{0,1\}, \Delta)$, and we set Σ to be its alphabet $\mathcal{P}(Q \times \{0,1\} \times Q)$. The subject of this section is to prove Theorem 4 that establishes our lower bound.

The proof consists first in restricting ourselves to a constant set of reachable states, second to prove a lower bound in this restricted context, third to reconstruct our main result.

4.1 Fixing the Set of Reachable States

We define the set of states reachable by a word u, $\text{Reach}(u)$, by induction. $\text{Reach}(\varepsilon) = Q$ and $\text{Reach}(va) = \{q : p \in \text{Reach}(v),\ (p, b, q) \in a\}$ for $v \in \Sigma^*$ and $a \in \Sigma$. Let Σ_S be the set of letters $a \in \Sigma$ such that $S = \{q : (p, b, q) \in a,\ p \in S\}$, or equivalently $\text{Reach}(a) = \text{Reach}(aa) = S$. Let L_n^S be $L_n \cap \Sigma_S^\omega$.

Of course, each history tree T maintains in $T(\varepsilon)$ the set of states reachable by the original automaton at the current position in the word. Hence it is natural for all $S \subseteq Q$ to consider $\mathcal{H}_S = \{T \in \mathcal{H} \ : \ T(\varepsilon) = S\}$. We have for all words $u \in \Sigma_S^*$ and all $T \in \mathcal{H}_S$ that $\delta(T, u) \in \mathcal{H}_S$.

4.2 The Game

Let us fix ourselves a set $S \subseteq Q$. We establish in this section Theorem 3 which provides a lower bound for the size of a deterministic Rabin automaton accepting L_n^S. For this, we define a game $\mathcal{G}$ such that Eva wins $\mathcal{G}$ but she cannot win with memory less than $|\mathcal{H}_S|$. That proves, by Lemma 2, that any deterministic Rabin automaton accepting L_n^S has at least $|\mathcal{H}_S|$ states.

We order history trees: we say that T is *strictly smaller* than T' at position x, written $T <_x T'$, if $T'(x) \subsetneq T(x)$ and for all $y <_{\text{lex}} x$, $T'(y) = T(y)$. We can remark that if $T <_x T'$, x may not be a node of T' (i.e., $T'(x) = \emptyset$), but in any case it is a node of T.

We construct the L_n^S-game $\mathcal{G} = (V, V_E, V_A, p_I, \Sigma_S^+, \text{Move}, L_n^S)$, where V_E is a singleton set $\{p_E\}$ and V_A consists of the initial position in the game p_I and one position p_T for each history tree $T \in \mathcal{H}_S$. The moves in $\mathcal{G}$ are the following:

1. $(p_I, u, p_E) \in \text{Move}$, for all $u \in \Sigma_S^+$,
2. $(p_E, \text{id}_S, p_T) \in \text{Move}$, for each history tree T, where $\text{id}_S = \{(q, 1, q) : q \in S\}$,
3. $(p_T, u, p_E) \in \text{Move}$, for each history tree T and word $u \in \Sigma_S^+$ if there exists a node x in $\delta(T, u)$ such that either
   - $(x, E) \in \mathcal{E}(T, u)$, and for all $y \leq_{\text{lex}} x$, $(y, A) \notin \mathcal{E}(T, u)$ and $\delta(T, u)(y) = T(y)$ or;
   - $\delta(T, u) <_x T$ and for all $y <_{\text{lex}} x$, $(y, A) \notin \mathcal{E}(T, u)$.

Essentially, this game has a flower shape. The central node is controlled by Eva, and from this node, she can decide to go to any of the petals of the flower, each one corresponding to a history tree p_T. Then, it is Adam's turn to choose a word u, and come back to the center. Adam's moves are restricted in the following way: playing a word u in petal p_T is valid if it is possible to witness in the behaviour of $\mathcal{R}$ from the state T when reading u that it is profitable for Eva. This witness takes the form of a position x in the tree T such that nothing happened above or to the left of x, and something good for Eva happened in x: either some Eva-good event in $\mathcal{E}$, or a local advance of the $<_x$ ordering.

It should be clear that both players can always perform a move from their positions. Thus, every completed play has the length ω.

Lemma 4. *Eva has a winning strategy in $\mathcal{G}$.*

To prove Lemma 4 we show that a winning strategy for Eva is as follows: if a word u was played after a finite play and Eva is to make a move from p_E, then she chooses to go to a position indexed by $\delta(T_0, u)$. Then, possible moves for Adam forces him to generate infinitely often an event (x, E) without generating infinitely many (x, A).

4.3 Memory Lower Bound for the Game

Now, for each history tree $T \in \mathcal{H}_S$ we define a game $\mathcal{G}_T$ which is a modification of $\mathcal{G}$ by removing a petal p_T, i.e., removing one of Adam's positions, and hence removing to Eva the ability to take the corresponding move. All other elements in the game are unchanged. Our crucial Lemma 5 states that for any two history trees $T \neq T'$ in $\mathcal{H}_S$, there exists always a word u such that $(p_{T'}, u, p_E)$ is a valid move in the game, and such that nothing good for Eva happens when $\mathcal{R}$ reads u from state T. This is formalized as follows:

Lemma 5. *Let $T \neq T'$ be history trees in $\mathcal{H}_S$. There exists a word u such that*

1. *$(p_{T'}, u, p_E)$ is a move in $\mathcal{G}_T$,*
2. *$T = \delta(T, u)$,*
3. *for all x, $(x, E) \notin \mathcal{E}(T, u)$.*

This lemma is the technical core of our proof. It requires an analysis of the differences between the two trees. This results in many situations of very different nature. With the help of Lemma 5 we can prove the following.

Lemma 6. *For every $T \in \mathcal{H}_S$ Adam has a winning strategy in $\mathcal{G}_T$.*

A winning strategy for Adam is as follows. From p_I he plays a word u such that $\delta(T_0, u) = T$, where T_0 is the initial state of the deterministic Rabin automaton $\mathcal{R}$ from Theorem 2. The good answer for Eva would be to move to p_T (according to the proof of Lemma 4), but since the petal is removed, she has to choose another move, say to $p_{T'}$ for some $T' \neq T$. Then Adam answers according to Lemma 5. Of course, using this strategy, Adam maintains the property that if a word w has been played so far, then $\delta(T_0, w) = T$, and hence Adam can always answer to Eva's proposal using Lemma 5. Now, when the play gets infinite, say producing an infinite word u, we can see from the property of the words u in Lemma 5 that $\mathcal{R}$ does not accept this word (no events good for Eva are ever produced). Hence, $u \notin L_n^S$ and Adam wins.

From Lemma 6 we may easily infer the following.

Corollary 1. *Eva has no winning strategy with memory $|\mathcal{H}_S| - 1$ in $\mathcal{G}$.*

Indeed, if Eva had a strategy with memory $|\mathcal{H}_S| - 1$, then there would be a position p_T which is never visited by this strategy. But this would mean that Eva wins $\mathcal{G}_T$ with the exact same strategy.

Using Lemma 2, we now get:

Theorem 3. *Every deterministic Rabin automaton accepting L_n^S has size at least $|\mathcal{H}_S|$.*

4.4 Reduction to the General Case

What remains to be done is a reduction to the general case. We do this by decomposing a given deterministic Rabin automaton accepting L_n into disjoint sets of states. The following lemma gives an argument for such a decomposition.

Lemma 7. *Let $\mathcal{R}$ be a deterministic Rabin automaton which accepts L_n with a transition function δ and an initial state q_0. If $\delta(q_0, u) = \delta(q_0, v)$ then* $\mathrm{Reach}(u) = \mathrm{Reach}(v)$.

Now, it is possible to prove the main theorem.

Theorem 4. *Every deterministic Rabin automaton accepting L_n has size at least hist(n).*

The theorem is proved by defining for a given deterministic Rabin automaton $\mathcal{R}$ accepting L_n a partition of its states. For S a nonempty subset of Q, let R_S be the set of states $q \in Q$ such that there exists u with $\mathrm{Reach}(u) = S$ and $\delta(q_0, u) = q$, in which q_0 is the initial state of $\mathcal{R}$ and δ is its transition function. The automaton $\mathcal{R}$ restricted to R_S can be seen as a Rabin automaton which accepts the restriction of L_n to letters in Σ_S. Hence the lower bound of Theorem 3 holds for the automaton $\mathcal{R}$ restricted to R_S.

Theorem 4 then follows from the fact that the sets R_S are disjoint, and that

$$hist(n) = |\mathcal{H}| = \sum_{S \subseteq \{1,\dots,n\}} |\mathcal{H}_S| \ .$$

5 Discussion

We have proved an exact lower bound for the determinization of Büchi automata using a non-standard definition of automata in which the acceptance condition labels transitions rather than states as it is usual.

State-labeled Büchi automaton as input. A *state-labeled Büchi automaton* over alphabet Σ and of states Q, has transitions $\Delta \subseteq Q \times \Sigma \times Q$ (the Büchi label has disappeared from transitions), and a set of final states $F \subseteq Q$. A run of the automaton is accepting if it visits a state in F infinitely often. Given a (transition-labeled) Büchi automaton with n states $\mathcal{A}$, one can transform it into a state-labeled Büchi automaton with $2n$ states $\mathcal{B}$ in such a way that the action of every letter in $\mathcal{A}$ can be simulated by a sequence of two letters on $\mathcal{B}$. This is sufficient for proving that translating a state-labeled automaton of size $2n$ into a deterministic Rabin-automaton requires at least $hist(n)$ states.

However it is possible to do much better if one inspects the proof of our theorem more closely. For this one remarks that in our proof of optimality one does not need to use the whole alphabet Σ. By a close inspection of all the cases in the proof of Lemma 5, one can remark that all the letters appearing in the word u described by this lemma do satisfy the following property: either there is no 0-transitions in a letter, or all transitions labeled by 0 have the same state as origin. Using this remark, one can optimise the above argument and get the following lower bound.

Theorem 5. *There is a language L'_n accepted by a state-labeled Büchi automaton with n-states such that every deterministic Rabin automaton accepting L'_n has size at least $hist(n-1)$.*

State-labeled Rabin automaton as output. It is very simple to transform a state-labeled automaton accepting some language (whatever is its acceptance condition) into a transition-labeled one using the same acceptance condition and the same number of states. The idea is simply to use transitions that have as label the label of the origin state in the original automaton. From this we deduce that all the lower bounds presented above can be directly transferred to transition-labeled automata as output.

Acknowledgements

We are grateful to Mathilde Bouvel and Dominique Rossin for their proof of Lemma 3. We also thank Christof Löding for his insightful comments.

References

[Büc62] Richard Büchi, J.: On a decision method in restricted second-order arithmetic, pp. 1–11 (1962)

[Kla94] Klarlund, N.: Progress measures, immediate determinacy, and a subset construction for tree automata. Annals of Pure and Applied Logic 69(2–3), 243–268 (1994)

[KW08] Kähler, D., Wilke, T.: Complementation, disambiguation, and determinization of Büchi automata unified. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part I. LNCS, vol. 5125, pp. 724–735. Springer, Heidelberg (2008)

[Löd99] Löding, C.: Optimal bounds for transformations of ω-automata. In: Pandu Rangan, C., Raman, V., Ramanujam, R. (eds.) FSTTCS 1999. LNCS, vol. 1738, pp. 97–109. Springer, Heidelberg (1999)

[LW] Liu, W., Wang, J.: A tighter analysis of Piterman's Büchi determinization (submitted)

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

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

[MS95] Muller, D.E., Schupp, P.E.: Simulating alternating tree automata by nondeterministic automata: New results and new proofs of the theorems of Rabin, McNaughton and Safra. Theor. Comput. Sci. 141(1&2), 69–107 (1995)

[Pit07] Piterman, N.: From nondeterministic Büchi and Streett automata to deterministic parity automata. Logical Methods in Computer Science 3(3) (2007)

[RS64] Rabin, M.O., Scott, D.: Finite automata and their decision problems. Technical report (1964)

[Saf88] Safra, S.: On the complexity of ω-automata. In: FOCS, pp. 319–327 (1988)

[Sch09a] Schewe, S.: Büchi complementation made tight. In: STACS 2009 (2009)

[Sch09b] Schewe, S.: Tighter bounds for the determinisation of Büchi automata. In: de Alfaro, L. (ed.) FoSSaCS 2009. LNCS, vol. 5504, pp. 167–181. Springer, Heidelberg (2009)

[Tho97] Thomas, W.: Languages, automata and logic. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of Formal Languages, vol. 3. Springer, Heidelberg (1997)

[Yan06] Yan, Q.Q.: Lower bounds for complementation of ω-automata via the full automata technique. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006. LNCS, vol. 4052, pp. 589–600. Springer, Heidelberg (2006)

[Yan08] Yan, Q.Q.: Lower bounds for complementation of ω-automata via the full automata technique. Logical Methods in Computer Science 4 (2008)

[Zie98] Zielonka, W.: Infinite games on finitely coloured graphs with applications to automata on infinite trees. Theoretical Computer Science 200(1–2), 135–183 (1998)