Lower Bounds on Witnesses for Nonemptiness of Universal co-Büchi Automata

Orna Kupferman[1] and Nir Piterman[2⋆]

[1] Hebrew University
[2] Imperial College London

Abstract. The nonemptiness problem for nondeterministic automata on infinite words can be reduced to a sequence of reachability queries. The length of a shortest witness to the nonemptiness is then polynomial in the automaton. Nonemptiness algorithms for alternating automata translate them to nondeterministic automata. The exponential blow-up that the translation involves is justified by lower bounds for the nonemptiness problem, which is exponentially harder for alternating automata. The translation to nondeterministic automata also entails a blow-up in the length of the shortest witness. A matching lower bound here is known for cases where the translation involves a $2^{O(n)}$ blow up, as is the case for finite words or Büchi automata.

Alternating co-Büchi automata and witnesses to their nonemptiness have applications in model checking (complementing a nondeterministic Büchi word automaton results in a universal co-Büchi automaton) and synthesis (an LTL specification can be translated to a universal co-Büchi tree automaton accepting exactly all the transducers that realize it). Emptiness algorithms for alternating co-Büchi automata proceed by a translation to nondeterministic Büchi automata. The blow up here is $2^{O(n \log n)}$, and it follows from the fact that, on top of the subset construction, the nondeterministic automaton maintains ranks to the states of the alternating automaton. It has been conjectured that this super-exponential blow-up need not apply to the length of the shortest witness. Intuitively, since co-Büchi automata are memoryless, it looks like a shortest witness need not visit a state associated with the same set of states more than once. A similar conjecture has been made for the width of a transducer generating a tree accepted by an alternating co-Büchi tree automaton. We show that, unfortunately, this is not the case, and that the super-exponential lower bound on the witness applies already for universal co-Büchi word and tree automata.

1 Introduction

Finite automata on infinite objects were first introduced in the 60's. Motivated by decision problems in mathematics and logic, Büchi, McNaughton, and Rabin developed a framework for reasoning about infinite words and trees [2, 11, 16]. The framework has proven to be very powerful. Automata, and their tight relation to second-order monadic logics were the key to the solution of several fundamental decision problems in mathematics and logic [17]. Indeed, for many highly expressive logics, it is possible to translate

⋆ Supported by the UK EPSRC project *Complete and Efficient Checks for Branching-Time Abstractions* (EP/E028985/1).

a formula in the logic to an automaton accepting exactly all the models satisfying the formula. The formula is then satisfiable iff the language of the automaton is not empty. Thus, decidability can be reduced to the emptiness problem.

Today, automata on infinite objects are used for specification and verification of nonterminating systems [18, 9, 19]. The emptiness problem plays a key role also in these more modern applications. Two important examples are model checking and synthesis. Model checking a system with respect to a specification is reduced to checking the emptiness of the product of the system with an automaton accepting exactly all models that violate the specification [19]. Synthesis of a reactive system that satisfies a desired specification is reduced to checking the emptiness of a tree automaton accepting all possible strategies that realize the specification [15].

In the case of finite nondeterministic automata on finite words, the emptiness problem is simple: The automaton accepts some word if there is a path from an initial state to an accepting state (c.f., [4]). Thus, the automaton is viewed as a graph, its alphabet is ignored, and emptiness is reduced to reachability in finite graphs. An important and useful outcome of this simplicity is the fact that when the language of the automaton is not empty, it is easy to return a witness to the nonemptiness — a word v that labels a path from an initial state to a final states. Clearly, reachability may be checked only along simple paths, thus the length of a witness is bounded by the number of states of the automaton.

The case of finite nondeterministic automata on infinite words is similar. Acceptance in such automata depends on the set of states that a run visits infinitely often. For example, in the Büchi acceptance condition, some states are designated as accepting, and in order for a run to be accepting it has to visit at least one of these states infinitely often. Nonemptiness is slightly more complicated, but again, the automaton is viewed as a graph, its alphabet is ignored, and emptiness is reduced to a sequence of reachability queries in finite graphs. Now, the witness to the nonemptiness is a word of the form $v \cdot u^\omega$, where the word v labels a path from an initial state to some accepting state, and the word u labels a path from this accepting state to itself. Since both v and u are extracted from reachability queries on the graph, their lengths are bounded by the number of states of the automaton.[1] For acceptance conditions more complicated than Büchi, the emptiness test is more involved, but still, as long as we consider nondeterministic automata, emptiness can be reduced to a sequence of reachability queries on the graph of the automaton, and a nonempty automaton has a witness of the form $v \cdot u^\omega$ for v and u polynomial in the number of states of the automaton.

Alternating automata enrich the branching structure of the automaton by combining universal and existential branching. In the presence of alternation, we can no longer ignore the alphabet when reasoning about emptiness. Indeed, the different copies of the automaton have to agree on the letters they read on the same position of the word. The standard solution is to remove alternation by translating the automaton to an equivalent nondeterministic automaton, and checking the emptiness of the latter. This simple solution is optimal, as the exponential blow-up that the translation involves is justified by lower bounds for the nonemptiness problem, which is exponentially harder in the

[1] In fact, it can be shown that even the sum of their lengths is bounded by the number of states of the automaton [6].

alternating setting (c.f., NLOGSPACE vs. PSPACE for nondeterministic vs. alternating automata on finite words).

The translation to nondeterministic automata also entails an exponential blow-up in the length of the shortest witness. Can this blow up be avoided? A negative answer for this question is known for alternating automata on finite words and alternating Büchi automata. There, removing alternation from an alternating automaton with n states results in a nondeterministic automaton with $2^{O(n)}$ states [3, 12], and it is not hard to prove a matching lower bound [1]. Note also that a polynomial witness would have led to the nonemptiness problem being in NP, whereas it is known to be PSPACE-complete.

Things become challenging when the removal of alternation involves a super-exponential blow up. In particular, emptiness algorithms for alternating co-Büchi automata proceed by a translation to nondeterministic Büchi automata, and the involved blow up is $2^{O(n \log n)}$. Alternating co-Büchi automata have been proven useful in model checking (complementing a nondeterministic Büchi word automaton results in a universal co-Büchi automaton) and synthesis (an LTL specification can be translated to a universal co-Büchi tree automaton accepting exactly all the transducers that realize it [8, 5]). In the case of model checking, the witness to the nonemptiness is a computation that violates the property. In the case of synthesis, the witness is a system that realizes the specification). Thus, we clearly seek shortest witnesses.

The $2^{O(n \log n)}$ blow up follows from the fact that, on top of the subset construction, the nondeterministic automaton maintains ranks to the states of the alternating automaton. It has been conjectured that this super-exponential blow-up need not apply to the length of the shortest witness. Intuitively, since co-Büchi automata are memoryless, it seems as if a shortest witness need not visit a state associated with the same set of states more than once. This intuition suggests that a shortest witness need not be longer than $2^{O(n)}$. A similar conjecture has been made for the width of a transducer[2] generating a tree accepted by an alternating co-Büchi tree automaton [8].

In this paper we show that, unfortunately, this is not the case, and the super-exponential blow-up in the translation of alternating co-Büchi automata to nondeterministic Büchi automata is carried over to a super-exponential lower bound on the witness to the nonemptiness. In fact, the lower bound applies already for universal co-Büchi automata. We start with the linear framework. There, we show that for every odd integer $n \geq 1$, there exists a universal co-Büchi word automaton $\mathcal{A}_n$ with n states such that the shortest witness to the nonemptiness of $\mathcal{A}_n$ has a cycle of length $\frac{n+1}{2}!$.

In the branching framework, the witness to the nonemptiness is a transducer that generates a tree accepted by the automaton. The linear case trivially induces a lower bound on the size of such a transducer. In the branching framework, however, it is interesting to consider also the width of the witness transducer. In particular, the LTL synthesis algorithm in [8], which is based on checking the nonemptiness of a universal co-Büchi tree automaton, is incremental, and it terminates after k iterations, with k being an upper bound on the width of a transducer generating a tree accepted by the automaton. The bound used in [8] is super-exponential, and has been recently tightened to $2n(n!)^2$ [14, 10]. It is conjectured in [8] that the bound can be improved to $2^{O(n)}$. As in the word case, the intuition is convincing: The alternating automaton may send a set of states to a sub-

[2] Essentially, the width of a transducer is the number of different states that the transducer may be at after reading different input sequences of the same length.

tree of the input tree, in which case the subtree should be accepted by all the states in the set. The memoryless nature of the co-Büchi condition suggests that if in an accepting run of the automaton the same set of states is sent to different subtrees, then there is also an accepting run on a tree in which these subtrees are identical. Thus, we do not need more than 2^n different subtrees in a single level of the input tree. We show that, unfortunately, this intuition fails, and there is a lower bound of $\frac{n+1}{2}!$ on the width of the transducer. Formally, we show that for every odd integer $n \geq 1$, there exists a universal co-Büchi tree automaton $\mathcal{B}_n$ with n states such that every tree accepted by $\mathcal{B}_n$ is such that, all levels beyond a finite prefix have at least $\frac{n+1}{2}!$ different subtrees. Thus, the minimal width of a transducer that generate a tree accepted by $\mathcal{B}_n$ has width at least $\frac{n+1}{2}!$.

Our constructions use a very large alphabet. Indeed, the alphabet of the automata $\mathcal{A}_n$ and $\mathcal{B}_n$ has $\frac{n+1}{2}!$ letters. In the case of words, the word accepted by the automaton is a cycle consisting of all these letters ordered in some fixed order (say, lexicographically). The case of trees is similar. We were not able to reduce the size of the alphabet. While the question of a smaller alphabet is very interesting, it is of less practical importance: Constructions for removal of alternation introduce an exponential alphabet in an intermediate step (where the exponent is quadratic in the number of states). The larger alphabet is discarded at a later stage but the degree of nondeterminism induced by it remains in the resulting nondeterministic automaton. Furthermore, the size of the alphabet does not play a role in these constructions, and obviously does not play a role when checking the emptiness of a nondeterministic automaton.

2 Universal co-Büchi Word Automata

A *word automaton* is $\mathcal{A} = \langle \Sigma, Q, \delta, Q_{in}, \alpha \rangle$, where Σ is the input alphabet, Q is a finite set of states, $\delta : Q \times \Sigma \to 2^Q$ is a transition function, $Q_{in} \subseteq Q$ is a set of initial states, and α is an acceptance condition that defines a subset of Q^ω.

Given an input word $w = \sigma_0 \cdot \sigma_1 \cdots$ in Σ^ω, a *run* of $\mathcal{A}$ on w is a word $r = q_0, q_1, \ldots$ in Q^ω such that $q_0 \in Q_{in}$ and for every $i \geq 0$, we have $q_{i+1} \in \delta(q_i, \sigma_i)$; i.e., the run starts in the initial state and obeys the transition function. Since the transition function may specify many possible transitions for each state and letter, $\mathcal{A}$ may have several runs on w. A run is accepting iff it satisfies the acceptance condition α. We consider here the *Büchi* acceptance condition, where $\alpha \subseteq Q$ is a subset of Q. For a run r, let $inf(r)$ denote the set of states that r visits infinitely often. That is, $inf(r) = \{q \in Q : q_i = q \text{ for infinitely many } i \geq 0\}$. A run r is accepting iff $inf(r) \cap \alpha \neq \emptyset$. That is, r is accepting if some state in α is visited infinitely often. The *co-Büchi* acceptance condition dualizes the Büchi condition. Thus, again α is a subset of Q, but a run r is accepting if $inf(r) \cap \alpha = \emptyset$. Thus, r visits all the states in α only finitely often.

If the automaton $\mathcal{A}$ is *nondeterministic*, then it accepts an input word w iff it has an accepting run on w. If $\mathcal{A}$ is *universal*, then it accepts w iff all its runs on w are accepting. The *language* of $\mathcal{A}$, denoted $\mathcal{L}(\mathcal{A})$ is the set of words that $\mathcal{A}$ accepts. Dualizing a nondeterministic Büchi automaton (NBW, for short) amounts to viewing it as a universal co-Büchi automaton (UCW, for short). It is easy to see that by dualizing $\mathcal{A}$, we get an automaton that accepts its complementary language.

In [7], Kupferman and Vardi analyze runs of UCW in terms of a ranking function one can associate with their run DAG. In the rest of this section, we describe their analysis.

Let $\mathcal{A} = \langle \Sigma, Q, Q_{in}, \delta, \alpha \rangle$ be a universal co-Büchi automaton with α. Let $|Q| = n$. The runs of $\mathcal{A}$ on a word $w = \sigma_0 \cdot \sigma_1 \cdots$ can be arranged in an infinite DAG (directed acyclic graph) $G = \langle V, E \rangle$, where

- $V \subseteq Q \times \mathbb{N}$ is such that $\langle q, l \rangle \in V$ iff some run of $\mathcal{A}$ on w has $q_l = q$. For example, the first level of G contains the vertices $Q_{in} \times \{0\}$.
- $E \subseteq \bigcup_{l \geq 0} (Q \times \{l\}) \times (Q \times \{l+1\})$ is such that $E(\langle q, l \rangle, \langle q', l+1 \rangle)$ iff $\langle q, l \rangle \in V$ and $q' \in \delta(q, \sigma_l)$.

Thus, G embodies exactly all the runs of $\mathcal{A}$ on w. We call G the *run* DAG of $\mathcal{A}$ on w. We say that a vertex $\langle q, l \rangle$ in G is an *α-vertex* iff $q \in \alpha$. We say that G is *accepting* if each path p in G contains only finitely many α-vertices. It is easy to see that $\mathcal{A}$ accepts w iff G is accepting.

Let $[2n]$ denote the set $\{0, 1, \ldots, 2n\}$. A *ranking* for G is a function $f : V \to [2n]$ that satisfies the following conditions:

1. For all vertices $\langle q, l \rangle \in V$, if $f(\langle q, l \rangle)$ is odd, then $q \notin \alpha$.
2. For all edges $\langle \langle q, l \rangle, \langle q', l+1 \rangle \rangle \in E$, we have $f(\langle q', l+1 \rangle) \leq f(\langle q, l \rangle)$.

Thus, a ranking associates with each vertex in G a rank in $[2n]$ so that ranks along paths decrease monotonically, and α-vertices cannot get an odd rank. Note that each path in G eventually gets trapped in some rank. We say that the ranking f is an *odd ranking* if all the paths of G eventually get trapped in an odd rank. Formally, f is odd iff for all paths $\langle q_0, 0 \rangle, \langle q_1, 1 \rangle, \langle q_2, 2 \rangle, \ldots$ in G, there is $l \geq 0$ such that $f(\langle q_l, l \rangle)$ is odd, and for all $l' \geq l$, we have $f(\langle q_{l'}, l' \rangle) = f(\langle q_l, l \rangle)$. Note that, equivalently, f is odd if every path of G has infinitely many vertices with odd ranks.

We now analyze the form of accetping run DAGs. The following three lemmata relate to DAGs induced by words accepted by $\mathcal{A}$. Consider a (possibly finite) DAG $G' \subseteq G$. We say that a vertex $\langle q, l \rangle$ is *finite* in G' iff only finitely many vertices in G' are reachable from $\langle q, l \rangle$. We say that a vertex $\langle q, l \rangle$ is *α-free* in G' iff all the vertices in G' that are reachable from $\langle q, l \rangle$ are not α-vertices. Note that, in particular, $\langle q, l \rangle$ is not an α-vertex.

We define an infinite sequence of DAGs $G_0 \supseteq G_1 \supseteq G_2 \supseteq G_3 \supseteq \ldots$ as follows.

- $G_0 = G$.
- $G_{2i+1} = G_{2i} \setminus \{\langle q, l \rangle \mid \langle q, l \rangle \text{ is finite in } G_{2i}\}$.
- $G_{2i+2} = G_{2i+1} \setminus \{\langle q, l \rangle \mid \langle q, l \rangle \text{ is } \alpha\text{-free in } G_{2i+1}\}$.

Lemma 1. *For every $i \geq 0$, there exists l_i such that for all $l \geq l_i$, there are at most $n - i$ vertices of the form $\langle q, l \rangle$ in G_{2i}.*

Lemma 1 implies that G_{2n} is finite, and G_{2n+1} is empty.

Each vertex $\langle q, l \rangle$ in G has a unique $i \geq 1$ such that $\langle q, l \rangle$ is either finite in G_{2i} or α-free in G_{2i+1}. This induces a function $f : V \to [2n]$ defined as follows.

$$f(\langle q, l \rangle) = \begin{cases} 2i & \text{If } \langle q, l \rangle \text{ is finite in } G_{2i}. \\ 2i+1 & \text{If } \langle q, l \rangle \text{ is } \alpha\text{-free in } G_{2i+1}. \end{cases}$$

Lemma 2. *For every two vertices $\langle q, l \rangle$ and $\langle q', l' \rangle$ in G, if $\langle q', l' \rangle$ is reachable from $\langle q, l \rangle$, then $f(\langle q', l' \rangle) \leq f(\langle q, l \rangle)$.*

Lemma 3. *For every infinite path in G, there exists and a vertex $\langle q, l\rangle$ such that all the vertices $\langle q', l'\rangle$ on the path that are reachable from $\langle q, l\rangle$ have $f(\langle q', l'\rangle) = f(\langle q, l\rangle)$.*

We can now conclude with Theorem 1 below.

Theorem 1. [7] *The* DAG *G is accepting iff it has an odd ranking.*

Proof. Assume first that there is an odd ranking for G. Then, every path in G eventually gets trapped in some odd rank. Hence, as α-vertices cannot get this rank, the path visits α only finitely often, and we are done.

For the other direction, note that Lemma 2, together with the fact that a vertex gets an odd rank only if it is α-free, imply that the function f described above is a ranking. Lemma 3 then implies that the ranking is odd. □

3 Lower Bound on Length of Accepted Words

In this section we construct, for every odd $n \geq 1$, a UCW $\mathcal{A}_n$ with n states such that the shortest words accepted by $\mathcal{A}$ have a cycle of length $\frac{n+1}{2}!$. The alphabet Σ_n of $\mathcal{A}_n$ has $\frac{n+1}{2}!$ letters, and there is an ordering $\leq$ of all the letters in Σ_n such that $\mathcal{A}_n$ accepts exactly all words vu^ω, where $v \in \Sigma_n^*$ and $u \in (\Sigma_n)^{\frac{n+1}{2}!}$ has all the letters in Σ_n ordered according to $\leq$.

Formally, given an odd $n \geq 1$, let $\mathcal{A}_n = \langle \Sigma_n, Q_n, \delta_n, Q_n, \alpha_n\rangle$, where

- Let Π_n be the set of permutations on $\{1, 3, 5, \ldots, n\}$ (the odd members of $\{1, \ldots, n\}$), and let $\leq$ be the lexicographic ordering[3] on the members of Π_n. Then, $\Sigma_n \subseteq \Pi_n \times \Pi_n$ is such that $\langle \pi, \pi'\rangle \in \Sigma_n$ iff π' is the (cyclic) successor of π in the order $\leq$. Thus, each letter of Σ_n is a pair $\langle \pi, \pi'\rangle$ of permutations, such that π' is the successor of π in the lexicographic order of Π_n. Note we refer to the order in a cyclic way, thus $\langle n \ldots 31, 13 \ldots n\rangle$ is a letter in Σ_n. For example, $\Pi_5 = \{135, 153, 315, 351, 513, 531\}$ and $\Sigma_5 = \{\langle 135, 153\rangle, \langle 153, 315\rangle, \langle 315, 351\rangle, \langle 351, 513\rangle, \langle 513, 531\rangle, \langle 531, 135\rangle\}$. Note that each permutation in Π_n contributes to Σ_n one letter, thus $|\Sigma_n| = |\Pi_n| = \frac{n+1}{2}!$.
- $Q_n = \{1, \ldots, n\}$.
- Consider a permutation $\pi \in \Pi_n$. An *even-extension* of π is a permutation σ of $\{1, 2, 3, \ldots, n\}$ obtained from π by using π for the odd positions and inserting in each even position e the even number e. For example, if $\pi = 153$, then $\sigma = 12543$. Let π and π' be such that $\langle \pi, \pi'\rangle \in \Sigma_n$, and let $\sigma = i_1 \cdots i_n$ and $\sigma' = j_1 \cdots j_n$ be the even extensions of π and π'. Then, for every $1 \leq k \leq n$, we define

$$\delta_n(i_k, \langle \pi, \pi'\rangle) = \begin{cases} \{j_1, \ldots, j_k\} & \text{if } k \text{ is odd} \\ \{j_1, \ldots, j_{k-1}\} & \text{if } k \text{ is even.} \end{cases}$$

  That is, when a state $h \in Q_n$ reads $\langle \pi, \pi'\rangle$, it checks its location in σ (this is the k for which $h = i_k$) and sends copies to all states in smaller (or equal, if k is odd) locations in σ' (these are the states h' for which $h' = j_{k'}$ for k' smaller than (or equal to) k. Note that for all even k's, we have $\delta_n(i_k, \langle \pi, \pi'\rangle) = \delta_n(i_{k-1}, \langle \pi, \pi'\rangle)$.

[3] The proof stays valid with every ordering.

For example, $\delta_5(3, \langle 135, 153\rangle) = \{1, 2, 5\}$. Indeed, the location of 3 in 12345 is 3 and the states located in the first three positions in 12543 are 1, 2, and 5. The other transitions on the letter $\langle 135, 153\rangle$ are defined similarly:

- $\delta_5(1, \langle 135, 153\rangle) = \delta_5(2, \langle 135, 153\rangle) = \{1\}$,
- $\delta_5(3, \langle 135, 153\rangle) = \delta_5(4, \langle 135, 153\rangle) = \{1, 2, 5\}$, and
- $\delta_5(5, \langle 135, 153\rangle) = \{1, 2, 3, 4, 5\}$.

– $\alpha_n = \{i \mid i \text{ is even}\}$. Thus, every infinite run of $\mathcal{A}_n$ has to visit only finitely many even states.

Note that for every word $v \in \Sigma^\omega$, the run DAG of $\mathcal{A}_n$ on v has all the states in Q_n appearing in every level of the DAG. This follows from the set of initial states of $\mathcal{A}_n$ being Q_n and the fact that for every letter $a = \langle \pi, \pi' \rangle \in \Sigma_n$, there exists one state q in Q_n (q is last number in π) for which the transition from q on a contains all the states in Q_n.

Let u be the word in $(\Sigma_n)^{\frac{n+1}{2}!}$ that contains all the letters in Σ_n ordered lexicographically. For example, when $n = 5$, we have that $u = \langle 135, 153\rangle\ \langle 153, 315\rangle\ \langle 315, 351\rangle\langle 351, 513\rangle\ \langle 513, 531\rangle\langle 531, 135\rangle$. We prove that $\mathcal{A}_n$ accepts the word u^ω. It follows that $\mathcal{A}_n$ accepts vu^ω for every word $v \in \Sigma^*$.

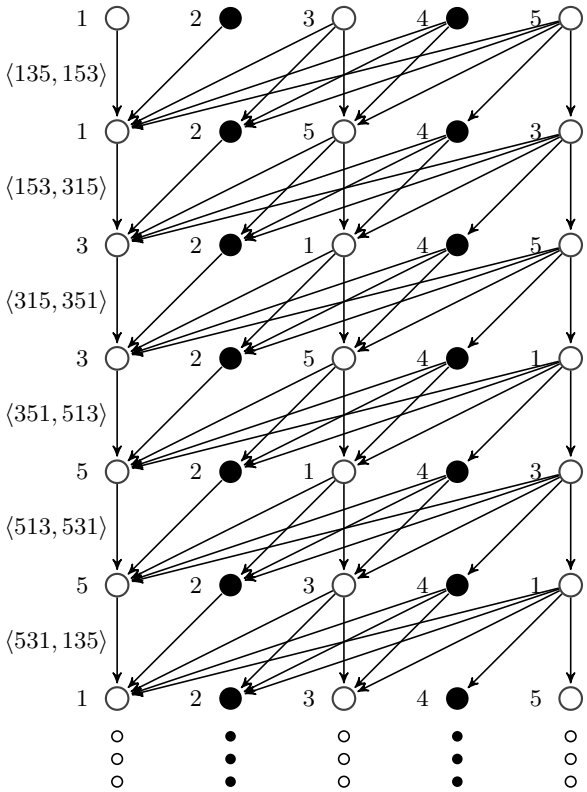

Fig. 1. The accepting run of $\mathcal{A}_5$ on u^ω.

Lemma 4. $u^\omega \in L(\mathcal{A}_n)$.

Proof. Consider the run DAG G of $\mathcal{A}_n$ on u^ω. In Figure 1, we describe the accepting run DAG of $\mathcal{A}_5$ on u^ω. As argued above, each level l of G consists of all the vertices in $Q_n \times \{l\}$. We arrange the vertices of G in columns numbered 1 to n. In the level that reads $\langle \pi, \pi' \rangle$, we arrange the vertices according to the position of the state component of each vertex in the even extension σ of π. For example, when we read $\langle 135, 153 \rangle$ in level 0, we consult the even extension 12345 of 135 and put the vertex $\langle 1, 0 \rangle$ in Column 1 (the leftmost), put $\langle 2, 0 \rangle$ in Column 2, and so on. Since u contains all the letters in Σ_n ordered lexicographically, the letter to be read in the next level is $\langle \pi', \pi'' \rangle$, and the vertices are arranged in columns in this level according to π'. By the definition of δ_n, the above implies that the edges in G go from columns to smaller or equal columns. Accordingly, all α-vertices appear in even columns and all other vertices appear in odd columns.

We prove that G has an odd ranking. For that, we prove, by induction on i, that the vertices in Column i, for $1 \leq i \leq n$, get rank i (independent of their level).

By definition, the set of successors of a vertex in Column 1 is a singleton containing the next vertex in Column 1. As all vertices in this column are not α-vertices, they are all α-free and they get rank 1. The set of successors of vertices in Column 2 is again a singleton containing only the next vertex in Column 1. Since vertices in Column 2 are α-vertices, they do not get rank 1. In the DAG G_2, however, these vertices have no successors. Thus, they are finite, and get rank 2.

The induction step is similar: the DAG G_i contains only vertices in Columns i to n. When i is odd, the vertices in Column i are α-free, and get rank i. When i is even, the vertices in Column i are finite, and get rank i too. □

Consider two letters $\langle \pi_1, \pi_1' \rangle$ and $\langle \pi_2, \pi_2' \rangle$ in Σ_n. We say that $\langle \pi_1, \pi_1' \rangle$ and $\langle \pi_2, \pi_2' \rangle$ are *gluable* if $\pi_1' = \pi_2$. Otherwise, $\langle \pi_1, \pi_1' \rangle$ and $\langle \pi_2, \pi_2' \rangle$ are *non-gluable*. We say that location $i \in \mathbb{N}$ is an *error* in w if letters i and $i+1$ in w are non-gluable. A word w is *bad* if w has infinitely many errors. The definition of non-gluable is extended to finite words in the obvious way. Consider a word $v \in \Sigma_n^*$. We denote by $first(v)$ the permutation $\pi \in \Pi_n$ such that the first letter of v is $\langle \pi, \pi' \rangle$, for the (lexicographic) successor π' of π. Similarly, we denote by $last(v)$ the permutation π' such that the last letter of v is $\langle \pi, \pi' \rangle$ for the predecessor π of π'. Given an even-extension $\sigma = i_1 \cdots i_n$ of a permutation, we say that the state i_k is the k-th state appearing in σ.

Consider a fragment of a run that starts in permutation π and ends in permutation π'. That is, the fragment reads the word v, the permutation π is $first(v)$, and the permutation π' is $last(v)$. We arrange the states in Q_n according to their order in the even extensions σ and σ' of π and π'. In the following lemma, we show that if q is the k-th state in σ, q' is the k'-th state in σ', and $k' \leq k$, then q' is reachable from q in this run fragment. Furthermore, if $k' < k$ then q' is reachable from q along a run that visits α.

Lemma 5. *Consider an infinite word $\sigma_0 \sigma_1 \cdots$ and a run DAG G of $\mathcal{A}_n$ on it. Let l be a level of G, let $l' > 0$ be an integer, and let $v = \sigma_l \cdots \sigma_{l+l'}$ be the subword of length l' read at the level l. Let k and k' be such that k is odd and $1 \leq k' \leq k \leq n$. Let q be the k-th state in the even extension of $first(v)$, and let q' be the k'-th state in the even extension of $last(v)$. Then, the vertex $\langle q', l + l' \rangle$ is reachable from the vertex $\langle q, l \rangle$ of G. Moreover, if $l' > 1$ and $k' < k$, then $\langle q', l + l' \rangle$ is reachable from $\langle q, l \rangle$ along a path that visits α.*

Proof. We start with the first part of the lemma and prove it by induction on l' (that is, the length of v). For $l' = 1$, the lemma follows from the definition of the transition function. For the induction step, consider a word $v = wa$. Let $\mathit{first}(w) = \pi_1$, $\mathit{last}(w) = \pi_2$ and $a = \langle \pi_3, \pi_4 \rangle$. Let $i_1 \cdots i_n$, $j_1 \cdots j_n$, $c_1 \cdots c_n$, and $d_1 \cdots d_n$ be the even extensions of π_1, π_2, π_3, and π_4, respectively.

Consider the run DAG G of $\mathcal{A}_n$ on the input word. By the induction hypotheses, which holds for w, we know that for every odd k and for all $k' \leq k$, the vertex $\langle j_{k'}, l + |w| \rangle$ is reachable from the vertex $\langle i_k, l \rangle$. We consider now the edges of G reading the last letter a. We distinguish between two cases. If $\pi_2 = \pi_3$, the lemma follows from the definition of the transition function. If $\pi_2 \neq \pi_3$, consider the state c_k appearing in the k-th position in even extension of π_3. Let m be such that $j_m = c_k$. We again distinguish between two cases. If $m \leq k$, the lemma follows from the definition of the transition function. If $m > k$, then there exist $m' \leq k$ and $m'' > k$ such that $c_{m''} = j_{m'}$. By the induction hypothesis, $\langle j_{m'}, l + |w| \rangle$ is reachable from $\langle i_k, l \rangle$. As $j_{m'} = c_{m''}$, the transition of $c_{m''}$ reading $\langle \pi_3, \pi_4 \rangle$ implies that for every $k' < m''$ (and in particular for every $k' < k$) the vertex $\langle d_{k'}, l + |w| + 1 \rangle$ is reachable from $\langle i_k, l \rangle$.

We now prove the second part of the lemma. By the first part, the vertex $\langle j_{k-1}, l + l' - 1 \rangle$ is reachable from $\langle i_k, l \rangle$. As k is odd, $k - 1$ is even, thus, by the definition of an even-extension, $c_{k-1} = k - 1$, thus $\langle c_{k-1}, l + l' - 1 \rangle$ is an α-vertex. By the definition of the transition function, for every $k' < k - 1$, there is an edge from $\langle c_{k-1}, l + l' - 1 \rangle$ to $\langle d_{k'}, l + l' \rangle$. It follows that there is a path that visits α from $\langle i_k, l \rangle$ to $\langle d_{k'}, l + l' \rangle$. □

We use this result to show that bad words cannot be accepted by $\mathcal{A}_n$. Indeed, whenever there is a mismatch between the permutations, we find a state that reduces its position in the permutations. This state, gives rise to a fragment that visits α. If this happens infinitely often, we get a run that visits α infinitely often.

Lemma 6. *Every bad word u is rejected by $\mathcal{A}_n$.*

Proof. We start with the case that $u = vw^\omega$. Assume that $|w| > 1$. Otherwise, we replace w by $w \cdot w$. By the definition of bad words, the word w^ω contains two successive letters $\langle \pi_1, \pi_1' \rangle$ and $\langle \pi_2, \pi_2' \rangle$ such that $\pi_1' \neq \pi_2$. Let l be a level in the run DAG of $\mathcal{A}_n$ on vw^ω such that $l > |v|$ is such that $\langle \pi_1, \pi_1' \rangle$ is being read in level $l - 1$ and $\langle \pi_2, \pi_2' \rangle$ is being read in level l. Note that $\langle \pi_1, \pi_1' \rangle$ is then being read again at level $l + |w| - 1$.

We show that there exists a vertex $\langle q, l + |w| \rangle$ reachable from $\langle q, l \rangle$ such that the path from $\langle q, l \rangle$ to $\langle q, l + |w| \rangle$ visits an α-vertex. Since $\mathcal{A}_n$ is universal, the block of $|w|$ levels of G that starts in level l repeats forever, thus it follows that G has a path with infinitely many α-vertices.

Let w' be the word read between levels l and $l + |w|$. Note that w' is w shifted so that $\mathit{first}(w') = \pi_2$, and $\mathit{last}(w') = \pi_1'$. Let $\sigma = i_1, \ldots, i_n$ and $\sigma' = j_1, \ldots, j_n$ be the even-extensions of π_2 and π_1', respectively. Since $\pi_2 \neq \pi_1'$, there exists some odd k and k' such that $i_k = j_{k'}$ and $k' < k$. Let q be the state $i_k = j_{k'}$. The state q satisfies the conditions of Lemma 5 with respect to level l and length $l' = |w|$: it is the k-th state in $\mathit{first}(w')$ for an odd k, and it is also the k'-th state in $\mathit{last}(w')$. Hence, since $|w'| > 1$ and $k' < k$, we have that $\langle q, l + |w| \rangle$ is reachable from $\langle q, l \rangle$ along a path that visits α.

Consider some bad word $u \in \Sigma^\omega$ such that u does not have a cycle. It follows that u can be partitioned to infinitely many finite subwords that are non-gluable. Consider two

such subwords w_1 and w_2. As w_1 and w_2 are non-gluable there exists some k and k' such that $k' < k$ and the k-th state in $last(w_1)$ is the k'-th state in $first(w)$. There are infinitely many subwords, we use Ramsey's Theorem to find infinitely many points that have the same k and k'. This defines a new partition to finite subwords. By using Lemma 5 we can show that the run on w contains a path with infinitely many visits to α. □

Corollary 1. *The language of $\mathcal{A}_n$ is $\{vu^\omega \mid v \in \Sigma_n^*\}$.*

In Figure 2, we describe a rejecting run of $\mathcal{A}_n$ on v^ω where v is obtained from u by switching the order of the letters $\langle 315, 351\rangle$ and $\langle 351, 513\rangle$. The pair $\langle 153, 315\rangle$ and $\langle 351, 513\rangle$ is non-gluable. In the run DAG G, the state 1 satisfies the conditions of Lemma 5 with $l = 2$ and $l' = 6$. To see this, note that the subword of v^ω of length 6 that is read at level 2 is $w = \langle 351, 513\rangle\langle 315, 351\rangle\ \langle 513, 531\rangle\ \langle 531, 135\rangle\ \langle 135, 153\rangle\langle 153, 315\rangle$, with $first(w) = 351$ and $last(w) = 315$. The state 1 is the 5-th state in the even extension 32541 of $first(w)$, thus $k = 5$, and is the 3-rd state in the even extension 32145 of $last(w)$, thus $k' = 3$. As promised in the lemma, the vertex $\langle 1, 8\rangle$ is reachable from the vertex $\langle 1, 2\rangle$ via a path that visits the α-vertex $\langle 2, 3\rangle$ — the rejecting path that is highlighted in bold in the figure.

We can now conclude with the statement of the lower bound for the linear case.

Theorem 2. *There is a $\frac{n+1}{2}!$ lower bound on the length of a witness accepted by a UCW with n states.*

Proof. Consider the sequence of UCWs $\mathcal{A}_1, \mathcal{A}_3, \ldots$ defined above. By the above, the language of $\mathcal{A}_n$ is $\{vu^\omega \mid v \in \Sigma_n^*\}$, where u is the word in $(\Sigma_n)^{\frac{n+1}{2}!}$ that contains all the letters in Σ_n ordered lexicographically. Thus, the length of witnesses is at least $\frac{n+1}{2}!$. □

4 Universal co-Büchi Tree Automata

Given an alphabet Σ and a set D of directions, a *Σ-labeled D-tree* is a pair $\langle T, \tau\rangle$, where $T \subseteq D^*$ is a tree over D and $\tau : T \to \Sigma$ maps each node of T to a letter in Σ. A *transducer* is a labeled finite graph with a designated start node, where the edges are labeled by D and the nodes are labeled by Σ. A Σ-labeled D-tree is *regular* if it is the unwinding of some transducer. More formally, a transducer is a tuple $\mathcal{T} = \langle D, \Sigma, S, s_{in}, \eta, L\rangle$, where D is a finite set of directions, Σ is a finite alphabet, S is a finite set of states, $s_{in} \in S$ is an initial state, $\eta : S \times D \to S$ is a deterministic transition function, and $L : S \to \Sigma$ is a labeling function. We define $\eta : D^* \to S$ in the standard way: $\eta(\varepsilon) = s_{in}$, and for $x \in D^*$ and $d \in D$, we have $\eta(x \cdot d) = \eta(\eta(x), d)$. Intuitively, a Σ-labeled D-tree $\langle D^*, \tau\rangle$ is regular if there exists a transducer $\mathcal{T} = \langle D, \Sigma, S, s_{in}, \eta, L\rangle$ such that for every $x \in D^*$, we have $\tau(x) = L(\eta(x))$. We denote by $\mathcal{T}_s$ the transducer $\langle D, \Sigma, S, s, \eta, L\rangle$, i.e., the transducer $\mathcal{T}$ with s as initial state. Given a transducer $\mathcal{T}$, let $reach_0(\mathcal{T}) = \{s_{in}\}$ and let $reach_{i+1}(\mathcal{T}) = \bigcup_{s \in reach_i(\mathcal{T})} \bigcup_{d \in D} \{\eta(s, d)\}$. The *width* of $\mathcal{T}$ is the minimal j such that $|reach_i(\mathcal{T})| = j$ for infinitely many i. That is, starting from some i_0, we have that $|reach_i(\mathcal{T})| \geq j$ for all $i \geq i_0$. Note that while the width of an infinite tree generated by a transducer is unbounded, the width of a transducer is always bounded by its number of states.

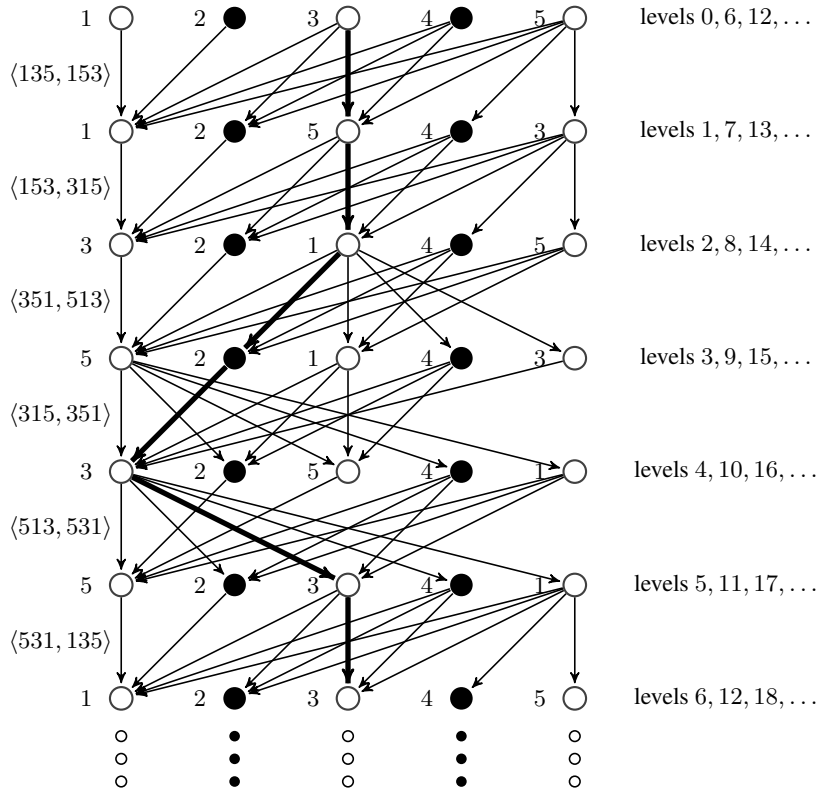

Fig. 2. The rejecting run of $\mathcal{A}_5$ on $(\langle 135, 153\rangle\langle 153, 315\rangle\langle 351, 513\rangle\langle 315, 351\rangle\langle 513, 531\rangle\langle 531, 135\rangle)^\omega$.

A universal co-Büchi tree automaton (UCT, for short) is a tuple $\mathcal{A} = \langle \Sigma, D, Q, Q_{in}, \delta, \alpha\rangle$, where Σ, Q, Q_{in}, and α are as in UCW, D is a set of directions, and $\delta : Q \times \Sigma \to 2^{(D\times Q)}$ is a transition function. When the language of $\mathcal{A}$ is not empty, it accepts a regular Σ-labeled D-tree [16, 13]. It is convenient to consider runs of $\mathcal{A}$ on transducers.

Consider a transducer $\mathcal{T} = \langle D, \Sigma, S, s_{in}, \eta, L\rangle$. A run of $\mathcal{A}$ on $\mathcal{T}$ can be arranged in an infinite DAG $G = \langle V, E\rangle$, where

- $V \subseteq S \times Q \times \mathbb{N}$.
- $E \subseteq \bigcup_{l\geq 0}(S \times Q \times \{l\}) \times (S \times Q \times \{l + 1\})$ is such that $E(\langle s, q, l\rangle, \langle s', q', l + 1\rangle)$ iff there is $d \in D$ such that $(d, q') \in \delta(q, L(s))$ and $\eta(s, d) = s'$.

The run DAG G is accepting iff every path in it has only finitely many vertices in $S \times \alpha \times \mathbb{N}$. A transducer is accepted by $\mathcal{A}$ if its run DAG is accepting. In the sequel we restrict attention to binary trees, i.e., $D = \{0, 1\}$ and $T = \{0, 1\}^*$. All our ideas apply to larger branching degrees as well.

5 Lower Bound on Width of Accepted Transducers

In [8], it is shown that if a UCT with n states is not empty, then it accepts a transducer of width bounded by $(2n!)n^{2n}3^n(n+1)/n!$. An improved upper bound for determinization shows that the width reduces to $2n(n!)^2$ [14, 10]. It is conjectured in [8] that this bound can be tightened to $2^{O(n)}$. Intuitively, it is conjectured there that if a UCT is not empty, then different states of a transducer it accepts that are visited by the same set of states of the UCT can be merged.

In this section we construct, for every odd $n \geq 1$, a UCT $\mathcal{B}_n$ with n states such that the language of $\mathcal{B}_n$ is not empty and yet the width of a transducer accepted by $\mathcal{B}_n$ is at least $\frac{n+1}{2}!$.

We extend the ideas in Section 3 to a tree automaton. The basic idea is to create a mismatch between the permutation the automaton has to send to the left successor of a node and the permutation the automaton has to send to the right successor. Doing so, we force the input tree to display all possible permutations in one level. Thus, the minimal width of a transducer generating such a tree is $\frac{n+1}{2}!$.

Recall the alphabet Σ_n defined in Section 3. We reuse this alphabet in the context of a tree. Whenever we refer to a letter $\langle \pi, \pi' \rangle \in \Sigma_n$ we assume that π' is the successor of π according to the lexicographic order. Consider a letter $\langle \pi, \pi' \rangle \in \Sigma_n$ and a node x labeled by $\langle \pi, \pi' \rangle$. Intuitively, when the automaton $\mathcal{B}_n$ reads the node x, it "sends" the permutation π' to the left successor of x and it "sends" the permutation π (i.e., the same permutation) to the right successor of x. Consider a Σ_n-labeled binary tree $\langle T, \tau \rangle$. Consider a node x and its two successors $x \cdot 0$ and $x \cdot 1$. Let $\tau(x)$ be $\langle \pi_x, \pi'_x \rangle$, $\tau(x \cdot 0)$ be $\langle \pi_{x0}, \pi'_{x0} \rangle$, and $\tau(x \cdot 1)$ be $\langle \pi_{x1}, \pi'_{x1} \rangle$. We say that the node x is *good* if $\pi_{x0} = \pi'_x$ and $\pi_{x1} = \pi_x$. That is, the left successor of x is labeled by the successor permutation π'_x (paired with its successor permutation) and the right successor of x is labeled by the same permutation π_x (paired with its successor permutation). A tree $\langle T, \tau \rangle$ is *good* if all vertices $x \in T$ are good. Given a permutation π there is a unique good tree whose root is labeled by $\langle \pi, \pi' \rangle$. We denote this tree by $\langle T, \tau_\pi \rangle$.

Lemma 7. *For every permutation π, the width of a transducer that generates $\langle T, \tau_\pi \rangle$ is $\frac{n+1}{2}!$.*

Proof. We can construct a transducer generating $\langle T, \tau_\pi \rangle$ with $\frac{n+1}{2}!$ states. Indeed, the states of such a transducer are the letters of Σ_n. The 0-successor of a state $\langle \pi, \pi' \rangle$ is the unique state $\langle \pi', \pi'' \rangle$, for the successor π'' of π', and its 1-successor is $\langle \pi, \pi' \rangle$.

Let $\pi_0, \ldots, \pi_{\frac{n+1}{2}!}$ be an enumeration of all permutations according to the lexicographic order. For simplicity we assume that $\pi = \pi_0$. We can see that $\langle \pi_0, \pi_1 \rangle$ appears in every level in $\langle T, \tau_\pi \rangle$. By induction, $\langle \pi_i, \pi_{i+1} \rangle$ appears for the first time in $\langle T, \tau_\pi \rangle$ in level $i-1$. It follows that $\langle \pi_i, \pi_{i+1} \rangle$ appears in $\langle T, \tau_\pi \rangle$ in all levels above $i-1$. In particular, in all levels after $\frac{n+1}{2}!$, all permutations appear. It follows that $|reach_j(\mathcal{T})| \geq \frac{n+1}{2}!$ for all transducers $\mathcal{T}$ that generate $\langle T, \tau_\pi \rangle$ and $j \geq \frac{n+1}{2}!$. □

Corollary 2. *Every transducer $\mathcal{T}$ that generates a tree that has a subtree $\langle T, \tau_\pi \rangle$, for some permutation π, has width at least $\frac{n+1}{2}!$.*

We now define $\mathcal{B}_n$ as a UCT variant of the UCW $\mathcal{A}_n$ constructed in Section 3. Essentially, every transducer accepted by $\mathcal{B}_n$ generates a tree that contains $\langle T, \tau_\pi \rangle$ as a subtree, for some permutation π of all the letters in Σ_n.

Let $\mathcal{B}_n = \langle \Sigma_n, \{0,1\}, Q_n, \delta_n, Q_n, \alpha_n \rangle$, where $Q_n = \{1, \ldots, n\}$, $\alpha_n = \{i \mid i \text{ is even}\}$, and $\delta : Q_n \times \Sigma_n \to 2^{\{0,1\} \times Q_n}$ is as follows. Let $\langle \pi, \pi' \rangle \in \Sigma_n$ and let $\sigma = i_1 \cdots i_n$ and $\sigma' = j_1 \cdots j_n$ be the even extensions of π and π'. Then, for every $1 \leq k \leq n$, we define

$$\delta_n(i_k, \langle \pi, \pi' \rangle) = \begin{cases} \{(0, j_1), \ldots, (0, j_k), (1, i_1), \ldots, (1, i_k)\} & \text{if } k \text{ is odd} \\ \{(0, j_1), \ldots, (0, j_{k-1}), (1, i_1), \ldots, (1, i_{k-1})\} & \text{if } k \text{ is even} \end{cases}$$

When going left, $\mathcal{B}_n$ treats the pair $\langle \pi, \pi' \rangle$ like the UCW $\mathcal{A}_n$ treats it. When going right, $\mathcal{B}_n$ mimics the same concept, this time, without changing the permutation. From state $q \in Q_n$, our automaton checks the location of q in σ and sends copies to all states in smaller (or equal, if k is odd) locations in σ' in direction 0 and all states in smaller (or equal) locations in σ in direction 1.

Consider a transducer $\mathcal{T} = \langle D, \Sigma_n, S, s_{in}, \eta, L \rangle$ accepted by $\mathcal{B}_n$. Given a permutation π, we say that π' is the 0-successor of π for the successor π' of π according to the lexicographic order (i.e., the unique π' such that $\langle \pi, \pi' \rangle \in \Sigma_n$) and we say that π is the 1-successor of π. Consider a path $p = s_0, a_0, s_1, a_1, \ldots \in (S \times D)^\omega$, where $s_{i+1} = \eta(s_i, a_i)$. We say that p is *good* if for all $i \geq 0$ we have $L(s_{i+1})$ is the a_i-successor of $L(s_i)$. We say that p is *bad* otherwise[4]. If p is bad, every location $i \in \mathbb{N}$ such that $L(s_i)$ is not the a_{i-1}-successor of $L(s_{i-1})$ is called an *error* in p.

Consider a transducer $\mathcal{T} = \langle D, \Sigma_n, S, s_{in}, \eta, L \rangle$ and an infinite path $p = s_0, a_0, s_1, a_1, \ldots \in (S \times D)^\omega$, where $s_{i+1} = \eta(s_i, a_i)$. Consider a sub-path $v = s_l, a_l, \ldots, s_{l'-1}, a_{l'-1}, s_{l'}$. We denote by $\mathit{first}(v)$ the permutation $\pi \in \Pi_n$ such that $\langle \pi, \pi' \rangle = L(s_l)$. We denote by $\mathit{last}(v)$ the permutation $\pi'' \in \Pi_n$ such that $L(s_{l'-1}) = \langle \pi, \pi' \rangle$ and $\pi'' = \pi$ if $a_{l'-1} = 1$ and $\pi'' = \pi'$ if $a_{l'-1} = 0$. That is, the last permutation read in v is determined by the last direction p takes in v.

Let G be the DAG run of $\mathcal{B}_n$ on $\mathcal{T}$, $p = s_0, a_0, s_1, a_1, \ldots$ an infinite path in T, and $v = s_l, a_l, \ldots, s_{l'-1}, a_{l'-1}, s_{l'}$ a sub-path of p. Consider the part of G consisting of all nodes in levels l to l' that read the states $s_l, \ldots, s_{l'}$. Let π be $\mathit{first}(v)$ and π' be $\mathit{last}(v)$. We arrange the states in Q according to their order in the even extensions σ and σ' of π and π'. The following lemma is the tree variant of Lemma 5. It shows that if q is the k-th state in σ and q' is the k'-th state in σ', then $k' \leq k$ implies that q' is reachable from q in this part of the run. Furthermore, if $k' < k$ then q' is reachable from q along a run that visits α. The proof is identical to that of Lemma 5.

Lemma 8. *Consider a transducer $\mathcal{T} = \langle D, \Sigma_n, S, s_{in}, \eta, L \rangle$ and the* DAG *run G of $\mathcal{B}_n$ on it. Let $p = s_0, a_0, s_1, a_1, \ldots$ be a path in T and let $v = s_l, a_l \ldots, s_{l'-1}, a_{l'-1}, s_{l'}$ be a sub-path of p. Let q be the k-th state in the even extension of $\mathit{first}(v)$ for an odd k, and let q' be the k'-th state in the even extension of $\mathit{last}(v)$, for $k' \leq k$. Then, the vertex $\langle s_{l'}, q', l' \rangle$ in G is reachable from the vertex $\langle s_l, q, l \rangle$. Moreover, if $l' - l > 1$ and $k' < k$, then a path connecting $\langle s_l, q, l \rangle$ to $\langle s_{l'}, q', l' \rangle$ visits α.*

The following Lemma resembles Lemma 6. It shows that in a transducer accepted by $\mathcal{B}_n$, every path has only finitely many errors.

Lemma 9. *For every path p in a transducer $\mathcal{T} \in L(\mathcal{B}_n)$, the path p contains finitely many errors.*

[4] Notice that the definition of bad here is slightly different from the definition of bad in Section 3.

Proof. Let G be an accepting run of $\mathcal{B}_n$ on $\mathcal{T} = \langle D, \Sigma_n, S, s_{in}, \eta, L\rangle$. Assume that $p = s_0, a_0, s_1, a_1, \ldots$, where $s_{i+1} = \eta(s_i, a_i)$, is a path in $\mathcal{T}$ with infinitely many errors. Let $s_{l_0}, s_{l_1}, \ldots$ denote the error locations in p. By definition, for every $m \geq 0$ we have $L(s_{l_m})$ is not the a_{l_m-1}-successor of $L(s_{l_m-1})$. With every index l_m we associate a triplet $\langle \pi_m, \pi'_m, d_m\rangle$ such that $L(s_{l_m-1}) = \langle \pi, \pi'\rangle$ and π_m is the a_{l_m-1}-successor of π (i.e., π' if $a_{l_m-1} = 0$ and π otherwise), $L(s_{l_m}) = \langle \pi'_m, \pi'''\rangle$, and $d_m = a_{l_m-1}$. That is, we record the permutation π'_m labeling s_{l_m}, the unmatching π_m, which is the a_{l_m-1}-successor of the label of s_{l_m-1}, and the direction that takes from s_{l_m-1} to s_{l_m}. There are infinitely many errors and finitely many triplets. There is a triplet $\langle \pi, \pi', d\rangle$ associated with infinitely many indices. We abuse notations and denote by $s_{l_0}, s_{l_1}, \ldots$ the sub-sequence of locations associated with $\langle \pi, \pi', d\rangle$. Without loss of generality, assume that for all $m \geq 0$ we have $l_{m+1} - l_m > 1$.

For $m, m' \geq 0$ such that $m \neq m'$, let $v_{m,m'}$ denote the sub-path of p that starts in s_{l_m} and ends in $s_{l_{m'}}$. Then $\pi' = \mathit{first}(v_{m,m'})$ and $\pi = \mathit{last}(v_{m,m'})$. By assumption π' is not the d-successor of π. Let $\sigma = i_1, \ldots, i_n$ be the even extension of the d-successor of π and let $\sigma' = j_1, \ldots, j_n$ be the even extension of π'. Then there exists some odd k and k' such that $j_k = i_{k'}$ and $k' < k$. Let q be the state $j_k = i_{k'}$. The state q satisfies the condition of Lemma 8 with respect to $v_{m,m'}$: it is the k-th state in $\mathit{first}(v_{m,m'})$ for an odd k, and it is also the k'-th state in $\mathit{last}(v_{m,m'})$. Hence, since $l_{m'} - l_m > 1$ and $k' < k$, the node $\langle s_{l_{m'}}, q, l_{m'}\rangle$ in G is reachable from the node $\langle s_{l_m}, q, l_m\rangle$ along a path that visits α.

For every two different integers m and m' we identify one such state $q_{m,m'}$. By Ramsey's Theorem, there exist a state q and a sequence $l'_0, l'_1, \ldots$ such that for every $m \geq 0$ the sub-path $v_{l'_m, l'_{m+1}}$ connects state q to itself with a path that visits α. We have found a path in G that visits α infinitely often. □

We now show that every tree generated by $\mathcal{T}$ contains $\langle T, \tau_\pi\rangle$ for some π as a subtree.

Lemma 10. *For every $\mathcal{T} \in L(\mathcal{B}_n)$, there exists a permutation π and a state s reachable from s_{in} such that the transducer $\mathcal{T}_s$ generates $\langle T, \tau_\pi\rangle$.*

Proof. We add an annotation to the edges in $\mathcal{T}$. Every edge $s' = \eta(s, a)$ such that s' is an error in a path that contains s and s' is annotated by 1. Every other edge is annotated by 0. According to Lemma 9, every path in $\mathcal{T}$ is annotated by finitely many 1's.

We say that a state s is 1-free in $\mathcal{T}$ iff all the edges in $\mathcal{T}$ that are reachable from s are not labeled by 1. It is enough to find one such state s. Assume by contradiction that no such state s exists. We construct by induction a path that is labeled by infinitely many 1's.[5]

By assumption, s_{in} is not 1-free. Hence there is some state s_1 reachable from s_{in} and a direction a_1 such that the edge from s_1 to $\eta(s_1, a_1)$ is annotated by 1. By induction the path from s_{in} to $\eta(s_i, a_i)$ has at least i edges annotated by 1. By assumption $\eta(s_i, a_i)$ is not 1-free. There exists a node s_{i+1} reachable from $\eta(s_i, a_i)$ and a direction a_{i+1} such that the edge from s_{i+1} to $\eta(s_{i+1}, a_{i+1})$ is annotated by 1. It follows that the path from s_{in} to $\eta(s_{i+1}, a_{i+1})$ has at least $i + 1$ edges annotated by 1. In the limit, we get a path in $\mathcal{T}$ that has infinitely many edges labeled 1. In contradiction to Lemma 9.

[5] Notice the resemblance to the definition of α-free in Section 2. Indeed, the proof of the existence of a 1-free state follows closely the similar proof in [7].

It follows that there exists a state s in $\mathcal{T}$ such that s is 1-safe. As s is 1-safe, the subtree generated by $\mathcal{T}_s$ contains no errors. Let π be the permutation such that $L(s) = \langle \pi, \pi' \rangle$. Then $\mathcal{T}_s$ generates $\langle T, \tau_\pi \rangle$. □

We can now conclude with the statement of the lower bound for the branching case.

Theorem 3. *There is a $\frac{n+1}{2}!$ lower bound on the width of a transducer accepted by a UCT with n states.*

Proof. Consider the sequence of UCTs $\mathcal{B}_1, \mathcal{B}_3, \ldots$ defined above. For every permutation π, the transducer that generates $\langle T, \tau_\pi \rangle$ is accepted by $\mathcal{B}_n$. By Lemma 10 and Corollary 2, every transducer accepted by $\mathcal{B}_n$ is of width at least $\frac{n+1}{2}!$. □

References

1. J.A. Brzozowski and E. Leiss. Finite automata and sequential networks. *Theoretical Computer Science*, 10:19–35, 1980.
2. J.R. Büchi. On a decision method in restricted second order arithmetic. In *Proc. Int. Congress on Logic, Method, and Philosophy of Science. 1960*, pages 1–12. Stanford University Press, 1962.
3. A.K. Chandra, D.C. Kozen, and L.J. Stockmeyer. Alternation. *Journal of the Association for Computing Machinery*, 28(1):114–133, 1981.
4. J.E. Hopcroft, R. Motwani, and J.D. Ullman. *Introduction to Automata Theory, Languages, and Computation (2nd Edition)*. Addison-Wesley, 2000.
5. O. Kupferman, N. Piterman, and M.Y. Vardi. Safraless compositional synthesis. In *Proc 18th Int. Conf. on Computer Aided Verification*, volume 4144 of *Lecture Notes in Computer Science*, pages 31–44. Springer, 2006.
6. O. Kupferman and S. Sheinvald-Faragy. Finding shortest witnesses to the nonemptiness of automata on infinite words. In *17th Int. Conf. on Concurrency Theory*, volume 4137 of *Lecture Notes in Computer Science*, pages 492–508. Springer, 2006.
7. O. Kupferman and M.Y. Vardi. Weak alternating automata are not that weak. *ACM Transactions on Computational Logic*, 2(2):408–429, 2001.
8. O. Kupferman and M.Y. Vardi. Safraless decision procedures. In *Proc. 46th IEEE Symp. on Foundations of Computer Science*, pages 531–540, 2005.
9. R.P. Kurshan. *Computer Aided Verification of Coordinating Processes*. Princeton Univ. Press, 1994.
10. W. Liu. A tighter analysis of piterman determinization construction. http://nlp.nudt.edu.cn/ lww/pubs.htm, 2007.
11. R. McNaughton. Testing and generating infinite sequences by a finite automaton. *Information and Control*, 9:521–530, 1966.
12. S. Miyano and T. Hayashi. Alternating finite automata on ω-words. *Theoretical Computer Science*, 32:321–330, 1984.
13. D.E. Muller and P.E. Schupp. Simulating alternating tree automata by nondeterministic automata: New results and new proofs of theorems of Rabin, McNaughton and Safra. *Theoretical Computer Science*, 141:69–107, 1995.
14. N. Piterman. From nondeterministic Büchi and Streett automata to deterministic parity automata. *Logical Methods in Computer Science*, 3(3):5, 2007.
15. A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *Proc. 16th ACM Symp. on Principles of Programming Languages*, pages 179–190, 1989.

16. M.O. Rabin. Decidability of second order theories and automata on infinite trees. *Transaction of the AMS*, 141:1–35, 1969.
17. W. Thomas. Automata on infinite objects. *Handbook of Theoretical Computer Science*, pages 133–191, 1990.
18. M.Y. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification. In *Proc. 1st IEEE Symp. on Logic in Computer Science*, pages 332–344, 1986.
19. M.Y. Vardi and P. Wolper. Reasoning about infinite computations. *Information and Computation*, 115(1):1–37, 1994.