Unary NFAs with 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 unary finite automata employing limited nondeterminism. We show that for a unary regular language, a minimal finite tree width nondeterministic finite automaton (NFA) can always be found in Chrobak normal form. A similar property holds with respect to other measures of nondeterminism. The latter observation is used to establish relationships between classes of unary regular languages recognized by NFAs of given size where the nondeterminism is limited in various ways. Finally, we show that the branching measure of a unary NFA is always either bounded by a constant or has an exponential growth rate.

Keywords: finite automata, limited nondeterminism, state complexity, unary regular languages.

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. Good general surveys on the topic include [8,9] and as examples of early papers on state complexity of finite automata we mention [17,18,19].

Motivated by the well known exponential trade-off in the NFA (nondeterministic finite automaton) to DFA (deterministic finite automaton) conversion, the literature has considered various ways of quantifying the amount of nondeterminism in finite automata. The *degree of ambiguity* of an NFA refers to the number of accepting computations on a given input [15,23]. The *guessing measure*, roughly, counts the number of advice bits used by an accepting computation on a given input [7,11]. The *branching* of an NFA is the product of the degrees of nondeterministic choices on the best accepting computation [7,14] and the *trace* of an NFA is the corresponding worst-case measure [21]. The *tree width measure* [20] counts the total number of computation paths corresponding to a given input. This measure is called *leaf size* in [10,11], see also [1]. The reader is referred to [6] for more information and references on NFAs employing limited nondeterminism.

With a few exceptions, little is known about the interrelationships of the different nondeterminism measures from a descriptional complexity point of view. Directly based on the definitions it follows that the branching and guessing measure are exponentially related [7] and some further results can be found in [10,11,21]. The size trade-off between NFAs of finite branching and DFAs with multiple initial states has been considered in [13,22].

V. Geffert et al. (Eds.): SOFSEM 2014, LNCS 8327, pp. 443–454, 2014.

In this paper we study the interrelationships of the different nondeterminism measures for the special case of unary NFAs. We show that for a given $k \in \mathbb{N}$ and a unary regular language, a minimal NFA with tree width k or trace k can always be found in Chrobak normal form. An analogous result for unary NFAs with finite ambiguity is known from [12]. The above normal form result is used to show that the state complexity classes defined by bounded tree width and by bounded trace, respectively, coincide in the case of unary regular languages and a similar correspondence, with certain limitations, holds for state complexity classes of unary regular languages defined by bounded ambiguity. The situation is different for the branching measure. In contrast with the measures of tree width, trace and ambiguity, it remains open whether unary NFAs with finite branching could have a normal form with a simple nice structure.

In the literature it is known that the growth rate of the degree of ambiguity and of tree width can be either constant, polynomial or exponential and that the growth rate of the trace measure is always either constant or exponential [16,11,21]. As our main result in Section 4, we show that the branching function of a unary NFA is either constant or grows exponentially, and in the latter case give a lower bound for the exponential growth rate that depends only on the number of states. It remains open whether for an NFA defined over an arbitrary alphabet that has unbounded branching, the branching growth rate is always exponential.

2 Preliminaries

We assume that the reader is familiar with the basic definitions concerning finite automata [24,25] and descriptional complexity [6,9]. Here we just fix some notation needed in the following.

The set of strings, or words, 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 by $\mathbb{N}$. 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 is the initial 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 $\delta(q_0, w) \cap F \neq \emptyset$. An NFA A is called deterministic finite automaton (DFA) if for every state q of A and letter a of the input alphabet of A, the transition function goes to at most one state, i.e. $\#\delta(q, a) \leq 1$. 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. The *size of* A is the number of states of A, i.e. $\text{size}(A) = \#Q$.

A special case of an NFA $A = (Q, \Sigma, \delta, q_0, F)$ is when the alphabet Σ has a unique letter. In this case we call the NFA A *unary* and we omit the alphabet of its tuple notation. Similarly, the transition function δ of an unary NFA has one argument, i.e. $\delta : Q \to 2^Q$. For a unary NFA $A = (Q, \delta, q_0, F)$ over an alphabet

$\Sigma = \{a\}$ we say that the number m is accepted by the NFA A instead of the word a^m, when $a^m \in L(A)$. To avoid confusion between operations on numbers and strings we use the symbols $+, \times, \cup, \cdot$ for the operations addition, multiplication, union, and concatenation respectively.

Every unary regular language L has a period and a preperiod. The period and preperiod of a regular language L are natural numbers m and n_0, respectively, where for all $n > n_0$ we have $n \in L$ if and only if $n + m \in L$.

The minimal size of a DFA (respectively, an NFA) recognizing a regular language is called the state complexity (respectively, the nondeterministic state complexity) of L and denoted $\mathrm{sc}(L)$ (respectively, $\mathrm{nsc}(L)$). 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.

A *computation* of an NFA A from a state s_1 to a state s_2 is a sequence of transitions (q_i, a_i, p_i), $1 \leq i \leq k$, where $q_{i+1} = p_i$, $i = 1, \ldots, k-1$, and $s_1 = q_1$, $s_2 = p_k$. The underlying word of a computation (q_1, a_1, q_2) (q_2, a_2, q_3) $\cdots$ (q_m, a_m, q_{m+1}) is $a_1 a_2 \cdots a_m$. For $x \in \Sigma^*$, $\mathrm{comp}_A(x)$ denotes the set of all computation of A with underlying word x, starting from the initial state of A. We call a computation of A accepting if it starts from the initial state and it finishes at a final state. For $x \in \Sigma^*$, $\mathrm{acc\_comp}_A(x)$ denotes the set of all accepting computations of A with underlying word x.

We say that the computations C and C' on word w are equivalent if C and C' begin in the same state and they both end in the same state.

The *branching of a transition* (q, a, p) of an NFA A, denoted by $\beta_A((q, a, p))$, is the number $\#\delta(q, a)$ and the *branching of a computation* C, denoted by $\beta_A(C)$, is the product of the branching of each transition in C. The *branching of a word* $x \in L(A)$ is the minimum branching among all accepting computations by reading the word x, the branching of a word x is given by the formula $\beta_A(x) = \min\{\beta_A(C) \mid C \in \mathrm{acc\_comp}_A(x)\}$. The branching of an NFA A, denoted by $\beta(A)$, is the maximum branching of A on any string, assuming this quantity is bounded. More details on the branching measure can be found in [7].

We have also considered a worst-case variant of the above measure, so called *trace* [21]. The trace of an NFA A on a string x is the maximum branching among all computations reading the word x (accepting or not). The trace of a word x is given by the formula $\tau_A(x) = \max\{\beta_A(C) \mid C \in \mathrm{comp}_A(y), y \text{ is a prefix of } x\}$ (the prefixes of the word x are in the given formula to emphasize that we include also computation reading only an initial part of the word x). The trace of an NFA A, denoted by $\tau(A)$, is the maximum trace of A on any string, assuming this quantity is bounded.

The computation tree of an NFA A on string w is defined in the natural way and denoted as $T_{A,w}$. The tree width of A on w, $\mathrm{tw}_A(w)$, is the number of leaves of $T_{A,w}$ and the tree width of A, $\mathrm{tw}(A)$ (if it is finite) is the maximum tree width of A on any string w. The formal definitions associated with computation trees and tree width of an NFA can be found in [20,21].[1] The ambiguity of A

[1] Note that the tree width of an NFA is unrelated to the notion of tree width as used in graph theory [2].

on w, $\text{amb}_A(w)$, is the number of accepting leaves of $T_{A,w}$ and the ambiguity of A, $\text{amb}(A)$ (if it is finite) is the maximum ambiguity of A on any string w. Ambiguity is a well studied measure of nondeterminism, more details on ambiguity in NFAs can be found in [6].

Next we want to consider questions that involve the state complexity of classes of NFAs of limited nondeterminism. To formalize such question we have to define the following notation, where $sNFA$ is the set of all NFAs, α is a measure of nondeterminism, and c a constant.

$$nsc_{\alpha \leq c}(L) = \min_{A \in sNFA} \{size(A) \mid L = L(A) \text{ and } \alpha(A) \leq c\}$$

Now the numbers $nsc_{\beta \leq k}(L)$ and $nsc_{\tau \leq k}(L)$ have a meaning. The number $nsc_{\beta \leq k}(L)$ is the size of a smallest NFA A such that $L = L(A)$ and $\beta(A) \leq k$. The number $nsc_{\tau \leq k}(L)$ is the smallest number of states required from an automaton B such that $\tau(B) \leq k$. It is easy to see that $nsc_{\beta \leq k}(L) \leq nsc_{\tau \leq k}(L)$ by the definitions of the measures branching and trace.

Let us remind to the reader the Chrobak normal form [3]. A unary NFA A is in Chrobak normal form if initially the states of A form a 'tail' and later, at the end of the tail, are followed nondeterministically by disjoint deterministic cycles. Note, that the only state with nondeterministic choices is the last state of the tail. Formally, the NFA $M = (Q, \delta, q_0, F)$ is in Chrobak normal form if it has the following properties:

(i) $Q = \{q_0, \ldots, q_{t-1}\} \cup C_1 \cup \cdots \cup C_k$, where $C_i = \{p_{i,0}, p_{i,1}, \ldots, p_{i,y_i-1}\}$ for $i \in \{1, \ldots, k\}$,
(ii) $\delta = \{(q_i, q_{i+1}) \mid 0 \leq i \leq t-2\} \cup \{(q_{t-1}, p_{i,0}) \mid 1 \leq i \leq k\} \cup \{(p_{i,j}, p_{i,j+1}) \mid 1 \leq i \leq k, 1 \leq j \leq y_i - 2\} \cup \{(p_{i,y_{i-1}}, p_{i,0}) \mid 1 \leq i \leq k\}$.

We will use also a more relaxed normal form for unary NFAs which we call *a semi-Chrobak normal form.* A semi-Chrobak normal form NFA consists of a tail and a finite number of disjoint cycles. The only nondeterministic transitions are from the last state of the tail to the cycles, however, as opposed to the usual Chrobak normal form now there may be more than one transition from the last state of the tail to the same cycle. An example of a unary NFA in semi-Chrobak normal can be found in Figure 1.

3 Finite Tree Width and Chrobak Normal Form

For an NFA A in Chrobak normal form it is easy to determine the various nondeterminism measures of A.

Lemma 3.1. *Let A be a Chrobak normal form NFA with k cycles. Then $\beta(A) = \tau(A) = \text{tw}(A) = k$.*

Furthermore, if A is a minimal NFA for $L(A)$ and $m \geq k$, then

$$\text{size}(A) = \text{nsc}_{\beta \leq m}(L(A)) = \text{nsc}_{\tau \leq m}(L(A)) = \text{nsc}_{\text{tw} \leq m}(L(A)) = \text{nsc}(L(A)).$$

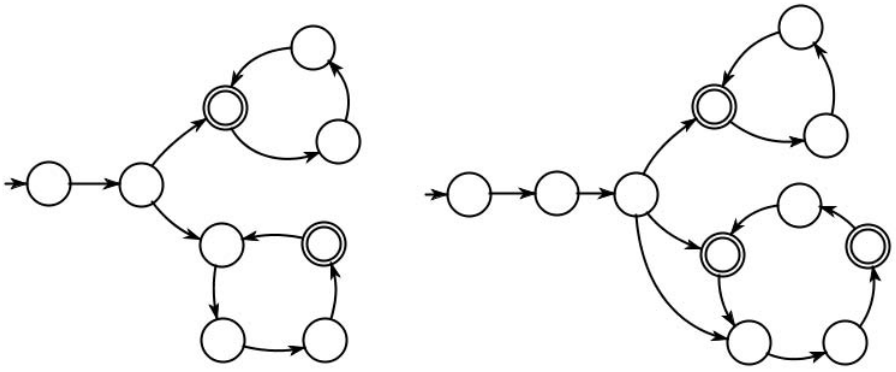

Fig. 1. Two unary NFAs in Chrobak normal form and semi-Chrobak normal form respectively

Proof. The first claim follows directly from the definition of Chrobak normal form. If A is minimal, $\text{size}(A) = \text{nsc}(L(A))$ and the chain of equalities follows because for any $m \geq k$ and $\varphi \in \{\beta, \tau, \text{tw}\}$, $\text{nsc}(L(A)) \leq \text{nsc}_{\varphi \leq m}(L(A)) \leq \text{size}(A)$. □

The semi-Chrobak normal form is a less restrictive variant of the Chrobak normal form. Lemma 3.2 shows that a semi-Chrobak NFA can be transformed to a Chrobak normal form NFA of the same size.

Lemma 3.2. *Every semi-Chrobak normal form NFA has an equivalent Chrobak normal form NFA of same size.*

Proof outline. A semi-Chrobak normal form NFA A with tail T and cycles $C_1, \ldots, C_k$ can be transformed to a Chrobak normal form NFA B with the same tail T and cycles $C_1, \ldots, C_k$. The NFA B has only one transition from the last state of T to each cycle C_i and to compensate for the omitted transitions we add new final states to the cycles. □

Chrobak showed in [3] that every unary NFA can be transformed into an equivalent NFA in Chrobak normal form with losing in efficiency in terms of the number of states. In the following lemma we show that we can transform any NFA with finite tree width into an NFA in Chrobak normal form without losing in efficiency.

Theorem 3.1. *Let A be a unary n-state NFA with tree width k. Then there exists an equivalent Chrobak normal form NFA B with at most n states and tree width k.*

Proof outline. Since the NFA A has finite tree width, we can divide its states into two groups. The states of the first group are the ones belonging in a cycle of A and the second group has the rest. The first group can have only disjoint cycles and its states can have only deterministic choices. We can replace the

states of the second group with a chain (the number of the new states is at most as the number of states in the second group). The tail of the NFA B is made from the new states and its cycles are the cycles of the NFA A. □

Theorem 3.1 speaks about NFAs with finite tree width, not only minimal automata. Sometimes we may use the minimality of an automaton so we want to emphazise that it also holds for minimal automata. We do that with the following corollary.

Corollary 3.1. *For any unary regular language, a state minimal finite tree width NFA is in Chrobak normal form.*

Moreover, Theorem 3.1 suggests a better comparison between NFA with finite tree width and a deterministic finite automaton with multiple initial states (MDFA) [5]. Recall that in [22] we have seen that the size of an MDFA can be exponentially larger than the size of a finite tree width NFA as a function of the degree of its tree width. In the next corollary we show that this is not the case for unary languages.

Corollary 3.2. *Let B be an n-state unary NFA with tree width $k \geq 2$. Then, there is an MDFA B' equivalent with B such that it has at most $k \times n - 5 \times (k-1)$ states.*

Corollary 3.2 gives an upper bound on the size of MDFAs in terms of the size of equivalent NFAs with finite tree width. The size of an MDFA can be linearly more than the size of an equivalent finite tree width NFA. However, note that the limited state complexity of a regular language L for finite tree width k is at most the limited state complexity of MDFAs having k initial states plus one. We can not do better than this, since there are languages that make these quantities equal. Such a language is $L = (p_1)^* \cup \ldots \cup (p_k)^*$, where the numbers p_j are prime, for $1 \leq j \leq k$.

In [21] we have seen that every NFA has finite tree width if and only if it has finite trace. In that paper we have also seen that the trace of an NFA can be as small as its tree width, but the trace can also be exponentially larger than the tree width. In the next corollary, we show that these two measures are equivalent for unary minimal NFAs. Its proof comes from Theorem 3.1 and Lemma 3.1.

Corollary 3.3. *For every unary regular language L and every natural number k, we have the following equality,*

$$\mathrm{nsc}_{\tau \leq k}(L) = \mathrm{nsc}_{\mathrm{tw} \leq k}(L)$$

Moreover, there is an NFA A with tree width k and trace k such that $L = L(A)$ and $size(A) = \mathrm{nsc}_{\tau \leq k}(L) = \mathrm{nsc}_{\mathrm{tw} \leq k}(L)$.

Corollary 3.3 compares minimal NFAs with finite tree width and NFAs with finite trace. The corollary says that the size of a minimal NFA with tree width k is the same as the size of a minimal NFA with trace k, and vice versa. The question here is whether this is true comparing NFAs with finite branching with

NFAs with finite trace or finite tree width. This question seems more difficult since the branching and tree width measures are not comparable, in general. For example take the automaton A of Figure 2, then, for all $m \in \mathbb{N}$, we have that $\mathrm{tw}_A(m) = m + 1$ and $\beta_A(m) = 2$. For the automaton B of the same figure, for all $m \in \mathbb{N}$, we have $\mathrm{tw}_B(m) = m + 1$ and $\beta_B(m) = 2^m$. Both NFAs A and B are minimal for the language $L = \{i \in \mathbb{N} \mid i \geq 1\}$. However, from Corollary 3.3 we have that for bounded branching and bounded tree width we can show an inequality between $\mathrm{nsc}_{\mathrm{tw} \leq k}(L)$ and $\mathrm{nsc}_{\beta \leq k}(L)$. Since for any NFA A, $\beta(A) \leq \tau(A)$, as a consequence of Corollary 3.3 we have Corollary 3.4.

Fig. 2. The NFA A is on the left, the NFA B is on the right

Corollary 3.4. *For every unary regular language L and every natural number k, we have that* $\mathrm{nsc}_{\beta \leq k}(L) \leq \mathrm{nsc}_{\mathrm{tw} \leq k}(L)$

In contrast to Corollary 3.3, the inequality for a unary language L, $\mathrm{nsc}_{\beta \leq k}(L) \leq \mathrm{nsc}_{\mathrm{tw} \leq k}(L)$ of Corollary 3.4 cannot be replaced by an equality. Let A be the unary NFA depicted in Figure 3. On any accepted input, the NFA A needs to go through the first cycle at most two times and, hence, $\beta_A = 4$. On the other hand, it is easy to verify that any NFA with finite tree width for the language $L(A)$ needs at least 6 states.

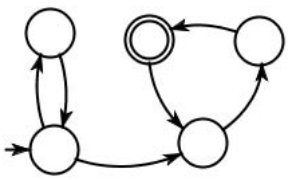

Fig. 3. A unary NFA recognizing the language $2^* \cdot 3^+$

Another consequence of Theorem 3.1 is a connection between finite tree width NFAs and finite ambiguous NFAs. Recall in [20] we have seen that the limited state complexity of finite tree width NFAs can be exponentially larger than the size of unambiguous equivalent NFAs. The next theorem, from [12], is a similar with Theorem 3.1 for finite ambiguity.

Theorem 3.2 ([12]). *Let A be a unary n-state NFA with ambiguity k. There is an equivalent Chrobak normal form NFA B with at most n states and ambiguity k.*

As a result of Theorem 3.1 and Theorem 3.2 we have the following corollary.

Corollary 3.5. *Let A be a unary NFA with n states, for every $k \geq \frac{n}{2}$ we have* $\mathrm{nsc}_{\mathrm{amb}\leq k}(L(A)) = \mathrm{nsc}_{\mathrm{tw}\leq k}(L(A))$.

Our final result for this section shows that there is a strict hierarchy for the state complexity of NFA with finite tree width and respectively with finite trace.

Lemma 3.3. *For any $k \in \mathbb{N}$ there exists a unary regular language L such that*

$$\mathrm{nsc}_{\tau\leq k+1}(L) < nsc_{\tau\leq k}(L) \quad \textit{and} \quad \mathrm{nsc}_{\mathrm{tw}\leq k+1}(L) < nsc_{\mathrm{tw}\leq k}(L).$$

4 Growth Rate of the Branching Measure

In this section we study the growth rate of the branching function for unary automata. We show that the β-function of a unary NFA is either bounded by a constant or grows exponentially.

Before we show our result on the growth rate of the β-function, we will give two lemmas which we are going to use in the proof of the main theorem of this section. The first lemma shows that every computation going through a deterministic cycle S can be transformed into an equivalent computation that repeats cycles outside of S at most a fixed number of times.

Lemma 4.1. *Let A be a unary NFA and consider a computation C of A that contains a deterministic cycle S of length k. Then the computation C has an equivalent computation C' containing the deterministic cycle S, such that every state of A appearing in C' and not in S appears at most k times.*

Proof. Let $A = (Q, \Sigma, q_0, F)$ be a unary NFA and a computation C containing a deterministic cycle S of length k.

Consider a state q appearing in C but not in the cycle S. Let us assume that the state q appears at least $k+1$ times in the computation C, notice here that since cycle S is deterministic if the computation C enter the cycle S then it stays inside the cycle S.

Let d_i be the length of the computation until i-th occurrence of the state q in the computation C, for $i = 1, \ldots, k+1$. Two of the d_i numbers must be in same congruence class modulo k. Then, the steps of the computation C between these two occurrences of q can be shifted into the cycle S.

Continuing similar for all the states of A appearing in the computation C but not in the cycle S, we end up with a computation C', equivalent with the computation C, such that every state before the cycle S appears at most k times.

□

In [3] Chrobak showed that for every unary NFA A with n states, there is a unary NFA A' in Chrobak normal form with n states participating in cycles and $O(n^2)$ states in its tail. For our purposes, we need a more accurate estimation on the size of the tail, which is due to Gawrychowski in [4].

Lemma 4.2 ([4]). *For each unary NFA A with n states, there is a unary NFA A' in Chrobak normal form with at most n states participating in cycles and with a tail with at most $n \times (n-1)$states.*

Now we are ready to give the main result of this section.

Theorem 4.1. *Let A be a unary NFA with n states. Then either for every natural number m, $\beta_A(m) \leq n^{n\times(n-1)}$, or for every natural number $m > n \times (n-1)$,*

$$\beta_A(m) \geq 2^{\lfloor \frac{m}{e\sqrt{n\times\log n}} \rfloor}$$

Proof. Let $A = (Q, \delta, q_0, F)$ be a unary NFA with n states. Let the sets S_N and S_D be

$S_D = \{i \in \mathbb{N} \mid i$ is accepted by a path that enters a deterministic cycle$\}$
$S_N = \{i \in \mathbb{N} \mid i$ is accepted by a path that does not enter a deterministic cycle$\}$

We have that $L(A) = S_D \cup S_N$, note here that the sets S_D and S_N do not need to be disjoint.

Let us have now the NFA $D = (Q, \delta, q_0, F')$ which is exactly like the NFA A except the final states. The final states of D are the final states of A which are in a deterministic cycle. Since, if a computation enters a deterministic cyclic cannot exit this cycle, we have that $L(D) = S_D$. Similarly, we define the NFA $N = (Q, \delta, q_0, F/F')$ which we get by the NFA A by changing the final states appearing in deterministic cycles to non-final states. The NFA N recognizes the language S_N.

Since the unary languages S_N and S_D are regular, they both have a period and a preperiod. From Lemma 4.2 there are unary NFAs N' and D' in Chrobak normal form, respectively equivalent to the NFAs N and D, with tails of size at most $n \times (n-1)$. Then, the number $n \times (n-1)$ is a preperiod for both of the sets S_D and S_N.

Now we are interested in the relationship between S_D and S_N after their preperiod $n \times (n-1)$. To simplify things, we denote $S^{(k)} = \{x \in S \mid x > k\}$. Then, we are interested in the sets $S_N^{(n\times(n-1))}$ and $S_D^{(n\times(n-1))}$. Here we can have two cases, in the first case we have $S_N^{(n\times(n-1))} \subseteq S_D^{(n\times(n-1))}$, and in the second case we have $S_N^{(n\times(n-1))} \setminus S_D^{(n\times(n-1))} \neq \emptyset$.

In the former case, where $S_N^{(n\times(n-1))} \subseteq S_D^{(n\times(n-1))}$, we have that for every number k in $L(A)$ greater than $n \times (n-1)$ there is a computation C_k which enters a deterministic cycle and accepts k. In this case we argue that for every $m \in \mathbb{N}$, we have $\beta_A(m) \leq n^{n\times(n-1)}$. For every computation C with length at most $n \times (n-1)$ the maximum branching that C can have is $n^{n\times(n-1)}$. For

any other accepting computation C with length greater than $n \times (n-1)$ there is an equivalent computation C' which enters in a deterministic cycle. From Lemma 4.1, we can safely assume that the computation C' enters a deterministic cycle and the number of states before this cycle is at most $\frac{(n)^2}{4}$, which implies that the branching of C' is at most $n^{n\times(n-1)}$.

The latter case is a bit more complicated. In this case we have that $S_N^{(n\times(n-1))} \setminus S_D^{(n\times(n-1))} \neq \emptyset$ which means that there is an accepting computation C which does not enter a deterministic cycle and there is not an equivalent computation with C such that enters a deterministic cycle. In the rest of this proof we will argue that since the computation C exists, there are infinitely many such computations and additionally the distance between two consecutive computations is at most $e^{\sqrt{n\times\log n}}$.

From Lemma 4.2 both of the NFAs N' and D', which are in Chrobak normal form, have disjoint cycles with the sum of sizes at most n. Consider the least common multiple of the sizes of the cycles of the NFAs N' and D' combined, denote this least common multiple by z. Then the number z is a period for both of the languages S_D and S_N (which implies also the same for the language $L(A)$). To justify this, consider that there is an accepting computation C_1 which has finished in a final state q inside of one of these cycles, call it S_1, in one of these NFAs (same argument applies to all the cycles of N' and D'). From the definition of the number z the size of the cycle S_1 divides z, then by continuing the computation C_1 after it has reached state q for z more steps, we end up again at the state q. This is true for all final states that are in any cycle of these NFAs. Vice versa, if there is an accepting computation C_2 greater in length than $n\times(n-1)+z$, then it finishes at a final state p which appears in one of these cycles, call it S_2. Considering the computation C_2' which follows computation C_2 and stops exactly z steps before. The computation C_2 is greater that $n\times(n-1)$ which implies that the computation C_2' finishes also in the same cycle S_2. Since the size of S_2 divides z the computation C_2' finishes at the state p as the computation C_2. From the Landau's function we know that z is at most $e^{\sqrt{n\times\log n}}$.

Since the number z is a period for the sets S_N and S_D and there is a number i in $S_N^{(n\times(n-1))} \setminus S_D^{(n\times(n-1))} \neq \emptyset$, we have that for every integer m, such that $i + m \times z > n \times (n-1)$, the number $i + m \times z$ is in S_N but not in S_D. Since we know that $z \leq e^{\sqrt{n\times\log n}}$ we get that for every $e^{\sqrt{n\times\log n}}$ consecutive values (or even more frequently) we have an integer in $S_N \setminus S_D$. The computation on such an integer has at least one nondeterministic step (i.e. the transition function has a choice of at least 2) for every n steps of the computation. We know that the β function is monotone, but we do not know if there are numbers in S_D between two consecutive numbers in $S_N \setminus S_D$. Hence, we have that for every number $m > n \times (n-1)$ the function $\beta_A(m)$ is at least as the function

$$f(m) = \begin{cases} 2^{\frac{m}{n}} & \text{if } m \equiv 0 \bmod \lceil e^{\sqrt{n\times\log n}} \rceil, \\ f(m-1) & \text{otherwise .} \end{cases}$$

It is easy to see that for every value m the function $f(m)$ is at least $2^{\lfloor \frac{m}{e\sqrt{n\times\log n}} \rfloor}$, which implies that the branching $\beta_A(m)$ for every number $m \geq n \times (n-1)$ is at least $2^{\lfloor \frac{m}{e\sqrt{n\times\log n}} \rfloor}$. □

In Theorem 4.1 the values, given as a function of the number of states, are not intended to be the best possible. With a more careful analysis, especially the constant upper bound for $\beta_A(m)$ in the first case, $n^{n\times(n-1)}$, could be significantly improved. It is somewhat less clear whether, in the second case, the factor $e^{\sqrt{n\times\log n}}$ in the exponent can be improved.

5 Conclusion and Open Problems

We have seen that for a unary regular languages, a minimal finite tree width NFA can always be found in Chrobak normal form. It remains open whether there is a similar simple structure for a minimal finite branching unary NFA.

We also studied the growth rate of branching for unary NFAs. We have seen that the branching function of unary NFAs is either bounded by a constant or grows exponentially. The characterization of possible growth rates of the branching function of an NFA defined over an arbitrary alphabet remains open. Here techniques used for our result dealing with the unary case seem not directly applicable.

References

1. Björklund, H., Martens, W.: The tractability frontier for nfa minimization. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 27–38. Springer, Heidelberg (2008)
2. Bondy, J., Murty, U.: Graph theory. Graduate texts in mathematics, vol. 244. Springer (2008)
3. Chrobak, M.: Finite automata and unary languages. Theor. Comput. Sci. 47(3), 149–158 (1986)
4. Gawrychowski, P.: Chrobak normal form revisited, with applications. In: Bouchou-Markhoff, B., Caron, P., Champarnaud, J.-M., Maurel, D. (eds.) CIAA 2011. LNCS, vol. 6807, pp. 142–153. Springer, Heidelberg (2011)
5. Gill, A., Kou, L.T.: Multiple-entry finite automata. J. Comput. Syst. Sci. 9(1), 1–19 (1974)
6. Goldstine, J., Kappes, M., Kintala, C.M.R., Leung, H., Malcher, A., Wotschke, D.: Descriptional complexity of machines with limited resources. J. UCS 8(2), 193–234 (2002)
7. Goldstine, J., Kintala, C.M.R., Wotschke, D.: On measuring nondeterminism in regular languages. Inf. Comput. 86(2), 179–194 (1990)
8. Holzer, M., Kutrib, M.: Nondeterministic finite automata - recent results on the descriptional and computational complexity. Int. J. Found. Comput. Sci. 20(4), 563–580 (2009)
9. Holzer, M., Kutrib, M.: Descriptional and computational complexity of finite automata - a survey. Inf. Comput. 209(3), 456–470 (2011)

10. Hromkovič, J., Karhumäki, J., Klauck, H., Schnitger, G., Seibert, S.: Measures of nondeterminism in finite automata. In: Montanari, U., Rolim, J.D.P., Welzl, E. (eds.) ICALP 2000. LNCS, vol. 1853, pp. 199–210. Springer, Heidelberg (2000)
11. Hromkovic, J., Seibert, S., Karhumäki, J., Klauck, H., Schnitger, G.: Communication complexity method for measuring nondeterminism in finite automata. Inf. Comput. 172(2), 202–217 (2002)
12. Jiang, T., McDowell, E., Ravikumar, B.: The structure and complexity of minimal nfa's over a unary alphabet. In: Biswas, S., Nori, K.V. (eds.) FSTTCS 1991. LNCS, vol. 560, pp. 152–171. Springer, Heidelberg (1991)
13. Kappes, M.: Descriptional complexity of deterministic finite automata with multiple initial states. Journal of Automata, Languages and Combinatorics 5(3), 269–278 (2000)
14. Kintala, C.M.R., Wotschke, D.: Amounts of nondeterminism in finite automata. Acta Inf. 13, 199–204 (1980)
15. Leung, H.: On finite automata with limited nondeterminism. Acta Inf. 35(7), 595–624 (1998)
16. Leung, H.: Separating exponentially ambiguous finite automata from polynomially ambiguous finite automata. SIAM J. Comput. 27(4), 1073–1082 (1998)
17. Lupanov, O.B.: A comparison of two types of finite sources. Problemy Kibernetiki 9, 328–335 (1963)
18. Meyer, A.R., Fischer, M.J.: Economy of description by automata, grammars, and formal systems. In: SWAT (FOCS), pp. 188–191. IEEE Computer Society (1971)
19. Moore, F.R.: On the bounds for state-set size in the proofs of equivalence between deterministic, nondeterministic, and two-way finite automata. IEEE Transactions on Computers C-20(10), 1211–1214 (1971)
20. Palioudakis, A., Salomaa, K., Akl, S.G.: State complexity and limited nondeterminism. In: Kutrib, M., Moreira, N., Reis, R. (eds.) DCFS 2012. LNCS, vol. 7386, pp. 252–265. Springer, Heidelberg (2012)
21. Palioudakis, A., Salomaa, K., Akl, S.G.: Comparisons between measures of nondeterminism on finite automata. In: Jurgensen, H., Reis, R. (eds.) DCFS 2013. LNCS, vol. 8031, pp. 217–228. Springer, Heidelberg (2013)
22. Palioudakis, A., Salomaa, K., Akl, S.G.: Finite nondeterminism vs. dfas with multiple initial states. In: Jurgensen, H., Reis, R. (eds.) DCFS 2013. LNCS, vol. 8031, pp. 229–240. Springer, Heidelberg (2013)
23. 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)
24. Shallit, J.O.: A Second Course in Formal Languages and Automata Theory. Cambridge University Press (2008)
25. Yu, S.: Regular Languages. In: Handbook of Formal Languages, vol. 1, pp. 41–110. Springer (1998)