Nondeterminism Growth and State Complexity

Chris Keeler, Kai Salomaa

Nondeterminism Growth and State Complexity

Chris Keeler and Kai Salomaa

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

Abstract. Tree width (respectively, string path width) measures the maximal number of partial (respectively, complete) computations of a nondeterministic finite automaton (NFA) on an input of given length. We study the growth rate of the tree width and string path width measures. As the main result we show that the degree of the polynomial bounding the tree width of an NFA differs by at most one from the degree of the polynomial bounding the string path width. Also we show that for $m \geq 4$ there exists an m-state NFA with finite string path width such that any equivalent finite tree width NFA needs $2^{m-2} + 1$ states.

1 Introduction

Deterministic and nondeterministic finite automata (DFA and NFA) define the class of regular languages, and have been systematically studied for over 60 years. More recently there has been much interest in automata employing limited nondeterminism [2, 5, 13]. Different *measures of nondeterminism* allow us to quantify the amount and type of nondeterminism present in an NFA's computations.

Ambiguity is probably the first nondeterminism measure to be studied systematically. The *ambiguity* [9, 10, 13, 15] of an NFA A on an input string counts the number of accepting computations of A on that string and the *tree width* [5, 11] of A counts the number of all computations of A on the input string. The *string path width* [6, 7] lies between tree width and ambiguity, as it does not count the partial computations counted by tree width, but it does count complete nonaccepting computations. We extend each of these measures of nondeterminism to functions on integers, and consider the growth rate of the corresponding function with respect to worst-case inputs of given length. The growth rate of ambiguity and tree width has been considered, respectively, e.g., in [9, 13, 15] and in [5].

State complexity is another topic in automata theory with much recent research [4, 8]. The study of state complexity aspects of limited nondeterminism was initiated by Goldstine et al. [2] who showed that there exists an m-state NFA A such that any finite *branching* NFA equivalent to A needs 2^{m-1} states. More generally, Goldstine et al. [2] gave a spectrum result which establishes that there exist regular languages for which different finite amounts of nondeterminism yield incremental savings in the number of states. Hromkovič et al. [5] have shown that there exist m-state NFAs with linear ambiguity such that any equivalent finitely ambiguous NFA needs close to 2^m states. Palioudakis et al. [11] showed that there exists an m-state unambiguous NFA such that an equivalent

NFA with finite tree width needs 2^{m-1} states. Here we establish a similar state complexity blow-up between NFAs of finite string path width and finite tree width, respectively.

The contents of this paper are as follows. Section 2 recalls definitions and fixes the notation. Section 3.1 shows that an NFA has polynomial string path width if and only if its tree width is polynomial and, furthermore, that the degrees of the polynomials differ by at most one. Also, it is shown that for NFAs without useless states the degree of the polynomial bounding the growth rate of ambiguity coincides with the polynomial bounding string path width. Section 3.2 shows that an NFA has exponential string path width if and only if it has exponential tree width. Section 4 gives polynomial-time algorithms to decide whether the growth rate of an NFA's tree width (respectively, string path width) is polynomial or exponential, and additionally, to compute the degree of the polynomial bounding the growth. These algorithms utilize existing algorithms from the literature [9, 15]. Section 5 shows that for $m \geq 4$ there exists an m-state NFA A with finite string path width such that any finite tree width NFA equivalent to A needs at least $2^{m-2} + 1$ states.

2 Preliminaries

Here we recall definitions and notation needed in later sections. We assume that the reader is familiar with finite automata, and point them to resources such as [14]. The set of strings over an alphabet Σ is denoted as Σ^*, Σ^+ is the set of nonempty strings and the set of strings of length ℓ over Σ is Σ^ℓ. We use $|S|$ to mean the cardinality of a finite set S, ε to mean the empty string, and $\mathbb{N}$ to mean the set of positive integers. Consider a function $f(\ell) : \mathbb{N} \to \mathbb{N}$. If for $d \in \mathbb{N}$, $f(\ell) \in O(\ell^d)$, (respectively, $\in \Theta(\ell^d)$), then we say that f has *polynomial growth degree* d (respectively, *strict* polynomial growth degree d). If $f(\ell) \in 2^{\Theta(\ell)}$, then we say that $f(\ell)$ has *exponential growth*.

A *nondeterministic finite automaton* (NFA) is a 5-tuple $A = (Q, \Sigma, \delta, q_0, F)$ consisting of a finite set of states Q, a finite alphabet Σ, a transition function $\delta : Q \times \Sigma \to 2^Q$, an initial state $q_0 \in Q$, and a set of final states $F \subseteq Q$. The transition function is extended in the usual way as a function $\delta : Q \times \Sigma^* \to 2^Q$ and the language recognized by A is $L(A) = \{w \in \Sigma^* \mid \delta(q_0, w) \cap F \neq \emptyset\}$.

The NFA A is a *deterministic* finite automaton (DFA) if $|\delta(q, c)| \leq 1$ for all $q \in Q$ and $c \in \Sigma$. If $\delta(q, c)$ is always a singleton set, A is a complete DFA.

Without loss of generality, we assume that all states of an NFA are *reachable* from the start state. In the event that there are non-reachable states, they can simply be removed. Unless otherwise specified, we do not assume that every state can reach a final state, i.e., states need not be *co-reachable*.

A *path* of A from $p_1 \in Q$ to $p_{\ell+1} \in Q$ on a string $a_1 \cdots a_\ell \in \Sigma^*$ is an ordered sequence of the form $(p_1, a_1, p_2, a_2, \ldots, p_\ell, a_\ell, p_{\ell+1})$ such that $p_{i+1} \in \delta(p_i, a_i)$, $1 \leq i \leq \ell$. The set of all paths from state q to state p on string $w \in \Sigma^*$ is denoted as $\text{paths}(q, w, p)$.

A path of A from the initial state q_0 to a state p on string w is a *(complete) computation* on w and it is an *accepting computation* if $p \in F$. A path of A from the initial state q_0 to a state p on a prefix w_1 of w is a *partial computation* of A on w if either $w_1 = w$ (complete computation) or $w = w_1 b w_2$, $b \in \Sigma$, and $\delta(p, b) = \emptyset$. That is, a partial computation on w must read the string as far as it can, until it encounters an undefined transition.

For $L \subseteq \Sigma^*$, the Myhill-Nerode equivalence relation of L, $\equiv_L \subseteq \Sigma^* \times \Sigma^*$, is defined by setting for $x, y \in \Sigma^*$:

$$x \equiv_L y \quad \text{iff} \quad (\forall z \in \Sigma^*)\ xz \in L \Leftrightarrow yz \in L$$

For a regular language L, the number of equivalence classes of $\equiv_L$ gives the number of states of the minimal complete DFA recognizing L [14].

The *ambiguity* of an NFA A on a string w, denoted $\mathrm{da}(A, w)$, is the number of accepting computations of A on w. The ambiguity of A on all strings of length ℓ is $\mathrm{da}(A, \ell) = \max\{\mathrm{da}(A, w) \mid w \in \Sigma^\ell\}$, and the ambiguity of A is $\mathrm{da}(A) = \sup_{\ell \in \mathbb{N}}\{\mathrm{da}(A, \ell)\}$. Note that $\mathrm{da}(A)$ may be infinite.

The *string path width* [6, 3], roughly speaking, counts the number of complete computations and tree width [11] counts the number of partial computations. Formally this is defined as follows. For an NFA A and string w, the *string path width* of A on w, $\mathrm{spw}(A, w)$, is the number of complete computations of A on w and the *tree width* of A on w, $\mathrm{tw}(A, w)$, is the number of partial computations of A on w. Again for $\ell \in \mathbb{N}$, we define $\mathrm{spw}(A, \ell) = \max\{\mathrm{spw}(A, w) \mid w \in \Sigma^\ell\}$ and $\mathrm{tw}(A, \ell) = \max\{\mathrm{tw}(A, w) \mid w \in \Sigma^\ell\}$, as well as, $\mathrm{spw}(A) = \sup_{\ell \in \mathbb{N}}\{\mathrm{tw}(A, \ell)\}$ and $\mathrm{spw}(A) = \sup_{\ell \in \mathbb{N}}\{\mathrm{spw}(A, \ell)\}$.

An NFA's tree width will grow unboundedly with respect to the length of the input string if and only if some cycle has a nondeterministic transition [11]. Figure 1 gives a visual abstraction of this condition.

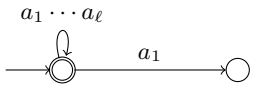


Fig. 1. NFA with Infinite Tree Width

Directly by the definitions, for any NFA A and $\ell \in \mathbb{N}$ we have $\mathrm{da}(A, \ell) \leq \mathrm{spw}(A, \ell) \leq \mathrm{tw}(A, \ell)$. If all states of A are final, then the ambiguity and string path width of A are equal on any string. If A has no undefined transitions, then its string path width and tree width are equal.

The *completion* of an NFA $A = (Q, \Sigma, \delta, q_0, F)$ is obtained from A by adding a sink state that is the target of all previously undefined transitions. Formally, the *completion* of A is $\widehat{A} = (Q', \Sigma, \delta', q_0, F)$, where

$$Q' = \begin{cases} Q, & \text{if } \delta(q, b) \neq \emptyset \text{ for all } q \in Q \text{ and } b \in \Sigma; \\ Q \cup \{q_{\mathrm{sink}}\}, & \text{otherwise,} \end{cases}$$

and the transitions are defined by setting $\delta'(q_{\text{sink}}, b) = \{q_{\text{sink}}\}$, for all $b \in \Sigma$,

$$\delta'(q,b) = \begin{cases} \delta(q,b), & \text{if } \delta(q,b) \neq \emptyset; \\ \{q_{\text{sink}}\}, & \text{if } \delta(q,b) = \emptyset, \end{cases} \quad q \in Q, b \in \Sigma.$$

Directly from the definition of string path width and tree width we get:

Lemma 1. *For any NFA A and string $w \in \Sigma^*$,*

$$\text{spw}(\widehat{A}, w) = \text{tw}(A, w).$$

3 Growth of string path width and tree width

Weber and Seidl [15], based on earlier work, developed structural criteria to determine the growth rate of ambiguity of an NFA as a function of input length. We extend and modify these conditions to study the growth rate of string path width and tree width. In the following $A = (Q, \Sigma, \delta, q_0, F)$ is always an NFA.

3.1 Polynomial Growth

For an NFA A, $\text{da}(A, \ell) \in \Omega(\ell^d)$, $d \in \mathbb{N}$, if and only if A complies with a condition **(IDA$_d$)** [15] which means that A can be viewed to have a "subgraph" of a certain type. We refer to such subgraphs as *widgets*. The widget **(IDA$_d$)** is almost the same as **(ISPW$_d$)** defined below and represented in Figure 2. The only difference is that **(IDA$_d$)** additionally requires that the state c_d must be able to reach a final state.

To characterize polynomial string path width, we use a widget, **(ISPW$_d$)**, modified from **(IDA$_d$)** [15].

(ISPW$_d$): There exist states $s_1, c_1, \ldots, s_d, c_d \in Q$, strings $v_1, \ldots, v_d \in \Sigma^+$, and strings $u_2, \ldots, u_d \in \Sigma^*$ such that:

- For all $1 \leq i \leq d$: $s_i \neq c_i$, and $\text{paths}(s_i, v_i, s_i)$, $\text{paths}(s_i, v_i, c_i)$, and $\text{paths}(c_i, v_i, c_i)$ are all non-empty.
- For all $2 \leq j \leq d$: $\text{paths}(c_{j-1}, u_j, s_j)$ is non-empty.

We call the cycle on s_i a "seeding cycle" and the cycle on c_i a "catching cycle". The states s_i and c_i must be connected by the same string v_i occurring in the cycles. Note that the u_j-strings can be empty, but the v_i-strings cannot. When a string u_j is empty for some $2 \leq j \leq d$, states c_{j-1} and s_j must be the same state. That is, there exists a state which is involved in a cycle on each of the strings v_{j-1} and v_j (which may be the same string).

Lemma 2. *An NFA A has string path width $\Omega(\ell^d)$ if and only if A admits a widget* **(ISPW$_d$)**, *for some $d \in \mathbb{N}$.*

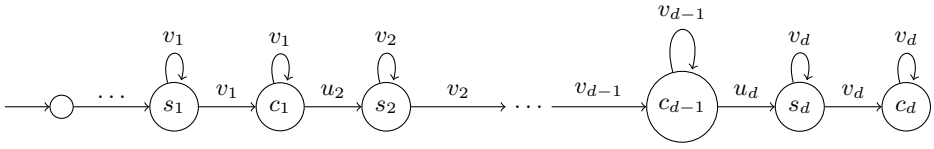


Fig. 2. Widget $\mathbf{(ISPW}_d\mathbf{)}$

Proof. Let A' be obtained from A by making all of its states final. From [15] we know that A' has degree of ambiguity $\Omega(\ell^d)$ iff A' admits the widget $\mathbf{(IDA}_d\mathbf{)}$, and the latter holds iff A admits the widget $\mathbf{(ISPW}_d\mathbf{)}$. For any string w, $\mathrm{da}(A', w) = \mathrm{spw}(A, w)$. □

An NFA's string path width will be finite if and only if it does not admit a widget $\mathbf{(ISPW}_1\mathbf{)}$.

Corollary 1. *For an NFA* A, $\mathrm{spw}(A) \in O(1)$ *iff for all* $q, q' \in Q$ *and* $w \in \Sigma^*$ *such that* $q \neq q'$*:*

(i) $|\mathrm{paths}(q, w, q)| \leq 1$
(ii) $(|\mathrm{paths}(q, w, q)| = 1$ *and* $|\mathrm{paths}(q', w, q')| = 1)$ *implies* $|\mathrm{paths}(q, w, q')| = 0$.

Lemma 2 implies that an NFA A has tree width $\Theta(\ell^d)$ exactly when A admits a widget $\mathbf{(ISPW}_d\mathbf{)}$, but does not admit a widget $\mathbf{(ISPW}_{d+1}\mathbf{)}$.

For an NFA A, the ambiguity and string path width of A on a particular string, or on strings of given length, can be very different, even assuming that A has no useless states. For example, if $\mathrm{spw}(A, \ell) \neq 0$, the string path width of A on all lengths $1, \ldots, \ell - 1$ must be positive. On the other hand, it is possible that $\mathrm{da}(A, \ell) = 0$ and $\mathrm{da}(A, \ell + 1) \neq 0$.

However, as a consequence of Lemma 2 and [15], we see that, for NFAs without useless states, the polynomial growth rates of ambiguity and string path width must coincide. Note that when A has no useless states, an $\mathbf{(ISPW}_d\mathbf{)}$ widget is also an $\mathbf{(IDA}_d\mathbf{)}$ widget.

Corollary 2. *If* A *is an NFA without useless states, then* $\mathrm{spw}(A, \ell) \in \Theta(\ell^d)$ *if and only if* $\mathrm{da}(A, \ell) \in \Theta(\ell^d)$*, for* $d \in \mathbb{N}$.

To characterize polynomial tree width, we define a new widget, $\mathbf{(ITW}_d\mathbf{)}$, which is the same as $\mathbf{(ISPW}_d\mathbf{)}$, except:

(i) For the "last state" c_d we remove the condition that $|\mathrm{paths}(c_d, v_d, c_d)| > 0$. That is, the final "catching loop" on state c_d does not need to be present.
(ii) For the final pair of states we modify the condition $|\mathrm{paths}(s_d, v_d, c_d)| > 0$ to be $|\delta(s_d, a)| > 0$ instead, where a is the first letter of v_d. That is, the state s_d with the final "seeding loop" only needs a nondeterministic transition on the first symbol of v_d.

We give a visual abstraction of these alterations in Figure 3 (cf. Figure 2).

Next we establish the correctness result for our new criterion.

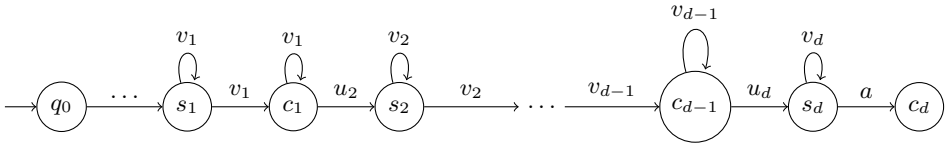


Fig. 3. Widget $\mathbf{(ITW}_d\mathbf{)}$. The letter a must be the first symbol in v_d.

Lemma 3. *An NFA A has tree width $\Omega(\ell^d)$, $d \in \mathbb{N}$, if and only if A admits a widget $\mathbf{(ITW}_d\mathbf{)}$.*

Proof. Suppose that A has tree width $\Omega(\ell^d)$. By Lemma 1, the completion of A, $\widehat{A}$, has string path width $\Omega(\ell^d)$ and hence by Lemma 2, $\widehat{A}$ admits the widget $\mathbf{(ISPW}_d\mathbf{)}$. Since $s_i \neq c_i$ for $i = 1, \ldots, d$, none of the states s_i $1 \leq i \leq d$, or c_j, $1 \leq j \leq d-1$ can be the sink state of $\widehat{A}$ because the only state reachable from the sink state is itself. Thus, in the widget $\mathbf{(ISPW}_d\mathbf{)}$ of $\widehat{A}$, only the state c_d may be the sink state and all transitions "before" entering c_d must exist also in the original NFA A. This means that A admits $\mathbf{(ITW}_d\mathbf{)}$.

Conversely, assume that A admits a widget $\mathbf{(ITW}_d\mathbf{)}$ and let $\widehat{A}$ be the completion of A. Using the notation of Figure 3, denote $v_d = a \cdot v'_d$, $a \in \Sigma$, and choose $p \in \delta(c_d, v'_d)$, that is, p is reached from the state s_d on string v_d and first making the transition to state c_d. If $p = s_d$, the NFA A has exponential tree width (cf. Section 3.2) and we are done. Thus, we can assume $p \neq s_d$. Since $\widehat{A}$ has no undefined transitions, there exist $i, k \geq 1$ and a state r of $\widehat{A}$ such that $r \in \delta'(p, v_d^i)$ and $r \in \delta'(r, v_d^k)$. Choose $0 \leq m < k$ such that

$$1 + i + m \equiv 0 \pmod{k}.$$

In the cycle from r to r on string v_d^k, let t be the state reached by v_d^m. Now the state t can be reached from s_d on string v_d^{1+i+m} and $\text{paths}(t, v_d^k, t) \neq \emptyset$. Since $1 + i + m$ is a multiple of k, the $\mathbf{(ITW}_d\mathbf{)}$ widget can be completed to an $\mathbf{(ISPW}_d\mathbf{)}$ widget as follows:

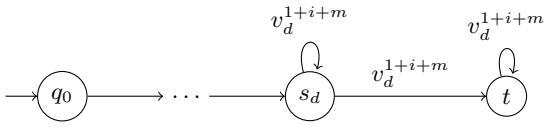


Fig. 4. Completing the $\mathbf{(ITW}_d\mathbf{)}$ widget to a $\mathbf{(ISPW}_d\mathbf{)}$ widget

Thus, by Lemma 2, $\widehat{A}$ has string path width $\Omega(\ell^d)$. By Lemma 1, $\text{tw}(A, \ell) \in \Omega(\ell^d)$. □

Again, Lemma 3 implies that $\text{tw}(A, \ell) \in \Theta(\ell^d)$ exactly when A admits a widget $\mathbf{(ITW}_d\mathbf{)}$ and does not admit a widget $\mathbf{(ITW}_{d+1}\mathbf{)}$.

Intuitively, tree width and string path width can seem very different measures because the former counts all computations while the latter counts only complete computations. As a consequence of Lemmas 2 and 3, we see that the degrees of polynomials bounding the growth rate of, respectively, the tree width and the string path width of an NFA differ by at most one.

Theorem 1. *Let A be an NFA and $d \in \mathbb{N}$.*

(i) If $\mathrm{spw}(A, \ell) \in \Theta(\ell^d)$, *then* $\mathrm{tw}(A, \ell)$ *is in* $\Theta(\ell^d)$ *or in* $\Theta(\ell^{d+1})$.
(ii) If $\mathrm{tw}(A, \ell) \in \Theta(\ell^d)$, *then* $\mathrm{spw}(A, \ell)$ *is in* $\Theta(\ell^d)$ *or in* $\Theta(\ell^{d-1})$.

From the characterization of Lemmas 2 and 3 it follows also that, for all $d \in \mathbb{N}$, there exists an NFA A such that $\mathrm{spw}(A, \ell) \in \Theta(\ell^d)$ and $\mathrm{tw}(A, \ell) \in \Theta(\ell^{d+1})$.

To conclude this section we observe that the criteria **(ISPW$_d$)** and **(ITW$_d$)** yield lower bounds for the number of states or the number of transitions of an NFA having polynomial string path (or tree) width.

Proposition 1. *Let $A = (Q, \Sigma, \delta, q_0, F)$ be an NFA and $d \in \mathbb{N}$.*

(i) If $\mathrm{tw}(A, \ell) \in \Theta(\ell^d)$, *then* $1 \leq d < |Q|$, *and* $2 \cdot d \leq |\delta|$.
(ii) If $\mathrm{tw}(A, \ell) \in \Theta(\ell^d)$, *and* $L(A) = w^*$, $w \in \Sigma^k$, $k \in \mathbb{N}$, *then*

$$1 \leq (d \cdot k) + 1 \leq |Q| \text{ and } d \cdot (k+1) \leq |\delta|.$$

Furthermore, in case (i) (respectively, in case (ii)) there exists an NFA A such that $d = |Q| - 1$ and $2 \cdot d = |\delta|$ (respectively, $d \cdot k + 1 = Q$ and $d \cdot (k+1) = |\delta|$).

3.2 Exponential Growth

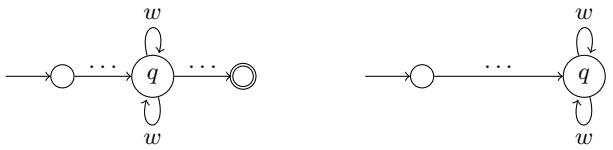
In the following $A = (Q, \Sigma, \delta, q_0, F)$ is again always an NFA. If NFA A complies with **(EDA)** (see Figure 5), then it has exponential ambiguity [15]. This condition requires that for some state q there exist two distinct paths from q to q with the same underlying string.

(EDA): There exists a co-reachable state $q \in Q$ and a string $w \in \Sigma^*$ such that $|\mathrm{paths}(q, w, q)| \geq 2$.

The difference between ambiguity and string path width is that the latter measure does not require computations to be accepting. Based on this observation, we can formulate the widget **(ECOMP)** to characterize exponential string path width, and the correctness of **(ECOMP)** is verified in Lemma 4.

In the definition of the widget **(EDA)** or **(ECOMP)** the string w must have a length of at least two because there can be only one path from q to q on an individual alphabet symbol. Since the cycles must have a length of at least two, it follows immediately that an NFA admitting widget **(ECOMP)** has to admit **(ISPW$_d$)** for all $d \in \mathbb{N}$ (as should be the case).

(ECOMP): There exists a state $q \in Q$ and a string $w \in \Sigma^*$ such that $|\mathrm{paths}(q, w, q)| \geq 2$.



(a) Widget **(EDA)** (b) Widget **(ECOMP)**

Fig. 5. Exponential Nondeterminism Growth Rate Widgets

Lemma 4. *An NFA A satisfies* **(ECOMP)** *if and only if* $\mathrm{spw}(A, \ell) \in 2^{\Theta(\ell)}$.

An NFA has exponential tree width if and only if it has exponential string path width. The "if"-direction of the following proposition is proved by applying Lemma 4 to the completion of the NFA.

Proposition 2. *Let A be an NFA. Then A has exponential tree width if and only if A has exponential string path width.*

4 Algorithms for Deciding the Growth Rate

From Theorem 1, we know that there is a relationship between the polynomial degrees of an NFA's string path width and tree width. An algorithm which decides the polynomial degree of an NFA's string path width or tree width, roughly speaking, also decides the degree of the other measure.

Palioudakis et. al [11] have shown that for an NFA A we can decide in polynomial time whether A has finite tree width, but did not give a more precise time bound for the algorithm.

Theorem 2. *Let $A = (Q, \Sigma, \delta, q_0, F)$ be an NFA. Then we can decide whether or not A has finite tree width in $O(|Q|^4 \cdot |\Sigma|)$ time.*

Proof. For $q \in Q$ and $a \in \Sigma$, we denote $A_{q,a} = (Q, \Sigma, \delta_{q,a}, q, F)$ where the transitions are defined for $p \in Q$, $b \in \Sigma$,

$$\delta_{q,a}(p, b) = \begin{cases} \emptyset, & \text{if } p = q \text{ and } b \neq a; \\ \delta(p, b), & \text{otherwise.} \end{cases} \tag{1}$$

In Algorithm 1, for states q and p of an NFA, the *distance* from q to p is the length of the shortest nonempty string that takes q to p. If p is not reachable from q, then the distance is ∞. The distance from q to itself is ∞ unless q is involved in a cycle.

Complexity analysis of Algorithm 1: Creating each NFA takes $O(|Q| + |\delta|)$ time and space, and creating each distance matrix takes $\Theta(|Q|^3)$ time and $\Theta(|Q|^2)$ space using the Floyd-Warshall reachability algorithm [1]. There are $|Q| \cdot |\Sigma|$ NFAs $A_{q,a}$ and matrices which are created. In lines 8-16, we again check each combination of state and character, but the inner work on lines 10-14 is done

Algorithm 1 Deciding if an NFA has finite tree width

Input: $A = (Q, \Sigma, \delta, q_0, F)$
Output: finite $\in \{\text{true}, \text{false}\}$

for all $q \in Q$ **do**
  for all $a \in \Sigma$ **do**
    Create $A_{q,a} = (Q, \Sigma, \delta_{q,a}, q, F)$ as in (1).
    Create the distance matrix $M_{q,a}[p, p']$, where $M_{q,a}[p, p']$ is the minimum distance from state $p \in Q$ to state $p' \in Q$ in the NFA $A_{q,a}$.
  end for
end for
finite = true
for all $q \in Q$ **do**
  for all $a \in \Sigma$ **do**
    if $M_{q,a}[q, q] \neq \infty$ **then** $\#q$ can reach itself starting with a
      if $|\delta(q, a)| \geq 2$ **then** $\#q$ is a nondeterministic branching point
        finite = false
      end if
    end if
  end for
end for
return finite

in constant time with the help of the reachability matrices. The algorithm then runs in $O(|Q|^4 \cdot |\Sigma|)$ time.

Correctness analysis of Algorithm 1: Recall that an NFA has infinite tree width if and only if some cycle has a nondeterministic transition. The diagonal of each distance matrix gives the self-reachability for each combination of state and character. That is, for all $q \in Q$ and $a \in \Sigma$, $M_{q,a}[q, q] \neq \infty$ means that q can reach itself in A on a string that begins with a. The algorithm returns false ("unbounded tree width") if and only if there exists a state $q \in Q$ and character $a \in \Sigma$ such that $M_{q,a}[q, q] \neq \infty$ and $|\delta(q, a)| \geq 2$. □

Note that while Algorithm 1 could be modified to decide finiteness of string path width, its complexity would be worse than existing algorithms [15]. This is because the algorithm checks only for nondeterministic characters that initiate cycles, and does not check for strings across cycles.

To design efficient decision algorithms for deciding the growth rate of an NFA's tree width, we leverage existing algorithms [9, 15] for deciding the growth rate of ambiguity.

Lemma 5. [modified from [15]] *Let $A = (Q, \Sigma, \delta, q_0, F)$ be an NFA. Then we can decide whether or not A's tree width is exponential in $O(|Q|^4 \cdot |\Sigma|)$ time.*

Using Algorithm 1 and Lemma 5, we can decide whether the growth rate of tree width of A is finite or exponential, and the remaining possibility is polynomial.

Corollary 3. *For an NFA $A = (Q, \Sigma, \delta, q_0, F)$, we can decide whether A's tree width is finite, polynomial, or exponential in $O(|Q|^4 \cdot |\Sigma|)$ time.*

Using another algorithm by Weber and Seidl [15], we can also determine the minimum degree of the polynomial bounding an NFA's tree width.

Proposition 3. [modified from [15]] *For an NFA $A = (Q, \Sigma, \delta, q_0, F)$ we can decide in $O(|Q|^6 \cdot |\Sigma|)$ time whether A's tree width is polynomial, and if so, the degree $d > 0$ such that $tw(A, \ell) \in \Theta(\ell^d)$.*

5 State Complexity

It is known that unambiguous NFAs may be significantly more succint than finite tree width NFAs [11]. Since string path width lies in between the measures of ambiguity and tree width, we initiate here a descriptional complexity comparison of NFAs with finite string path width and finite tree width, respectively. We show that there exist finite string path width NFAs with m states such that an equivalent finite tree width NFA needs close to 2^m states.

Goldstine et al. [2] establishes that for certain regular languages an NFA with finite *branching* has to be as large as a DFA. Instead of the branching measure [2] the below lemma considers tree width.

Lemma 6. [modified from [2]] *Let $L \subseteq \Sigma^*$ be a regular language and $c \notin \Sigma$. If A is a finite tree width NFA for $(cLc)^*$, then A needs as many states as the minimal incomplete DFA for $(cLc)^*$.*

Let $\Sigma = \{a, b, c\}$ and for $m \in \mathbb{N}$ define

$$L_m = (a + b)^* a (a + b)^m.$$

Set $Q_m = \{0, 1, \ldots, m+2\}$ and define $A_m = (Q_m, \Sigma, \delta, 0, \{0\})$ where δ is defined by setting:

(i) $\delta(0, c) = \{1\}$, $\delta(m + 2, c) = \{0\}$,
(ii) $\delta(1, a) = \{1, 2\}$, $\delta(1, b) = \{2\}$,
(iii) $\delta(i, z) = \{i + 1\}$, $z \in \{a, b\}$, $2 \leq i \leq m + 1$.

All transitions not defined above are undefined. Figure 6 depicts the NFA A_m and it is clear that $L(A_m) = (cL_mc)^*$.

Lemma 7. $\mathrm{spw}(A_m) = m + 2$.

It is well known that the minimal DFA B_m for L_m has 2^{m+1} states — the DFA B_m just remembers the positions of symbols a among the last $m+1$ symbols read. Based on B_m it is easy to construct an incomplete DFA C_m for $(cL_mc)^*$ by adding a new start state q_{new} that is the sole accepting state of C_m, a transition on c from q_{new} to the start state of B_m and a transition on c from each accepting state of B_m to q_{new}. The DFA C_m has $2^{m+1} + 1$ states.

Next using Lemma 6 we show that any finite tree width NFA for $(cL_mc)^*$ needs $2^{m+1} + 1$ states.

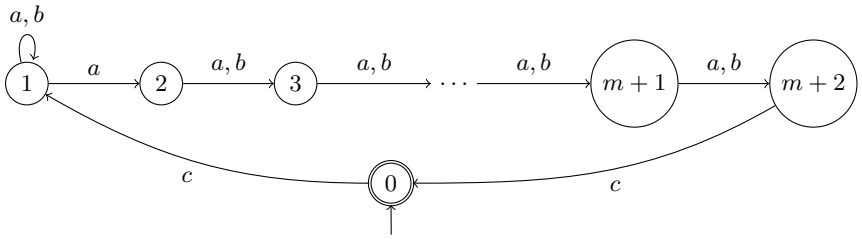


Fig. 6. NFA A_m recognizing the language $(cL_mc)^*$

Lemma 8. *A finite tree width NFA recognizing the language* $(cL_mc)^*$ *needs* $2^{m+1}+1$ *states,* $m \geq 1$.

Proof. We claim that all strings of $K = \{\varepsilon, b\} \cup c \cdot \{a,b\}^{m+1}$ belong to distinct congruence classes of $\equiv_{(cL_mc)^*}$. The empty string is inequivalent to all other strings of K because ε is the only string of K in $(cL_mc)^*$. The string b is inequivalent to all other string of K because it is the only string of K that is not a prefix of any string of $(cL_mc)^*$.

Consider two distinct strings $w_1, w_2 \in c \cdot \{a,b\}^{m+1}$. Since $w_1 \neq w_2$ and both strings have length $m+2$, without loss of generality we can write

$$w_1 = cuav_1,\ w_2 = cubv_2, \quad \text{where} \quad |v_1| = |v_2|, \quad u, v_1, v_2 \in \{a,b\}^*.$$

This means that $w_1 \cdot b^{m-|v_1|}c \in (cL_mc)^*$ and $w_2 \cdot b^{m-|v_1|}c \notin (cL_mc)^*$.

The above means that the minimal complete DFA for $(cL_mc)^*$ has (at least) $2^{m+1}+2$ states, and hence the minimal incomplete DFA for $(cL_mc)^*$ needs $2^{m+1}+1$ states. The claim follows from Lemma 6. □

As a consequence of Lemmas 7 and 8 we have:

Theorem 3. *For* $m \geq 1$ *there exists an* $(m+3)$*-state NFA with finite string path width such that any equivalent finite tree width NFA needs* $2^{m+1}+1$ *states.*

Note that Lemma 6 uses the language $(cLc)^*$, while it seems that the same claim should hold also for $(Lc)^*$ and using the latter language would slightly improve the state complexity blow-up of Theorem 3. However, we have not been able to modify the original construction from [2] (that is needed for Lemma 6) to work with the simpler language. We can note that the NFA A_m is also unambiguous. The size blow-up given by Theorem 3 is slightly worse than the known blow-up of converting an unambiguous NFA to a finite tree width NFA [11].

6 Conclusion

As the main result we have shown that the tree width of an NFA A grows polynomially if and only if the string path width of A is polynomial and the

degrees of the polynomials differ by at most one. Furthermore, if A has no useless states then the degree of the polynomial growth rate of string path width coincides with the degree of the polynomial bounding the ambiguity of A. We have also initiated a descriptional complexity comparison of NFAs with finite string path width and finite tree width, respectively.

A topic for further research could include the descriptional complexity comparison of NFAs with different growth rates of tree width (respectively, string path width). For example, it is known that the size of NFAs with polynomial ambiguity is exponentially separated from the size of general NFAs [9, 5].

References

1. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, 3rd Edition. MIT Press (2009), http://mitpress.mit.edu/books/introduction-algorithms
2. Goldstine, J., Kintala, C.M.R., Wotschke, D.: On measuring nondeterminism in regular languages. Inform. Comput **86**(2), 179–194 (1990)
3. Han, Y.S., Salomaa, A., Salomaa:, K.: Ambiguity, nondeterminism and state complexity of finite automata. Acta Cybern **23**(1), 141–157 (2017)
4. Holzer, M., Kutrib, M.: Descriptional and computational complexity of finite automata - a survey. Inform. Comput **209**(3), 456–470 (2011)
5. 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)
6. Keeler, C., Salomaa, K.: Branching measures and nearly acyclic NFAs. Proceedings of DCFS'17, Lect. Notes Comput. Sci. 10316 pp. 202–213 (2017)
7. Keeler, C., Salomaa, K.: Cycle height of finite automata. Proceedings of DCFS'18, Lect. Notes Comput. Sci. 10952 pp. 200–211 (2018)
8. Kutrib, M., Pighizzini, G.: Recent trends in descriptional complexity of formal languages. Bulletin of the EATCS **111** (2013)
9. Leung, H.: Separating exponentially ambiguous finite automata from polynomially ambiguous finite automata. SIAM J. Comput **27**(4), 1073–1082 (1998)
10. Leung, H.: Descriptional complexity of NFA of different ambiguity. Int. J. Found. Comput. Sci **16**(5), 975–984 (2005)
11. Palioudakis, A., Salomaa, K., Akl, S.G.: State complexity of finite tree width NFAs. Journal of Automata, Languages and Combinatorics **17**(2-4), 245–264 (2012)
12. Palioudakis, A., Salomaa, K., Akl, S.G.: Comparisons between measures of nondeterminism on finite automata. In: Proceedings of DCFS'13, Lect. Notes Comput. Sci. 8031. pp. 217–228 (2013)
13. Ravikumar, B., Ibarra, O.H.: Relating the type of ambiguity of finite automata to the succinctness of their representation. SIAM J. Comput **18**(6), 1263–1282 (1989)
14. Shallit, J.: A Second Course in Formal Languages and Automata Theory. Cambridge University Press (2008), http://cambridge.org/gb/knowledge/isbn/item1173872/
15. Weber, A., Seidl, H.: On the degree of ambiguity of finite automata. Theoret. Comput. Sci. **88**(2), 325–349 (1991)