State Complexity and Limited Nondeterminism

Alexandros Palioudakis, Kai Salomaa, and Selim G. Akl

School of Computing, Queen's University,
Kingston, Ontario K7L 3N6, Canada
{alex,ksalomaa,akl}@cs.queensu.ca

Abstract. We consider nondeterministic finite automata having finite tree width (ftw-NFA) where the computation on any input string has a constant number of branches. We give effective characterizations of ftw-NFAs and a tight worst-case state size bound for determinizing an ftw-NFA A as a function of the tree width and the number of states of A. We introduce a lower bound technique for ftw-NFAs and consider the operational state complexity of this model.

Keywords: finite automata, limited nondeterminism, state complexity, language operations.

1 Introduction

The descriptional complexity of finite automata has been studied for over half a century, and there has been particularly much work done over the last two decades. Various ways of quantifying the nondeterminism of finite automata have been considered since the 70's. The ambiguity of a nondeterministic finite automaton (NFA) refers to the number of accepting computations, and other nondeterminism measures count, roughly speaking, the amount of guessing in both accepting and non-accepting computations.

Schmidt [23] first established an exponential trade-off between the size of unambiguous finite automata and general NFAs, and the bound was later refined in [19,25]. The relationship between the degree of ambiguity and succinctness of finite automata was considered in [13,18,22] and the state complexity of unambiguous finite automata over a unary alphabet has been recently studied in [21].

The branching measure of an NFA is the product of the degrees of nondeterministic choices on the best accepting computation [6,17], and the guessing measure counts the minimum (or maximum) number of advice bits used on a computation on a given input [6,13]. The reader is referred to [5] for more details on the different notions of limited nondeterminism.

Here we focus on the state complexity of NFAs having finite *tree width.* By the tree width of an NFA A on input w we mean the total number of nondeterministic branches the computation tree of A on w has, and A has finite tree width if the number is bounded for all inputs. The tree width of an NFA A is called in [13] the *leaf size* of A and in [2] it is denoted as "*computations*(A)". It is known that the tree width of an NFA on inputs of length m is either constant (which

M. Kutrib, N. Moreira, and R. Reis (Eds.): DCFS 2012, LNCS 7386, pp. 252–265, 2012.

situation will be our focus here), or in between linear and polynomial in m, or otherwise $2^{\Theta(m)}$ [13].

The tree width measure can be viewed to be more restrictive than the other nondeterminism measures for NFAs considered in the literature, with the exception of the δNFAs of [2] that are a special case of tree width 2 NFAs. Note that while a DFA equivalent to an n state unambiguous NFA may need $2^n - 1$ states [19], the size blow-up of determinizing a constant tree width NFA is clearly polynomial. Here we give a tight upper bound for the conversion as a function of the size of an NFA and its tree width.

We show that there exist regular languages for which any n-state NFA needs finite tree width 2^{n-2}. We establish that an NFA A has finite tree width if and only if A can be subdivided into deterministic components that are connected by the reachability relation in an acyclic way. We introduce a separator set technique that gives considerably better lower bounds for the size of finite tree width automata than the known fooling set and biclique edge-cover methods [1,4,8] for general NFAs. However, even the separator set technique is not powerful enough to handle, for example, the constructions we use to give lower bounds for the state complexity of union and intersection of finite tree width NFAs. Due to the inherent hardness of minimization of very restricted NFAs [2,9,20], it can perhaps be expected that general techniques do not give optimal lower bounds even for finite tree width NFAs.

Following the convention used by the overwhelming majority of the literature on finite automata, we use an NFA model that allows only one initial state. Some of our state complexity results could be stated in a nicer form if the NFA model were to allow multiple initial states; see Remark 4.1.

Many proofs are omitted due to the limitation on the number of pages.

2 Preliminaries

We assume that the reader is familiar with the basic definitions concerning finite automata [24,26] and descriptional complexity [5,11]. Here we just fix some notation needed in the following.

The set of strings over a finite alphabet Σ is Σ^*, the length of $w \in \Sigma^*$ is $|w|$ and ε is the empty string. The set of positive integers is denoted $\mathbb{N}$. The cardinality of a finite set S is $\#S$.

A nondeterministic finite automaton (NFA) is a 5-tuple $A = (Q, \Sigma, \delta, q_0, F)$, where Q is a finite set of states, Σ is a finite alphabet, $\delta : Q \times \Sigma \to 2^Q$ is the transition function, $q_0 \in Q$ is the start state and $F \subseteq Q$ is the set of accepting states. The function δ is extended in the usual way as a function $Q \times \Sigma^* \to 2^Q$ and the language recognized by A, $L(A)$, consists of strings $w \in \Sigma^*$ such that $\delta(q_0, w) \cap F \neq \emptyset$.

If δ is as above, we denote $\mathrm{rel}(\delta) = \{(q, a, p) \mid p \in \delta(q, a), q, p \in Q, a \in \Sigma\}$. We call $\mathrm{rel}(\delta)$ as the *transition relation* of A. A transition of A is an element $\mu = (q, a, p) \in \mathrm{rel}(\delta)$. The transition μ is nondeterministic if there exists $p' \neq p$ such that $p' \in \delta(q, a)$ and otherwise μ is deterministic. The NFA A is deterministic

(a DFA) if all transitions of A are deterministic, this is equivalent to saying that $\#\delta(q,a) \leq 1$ for all $q \in Q$ and $a \in \Sigma$. Note that we allow DFAs to have undefined transitions. Unless otherwise mentioned, we assume that any state q of an NFA A is reachable from the start state and some computation originating from q reaches a final state.

A *computation chain* of A from state s_1 to s_2 is a sequence of transitions (q_i, a_i, p_i), $1 \leq i \leq k$, where $p_{i+1} = q_i$, $i = 1, \ldots, k-1$, and $s_1 = q_1$, $s_2 = p_k$. A *cycle* of A is a computation chain from state s to the same state s.

Note that we can specify an NFA either as a tuple $(Q, \Sigma, \delta, q_0, F)$ or as $(Q, \Sigma, \mathrm{rel}(\delta), q_0, F)$. In the former notation δ is a function with domain $Q \times \Sigma$ and when we want to specify the transitions as a subset of $Q \times \Sigma \times Q$ we write $\mathrm{rel}(\delta)$. By the *size of* A we mean the number of states of A, $\mathrm{size}(A) = \#Q$.

The minimal size of a DFA or an NFA recognizing a regular language L is called the (nondeterministic) state complexity of L and denoted, respectively, $\mathrm{sc}(L)$ and $\mathrm{nsc}(L)$. Note that we allow DFAs to be incomplete and, consequently, the deterministic state complexity of L may differ by one from a definition using complete DFAs. If $A = (Q, \Sigma, \delta, q_0, F)$ is an NFA (respectively, DFA), the triple (Q, Σ, δ) without the start state and the set of accepting states is called a semi-NFA (respectively, semi-DFA).

For $q \in Q$ and $w \in \Sigma^*$, the *q-computation tree of A on w*, $T_{A,q,w}$, is a finite tree where the nodes are labeled by elements of $Q \times (\Sigma \cup \{\varepsilon, \natural\})$. We define $T_{A,q,w}$ inductively by first setting $T_{A,q,\varepsilon}$ to consist of one node labeled by (q, ε). When $w = au$, $a \in \Sigma$, $u \in \Sigma^*$ and $\delta(q,a) = \emptyset$ we set $T_{A,q,w}$ to be the singleton tree where the only node is labeled by $(q, \natural)$. Then assuming $\delta(q,a) = \{p_1, \ldots, p_m\}$, $m \geq 1$, we define $T_{A,q,w}$ as the tree where the root is labeled by (q,a) and the root has m children where the subtree rooted at the ith child is $T_{A,p_i,u}$, $i = 1, \ldots, m$. For our purposes the order of children of a node is not important and we can assume that the elements of $\delta(q,a)$ are ordered by a fixed but arbitrary linear order. Note that in $T_{A,q,w}$ every path from the root to a leaf has length at most $|w|$. A path may have length less than w because the corresponding computation of A may become blocked at a node labeled by a pair (p,b) where $\delta(p,b) = \emptyset$.

The tree $T_{A,q_0,w}$ is called the *computation tree of A on w* and denoted simply as $T_{A,w}$. The NFA A accepts w if and only if $T_{A,w}$ contains a leaf labeled by an element (q, ε), where $q \in F$. Note that a node label (q, ε) can occur in $T_{A,w}$ only after the corresponding computation branch has consumed the entire string w.

Remark 2.1. We mostly refer to a node of a computation tree of A to be labeled simply by an element $q \in Q$. This is taken to mean that the node is labeled by a pair (q, x) where $x \in \Sigma \cup \{\varepsilon, \natural\}$ is arbitrary.

To conclude the preliminaries, we recall and introduce a few definitions concerning NFAs for unary languages. The below notion of *simple normal form* is from [14] and it is a special case of the Chrobak normal form [3,14]. Note that simple normal form automata do not recognize all unary regular languages.

Definition 2.1. *Let $\Sigma = \{a\}$ and let $(m_1, \ldots, m_k) \in \mathbb{N}^k$. An NFA A over Σ is said to be in $(m_1, \ldots, m_k)$-simple normal form if A consists of an initial state*

q_0 and k disjoint cycles C_i, where the length of C_i is m_i, $1 \leq i \leq k$. The initial state q_0 is not part of any of the cycles and from q_0 there is a transition to one state of C_i, for each $1 \leq i \leq k$. An NFA is in k-simple normal form if it is in $(m_1, \ldots, m_k)$-simple normal form for some positive integers $m_1, \ldots, m_k$, and it is in simple normal form *if it is in k-simple normal form for some $k \geq 1$.*

3 Nondeterministic Tree Width

Various ways to measure the nondeterminism in finite automata have been considered in [6,13,17,19]. The tree width measure was used in [13] under the name of leaf size.

Definition 3.1. *Let $A = (Q, \Sigma, \delta, q_0, F)$ be an NFA and $w \in \Sigma^*$. The* tree width *of A on w, $\mathrm{tw}_A(w)$, is the number of leaves of the tree $T_{A,w}$. The tree width of A is defined as $\mathrm{tw}(A) = \sup\{\mathrm{tw}_A(w) \mid w \in \Sigma^*\}$. We say that A has* finite tree width *if $\mathrm{tw}(A)$ is finite.*

Note that $\mathrm{tw}_A(w)$ is simply the number of all (accepting and non-accepting) branches in the computation tree of A on w. An NFA in k-simple normal form has tree width k. In the following, ftw-NFA (respectively, tw(k)-NFA) stands for a finite tree width NFA (respectively, an NFA having tree width at most k, $k \in \mathbb{N}$).

We will be mainly concerned with the state complexity of NFAs of given finite tree width and for this purpose define, for a regular language L and $k \in \mathbb{N} \cup \{\infty\}$,

$$\mathrm{sctw}_k(L) = \inf\{\mathrm{size}(A) \mid A \text{ is a tw}(k)\text{-NFA and } L(A) = L\}.$$

Above $\mathrm{sctw}_k(L)$ is the smallest number states needed by an NFA of tree width at most k to recognize the language L. In particular, $\mathrm{sctw}_1(L) = \mathrm{sc}(L)$ and $\mathrm{sctw}_\infty(L) = \mathrm{nsc}(L)$.

Definition 3.2. *We say that an NFA A has* optimal tree width *if $L(A)$ is not recognized by any NFA B where $\mathrm{size}(B) \leq \mathrm{size}(A)$ and $\mathrm{tw}(B) \leq \mathrm{tw}(A)$ where one of the inequalities is strict. We say that a regular language L has* tree width *k if L has an optimal $\mathrm{tw}(k)$-NFA A such that A has a minimal number of states among all ftw-NFAs for L.*

When A has optimal tree width, $\mathrm{sctw}_{\mathrm{tw}(A)}(L(A)) = \mathrm{size}(A)$. A language L having tree width k means, roughly speaking, that an NFA recognizing L can "make use of" limited nondeterminism of tree width k, that is, the tree width k nondeterminism allows us to reduce the number of states, but any additional finite nondeterminism would not allow a further reduction of the number of states. If L has tree width 1 this means any ftw-NFA for L cannot be smaller than the minimal (incomplete) DFA.

For a given NFA A, the tree width of the language $L(A)$ can be determined by a brute-force algorithm that checks all NFAs of size less than the size of the minimal DFA for $L(A)$. The problem of finding the tree width of a regular language is intractable because it is known that minimizing bounded tree width NFAs is NP-hard [2,20].

Lemma 3.1. *A regular language L has tree width k ($k \geq 1$) iff L has a* $\mathrm{tw}(k)$*-NFA with optimal tree width and, for any $\ell > k$, L does not have a* $\mathrm{tw}(\ell)$*-NFA with optimal tree width.*

3.1 Characterizations of ftw-NFAs

An immediate observation is that if some cycle of an NFA A contains a nondeterministic transition, then the tree width of A on words that repeat the cycle cannot be bounded.

Lemma 3.2. *If A is an ftw-NFA, then no cycle of A can contain a nondeterministic transition.*

We want to show that the condition of Lemma 3.2 is also sufficient to guarantee that an NFA has finite tree width. For this purpose we establish an upper bound for the tree width of an NFA where no cycle contains a nondeterministic transition.

Theorem 3.1. *If A is an NFA with n states and no cycle contains a nondeterministic transition, then*

$$\mathrm{tw}(A) \leq 2^{n-2}$$

From Lemma 3.2 and Theorem 3.1 we get a characterization of finite tree width NFAs.

Corollary 3.1. *An NFA A has finite tree width iff no cycle of A contains a nondeterministic transition. We can decide in polynomial time whether or not a given NFA has finite tree width.*

The upper bound of Theorem 3.1 is tight as seen in the below example which gives an n-state NFA having optimal tree width 2^{n-2}.

Example 3.1. Let $n \geq 2$, $\Sigma = \{a, b\}$ and $A = (Q, \Sigma, \delta, 1, \{n\})$ where $Q = \{1, \ldots, n\}$ and we define for $i \in \{1, \ldots, n-1\}$, $\delta(i, a) = \{i+1, i+2, \ldots, n\}$, $\delta(i, b) = \{i+1\}$, $\delta(n, a) = \delta(n, b) = \emptyset$.

Since $L(A) \cap b^* = \{b^{n-1}\}$, in any n-state NFA $B = (\{1, \ldots, n\}, \Sigma, \gamma, 1, F_B)$ for the language $L(A)$ there can be only one final state. Without loss of generality, we can set $F_B = \{n\}$ and $\gamma(i, b) = \{i+1\}$, $1 \leq i \leq n-1$. Once the b-transitions are fixed, we observe that

$$(\forall 1 \leq i < j \leq n) : \; b^{i-1}ab^{n-j} \in L(B) \text{ iff } j \in \gamma(i, a).$$

The above means that B must be an isomorphic copy of A and, consequently, A has optimal tree width. It is easy to calculate that $\mathrm{tw}(A) = 2^{n-2}$. Furthermore, since A is a minimal NFA for $L(A)$ the tree width of the language $L(A)$ is 2^{n-2}.

An NFA A with finite tree width can be thought to consist of a finite number of deterministic "components" that are connected by the reachability relation in an acyclic way. This can be formalized as follows.

Definition 3.3. *Let $A = (Q, \Sigma, \mathrm{rel}(\delta), q_0, F)$ be an NFA. Let $B_i = (P_i, \Sigma, \mathrm{rel}(\gamma_i))$, $P_i \subseteq Q$, $1 \leq i \leq k$, be a collection of semi-DFAs. We say that the tuple $\mathcal{D} = (B_1, \ldots, B_k)$ is a* deterministic decomposition of A *if the sets $P_1, \ldots, P_k$ form a partition of Q, $\mathrm{rel}(\gamma_i)$ is the restriction of $\mathrm{rel}(\delta)$ to $P_i \times \Sigma \times P_i$, and every transition of A of the form $(p_1, a, p_2) \in \mathrm{rel}(\delta)$, $p_1, p_2 \in P_i$, $a \in \Sigma$, $1 \leq i \leq k$ is deterministic.*

Note that the above definition says that the transitions of $\mathrm{rel}(\gamma_i)$ have to be deterministic as transitions of the original NFA A, which is more restrictive than just requiring that the transition relation $\mathrm{rel}(\gamma_i)$ of each B_i is deterministic.

For a decomposition $\mathcal{D} = (B_1, \ldots, B_k)$ we can order the components by setting $B_i <_{\mathcal{D}} B_j$, $i \neq j$, if some state of B_j is reachable in A from a state of B_i. We say that the decomposition $\mathcal{D} = (B_1, \ldots, B_k)$ is *acyclic* if $<_{\mathcal{D}}$ is a partial order.

An NFA A where no state q has a nondeterministic transition that is a self-loop on q has always a trivial deterministic decomposition where each component consists of a single state. The property of having an acyclic deterministic decomposition exactly characterizes NFAs having finite tree width.

Theorem 3.2. *An NFA A has finite tree width iff A has an acyclic deterministic decomposition.*

3.2 Converting Finite Tree Width NFAs to DFAs

When applying the subset construction to an NFA with tree width k, only sets of states of size at most k can be reachable. This idea, together with some refinements, yields the following upper bound:

Lemma 3.3. *Let L be a regular language where $\mathrm{sctw}_k(L) = n$ for some $k \leq n-1$. Then $\mathrm{sc}(L) \leq 1 + \sum_{j=1}^{k} \binom{n-1}{j}$.*

A k-entry DFA [12,16] ($k \geq 1$) has a deterministic transition function but allows k-initial states. The computation of a k-entry DFA has at most k branches, however, strictly speaking a k-entry DFA is not a special case of an ftw-NFA because, following usual conventions, our NFA model allows only one initial state. By using a modification of the worst-case construction from [12] that converts a k-entry DFA to an ordinary DFA we see that the upper bound of Lemma 3.3 can be reached.

Theorem 3.3. *For every $1 \leq k \leq n-1$ there exists an n-state NFA $A_{n,k}$ such that $\mathrm{tw}(A_{n,k}) = k$ and $\mathrm{sc}(L(A_{n,k})) = 1 + \sum_{j=1}^{k} \binom{n-1}{j}$.*

Note that, in general, the finite tree width of an n-state NFA is bounded only by 2^{n-2} (Theorem 3.1), however, in cases where the tree width is greater than the number of states, estimates for the size of an equivalent DFA analogous to Lemma 3.3 would clearly not be useful.

As a consequence of results and constructions in [6,19] we observe that there exist regular languages where the unambiguous NFA to ftw-NFA conversion causes an exponential size blow-up.

Proposition 3.1. *For $n \geq 4$, there exists an unambiguous NFA A with n states such that, for any $k \in \mathbb{N}$, $\mathrm{sctw}_k(L(A)) = 2^{n-1}$.*

Proof. Let L_n^{Leung}, $n \geq 3$, be the language having an n-state unambiguous NFA (UFA) A_n used in Theorem 1 of [19]. From there we know that any incomplete DFA for L_n^{Leung} needs $2^n - 1$ states.

As in Theorem 4.2 of [6] we now define $L_n = (\$L_{n-1}^{\mathrm{Leung}}\$)^*$, $n \geq 4$. The language L_n has an n-state UFA. (It is easy to verify that the NFA constructed as in the proof of Theorem 4.2 of [6] is unambiguous.)

As in Theorem 4.2 of [6] it is seen that any NFA B with *finite branching* for the language L_n has at least 2^{n-1} states. Note that the construction of Theorem 4.2 of [6] relies only on the size of the minimal DFA for the original regular languages used to define L_n, and exactly the same construction works when L_n is defined in terms of L_{n-1}^{Leung}. The branching measure of an NFA B, β_B, is not the same as tree width, but it is immediate that any NFA with finite tree width has finite branching. (The converse does not hold in general.) □

3.3 Lower Bounds for the Size of ftw-NFAs

Recently Björklund and Martens [2] have shown that minimization is NP-hard for any class of finite automata that contains the so called δNFAs, which are a restriction of NFAs with tree width 2. Thus, in order to establish lower bounds for the size of ftw-NFAs we need to rely on ad hoc methods inspired by the fooling set techniques used for general NFAs [1,4,8,11].

Let S be a finite set. By an (S, k)-family of sets we mean a family $\mathcal{C} \subseteq 2^S$ such that all sets in $\mathcal{C}$ have cardinality at most k, and any two sets in $\mathcal{C}$ are incomparable as subsets of S.

Lemma 3.4. *Let S be a finite set of cardinality n and $k \leq \frac{n}{2}$. If $\mathcal{C}$ is an (S, k)-family, then $\#\mathcal{C} \leq \binom{n}{k}$. (Here $\#\mathcal{C}$ is the number of sets in the family $\mathcal{C}$.)*

Let L be a regular language over Σ. We say that a finite set of strings $\{u_1, \ldots, u_t\}$, $u_i \in \Sigma^*$, $1 \leq i \leq t$, is a t-separator set for the language L if

$$(\forall 1 \leq i, j \leq t,\ i \neq j)(\exists z \in \Sigma^*)\ u_i z \in L \text{ and } u_j z \notin L. \tag{1}$$

Note that the above definition treats (i, j) as an ordered pair, and it is not sufficient if $z \in \Sigma^*$ satisfies the corresponding condition where i and j are interchanged. On the other hand, the condition (1) allows z to depend on the pair (i, j) and, consequently, for a given regular language we can typically find a much larger separator set than a fooling set; see Example 3.2.

Lemma 3.5. *Suppose that a regular language L has a t-separator set $\{u_1, \ldots, u_t\}$, $t \geq 1$. If L has a tw(k)-NFA A with n states, where $k \leq \frac{n}{2}$, then*

$$\binom{n}{k} \geq t.$$

Proof. Let $A = (Q, \Sigma, \delta, q_0, F)$ be an NFA with $\#Q = n$ and $\mathrm{tw}(A) = k$. Define $P_i = \delta(q_0, u_i)$, $i = 1, \ldots, t$. Since A has tree width k, $\#P_i \leq k$. Consider $1 \leq i, j \leq t$, $i \neq j$, and suppose $P_i \subseteq P_j$. This means that, for any $z \in \Sigma^*$, $u_i z \in L$ implies $u_j z \in L$. Thus the sets P_i, $i = 1, \ldots, t$, have to be pairwise incomparable as subsets of Q and the claim follows from Lemma 3.4. □

Example 3.2. Consider the unary language $L_m = \{w \in a^* \mid |w| \not\equiv 0 \pmod m\}$, $m \geq 1$. The language L_m has a separator set $\{\varepsilon, a, a^2, \ldots, a^{m-1}\}$. According to Lemma 3.5, if L_m has a $\mathrm{tw}(k)$-NFA with n states where $k \leq \frac{n}{2}$, we have the estimation $\binom{n}{k} \geq m$. This is exponentially better than the lower bound obtained for the size of (general) NFAs with the bipartite dimension method (Theorem 9 in [8]). Recall that the bipartite dimension method is guaranteed to give at least as good bounds as (and often better bounds than) the fooling set methods [8,11].

However, also our bound does not coincide with the real lower bound except in the case $k = 1$.

Example 3.3. For $\{m_1, \ldots, m_k\} \subseteq \mathbb{N}$, $k \geq 1$, we define

$$L_{\{m_1,\ldots,m_k\}} = \{w \in a^* \mid (\exists 1 \leq j \leq k)\ |w| \equiv 0 \pmod{m_j}\}. \tag{2}$$

When $m_1, \ldots, m_k$ are pairwise relatively prime, $\{a^1, a^2, \ldots, a^{\prod_{i=1}^k m_i}\}$ is a separator set and, consequently, if $L_{\{m_1,\ldots,m_k\}}$ has a $\mathrm{tw}(k)$-NFA A where $k \leq \frac{\mathrm{size}(A)}{2}$, we have

$$\binom{\mathrm{size}(A)}{k} \geq \prod_{i=1}^{k} m_i.$$

Again this is not an optimal bound and, in the next section, we will see that under the above assumptions for any $h \geq k \geq 2$, $\mathrm{sctw}_h(L_{\{m_1,\ldots,m_k\}}) = 1 + \sum_{i=1}^k m_i$.

4 The State Complexity of Union and Intersection

We consider the union and intersection of languages recognized by a $\mathrm{tw}(k_1)$-NFA and a $\mathrm{tw}(k_2)$-NFA. For union (respectively, intersection) we give a state complexity upper bound in terms of a $\mathrm{tw}(k_1 + k_2)$-NFA (respectively, $\mathrm{tw}(k_1 \cdot k_2)$-NFA), and almost matching lower bounds.

Lemma 4.1. *For regular languages L_i and $k_i \geq 1$, $i = 1, 2$,*

(i) $\mathrm{sctw}_{k_1+k_2}(L_1 \cup L_2) \leq \mathrm{sctw}_{k_1}(L_1) + \mathrm{sctw}_{k_2}(L_2) + 1$.
(ii) $\mathrm{sctw}_{k_1 \cdot k_2}(L_1 \cap L_2) \leq \mathrm{sctw}_{k_1}(L_1) \cdot \mathrm{sctw}_{k_2}(L_2)$.

Proof. Let $A_i = (Q_i, \Sigma, \delta_i, q_{0,i}, F_i)$ be a $\mathrm{tw}(k_i)$-NFA recognizing L_i, $i = 1, 2$.

(i) Without loss of generality we assume that $Q_1 \cap Q_2 = \emptyset$. We define $B = (P, \Sigma, \gamma, p_0, F_B)$ where $P = Q_1 \cup Q_2 \cup \{p_0\}$, $p_0 \notin Q_1 \cup Q_2$ is a new state,

$$F_B = \begin{cases} F_1 \cup F_2 & \text{if } q_{0,i} \notin F_i, i = 1, 2, \\ F_1 \cup F_2 \cup \{p_0\} & \text{otherwise,} \end{cases}$$

and for $b \in \Sigma$, $q \in Q_i$, we set $\gamma(q,b) = \delta_i(q,b)$, $i = 1,2$, and $\delta(p_0,b) = \delta_1(q_{0,1},b) \cup \delta_2(q_{0,1},b)$.

It is sufficient to verify that $\mathrm{tw}(B) \leq k_1 + k_2$. Consider $w = bw'$, where $b \in \Sigma$, $w \in \Sigma^*$. If $\delta_i(q_{0,i},b) = \emptyset$, $i = 1,2$, $T_{B,w}$ and $T_{A_i,w}$, $i = 1,2$, each are a singleton tree and we are done. (The same holds in the case when $w = \varepsilon$.)

If $\delta_1(q_{0,1},b) \neq \emptyset$ and $\delta_2(q_{0,2},b) = \emptyset$, the tree $T_{B,w}$ is obtained from $T_{A_1,w}$ by changing the label of the root.

Finally consider the case where $\delta_i(q_{0,i},b) \neq \emptyset$, $i = 1,2$. Now the sequence of immediate subtrees of the root of $T_{B,w}$ consists of the immediate subtrees of the root of $T_{A_1,w}$ and of $T_{A_2,w}$.[1] This means that the number of leaves of $T_{B,w}$ is the sum of the numbers of leaves of $T_{A_1,w}$ and of $T_{A_2,w}$, respectively.

(ii) We define $C = (Q_1 \times Q_2, \Sigma, \gamma, (q_{0,1}, q_{0,2}), F_1 \times F_2)$ where for $b \in \Sigma$, $q_i \in Q_i$, $i = 1,2$, $\gamma((q_1,q_2),b) = \delta_1(q_1,b) \times \delta(q_2,b)$.

Again it is sufficent to verify the claimed bound for the tree width: $\mathrm{tw}(C) \leq k_1 \cdot k_2$. We denote the number of leaves of a tree T as $\#\mathrm{leaves}(T)$. We claim that for all $w \in \Sigma^*$ and $q_i \in Q_i$, $i = 1,2$,

$$\#\mathrm{leaves}(T_{C,(q_1,q_2),w}) \leq \#\mathrm{leaves}(T_{A_1,q_1,w}) \cdot \#\mathrm{leaves}(T_{A_2,q_2,w}). \tag{3}$$

We prove the claim using induction on the length of w. When $w = \varepsilon$, $T_{C,(q_1,q_2),\varepsilon}$, and $T_{A_i,q_i,\varepsilon}$, $i = 1,2$, each are a singleton tree.

Consider then $w = bw'$, where $b \in \Sigma$, $w \in \Sigma^*$. If $\delta_i(q_1,b) = \emptyset$ or $\delta_i(q_2,b) = \emptyset$, $T_{C,(q_1,q_2),w}$ is a singleton tree and we are done. Hence, without loss of generality, we can assume $\delta_i(q_i,b) \neq \emptyset$, $i = 1,2$. Now the subtrees corresponding to the children of the root of $T_{C,(q_1,q_2),w}$ are the trees $T_{C,(p_1,p_2),w'}$, $p_i \in \delta_i(q_i,b)$, $i = 1,2$. Thus, by the inductive assumption,

$$\begin{aligned}\#\mathrm{leaves}(T_{C,(q_1,q_2),w}) &= \sum_{p_i \in \delta_i(q_i,b), i=1,2} \#\mathrm{leaves}(T_{C,(p_1,p_2),w'}) \\ &\leq \sum_{p_i \in \delta_i(q_i,b), i=1,2} \#\mathrm{leaves}(T_{A_1,p_1,w'}) \cdot \#\mathrm{leaves}(T_{A_2,p_2,w'}).\end{aligned}$$

Since the number of leaves of $T_{A_i,q_i,w}$ is equal to $\sum_{p_i \in \delta_i(q_i,b)} \#\mathrm{leaves}(T_{A_i,p_i,w'})$, $1 \leq i \leq 2$, we have verified that (3) holds for the string w.

□

The proof of Lemma 4.1 uses the "natural" constructions, respectively, for the union and intersection of two NFAs. Note that the cross-product of a $\mathrm{tw}(k_1)$- and a $\mathrm{tw}(k_2)$-NFA, typically, has tree width considerably less than $k_1 \cdot k_2$, however, as we will see, in the worst case the upper bound of Lemma 4.1 (ii) cannot be improved, at least not by much.

Our lower bound constructions are based on languages of the form $L_{\{m_1,\ldots,m_k\}}$ considered in Example 3.3. When the integers m_i, $1 \leq i \leq k$, are pairwise

[1] Recall that for our purposes the order of children of a node is not important and we assume that the elements of $\gamma(p_0,b)$ have an arbitrary, but fixed, order.

relatively prime, from Theorem 2.1 of [14] it follows easily that $L_{\{m_1,\ldots,m_k\}}$ has a minimal NFA in k-simple normal form. We will need a corresponding property also under slightly less restrictive conditions.

Before establishing lower bounds for the size of ftw-NFAs for the unary languages (2) we state the following corollary which follows from Lemma 3.2 using the observation that a unary NFA may exit a cycle only using a nondeterministic transition.

Corollary 4.1. *If A is an ftw-NFA over a unary alphabet, then no cycle C of A has a transition exiting C and, in particular, no two cycles of A can share a state. Furthermore, if A has more than one cycle, then the start state of A cannot be part of any cycle.*

Below we will show that, under certain assumptions on the numbers m_i, the language $L_{\{m_1,\ldots,m_k\}}$ has a unique minimal ftw-NFA in k-simple normal form. More generally, from Corollary 4.1 it follows that any unary regular language has a (not necessarily unique) minimal ftw-NFA that is in Chrobak normal form [3,14].

We say that a set of integers $\{m_1,\ldots,m_k\}$, $m_i \geq 2$, $1 \leq i \leq k$, is *pairwise relatively prime* if, for all $i \neq j$, m_i and m_j do not have any common factor greater than one. A set of integers P is said to be an *independent set of prime pairs* if we can write $P = \{p_1 \cdot p_2 \mid p_i \in P_i, i = 1, 2\}$, for two sets of prime numbers P_1 and P_2 where $P_1 \cap P_2 = \emptyset$. Note that the elements of an independent set of prime pairs P are not pairwise relatively prime, however, no element of P is a multiple of any other element.

Lemma 4.2. *Let $k \geq 2$ and assume that the set $\{m_1,\ldots,m_k\}$, $m_i \geq 2$, $1 \leq i \leq k$, is either a set of independent prime pairs, or a pairwise relatively prime set that is not $\{2,3\}$. Then the minimal ftw-NFA for $L_{\{m_1,\ldots,m_k\}}$ is unique (up to renaming of states) and it is in $(m_1,\ldots,m_k)$-simple form.*

Corollary 4.2. *Let $k \geq 2$. Assume that the set $\{m_1,\ldots,m_k\}$, $m_i \geq 2$, $1 \leq i \leq k$, is either an independent set of prime pairs or a pairwise relatively prime set. Then*

$$\mathrm{sctw}_k(L_{\{m_1,\ldots,m_k\}}) = 1 + \sum_{i=1}^{k} m_k.$$

Furthermore, assuming we exclude the case $k = 2$, $\{m_1, m_2\} = \{2, 3\}$, the language $L_{\{m_1,\ldots,m_k\}}$ has tree width k.

Note that if a regular language L has tree width k, the maximum amount of limited nondeterminism an NFA recognizing L can "make use of" is precisely k (see Definition 3.2). By relying on Corollary 4.2, in the following lemmas we give state complexity lower bounds for the union and intersection of ftw-NFAs.

Lemma 4.3. *For every $k_i, n_i \in \mathbb{N}$, $i = 1, 2$, there exist regular languages L_i of tree width k_i such that $\mathrm{sctw}_{k_i}(L_i) \geq n_i$ and*

$$\mathrm{sctw}_{k_1+k_2}(L_1 \cup L_2) \geq \mathrm{sctw}_{k_1}(L_1) + \mathrm{sctw}_{k_2}(L_2) - 1.$$

Furthermore, $L_1 \cup L_2$ has tree width $k_1 + k_2$.

Proof. First consider the case $k_1, k_2 \geq 2$. Choose a set of pairwise relatively prime integers $\{m_{1,i}, m_{2,j} \mid m_{1,i}, m_{2,j} \geq 2,\ 1 \leq i \leq k_1,\ 1 \leq j \leq k_2\}$, where $\sum_{i=1}^{k_h} m_{h,i} \geq n_h$, $h = 1, 2$. Define $L_h = L_{\{m_{h,1},\ldots,m_{h,k_h}\}}$, $h = 1, 2$. Directly from the definition (2) it follows that

$$L_1 \cup L_2 = L_{\{m_{1,1},\ldots,m_{1,k_1},m_{2,1},\ldots,m_{2,k_2}\}}.$$

Immediately from Corollary 4.2, we get that

$$\operatorname{sctw}_{k_1+k_2}(L_1 \cup L_2) = \operatorname{sctw}_{k_1}(L_1) + \operatorname{sctw}_{k_2}(L_2) - 1, \tag{4}$$

and L_i has tree width k_i, $i = 1, 2$, as well as, $L_1 \cup L_2$ has tree width $k_1 + k_2$.

When $k_1 \geq 2$ and $k_2 = 1$, the construction is the same but instead of (4) we get $\operatorname{sctw}_{k_1+1}(L_1 \cup L_2) = \operatorname{sctw}_{k_1}(L_1) + \operatorname{sctw}_1(L_2)$.

Finally, with $k_1 = k_2 = 1$, by choosing two relatively prime numbers $m_i \geq n_i$, $\{m_1, m_2\} \neq \{2, 3\}$, $L_i = L_{\{m_i\}}$, $i = 1, 2$, we have $\operatorname{sctw}_2(L_1 \cup L_2) = \operatorname{sctw}_1(L_1) + \operatorname{sctw}_1(L_2) + 1$, and the language $L_1 \cup L_2$ has tree width 2. Note that $\{m_1, m_2\} = \{2, 3\}$ is the one exception in Lemma 4.2, and the tree width of $L_{\{2,3\}}$ would be one. □

Above note that since $L_1 \cup L_2$ has tree width $k_1 + k_2$, it follows that for any $k < k_1 + k_2$, $\operatorname{sctw}_k(L_1 \cup L_2)$ is strictly larger than $\operatorname{sctw}_{k_1}(L_1) + \operatorname{sctw}_{k_2}(L_2) - 1$.

Lemma 4.4. *For every $k_i, n_i \in \mathbb{N}$, $i = 1, 2$, there exist regular languages L_i of tree width k_i such that $\operatorname{sctw}_{k_i}(L_i) \geq n_i$ and*

$$\operatorname{sctw}_{k_1 \cdot k_2}(L_1 \cap L_2) \geq (\operatorname{sctw}_{k_1}(L_1) - 1) \cdot (\operatorname{sctw}_{k_2}(L_2) - 1) + 1.$$

Furthermore, $L_1 \cap L_2$ has tree width $k_1 \cdot k_2$.

Proof. We begin by considering the case $k_1, k_2 \geq 2$. Let $p_{i,j}$, $1 \leq j \leq k_i$, $1 \leq i \leq 2$, be distinct prime numbers, where $\sum_{j=1}^{k_i} p_{i,j} \geq n_i$, $i = 1, 2$.

We define $L_i = L_{\{p_{i,1},\ldots,p_{i,k_i}\}}$, $i = 1, 2$. By Corollary 4.2 we know that

$$\operatorname{sctw}_{k_i}(L_i) = 1 + \sum_{j=1}^{k_i} p_{i,j}, \quad 1 \leq i \leq 2, \tag{5}$$

where the tree width of L_i is k_i. Define $P(k_1, k_2) = \{p_{1,j_1} \cdot p_{2,j_2} \mid 1 \leq j_i \leq k_i,\ i = 1, 2\}$. Here $P(k_1, k_2)$ is an independent set of prime pairs. Also we note that $L_1 \cap L_2 = L_{P(k_1,k_2)}$. Now, again using Corollary 4.2, the tree width of $L_1 \cap L_2$ is $\#P(k_1, k_2) = k_1 \cdot k_2$ and

$$\operatorname{sctw}_{k_1 \cdot k_2}(L_1 \cap L_2) = 1 + \sum_{1 \leq j_1 \leq k_1, 1 \leq j_2 \leq k_2} p_{1,j_1} \cdot p_{2,j_2}.$$

Comparing this with (5) we get that the inequality in the claim of the lemma holds as an equality.

In the case where $k_1 \geq 2$ and $k_2 = 1$, instead of (5), we have $\mathrm{sctw}_1(L_2) = p_{2,1}$ and, using Corollary 4.2, we get

$$\mathrm{sctw}_{k_1}(L_1 \cap L_2) = (\mathrm{sctw}_{k_1}(L_1) - 1) \cdot \mathrm{sctw}_1(L_2) + 1.$$

Finally, when $k_1 = k_2 = 1$, the inequality in the statement of the lemma holds due to the well-known lower bound for the deterministic state complexity of intersection. □

According to Lemmas 4.3 and 4.4 the lower bound can, corresponding to any tree widths k_i, $i = 1, 2$, be reached with automata of arbitrarily large sizes. In particular, in the case of intersection, it is not clear whether the construction could be modified to always allow us to choose $\mathrm{sctw}_{k_i}(L_i)$ to be exactly n_i.

The lower bound for union differs only by a constant 2 from the upper bound in Lemma 4.1 and for intersection the gap is larger. We conjecture that the bounds of Lemma 4.1 are optimal for the NFA model (used here and in most of the literature) that allows only a single initial state. However, we might need to use NFAs over a nonunary alphabet to get exactly matching lower bounds.

Remark 4.1. For an NFA model that allows multiple initial states, the upper bound (of Lemma 4.1) for union would be changed to be $\mathrm{sctw}_{k_1}(L_1)+\mathrm{sctw}_{k_2}(L_2)$, while the upper bound for intersection would stay the same. For NFAs with multiple initial states, the lower bound constructions of the proof of Lemma 4.3 and 4.4 will exactly match the corresponding state complexity upper bounds that are modified from Lemma 4.1 as described above.

4.1 Other Language Operations

A general NFA recognizing the concatenation of an n_1 and an n_2 state NFA needs only n_1+n_2 states [10]. For ftw-NFAs the situation is not equally simple because, with only limited nondeterminism available, the automaton is not able to guess when a string belonging to the first language ends. The following upper bound is inspired by the construction originally used for deterministic state complexity of concatenation [26,27]. Note however that, with $k = 1$, the result of Lemma 4.5 does not coincide with the deterministic state complexity of concatenation. This is because a tw(1)-NFA is an incomplete DFA, and the well-known deterministic state complexity results [26,27] are stated in terms of complete DFAs.

Lemma 4.5. *For regular languages L_i, $i = 1, 2$, and $k \geq 1$,*

$$\mathrm{sctw}_k(L_1 \cdot L_2) \leq \mathrm{sctw}_k(L_1) \cdot 2^{\mathrm{nsc}(L_2)} + 2^{\mathrm{nsc}(L_2)-1} - 1.$$

The bound of Lemma 4.5 is stated in terms of the nondeterministic state complexity of the second language L_2 because the construction used in the proof would result in the same bound even if L_2 were recognized by an ftw-NFA. We do not have a lower bound construction that would be close to the upper bound of Lemma 4.5 for general values $k \geq 2$.

Determining the worst-case state complexity of operations such as concatenation and Kleene-star for ftw-NFAs remains open. For the Kleene-star operation it seems that, in the worst-case, an ftw-NFA cannot be smaller than a DFA.

Note that using constructions inspired by [6], similar to the one used here in the proof of Proposition 3.1, it is, in fact, possible to construct regular languages L such that any ftw-NFA for L^* cannot be smaller than a DFA. However, the construction delimits the strings of L by end-markers, which makes it seem unsuitable to be used in worst-case constructions for the state complexity of Kleene-star (or concatenation) where we, roughly speaking, need to make it hard for the NFA to detect the end of a substring belonging to L.

5 Conclusion and Open Problems

The results of the previous section have barely scratched the surface of operational state complexity of ftw-NFAs and most questions in this area remain open. Below we mention some other possible future research directions.

Hromkovič et al. [13] have given upper and lower bounds for the tree width of an NFA A in terms of the ambiguity of A and the number of advice bits required by A. (The tree width is called "leaf size" in [13].) The *guessing measure* of A [6] is related to the advice measure, except that for a given input it considers the computation of A using the least (as opposed to the largest) number of advice bits. Also, Goldstine et al. [7] have studied the relationships between the ambiguity of an NFA A and the amount of nondeterminism used by A.

In the context of state complexity, similar questions could be asked for regular languages, as opposed to individual NFAs. For example, it would be interesting to know, for a given regular language L, how the optimal sizes of an NFA recognizing L, respectively, with advice measure k and with tree width k' relate to each other. Similar questions can be asked also for other combinations of the complexity measures. Note that from Proposition 3.1 we know that there exist regular languages for which an unambiguous NFA can be exponentially smaller than any ftw-NFA. On the other hand, the state complexity of converting an ftw-NFA to an unambiguous NFA remains, to the best of our knowledge, open.

References

1. Birget, J.C.: Intersection and union of regular languages and state complexity, Inform. Process. Lett. 43, 185–190 (1992)
2. Björklund, H., Martens, W.: The tractability frontier for NFA minimization. J. Comput. System Sci. 78, 198–210 (2012)
3. Chrobak, M.: Finite automata and unary languages. Theoret. Comput. Sci. 47, 149–158 (1986)
4. Glaister, I., Shallit, J.: A lower bound technique for the size of nondeterministic finite automata. Inf. Proc. Lett. 59, 75–77 (1996)
5. Goldstine, J., Kappes, M., Kintala, C.M.R., Leung, H., Malcher, A., Wotschke, D.: Descriptional complexity of machines with limited resources. J. Univ. Comput. Sci. 8, 193–234 (2002)

6. Goldstine, J., Kintala, C.M.R., Wotschke, D.: On measuring nondeterminism in regular languages. Inform. Comput. 86, 179–194 (1990)
7. Goldstine, J., Leung, H., Wotschke, D.: On the relation between ambiguity and nondeterminism in finite automata. Inform. Comput. 100, 261–270 (1992)
8. Gruber, H., Holzer, M.: Finding Lower Bounds for Nondeterministic State Complexity Is Hard. In: Ibarra, O.H., Dang, Z. (eds.) DLT 2006. LNCS, vol. 4036, pp. 363–374. Springer, Heidelberg (2006)
9. Gruber, H., Holzer, M.: Inapproximability of Nondeterministic State and Transition Complexity Assuming $\mathbf{P} \neq \mathbf{NP}$. In: Harju, T., Karhumäki, J., Lepistö, A. (eds.) DLT 2007. LNCS, vol. 4588, pp. 205–216. Springer, Heidelberg (2007)
10. Holzer, M., Kutrib, M.: Nondeterministic descriptional complexity of regular languages. Internat. J. Foundations of Comput. Sci. 14, 1087–1102 (2003)
11. Holzer, M., Kutrib, M.: Descriptional and computational complexity of finite automata — A survey. Inf. Comput. 209, 456–470 (2011)
12. Holzer, M., Salomaa, K., Yu, S.: On the state complexity of k-entry deterministic finite automata. J. Automata, Languages and Combinatorics 6, 453–466 (2001)
13. Hromkovič, J., Seibert, S., Karhumäki, J., Klauck, H., Schnitger, G.: Communication complexity method for measuring nondeterminism in finite automata. Inform. Comput. 172, 202–217 (2002)
14. Jiang, T., McDowell, E., Ravikumar, B.: The structure and complexity of minimal NFA's over a unary alphabet. Internat. J. Foundations Comput. Sci. 2, 163–182 (1991)
15. Jiang, T., Ravikumar, B.: Minimal NFA problems are hard. SIAM J. Comput. 22, 1117–1141 (1993)
16. Kappes, M.: Descriptional complexity of deterministic finite automata with multiple initial states. J. Automata, Languages, and Combinatorics 5, 269–278 (2000)
17. Kintala, C.M.R., Wotschke, D.: Amounts of nondeterminism in finite automata. Acta Inform. 13, 199–204 (1980)
18. Leung, H.: Separating exponentially ambiguous finite automata from polynomially ambiguous finite automata. SIAM J. Comput, 1073–1082 (1998)
19. Leung, H.: Descriptional complexity of NFA of different ambiguity. Internat. J. Foundations Comput. Sci. 16, 975–984 (2005)
20. Malcher, A.: Minimizing finite automata is computationally hard. Theoret. Comput. Sci. 327, 375–390 (2004)
21. Okhotin, A.: Unambiguous Finite Automata over a Unary Alphabet. In: Hliněný, P., Kučera, A. (eds.) MFCS 2010. LNCS, vol. 6281, pp. 556–567. Springer, Heidelberg (2010)
22. Ravikumar, B., Ibarra, O.: Relating the type of ambiguity of finite automata to the succinctness of their representation. SIAM J. Comput. 18, 1263–1282 (1989)
23. Schmidt, E.M.: Succinctness of description of context-free, regular and unambiguous languages. Ph.D. thesis, Cornell University (1978)
24. Shallit, J.: A Second Course in Formal Languages and Automata Theory. Cambridge University Press (2009)
25. Stearns, R., Hunt, H.: On the equivalence and containment problems for unambiguous regular expressions, regular grammars and finite automata. SIAM J. Comput. 14, 596–611 (1985)
26. Yu, S.: Regular languages. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of Formal Languages, vol. I, pp. 41–110. Springer (1997)
27. Yu, S., Zhuang, Q., Salomaa, K.: The state complexity of some basic operations on regular languages. Theoret. Comput. Sci. 125, 315–328 (1994)