Complementation of Finitely Ambiguous Büchi Automata

Alexander Rabinovich

Tel Aviv University

Abstract. A nondeterministic automaton is finitely ambiguous if for each input there is at most finitely many accepting runs. We prove that the complement of the ω-language accepted by a finitely ambiguous Büchi automaton with n states is accepted by an unambiguous Büchi automaton with 2×5^n states.

1 Introduction

Restricted Forms of Nondeterminism The relationship between deterministic and nondeterministic machines plays a central role in computer science. An important topic is comparison of expressiveness, succinctness and complexity of deterministic and nondeterministic models. Various restricted forms of nondeterminism were suggested and investigated (see [4, 7] for recent surveys).

Probably the oldest restricted form of nondeterminism is unambiguity. An automaton is unambiguous if for every input there is at most one accepting run. For automata over finite words there is a rich and well-developed theory about the relationship between deterministic, unambiguous and nondeterministic automata [7]. All three models have the same expressive power. Unambiguous automata are exponentially more succinct than deterministic ones, and nondeterministic automata are exponentially more succinct than unambiguous ones [10, 13].

Some problems are easier for unambiguous than for nondeterministic automata. As shown by Stearns and Hunt [18], the equivalence and inclusion problems for unambiguous automata are in polynomial time, while these problems are PSPACE-complete for nondeterministic automata.

The complexity of basic regular operations on languages represented by unambiguous finite automata was investigated in [14], and tight upper bounds on state complexity of intersection, concatenation and many other operations on languages represented by unambiguous automata were established. It is well-known that the tight bound on the state complexity of the complementation of nondeterministic automata is 2^n. In [14], it was shown that the complement of the language accepted by an n-state unambiguous automaton is accepted by an unambiguous automaton with $2^{0.79n+\log n}$ states.

Many other notions of ambiguity were suggested and investigated. An automaton is k-ambiguous if on every word it has at most k accepting runs. A

recent paper [7] surveys works on the degree of ambiguity and on various nondeterminism measures for finite automata on words.

The theory of finite automata on ω-words is more subtle than the theory of finite automata on words. Deterministic Büchi automata (DBA) are strictly less expressive than nondeterministic Büchi automata (NBA). The restricted notions of nondeterminism can be considered for finite automata on ω-words. Arnold [1] proved that unambiguous Büchi automata (UBA) have the same expressive power as nondeterministic Büchi automata over ω-words.

From the exponential succinctness gap between unambiguous and nondeterministic automata on words it is easy to derive that there at least an exponential succinctness gap between UBA and NBA. Kahler and Wilke [9] proved that a Büchi automaton with n states is equivalent to an UBA with $(3n)^n$. However, the tight bound on disambiguating NBA is unknown.

The complexity of universality and inclusion problems for UBA are long standing open problems; polynomial time algorithms are known only for subclasses of UBA [8]. It seems that the only problem which was proved to be easier for UBA than for NBA is the almost universality problem, which can be seen as a probabilistic variant of the universality problem [2].

Our Results In this paper we will show that the complementation of UBA is easier than the complementation of NBA. The complementation of Büchi automata is a classical problem which was extensively studied [3, 11, 17, 15, 6, 16, 22]. Yan [22] and Schewe [16] proved tight $\Omega((0.76n)^n)$ lower and upper bounds for state complexity of complementation of NBA. The main result of our paper implies $O(5^n)$ upper bound on complementation of UBA.

Actually, we introduce a more liberal notion of ambiguity. A nondeterministic automaton is called **finitely ambiguous** if for each input there is at most finitely many accepting runs. An automaton is k-ambiguous if for each input it has at most k accepting runs. It is clear that an unambiguous automaton is k-ambiguous for every $k > 0$, and a k-ambiguous automaton is finitely ambiguous. The reverse implications fail. The automaton in the left of Fig. 1 is finitely ambiguous, but for no k, it is k-ambiguous.

Fig. 1. Finitely ambiguous and 2-ambiguous Büchi automata

We prove that the complement of the ω-language accepted by an n-states finitely ambiguous automaton is accepted by a 2×5^n-states unambiguous semi-deterministic automaton (semi-deterministic automata are defined in Sect. 4).

Organization of the paper. The next section outlines the complementation procedure. This procedure has two clearly separated steps, respectively presented in Sect. 3 and Sect. 4. Section 5 analyzes the complexity. Section 6 investigates the degrees of ambiguity for Büchi automata and states some open problems.

2 Outline of the complementation procedure

Throughout the paper we use standard notations and terminology about automata on ω-words (see e.g. [19]). Let $\mathcal{B}$ be a Büchi automaton and let α be an ω-word. The nodes of the *computational forest* $FT_{\alpha,\mathcal{B}}$ of $\mathcal{B}$ on α are the runs of $\mathcal{B}$ on the finite prefixes of α. The runs (nodes) are arranged in a natural way: if r_1 is a prefix of r_2 then r_1 is an ancestor of r_2. $\mathcal{B}$ accepts α iff $FT_{\alpha,\mathcal{B}}$ has a branch with infinitely many accepting states - an accepting branch. The roots of $FT_{\alpha,\mathcal{B}}$ are the initial states of $\mathcal{B}$ (the runs on the empty word), and if $\mathcal{B}$ has one initial node, a computational forest is a computational tree.

Our complementation procedure has two steps (see Fig. 2) and is a variant of the construction used by Kahler and Wilke [9].

The first step of the construction is to extract from $FT_{\alpha,\mathcal{B}}$ a narrow forest $t_{\alpha,\mathcal{B}}$ which has an accepting branch iff $FT_{\alpha,\mathcal{B}}$ has an accepting branch. A forest is n-narrow if at every level it has at most n nodes. This implies that an n-narrow forest has at most n infinite branches, while a computational forest might have uncountable many infinite branches.

An n-narrow forest t can be coded by a string. The i-th letter of the string describes how nodes of level i are connected to the nodes of level $i-1$. This is the i-th slice of t.

Let us consider two runs r_1 and r_2 of $\mathcal{B}$ on a prefix of α which lead to the same state q. The nodes v_1 and v_2 of $FT_{\alpha,\mathcal{B}}$ which correspond to the runs are on the same level in the forest. The subtree of $FT_{\alpha,\mathcal{B}}$ rooted at v_1 is isomorphic to the subtree of $FT_{\alpha,\mathcal{B}}$ rooted at v_2. Hence, one of the subtrees contains an accepting branch iff the other contains such a branch. So, if we prune $FT_{\alpha,\mathcal{B}}$ at one of these nodes and grow up the computational forest from the other, the pruned tree will have an accepting branch iff $FT_{\alpha,\mathcal{B}}$ has such a branch. If we have a strategy to keep on each level of the computational forest for every state q at most one run that leads to q and to prune the others such that the existence of an accepting branch is preserved, then we obtain a $|Q_{\mathcal{B}}|$-narrow forest which has an accepting branch iff $FT_{\alpha,\mathcal{B}}$ has one. Surprisingly, it turns out that for finitely ambiguous Büchi automata every strategy works.

Relying on a standard powerset construction for determinization of automata on finite words, for every finitely ambiguous Büchi automaton $\mathcal{B}$ we construct a transducer of size $2^{|Q_{\mathcal{B}}|}$ that receives an ω-string α and outputs a code of $|Q_{\mathcal{B}}|$-narrow sub-forest $t_{\alpha,\mathcal{B}}$ of $FT_{\alpha,\mathcal{B}}$ such that $t_{\alpha,\mathcal{B}}$ has an accepting branch iff $FT_{\alpha,\mathcal{B}}$ has an accepting branch.

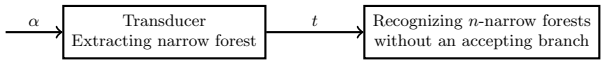

Fig. 2. Complementation Procedure

In the second step we construct an automaton of state complexity $2^{O(n)}$ that receives the code of an n-narrow forest t and accepts it iff t has no accepting branch. This step is based on the breakpoint construction [12], which is a variant of the construction used by Kahler and Wilke [9] and the resulting Büchi automaton is unambiguous. The product of the transducer constructed in the first step and the unambiguous Büchi automaton of the second step (see Fig. 2) is an unambiguous Büchi automaton that accepts the complement of the language of a finitely ambiguous automaton.

3 Extracting narrow forest

Let $\mathcal{A} := \langle Q_\mathcal{A}, \delta_\mathcal{A}, \Sigma_\mathcal{A}, Q^{init}_\mathcal{A}, F_\mathcal{A}\rangle$ be a Büchi automaton, where $Q_\mathcal{A}$ is its set of states, $\delta_\mathcal{A}$ is its transition relation, $Q^{init}_\mathcal{A}$ and $F_\mathcal{A}$ are the sets of initial and accepting states, and $\Sigma_\mathcal{A}$ is an alphabet. Let $\alpha = a_1a_2\ldots$ be an ω-string over $\Sigma_\mathcal{A}$. The computational dag of $\mathcal{A}$ on α is denoted by $Dag_{\alpha,\mathcal{A}} := \langle V_{\alpha,\mathcal{A}}, E_{\alpha,\mathcal{A}}\rangle$ and is defined as follows:

Nodes $V_{\alpha,\mathcal{A}} \subseteq Q_\mathcal{A} \times \mathbb{N}$ is the union $Q_l \times \{l\}$, where Q_0 is the set of initial nodes of $\mathcal{A}$, and $Q_{l+1} := \delta_\mathcal{A}(Q_l, a_{l+1})$.
Edges There is an edge from $\langle q, l\rangle$ to $\langle q', l'\rangle$ iff $l' = l + 1$ and $q' \in \delta_\mathcal{A}(q, a_{l+1})$.

A node $\langle q, l\rangle$ is said to be on the level l; there are at most $|Q_\mathcal{A}|$ nodes on each level. For $S \subseteq Q_\mathcal{A}$, a node $\langle q, l\rangle$ is an S-node if $q \in S$. There is a bijection between the set of ω-runs of $\mathcal{A}$ on α and the set of ω-branches (maximal ω-paths) in $Dag_{\alpha,\mathcal{A}}$. To an ω-run $\rho := q_0, \ldots q_i \ldots$ of $\mathcal{A}$ corresponds an ω-branch $\hat{\rho} := \langle q_0, 0\rangle, \ldots \langle q_i, i\rangle, \ldots$. The set of ω-runs on α as well as the corresponding set of ω-branches in $Dag_{\alpha,\mathcal{A}}$ might be uncountable. We often will not distinguish between ρ and its corresponding branch $\hat{\rho}$; e.g., we will say "a node is on ρ" instead of "a node is on the branch which corresponds to ρ."

A branch is called *accepting* if it contains infinitely many $F_\mathcal{A}$-nodes.

A *spanning forest* of $Dag_{\alpha,\mathcal{A}}$ is a subgraph of $Dag_{\alpha,\mathcal{A}}$ which has the same set of nodes and every node except the initial nodes has in-degree one. (If $\mathcal{A}$ has one initial state, its spanning forests are spanning trees.) The next proposition is our main technical result.

Proposition 1 *Assume that $\mathcal{A}$ is finitely ambiguous. Let $T_{\alpha,\mathcal{A}}$ be any spanning forest of $Dag_{\alpha,\mathcal{A}}$. Then, $T_{\alpha,\mathcal{A}}$ has an accepting branch iff $\mathcal{A}$ accepts α.*

First, we prove the following Lemma.

Lemma 2. *Let ρ be an ω-run on α. Then $T_{\alpha,\mathcal{A}}$ has an ω-branch π such that from every node of π a node on ρ is reachable in $T_{\alpha,\mathcal{A}}$.*

Proof. First, note that if ρ is unreachable from an i-th node of π, then for every $j > i$, ρ is unreachable from the j-th node of π. Note $T := T_{\alpha,\mathcal{A}}$ has finitely many (at most $|Q_{\mathcal{A}}|$) ω-branches. For the sake of contradiction assume that Lemma fails. Then, there is i such that no node of ρ is reachable from the nodes which are on ω-branches of T and are on the level $> i$. If u is not on an ω-branch of T then, by König Lemma, it has only finitely many descendants in T. Hence, from the nodes on the level $i+1$ of T only finitely many nodes of ρ are reachable. This contradicts the assumption that T is spanning which implies that from the nodes of level $i+1$ of T every node on every level $> i$ of $Dag_{\alpha,\mathcal{A}}$ is reachable. □

Proof. (of Proposition 1) Let ρ be an accepting run on α. Let π be as in Lemma 2. If there is i such that $\forall j > i(\pi(j) = \rho(j))$, then π is an accepting branch. Otherwise, there is an ω-sequence $g(0) < g(1) < \cdots$ such that $\pi(g(i)) \neq \rho(g(i))$ and there is a node in $\rho \setminus \pi$ on a level $< g(i+1)$ which is reachable in $T_{\alpha,\mathcal{A}}$ from $\pi(g(i))$ by a path r_i. Hence, the run which follows π up to $g(i)$ then follows r_i and then follows ρ is accepting. Since all these runs are different, we obtain infinitely many accepting runs on α. Contradiction. □

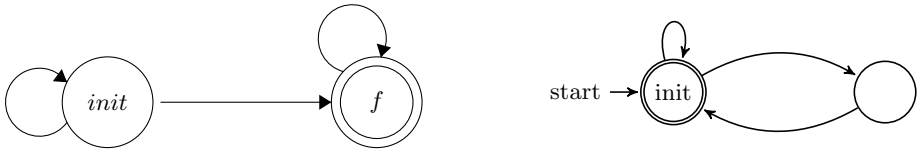

Fig. 3. Büchi automata over a unary alphabet with countably many and uncountably many accepting runs

Example 3. This example shows that the requirement that $\mathcal{A}$ is finitely ambiguous is necessary for Proposition 1. Let $\mathcal{A}$ be the automaton in the left of Fig. 3. The unique ω string over the unary alphabet is accepted by $\mathcal{A}$. One of the spanning trees of its computational DAG is the tree with one ω-branch $\langle init, 0\rangle, \ldots \langle init, i\rangle, \ldots$ of *init*-nodes, where each *init* node has two sons: one is a leaf labelled f and the second one is its son on the ω-branch. The unique ω-branch of this tree is not accepting.

Remark 4. Proposition 1 is stated for automata with Büchi acceptance conditions, yet it holds (with the same proof) for any prefix independent acceptance conditions; in particular, for Muller and Parity acceptance conditions.

Code of a narrow forest Let t be a forest such that for every $q \in Q_{\mathcal{A}}$ and i, t has at most one node labeled by q at distance (level) i from the roots. The edges between nodes on i-th and $i+1$-th levels of t can be described by a partial function $f_i : Q_{\mathcal{A}} \rightharpoonup Q_{\mathcal{A}}$ that maps a node at level $i+1$ to its father; there is an edge to node q at level $i+1$ from a node q' at level i if $f_i(q) = q'$. Hence, such a tree can be coded by a string $code(t) := f_0 f_1 \ldots$ over the alphabet of

the partial functions $Q_{\mathcal{A}} \rightharpoonup Q_{\mathcal{A}}$. Moreover, this code has the following *well-formedness* property: the image of f_{i+1} is a subset of the domain of f_i for all $i \in \mathbb{N}$.

A **transducer** with an input alphabet Σ and an output alphabet Γ is a complete deterministic automaton $\mathcal{D}$ over the alphabet Σ together with an output function $out : Q_{\mathcal{D}} \times \Sigma \to \Gamma$. Such a transducer defines a function $F : \Sigma^\omega \to \Gamma^\omega$ defined as follows: for $\alpha := a_1 a_2 \dots$ let $q_0 a_1 q_1 a_2 \dots$ be the unique run of $\mathcal{D}$ on α. Then, $F(\alpha) := b_1 b_2 \dots$ where $b_i := out(q_{i-1}, a_i)$.

The following proposition holds for arbitrary automata and its proof is based on a standard powerset construction.

Proposition 5 *For every Büchi automaton $\mathcal{A}$ there is a transducer $\mathcal{T}_{\mathcal{A}}$ of state complexity $2^{|Q_{\mathcal{A}}|}$ which for every $\alpha \in \Sigma_{\mathcal{A}}^\omega$ output (the code of) a spanning forest of $Dag_{\alpha,\mathcal{A}}$.*

Proof. The states of $\mathcal{T}_{\mathcal{A}}$ are the set of the subsets of $Q_{\mathcal{A}}$. The initial state is the set of the initial states of $\mathcal{A}$. After reading a string $s = a_1 \dots a_n$ its state will be the set of states reachable from the initial states in $\mathcal{A}$ after reading s. In a state B when reading a it goes to the state $B' := \cup_{q \in B} \delta_{\mathcal{A}}(q, a)$ and outputs any function g with domain B' and range B such that if $g(q') = q$ then $q' \in \delta_{\mathcal{A}}(q, a)$. □

Combining Propositions 1 and 5 we obtain:

Theorem 6. *For every finitely ambiguous Büchi automaton $\mathcal{A}$ there is a transducer $\mathcal{T}_{\mathcal{A}}$ of state complexity $2^{|Q_{\mathcal{A}}|}$ which for every $\alpha \in \Sigma_{\mathcal{A}}^\omega$ outputs (the code of) a $|Q_{\mathcal{A}}|$-narrow forest T such that T has an accepting branch iff $\mathcal{A}$ accepts α.*

4 Recognizing the narrow forests without accepting branches

In this section we construct a Büchi automaton $\mathcal{U}$ that accepts the code of n-narrow forest t iff t has no accepting ω-branch. Moreover, this automaton will be both unambiguous and semi-deterministic (semi-deterministic is defined on page 8).

Let $Q \rightharpoonup Q$ be the set of partial functions from Q to Q and $F \subseteq Q$. Recall that an ω-string $g_0 g_1 \dots$ over $Q \rightharpoonup Q$ is well-formed if for all i the image of g_{i+1} is a subset of the domain of g_i. The transducer $\mathcal{T}_{\mathcal{A}}$ from Theorem 6 outputs only well-formed ω-strings. We first describe auxiliary automata $\mathcal{C}$, $\mathcal{D}$ and $\mathcal{B}$ and then present our construction for $\mathcal{U}$.

Lemma 7. *The automaton $\mathcal{C}$ in Fig. 4 is deterministic and it accepts a well-formed ω-word α over $Q \rightharpoonup Q$ iff the forest described by α has at least one ω-branch which starts at an (initial) F-node, and has no other F-nodes.*

Proof. Let a forest t' be obtained from t by removing all the initial (root) nodes which are not in F and their descendants, and removing all the non-initial F-nodes and their descendants.

States: $\mathcal{C}$ has an initial state $init_{\mathcal{C}}$ and the non-empty subsets of $Q \setminus F$.
Transition relation $\delta_{\mathcal{C}}$**.** Let $g : Q \rightharpoonup Q$.

$$\delta_{\mathcal{C}}(init_{\mathcal{C}}, g) := \{q \in Q \setminus F \mid g(q) \in F\}$$

$$\delta_{\mathcal{C}}(L, g) := \{q \in Q \setminus F \mid g(q) \in L\}$$

Büchi Acceptance Condition: All states are accepting states.

Fig. 4. A Deterministic Safe Büchi automaton $\mathcal{C}$

A code of t should be accepted iff t' has at least one ω-branch. By König Lemma, the latter is equivalent to t' has a node on every level $i \in \mathbb{N}$.

Assume that after reading $g_1 \dots g_i$, the automaton is in a state L. Then $q \in L$ iff there is a path π to a node q on level i from an F-root such that π contains no other F-node. Therefore, there is an ω-run of $\mathcal{C}$ on α (i.e., L is always non-empty) iff t' has a node on every level. □

Lemma 8. *The automaton $\mathcal{D}$ in Fig. 5 is deterministic and it accepts a well-formed ω-word α over $Q \rightharpoonup Q$ iff the forest described by α has no ω-branch that passes through an F-node.*

States: $\mathcal{D}$ has an initial state $init_{\mathcal{D}}$ and states of the form $\langle A, D\rangle$ where $A \supseteq D$ are subsets of Q.
Transition relation $\delta_{\mathcal{D}}$**.** Let $g : Q \rightharpoonup Q$.

- $\delta_{\mathcal{D}}(init_{\mathcal{D}}, g) = \langle A, D\rangle$, where $A := D := (Dom(g) \cap F) \cup g^{-1}(F)$.
- $\langle A', D'\rangle := \delta_{\mathcal{D}}(\langle A, D\rangle, g)$ if

$$A' := g^{-1}(A) \cup (Dom(g) \cap F)$$

$$D' := \begin{cases} g^{-1}(D) & \text{if } D \neq \emptyset \\ A' & \text{otherwise} \end{cases}$$

Büchi Acceptance Condition: $F_{\mathcal{D}} := \{\langle A, \emptyset\rangle \mid A \subseteq Q\}$.

Fig. 5. The Deterministic Büchi automaton $\mathcal{D}$

Proof. By König Lemma, no ω-branch has an F node iff every F node has a finite number of descendants. We use a variant of the breakpoint construction [12].

After reading a string $g_1 g_2 \dots g_k$ the automaton records the set $A \subseteq Q$ of descendants (on the current level k) of the F-nodes visited so far. In addition to the set A, the automaton records a subset $D \subseteq A$ which is used to verify that every F-node has a finite set of descendants. The verification is done by phases. After reading the first letter g_1, D is the same as A and is equal to the set of the nodes on level one which have an F-node as ancestor. This starts the first phase; we want to verify that D has a finite number of descendants. When

reading $g_2 g_3 \ldots$ we update D to the set of descendants on the current level of the nodes initially assigned to D. We keep updating D until it becomes empty. If the nodes assigned initially to D are not on ω-branches, then D will become eventually empty, and this completes the first phase.

We begin the next phase by assigning to D the set of nodes in A (recall that A is updated all the time and always keeps the nodes at the current level that have an F-node as ancestor); we will continue to update D to keep the descendants of the nodes reassigned to D until it becomes empty. The automaton continues in this way: when D becomes empty, it starts a new phase by re-assigning to D the nodes in A, and computes the set of descendants of the nodes re-assigned to D in the beginning of the phase, which are at the current level.

We claim that $g_1 \ldots g_i \ldots$ has no F node on any ω-branch iff D is empty infinitely often. Indeed, assume that $g_1 \ldots g_i \ldots$ has an F-node u at level i on an ω-branch π. Then, this node will enter A at stage i. If D is not empty at any stage $j > i$, then we are done. If D becomes empty at a stage $j > i$, then A will contain $\pi(j)$ the j-th node of π which is a descendant of u on π. At stage j, the node $\pi(j)$ will enter D and at every stage $l > j$, D will contain $\pi(l)$ and therefore will be non-empty.

Conversely, if $g_1 \ldots g_i \ldots$ has no F-node on any ω-branch, then A and D contain only nodes with finitely many descendants. Hence, each time D is reassigned, it will become eventually empty. □

Now, from $\mathcal{D}$, a Büchi automaton $\mathcal{B}$ that accepts a narrow forest t iff t has no accepting ω-branch is obtained as follows. In addition to the states of $\mathcal{D}$, $\mathcal{B}$ has a new initial state $init_{\mathcal{B}}$. $\mathcal{B}$ waits in $init_{\mathcal{B}}$ until a guessed stage k from which no F nodes will appear on the ω-branches of t. Then, it moves to the initial state of $\mathcal{D}$. It is clear that $\mathcal{B}$ accepts $\alpha := g_1 \ldots g_k \ldots$ iff $\mathcal{D}$ accepts a suffix of α iff every ω-branch of α has finitely many F-nodes.

Recall that a Büchi automaton is **semi-deterministic**, if for every state q reachable from an accepting state and $b \in \Sigma$ there is at most one transition labelled by b from q. Semi-deterministic automata are sometimes called limit deterministic and were introduced by Vardi [20].

$\mathcal{B}$ is semi-deterministic. Its only nondeterministic transitions are in its initial state. However, it is ambiguous because if it accepts t by leaving the initial state after reading a prefix of length k, every run that leaves $init_{\mathcal{B}}$ after a prefix of length $> k$ is accepting. We will disambiguate $\mathcal{B}$ by leaving its initial state as early as possible, namely, when reading the last F-node on the ω-branches of t.

Lemma 9. *The following are equivalent:*

1. *The last occurrence of F-nodes on the ω-branches of t described by $\alpha := g_1 \ldots g_k \ldots$ is on level $k-1$.*
2. *(a) $g_{k+1} \ldots g_i \ldots$ has no F-node on the ω-branches, i.e., it is accepted by $\mathcal{D}$, and*
   (b) $g_k g_{k+1} \ldots g_i \ldots$ has an ω-branch that starts at an F-node and contains no other F-node, i.e., it is accepted by $\mathcal{C}$.

For $F_{\mathcal{A}} \subseteq Q$

$Lang(\mathcal{U}) = \{g_1 \dots g_i \dots \mid \text{no } F_{\mathcal{A}}\text{-accepting } \omega\text{- branch }\}$.

States: $init_{\mathcal{U}}$, $wait$, $Q_{\mathcal{D}}$, $Q_{\mathcal{C}} \times Q_{\mathcal{D}}$.
Transition relation δ. Let $g : Q \rightharpoonup Q$.

- $\delta(init_{\mathcal{U}}, g) := \begin{cases} \{wait, \delta_{\mathcal{D}}(init_{\mathcal{D}}, g)\} & \text{if the image of } g \cap F_{\mathcal{A}} = \emptyset \\ \{wait, \delta_{\mathcal{C}}(init_{\mathcal{C}}, g) \times init_{\mathcal{D}}\} & \text{otherwise} \end{cases}$
- $\delta(wait, g) := \{wait, \delta_{\mathcal{C}}(init_{\mathcal{C}}, g) \times init_{\mathcal{D}}\}$
- $\delta(q, g) := \delta_{\mathcal{D}}(q, g)$ for $q \in Q_{\mathcal{D}}$.
- $\delta(\langle q_1, q_2\rangle, g) := \langle \delta_{\mathcal{C}}(q_1, g), \delta_{\mathcal{D}}(q_2, g)\rangle$ for $\langle q_1, q_2\rangle \in Q_C \times Q_{\mathcal{D}}$.

Büchi Acceptance Condition: $F_{\mathcal{U}} := F_{\mathcal{D}} \cup Q_{\mathcal{C}} \times F_{\mathcal{D}}$ (recall that $Q_{\mathcal{C}} = F_{\mathcal{C}}$)

Fig. 6. The unambiguous Büchi automaton $\mathcal{U}$

Now, we are ready to describe a semi-deterministic unambiguous automaton $\mathcal{U}$ which accepts (a well-formed) $\alpha := g_1 \dots$ iff the forest described by α has no accepting ω-branch. $\mathcal{U}$ works as follows:

In its initial state $\mathcal{U}$ either (a) immediately wakes up $\mathcal{D}$ and then accepts if $\mathcal{D}$ accepts, or (b) waits for an arbitrary number $k \geq 0$ of steps, wakes up $\mathcal{C}$ which starts to work on $g_k \dots$ and in the next step wakes up $\mathcal{D}$ which works on $g_{k+1} \dots$, and accepts if both accept. The explicit description of $\mathcal{U}$ is in Fig. 6.

Note that $\mathcal{U}$ is semi-deterministic; its only nondeterministic states are $init_{\mathcal{U}}$ and $wait$. It is also unambiguous. If $\mathcal{U}$ accepts $\alpha := g_1 g_2 \dots$, then either forest described by α has no F-node on ω-branches and in this case the only accepting run on α goes to D, or the last occurrence of F-nodes on its ω-branches is on level $k-1$ and in this case its only accepting run enters $Q_{\mathcal{C}} \times Q_{\mathcal{D}}$ when reading k-th letter.

5 State Complexity

If $\mathcal{A}$ has n states, then the number of states $|Q_{\mathcal{U}}|$ of $\mathcal{U}$ is bounded by $2 + |Q_{\mathcal{D}}| + |Q_{\mathcal{D}}| \times |Q_{\mathcal{C}}| \leq 2|Q_{\mathcal{D}}| \times |Q_{\mathcal{C}}| \leq 2 \times 3^n \times 2^n$.

The automaton that accepts the complement of $\mathcal{A}$ is the product of the transducer $\mathcal{T}_{\mathcal{A}}$ and $\mathcal{U}$. Its state complexity is bounded by $2 \times 2^n \times 2^n \times 3^n$. However, not all the states of the product are reachable.

At least half of the set of reachable states in the product of $\mathcal{T}_{\mathcal{A}}$ and $\mathcal{U}$ are states which correspond to the tuples $\langle B, L, A, D\rangle$ where $B, L, A, D \subseteq Q_{\mathcal{A}}$, B is a state of $\mathcal{T}_{\mathcal{A}}$, $\langle L, A, D\rangle$ is a $Q_{\mathcal{C}} \times Q_{\mathcal{D}}$ state of $\mathcal{U}$. Moreover, if $\langle B, L, A, D\rangle$ is a reachable state, then B, L, A, D are subsets of $Q_{\mathcal{A}}$ which satisfy the following restrictions: $L \subseteq B \setminus F_{\mathcal{A}}$, $D \subseteq A \subseteq B$ and L is disjoint from A. The number of tuples which satisfy these restriction is bounded by the cardinality of the set of functions from $Q_{\mathcal{A}}$ into $\{1, 2, 3, 4, 5\}$ which is $5^{|Q_{\mathcal{A}}|}$.

Theorem 10. *The complement of the ω-language accepted by a finitely ambiguous Büchi automaton with n states is accepted by an unambiguous semi-deterministic automaton with $\leq 2 \times 5^n$ states.*

6 Further Results and Open Questions

We introduced a natural class of finitely ambiguous Büchi automata and proved that their complementation is easier than the complementation of general NBA. The class of finitely ambiguous Büchi automata includes the class of unambiguous and the class of k-ambiguous Büchi automata. It is incomparable with the class of semi-deterministic Büchi automata. The automaton in the left of Fig. 3 is semi-deterministic, but it is not finitely ambiguous. The automaton in the right of Fig. 1 is finitely ambiguous, but it is not semi-deterministic.

For nondeterministic ϵ-free automata over words, on every word there are at most finitely many accepting runs[1]; however, even over unary ω-words there are nondeterministic automata with uncountable many accepting runs (see the automaton on the right of Fig. 3). Hence, over ω-words one can consider a class of countably ambiguous automata which is a proper subclass of nondeterministic automata.

We do not know whether the complementation of countably ambiguous Büchi automata is easier than of NBA.

It is easy to check in polynomial time whether an automaton is unambiguous.

Weber and Seidl [21] investigated several classes of ambiguous automata on words and provided respective structural characterisations of the classes from which polynomial time algorithms are obtained for deciding the membership in each of these classes.

In particular, they proved that the following Bounded Ambiguity Criterion (BA) characterizes whether there is a bound k such that a nondeterministic automaton on words has at most k accepting runs on each word.

Forbidden Pattern for Bounded Ambiguity: There are distinct useful[2] states $p, q \in Q$ such that for some word u, there are runs on u from p to p, from p to q and from q to q (see Fig. 7.)

Theorem 11 (Weber and Seidl [21]). *Let $\mathcal{A}$ be an NFA on words. There is no bound $k \in \mathbb{N}$ such that on every word $\mathcal{A}$ has at most k accepting runs if and only if $\mathcal{A}$ contains the forbidden pattern for bounded ambiguity (see Fig. 7).*

It is easy to use the above theorem and to show that it holds when "NFA on words" is replaced by "NBA."

The following Forbidden Patterns for Countable Ambiguity (FPCA) and Finite Ambiguity (FPFA) characterize NBA for which on every ω-string there are at most countably many and finitely many accepting runs.

[1] unfortunately, an automaton on words is called finitely ambiguous if it is k-ambiguous for some k. Maybe a more appropriate name for such automata is "bounded ambiguous."

[2] A state is useful if it is on an accepting run.

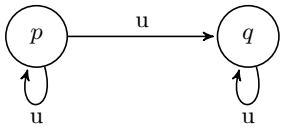

Fig. 7. Forbidden Pattern for Bounded Ambiguity

Forbidden Pattern for Countable Ambiguity: there is a final useful state f and there are two distinct runs on the same word u from f to f.

Forbidden Pattern for Finite Ambiguity: either contains the forbidden pattern for countable ambiguity or there is a final useful state f, a useful state $q \neq f$, and a string u such that there are runs on u from q to q, from q to f and from f to f (see Fig. 8).

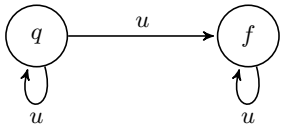

Fig. 8. Forbidden Pattern for Finite Ambiguity

Theorem 12. *Let $\mathcal{A}$ be a NBA.*

1. *$\mathcal{A}$ has uncountably many accepting runs on some ω-word if and only if $\mathcal{A}$ contains the forbidden pattern for countable ambiguity.*
2. *$\mathcal{A}$ has infinitely many accepting runs on some ω-word if and only if $\mathcal{A}$ contains the forbidden pattern for finite ambiguity.*

The proof of Theorem 12 is given in the full paper. As an immediate consequence, we obtain a polynomial time algorithm that decides whether NBA is bounded ambiguous, finitely ambiguous, or countable ambiguous.

Kahler and Wilke [9] proved that a nondeterministic Büchi automaton with n states is equivalent to an UBA with $(3n)^n$ states. This gives $(3n)^n$ upper bound on the succinctness gap between NBA and UBA. The 2^n lower bound is easily derived from the 2^n succinctness gap for automata over words [10]. The tight bounds on the succinctness gap between finitely ambiguous, unambiguous, k-ambiguous, countable ambiguous Büchi automata and NBA are open questions.

Acknowledgments. I would like to thank anonumous reviewers for their useful and insightful suggestions.

References

1. Arnold, A.: Rational ω-languages are non-ambiguous. Theoretical Computer Science 26 (1983) 221-223.
2. C. Baier and Stefan Kiefer, J. Klein, S. Klüppelholz, D. Müller and J. Worrell. Markov Chains and Unambiguous Büchi Automata, In CAV 2016, pp. 23–42.
3. J. R. Büchi. On a decision method in restricted second order arithmetic. In CLMPS 1960, pages 1-11. Stanford University Press, 1962.
4. Colcombet, T.: Unambiguity in automata theory. In DCFS 15, LNCS vol. 9118, pp. 3–18. Springer (2015).
5. Chan, T., Ibarra, O.H.: On the finite-valuedness problem for sequential machines. Theor. Comput. Sci. 23, 95–101 (1983).
6. E. Friedgut, O. Kupferman, and M. Y. Vardi. Büchi complementation made tighter. Inter. J. of Foundations of Computer Science, 17(4):851–868, 2006.
7. Y. Han and A. Salomaa and K. Salomaa, Ambiguity, Nondeterminism and State Complexity of Finite Automata, Acta Cybern., 23 1, 141–157, 2017.
8. D. Isaak and C. Loding. Efficient inclusion testing for simple classes of unambiguous ω-automata. IPL 112(14-15):578–582, 2012.
9. D. Kahler and T. Wilke. Complementation, disambiguation, and determinization of Büchi automata unified. In Proc. ICALP 2008, LNCS vol. 5125 pp. 724-735. Springer, 2008.
10. Leiss, E. Succinct representation of regular languages by Boolean automata. Theoret. Comput. Sci., 13: 323–330, 1981.
11. R. McNaughton. Testing and generating infinite sequences by a infnite automaton. Information and Control, 9(5):521–530, October 1966.
12. S. Miyano and T. Hayashi. Alternating Infite automata on ω-words. Theoretical Computer Science, 32(3):321–330, 1984.
13. Leung, H. Descriptional complexity of NFA of different ambiguity. Intern. J. Foundations Comput. Sci., 16(5): 975–984, 2005.
14. Jirasek J., Jiraskova G., Sebej J. (2016) Operations on Unambiguous Finite Automata. In DLT 2016, LNCS vol 9840. Springer, 2016.
15. S. Safra. On the complexity of omega-automata. In FOCS 1988, pages 319–327. IEEE Computer Society, 1988.
16. S. Schewe. Büchi complementation made tight. In TACS 2009, pp. 661–672, 2009.
17. A. P. Sistla, M. Y. Vardi, and P. Wolper. The complementation problem for Büchi automata with applications to temporal logic. TCS 49(3):217–239, 1987.
18. Stearns, R.E., Hunt, H.B.: On the equivalence and containment problems for un- ambiguous regular expressions, regular grammars and finite automata. SIAM J. Comput. 14(3), 598–611 (1985).
19. W. Thomas. Automata on infinite objects. In Handbook of Theoretical Computer Science, Vol. B: Formal Models and Sematics, pp. 133–192. 1990.
20. M. Y. Vardi. Automatic verification of probabilistic concurrent finite-state programs. In FOCS 1985, pages 327–338, 1985.
21. Weber, A. and Seidl, H.: On the degree of ambiguity of finite automata. Theor. Comput. Sci. 88(2), 32–349, 1991.
22. Q. Yan. Lower bounds for complementation of omega-automata via the full automata technique. J. of Logical Methods in Computer Science, 4(1:5), 2008.