Alternating Finite Automata with Limited Universal Branching

Chris Keeler(✉) and Kai Salomaa

School of Computing, Queen's University, Kingston, ON K7L 2N8, Canada
{keeler,ksalomaa}@cs.queensu.ca

Abstract. We consider measures that limit universal parallelism in computations of an alternating finite automaton (AFA). *Maximum pared tree width* counts the largest number of universal branches in any computation and *acceptance width* counts the number of universal branches in the best accepting computation, i.e., in the accepting computation with least universal parallelism. We give algorithms to decide whether the maximum pared tree width or the acceptance width of an AFA are bounded by an integer k. For a constant k the algorithm for maximum pared tree width operates in polynomial time. An AFA with m states and acceptance width k can be converted to an NFA with $(m+1)^k$ states. We consider corresponding lower bounds for the transformation. The *tree width* of an AFA counts the number of all (existential and universal) branches of the computation. We give upper and lower bounds for converting an AFA of bounded tree width to a DFA.

1 Introduction

Deterministic and nondeterministic finite automata (DFA and NFA) are well understood models for which a significant number of results are known. As a generalization of nondeterminism, *alternation* was introduced in [1], and has since been studied extensively for Turing machines [5,6,23], and pushdown automata [1,20].

The power of alternation in finite automata (AFAs) was first studied by Chandra, Kozen, and Stockmeyer [1], later by King [15] and Hromkovič [10], and state complexity trade-offs with NFAs and DFAs were given by Fellah et al. [3]. However, results on alternating finite automata remain relatively sparse compared to alternating pushdown automata and alternating (infinite) automata, and little effort has been made towards examining restricted computations within the context of alternation.

Restricted amounts of nondeterminism have been measured in various ways, including but not limited to *ambiguity* [19], *tree width* [11,22], and *string path width* [13]. These so-called "measures of nondeterminism" examine some aspect of an automaton's computations. For example, the number of partial, or accepting computations on a given string. For a particular regular language and model, the *state complexity* is a measure of how complicated it is for that model to capture that language. The state complexity is combined with these measures of

A. Leporati et al. (Eds.): LATA 2020, LNCS 12038, pp. 196–207, 2020.
https://doi.org/10.1007/978-3-030-40608-0_13

restricted nondeterminism, yielding tradeoffs between the amount of nondeterminism and the number of states required.

An automaton is said to *alternate* when it switches from an existential state to a universal state (or vice versa) [1]. There exists an exponential state complexity blow-up between two-way AFAs with at most k alternations and two-way AFAs with at most $k+1$ alternations, and in general this hierarchy is infinite [7]. The emptiness problem for AFAs was shown to be PSPACE-Complete for general alphabets [8,12]. More recently, the state complexity of various operations on AFAs has also been studied [9].

In this paper, we focus on the original model of AFAs (introduced by Chandra, Kozen, and Stockmeyer) where the states are either *existential* or *universal* [1,7,10,12,15,23], rather than the one where states are labeled with boolean functions [18]. However, both of these models recognize exactly the regular languages. We also do not consider states or transitions with negation, though there is only a linear blow-up between our model and the one which can perform negation [3].

This paper is organized as follows. Section 2 recalls several definitions, and fixes our model for alternating finite automata. Section 2.1 introduces the notions of acceptance width and maximal pared tree width, and provides several initial results and bounds for these new metrics. Section 3 gives a polynomial transformation for an NFA to simulate an AFA with bounded parallelism, shows that the decidability of several decision problems for AFAs with finite acceptance width, and gives algorithms to decide whether an AFA's maximal pared tree width or acceptance width is bounded by a given constant. Section 4 presents unary witness languages with finite acceptance width (with respect to the number of states) which require only a small number of states to be recognized by an AFA, but require an exponential number of states to be recognized by an NFA or DFA. Finally, Sect. 4.1 introduces a non-unary witness language, and provides another exponential state complexity blow-up; this time between AFAs with bounded tree width (with respect to the number of states), and NFAs and DFAs.

2 Preliminaries

An AFA is a 6-tuple, $A = (Q_e, Q_u, \Sigma, \delta, q_0, F)$ where Q_e (the existential state set) and Q_u (the universal state set) are finite sets of states such that $Q_e \cap Q_u = \emptyset$, Σ is the input alphabet, $\delta : (Q_e \cup Q_u) \times \Sigma \to 2^{Q_e \cup Q_u}$ is the transition function, $q_0 \in Q_e \cup Q_u$ is the initial state, and $F \subseteq Q_e \cup Q_u$ is the set of final states. We use ε to mean the empty string, and A_q to mean A with a different specified starting state, $q \in Q_e \cup Q_u$. Note that the standard NFA model can be seen as an AFA where Q_e contains all of the states, and Q_u is empty. We must further specify the *language* of an AFA, to account for the differences caused by universal states. We do this by defining them bottom-up with respect to their states.

Definition 1. *Let $A = (Q_e, Q_u, \Sigma, \delta, q_0, F)$ be an AFA, and A_q be a copy of the AFA with $q \in Q_e \cup Q_u$ as the initial state. We point out that $\varepsilon \in L(A_q)$ if $q \in F$.*

Consider $q \in Q_e \cup Q_u, a \in \Sigma$ *where* $\delta(q, a) = \{p_1, \ldots, p_n\}$. *Then for* $x \in \Sigma^*$, *define:*

- *If* $q \in Q_u$, *then* $ax \in L(A_q)$ *if and only if* $x \in L(A_{p_i})$ *for all* $1 \leq i \leq n$.
- *If* $q \in Q_e$, *then* $ax \in L(A_q)$ *if and only if* $x \in L(A_{p_i})$ *for some* $1 \leq i \leq n$.

The language of A *is defined as* $L(A) = L(A_{q_0})$.

The *computation tree* of an AFA A on ε from $q \in Q_e \cup Q_u$, denoted $T_{A,q,\varepsilon}$ is the singleton node (q, ε). The *computation tree* of an AFA A on cv from q, denoted $T_{A,q,cv}$, such that $q \in Q_e \cup Q_u, c \in \Sigma, v \in \Sigma^*$ is defined inductively as the tree:

- whose internal nodes are labeled by a tuple (p, a), for $p \in Q, a \in \Sigma$ (i.e., each internal node is labeled by a state and character)
- which is rooted by a node (q, c)
- where the trees rooted at the children of (q, c) are
  - the computation trees $(T_{A,p_1,v}, \ldots, T_{A,p_n,v})$ if $\delta(q, c) = \{p_1, \ldots, p_n\}$, and
  - the failure node $\perp$ if $\delta(q, c) = \emptyset$ (that is, if $\delta(q, c)$ is undefined).

If a computation tree of an AFA A on a string x starts on the initial state of A, then we omit the state label, denoting it as $T_{A,x}$. We use the notation leaves(T) to mean the (depth-first) ordered tuple of leaves in the computation tree T. The computation tree of an NFA is defined similarly, except its nodes are always labeled by existential states [11].

We define the *paring* of a computation tree, which serves as the transformation around which our new measures are defined. For an AFA $A = (Q_e, Q_u, \Sigma, \delta, q_0, F)$ and a string $x \in \Sigma^*$, a *pared computation tree* of $T_{A,x}$ is defined as a tree where for each node $(q, a) \in T_{A,x}$:

- if $q \in Q_e$ then keep only one child node, and
- if $q \in Q_u$ then keep all child nodes.

Since there is a choice made on each of the existential nodes, the same computation tree can result in many different pared computation trees. A pared tree represents a possible computation of the AFA A. At nodes labeled by existential states, the pared tree follows one (nondeterministically chosen) way to continue the computation. The nodes labeled by universal states have children labeled by all states reachable from that state in the next computation step. Note that every pared tree of an NFA will only have one leaf, since all of its states are existential. We denote the set of all pared computation trees on a tree T as $\Join(T)$. A pared computation tree is *accepting* if all of its leaves are labeled by accepting states (implying that no leaf is the failure node), and a string x is accepted by an AFA if and only if A has an accepting pared computation tree in $\Join(T_{A,x})$.

Without loss of generality, we assume that all of an AFA's universal states are reachable. However, since emptiness for AFAs is PSPACE-Complete, we cannot assume that all of an AFA's states are useful in the sense that they can be used in an accepting computation. Since a universal state with at most one outgoing transition per character is no different than using an existential state, we also

assume that every universal state has multiple outgoing transitions on at least one character.

For a regular language L, $\mathrm{sc}(L)$, (respectively, $\mathrm{nsc}(L), \mathrm{asc}(L)$), is the *state complexity*, (respectively, nondeterministic and alternating state complexity) of L.

2.1 Tree Width of Alternating Machines

The *tree width* [11] of an AFA A on a string x, denoted $\mathrm{tw}(A,x)$, is the number of leaves in the computation tree of A on x. That is, $\mathrm{tw}(A,x) = |\mathrm{leaves}(T_{A,x})|$.

Since the notion of tree width is originally based on the computation tree of an NFA, and our AFA definition extends the original notion of computation trees, it seems natural to look at "alternating tree widths".

Definition 2. *Let $A = (Q_e, Q_u, \Sigma, \delta, q_0, F)$ be an AFA. Then the* acceptance width *of A on a string $x \in \Sigma^*$, denoted* $\mathrm{aw}(A,x)$*, is the minimum number of leaves of any* accepting *pared computation tree of $T_{A,x}$. The* maximum pared tree width *of A on a string $x \in \Sigma^*$, denoted* $\mathrm{mptw}(A,x)$*, is the maximum number of leaves of* any *pared computation tree of $T_{A,x}$. Formally, these are:*

$$\mathrm{aw}(A,x) = \min\{|\mathit{leaves}(T)| \;\mid\; T \in \Join(T_{A,x}),\ \mathit{leaves}(T) \subseteq F\}$$

$$\mathrm{mptw}(A,x) = \max\{|\mathit{leaves}(T)| \;\mid\; T \in \Join(T_{A,x})\}$$

Since the (original) tree width does not perform the paring operation, we get that for any AFA A and string x, $\mathrm{aw}(A,x) \leq \mathrm{mptw}(A,x) \leq \mathrm{tw}(A,x)$. We also get the following condition for equality between the measures, which occurs when the paring operation does not change the computation tree.

Remark 1. Let A be an AFA, and x a string. Then $\mathrm{mptw}(A,x) = \mathrm{tw}(A,x)$ if and only if each node in $T_{A,x}$ with more than one child is labeled by some universal state in A.

We extend the acceptance width and maximum pared tree width functions as functions on integers in the normal manner:

$$\mathrm{aw}(A,\ell) = \max\{\mathrm{aw}(A,x) \mid x \in \Sigma^\ell\},$$

$$\mathrm{mptw}(A,\ell) = \max\{\mathrm{mptw}(A,x) \mid x \in \Sigma^\ell\}.$$

$$\mathrm{aw}(A) = \sup_{\ell\in\mathbb{N}}\{\mathrm{aw}(A,\ell)\},\ \text{and}\ \mathrm{mptw}(A) = \sup_{\ell\in\mathbb{N}}\{\mathrm{mptw}(A,\ell)\}.$$

If, for a string x, there are no accepting computation trees, then $\mathrm{aw}(A,x) = 0$. Since the emptiness problem is PSPACE-complete for AFAs [8], and these results hold even for unary languages, then we get the following equivalence.

Remark 2 ([8]). Let A be an AFA. Then it is PSPACE-complete to decide whether or not $\mathrm{aw}(A) = 0$.

If an m-state AFA has finite tree width, then its tree width is at most 2^{m-2} [22]. Since, on any string, the acceptance width and maximal pared tree width of an AFA are upper-bounded by the tree width, we get the following conditional upper bound.

Corollary 1 ([22]). *Let A be an m-state AFA with finite tree width. Then* $\mathrm{aw}(A) \leq \mathrm{mptw}(A) \leq 2^{m-2}$.

Alternatively, we could replace the computation trees by directed acyclic graphs by merging any nodes on the same state on the same level. However, in this case, the acceptance width and maximal pared tree width of an m-state AFA would be at most m.

3 Decision Problems for Pared Tree Width and Acceptance Width

Normally, an NFA may require an exponential state blow-up to simulate an AFA [3]. However, an NFA can simulate any finite acceptance width AFA with at most a polynomial blow-up in the number of states. An m-state AFA A with acceptance width k can be simulated by an NFA where the states are k-tuples of states of A and transitions of the NFA simulate at most k parallel computations of A.

Lemma 1. *Let A be an m-state AFA, such that* $\mathrm{aw}(A) \leq k$*, for some constant k. Then $(m+1)^k$ states are sufficient for an NFA to simulate A.*

It is known that the emptiness problem for NFAs can be solved in linear time, with respect to the number of states, using a breadth first search [4]. The transformation from Lemma 1 then yields a polynomial-time algorithm to decide emptiness for a finite acceptance width AFA.

Corollary 2. *Let A be an m-state AFA with finite acceptance width k, for some constant k. Then in $O(m^k)$ time we can decide whether $L(A) = \emptyset$.*

Using the transformation from Lemma 1, but modifying which states of the NFA are accepting, we can also decide whether the maximal pared tree width of an AFA is bounded.

Theorem 1. *Let A be an m-state AFA and k a constant. Then we can decide whether or not the maximal pared tree width of A is at most k in $O(m^k)$ time.*

Using similar ideas from the characterization of NFAs with finite tree width [22], we are able to characterize AFAs with finite maximal pared tree width.

Corollary 3. *Let $A = (Q_u, Q_e, \Sigma, \delta, q_0, F)$ be an AFA. Then* $\mathrm{mptw}(A) > 2^{m-2}$ *if and only if there exists some state $q \in Q_u$ and character $c \in \Sigma$ such that $|\delta(q, c)| \geq 2$ and q is involved in a cycle.*

Modifying existing algorithms for deciding finiteness of an NFA's tree width [14], we are also able to decide finiteness of an AFA's maximal pared tree width in polynomial time.

Corollary 4 ([14]). *Let $A = (Q_u, Q_e, \Sigma, \delta, q_0, F)$ be an m-state AFA. Then we can decide whether or not the maximal pared tree width of A is bounded by some constant k in $O(m^3 \cdot |\Sigma|)$ time*[1].

The general membership problem is P-complete for AFAs [12], and this holds even for finite unary languages. In fact, this P-completeness is even stronger, as it holds for all cycle-free AFAs.

Since an m-state cycle-free AFA has at most $m-1$ states being evaluated in parallel, then the membership problem for AFAs with bounded parallel computations is also P-complete.

Corollary 5 ([12]). *Let A be a finite maximal pared tree width AFA. Then for a string x, it is P-complete to decide whether $x \in L(A)$.*

We can also decide whether the pared acceptance width of an AFA is finitely bounded by some number.

Theorem 2. *Let A be an AFA, and $k \in \mathbb{N}$. Then it is decidable whether the acceptance width of A is bounded by k.*

While it is decidable whether the acceptance width of an AFA is bounded by an integer k, the algorithm presented in Theorem 2 is not an efficient one and we cannot expect to have an efficient algorithm for this problem[2]. For a given AFA A and $k \in \mathbb{N}$ we can construct an AFA A' that begins the computation by a universal step with $k+1$ choices, where the first computation simulates A and the remaining k computations always accept deterministically. Then $\mathrm{aw}(A') \leq k$ if and only if $L(A) = \emptyset$ and deciding the emptiness of an AFA is PSPACE-complete [8].

For any AFA A with finite tree width, the acceptance width of A must also be finite. Under this restriction, we can decide whether the acceptance width of A is finite using the construction from Theorem 2.

Corollary 6. *Let A be an m-state AFA with finite tree width. By Corollary 1, the acceptance width is then at most 2^{m-2}. Since the acceptance width of A is finite if and only if it is at most 2^{m-2}, then it is decidable whether the acceptance width of A is finite. We do this by using Theorem 2 with an input value of 2^{m-2}.*

Since the acceptance width of an AFA is only upper bounded by its tree width, it is possible that an AFA has infinite tree width and finite acceptance width. In this case, we do not have an upper bound for the acceptance width.

[1] The DCFS proceedings has a slightly worse bound of $O(m^4 \cdot |\Sigma|)$, and the specifics of the improved version will appear in a future paper.

[2] This observation, with a justification different from the below one, was suggested by an anonymous referee.

Question 1. *Let A be an m-state AFA with infinite tree width and finite acceptance width k. Is there any expression in m which bounds k?*

As a result, it is not immediately obvious whether the finiteness of an AFA's acceptance width is decidable in general.

Question 2. *For an AFA A such that* $\mathrm{tw}(A) \notin O(1)$*, does there exist an algorithm to decide whether or not* $\mathrm{aw}(A) \in O(1)$*?*

4 State Complexity

Let $\mathcal{I}$ be a set of integers, and $LCM(\mathcal{I})$ be the least common multiple of all elements in $\mathcal{I}$. We define $L_{\forall\mathcal{I}}$ as the set of all unary strings whose lengths are the product of all integers in $\mathcal{I}$.

$$L_{\forall\mathcal{I}} = \{a^y \mid (\forall i \in \mathcal{I})\ y \equiv 0 \pmod{i}\} \tag{1}$$

Equivalently, we have $L_{\forall\mathcal{I}} = \{a^{y \cdot z} \mid z \geq 0, y = LCM(\mathcal{I})\}$.

Lemma 2. *Let $\mathcal{I}$ be a set of integers. Then* $\mathrm{sc}(L_{\forall\mathcal{I}}) = \mathrm{nsc}(L_{\forall\mathcal{I}}) = LCM(\mathcal{I})$.

The state complexity is, of course, maximal with respect to the size of the input set when its elements are pairwise coprime.

Lemma 3. *Let $\mathcal{I} = \{p_1, \ldots, p_n\}$ be a set of n integers. If the elements of $\mathcal{I}$ are pairwise coprime, then there exists an AFA A recognizing $L_{\forall\mathcal{I}}$ with $1 + \sum_{i=1}^{n} p_i$ states and tree width n such that* $\mathrm{sc}(L(A)) = \mathrm{nsc}(L(A)) = \prod_{i=1}^{n} p_i$.

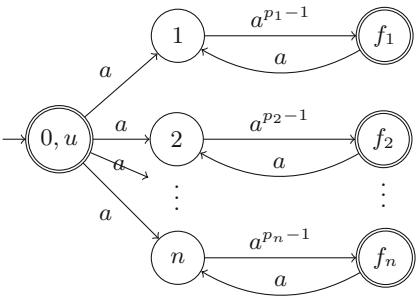

Fig. 1. AFA for $L_{\forall\mathcal{P}}$ where $\mathcal{P} = \{p_1, \ldots, p_n\}$. Universal states are marked with an additional label 'u', and existential states are given as normal.

Proof. Let $\mathcal{I} = \{p_1, \ldots, p_n\}$ be a set of integers whose elements are pairwise coprime. We give the AFA recognizing $L_{\forall\mathcal{I}}$ in Fig. 1, whose tree width and number of states matches the claim. Since $\mathcal{I}$'s elements are pairwise coprime, $LCM(\mathcal{I}) = \prod_{i=1}^{n} p_i$. And by Lemma 2, $\mathrm{sc}(L_{\forall\mathcal{I}}) = \mathrm{nsc}(L_{\forall\mathcal{I}}) = LCM(\mathcal{I})$. □

Recognizing that the state complexity blow-up in Lemma 3 is exactly Landau's function [2,21], we get the following exponential state complexity trade-off between AFAs with finite tree width (and therefore also finite acceptance width) and NFAs. A similar idea and result was also given by Kupferman et al. [17], though it was formulated to capture the unary language a^{n+i}, for $i \geq 0$.

Theorem 3 ([2,17,21]). *Let $\mathcal{I}$ be a set of pairwise coprime integers, and A be an $(m-1)$-state AFA recognizing $L_{\forall\mathcal{I}}$ with tree width $|\mathcal{I}|$. Then any NFA equivalent to A will require at least $e^{(1+o(1))\cdot\sqrt{m \ln m}}$ states.*

While Landau's function gives a lower bound for the state complexity blow-up of simulating a restricted tree width AFA with an NFA, it is only given in terms of the number of states.

Lemma 4. *Let $\mathcal{I} = \{p_1, \ldots, p_n\}$ be a set of pairwise coprime integers, for some $n \in \mathbb{N}$. Let A be an m-state AFA such that A has acceptance width n and recognizes $L_{\forall\mathcal{I}}$. Then any NFA equivalent to A will require at least $(\frac{m}{n \cdot p_n})^n$ states.*

In the general case, for every m, there exists an m-state AFA which cannot be simulated by any NFA with fewer than 2^m states [3], and any equivalent DFA needs 2^{2^m} states [1]. However, to get this double-exponential state complexity blow-up, the m-state AFA needs a tree width much larger than m.

Let $\mathcal{P} = \{p_1, \ldots, p_n\}$ be a set of n prime numbers. We define $L_{2\mathcal{P}}$, the set of all unary strings whose lengths are a product of *at least two* distinct primes from $\mathcal{P}$.

$$L_{2\mathcal{P}} = \{a^x \mid (\exists i, j)\ 1 \leq i < j \leq n, \text{ such that } p_i \text{ and } p_j \text{ divide } x\} \tag{2}$$

Lemma 5. *There exists an AFA A recognizing $L_{2\mathcal{P}}$ with $1 + \frac{n(n-1)}{2} + \sum_{i=1}^{n}(p_i - 1)$ states[3] and a maximal pared tree width of 2.*

We extend $L_{2\mathcal{P}}$, defining $L_{k\mathcal{P}}$ as the set of all unary strings whose lengths are a product of *at least* k distinct primes from $\mathcal{P}$, for some constant k.

$$L_{k\mathcal{P}} = \{a^y \mid (\exists r_1, \ldots, r_k)\ \{r_1, \ldots, r_k\} \subseteq \mathcal{P}, \text{ such that } (\forall i)\ 1 \leq i \leq k, y \equiv 0 \pmod{r_i}\} \tag{3}$$

Using similar ideas as the proof from Lemma 5 but operating on an arbitrary number of elements instead of only two, we get the following result.

Lemma 6. *For every $k \geq 2$, there exists an AFA A recognizing $L_{k\mathcal{P}}$ with $1 + \binom{n}{k} + \sum_{i=1}^{n}(p_i - 1)$ states and a maximal pared tree width of k.*

[3] We need one extra state each if 2 or 3 $\in \mathcal{P}$.

4.1 Universal Infix Language

For two strings $v, v' \in \Sigma^*$, we say that v and v' are *disjoint* if they do not share any symbols. We extend this notion to tuples of strings, such that a tuple of strings $\mathcal{W}$ is disjoint if and only if all pairs of strings $x, x' \in \mathcal{W}$ are disjoint.

A bitstring $b_1 \cdots b_n \in \{0,1\}^n$ is a string for representing some boolean value across a set of n elements. We define the cardinality of a bistring as the number of 1s appearing in that bitstring.

The *universal infix language* of an ordered string tuple $\mathcal{W}$ consists of strings that contain each $x \in \mathcal{W}$ as an infix. We define a labeling function $h_{\mathcal{W}} : \Sigma^* \to \{0,1\}^n$ which takes as input a string $s \in \Sigma^*$ and an n-tuple $\mathcal{W}$, and produces the bitstring $b_1 \cdots b_n$, where $b_i = 1$ if and only if the i^{th} element of $\mathcal{W}$ is an infix of s, for $1 \leq i \leq n$. More formally, the universal infix language over a tuple of strings $\mathcal{W}$ and an alphabet Σ is defined as:

$$L_{\alpha\mathcal{W}} = \{s \in \Sigma^* \mid (\forall x \in \mathcal{W})\ x \text{ is a substring of } s\} \tag{4}$$

An AFA with small amounts of alternation can recognize this language with relatively few states, and limited universal branching.

Lemma 7. *Let $\mathcal{W} = (x_1, \ldots, x_n)$ be an ordered, disjoint tuple of strings. Then there exists an AFA recognizing $L_{\alpha\mathcal{W}}$ with $2 + \sum_{i=1}^{n} |x_i|$ states and tree width n.*

Proof. Let $\mathcal{W} = (x_1, \ldots, x_n)$ be an ordered, disjoint tuple of strings, and let $x_i[j]$ be the j^{th} character of the i^{th} string. We give the general structure for an AFA in Fig. 2, which recognizes $L_{\alpha\mathcal{W}}$ with 1 universal and $1 + \sum_{i=1}^{n} |x_i|$ existential states. This AFA has tree width n, and only alternates between universal and existential states once. The only final state is the one at the end of all the branches. And, excepting the initial state, we define the transition function deterministically. If the machine is reading x_i, has read up to $x_i[j]$, and then encounters some mismatched symbol, then the computation path currently in state $i.j$ will return to state i, indicating that the infix must be restarted. □

However, a DFA for $L_{\forall\mathcal{W}}$ needs exponentially more states than an AFA.

Lemma 8. *Let $\mathcal{W} = (x_1, \ldots, x_n)$ be a disjoint tuple of strings. Then*

$$\text{sc}(L_{\alpha\mathcal{W}}) = 2^n + 2^{n-1} \cdot \sum_{i=1}^{n} (|x_i| - 1).$$

Furthermore, the addition of nondeterminism does not improve this bound.

Lemma 9. *Let $\mathcal{W} = (x_1, \ldots, x_n)$ be a disjoint tuple of strings. Then*

$$\text{nsc}(L_{\alpha\mathcal{W}}) = 2^n + 2^{n-1} \cdot \sum_{i=1}^{n} (|x_i| - 1).$$

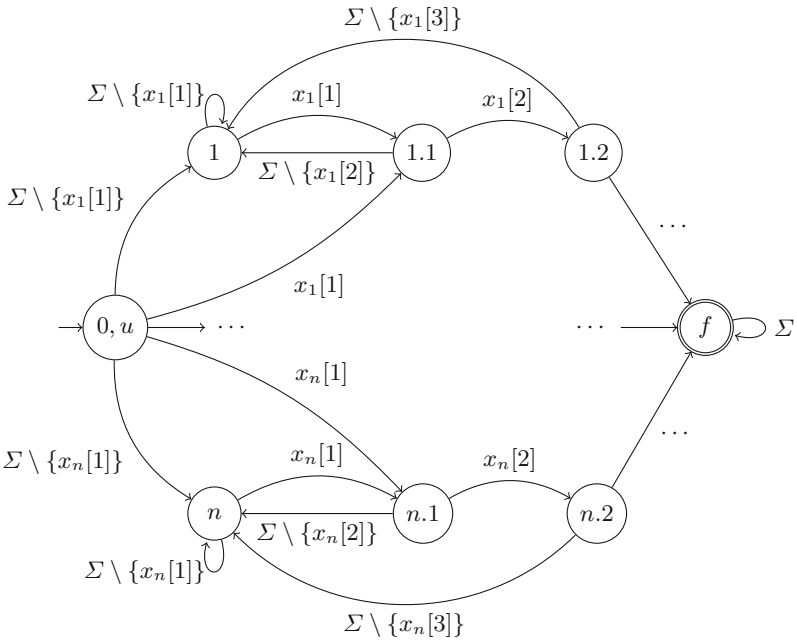

Fig. 2. AFA for a universal infix language over $(x_1, \ldots, x_n)$

Combining Lemmas 7, 8, and 9, we get the following theorem.

Theorem 4. *There exists an m-state AFA A (where m can be arbitrarily large) with tree width n such that any equivalent NFA needs $(m-n) \cdot 2^{n-1}$ states. The AFA A can be chosen to alternate only once between universal and existential states. We note that the alphabet size of A depends on n.*

We give the following constructive example to help clarify the state blow-up from Theorem 4.

Example 1. Let $\mathcal{W} = (aa, b, c)$, and $A = (Q, \{a, b, c\}, \delta, q_0, \{111\})$ be the DFA given in Fig. 3, which recognizes $L_{\alpha(aa,b,c)}$.

To make counting of states easier, below we assume that an AFA computation step always has at most two choices (i.e. computation step is either undefined, is deterministic, or has exactly two existential or universal choices). This assumption can be made with only a constant factor blow-up of the automaton's state complexity [16].

Lemma 10. *Let A be an m-state AFA with tree width n. Then A has an equivalent DFA B with at most $(m+1)^n \cdot (2^n - 1)$ states.*

Combining the upper and lower bounds from Lemmas 8, 9 and 10, we get the following state complexity range for simulating a finite tree width AFA with a DFA.

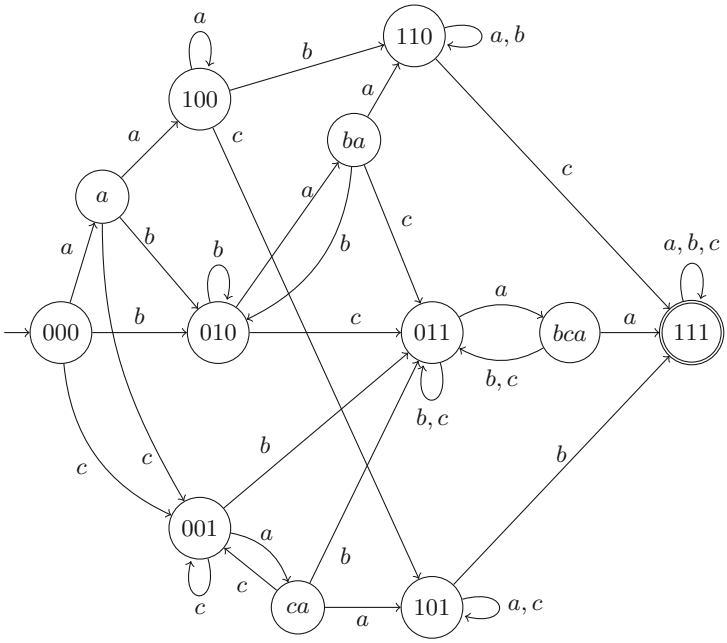

Fig. 3. 12-State DFA for $L_{\alpha(aa,b,c)}$

Corollary 7. *Let A be an m-state AFA with tree width n. Then*

$$2^{n-1} \cdot (m-n) \leq \mathrm{sc}(L(A)) \leq (2^n - 1) \cdot (m+1)^n.$$

Acknowledgments. Research supported by NSERC grant OGP0147224.

We thank the referees for their helpful and thoughtful comments. But, due to the short deadline for submitting the proceedings version, we will try to implement some revisions for a later journal version.

References

1. Chandra, A.K., Kozen, D.C., Stockmeyer, L.J.: Alternation. J. ACM **28**(1), 114–133 (1981)
2. Chrobak, M.: Finite automata and unary languages. Theoret. Comput. Sci. **47**, 149–158 (1986)
3. Fellah, A., Jürgensen, H., Yu, S.: Constructions for alternating finite automata. Int. J. Comput. Math. **35**(1–4), 117–132 (1990)
4. Fernau, H., Krebs, A.: Problems on finite automata and the exponential time hypothesis. Algorithms **10**(1), 24 (2017)
5. Fijalkow, N.: The state complexity of alternating automata. In: Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, 09–12 July 2018, pp. 414–421 (2018)

6. Finkbeiner, B., Sipma, H.: Checking finite traces using alternating automata. Formal Methods Syst. Des. **24**(2), 101–127 (2004)
7. Geffert, V.: An alternating hierarchy for finite automata. Theor. Comput. Sci. **445**, 1–24 (2012)
8. Holzer, M.: On emptiness and counting for alternating finite automata. In: Developments in Language Theory II, At the Crossroads of Mathematics, Computer Science and Biology, Magdeburg, Germany, 17–21 July 1995, pp. 88–97 (1995)
9. Hospodár, M., Jirásková, G., Krajňáková, I.: Operations on boolean and alternating finite automata. In: Fomin, F.V., Podolskii, V.V. (eds.) CSR 2018. LNCS, vol. 10846, pp. 181–193. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-90530-3_16
10. Hromkovič, J.: On the power of alternation in automata theory. J. Comput. Syst. Sci. **31**(1), 28–39 (1985)
11. Hromkovič, J., Seibert, S., Karhumäki, J., Klauck, H., Schnitger, G.: Communication complexity method for measuring nondeterminism in finite automata. Inform. Comput. **172**(2), 202–217 (2002)
12. Jiang, T., Ravikumar, B.: A note on the space complexity of some decision problems for finite automata. Inf. Process. Lett. **40**(1), 25–31 (1991)
13. Keeler, C., Salomaa, K.: Branching measures and nearly acyclic NFAs. In: Pighizzini, G., Câmpeanu, C. (eds.) DCFS 2017. LNCS, vol. 10316, pp. 202–213. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-60252-3_16
14. Keeler, C., Salomaa, K.: Nondeterminism growth and state complexity. In: Hospodár, M., Jirásková, G., Konstantinidis, S. (eds.) DCFS 2019. LNCS, vol. 11612, pp. 210–222. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-23247-4_16
15. King, K.N.: Alternating multihead finite automata (extended abstract). In: Even, S., Kariv, O. (eds.) ICALP 1981. LNCS, vol. 115, pp. 506–520. Springer, Heidelberg (1981). https://doi.org/10.1007/3-540-10843-2_40
16. King, K.N.: Measures of parallelism in alternating computation trees (extended abstract). In: Proceedings of the 13th Annual ACM Symposium on Theory of Computing, 11–13 May 1981, Milwaukee, Wisconsin, USA, pp. 189–201 (1981)
17. Kupferman, O., Ta-Shma, A., Vardi, M.Y.: Counting with automata. Short Paper Presented at the 15th Annual IEEE Symposium on Logic in Computer Science (LICS 2000) (2000)
18. Leiss, E.L.: Succinct representation of regular languages by boolean automata. Theor. Comput. Sci. **13**, 323–330 (1981)
19. Leung, H.: Descriptional complexity of nfa of different ambiguity. Int. J. Found. Comput. Sci. **16**(5), 975–984 (2005)
20. Moriya, E.: A grammatical characterization of alternating pushdown automata. Theor. Comput. Sci. **67**(1), 75–85 (1989)
21. Okhotin, A.: Unambiguous finite automata over a unary alphabet. Inf. Comput. **212**, 15–36 (2012)
22. Palioudakis, A., Salomaa, K., Akl, S.G.: State complexity of finite tree width nfas. J. Autom. Lang. Comb. **17**(2–4), 245–264 (2012)
23. Ruzzo, W.L.: Tree-size bounded alternation. J. Comput. Syst. Sci. **21**(2), 218–235 (1980)