Technical Report 2012-596

Nondeterministic State Complexity and Quantifying Non-Determinism in Finite Automata

Alexandros Palioudakis

October 30, 2012
School of Computing
Queen's University
Kingston, ON, Canada.

Contents

1 Introduction

Nondeterminism plays a profound role in the theory of computation. Even more generally, some of the most interesting problems in computer science arise from nondeterminism, some examples are P vs. NP, NP vs. co-Np, NP vs. PP and many others. Nondeterminism enhances the power of some computational models, like pushdown automata. In others, such as Turing machines and finite automata, nondeterminism does not enhance computational power but may increase efficiency. For Turing machines, the relation between determinism and nondeterminism can lead to questions about the difference between computational complexity of these different models of computation. The computational complexity of Turing machines is normally measured in time complexity. In the world of automata theory, a time measure makes no sense. Hence, the computational complexity in finite automata is measured by descriptional complexity.

The area of descriptional complexity of finite automata investigates the sizes of descriptions of automata [15], deterministic or not. Most commonly, descriptional complexity measures the number of states, also another obvious measure could be the number of transitions. In other words, for a given regular language we try to minimize the number of states for deterministic and nondeterministic automata. We know that nondeterminism can produce as much as an exponential savings [39, 40]. Furthermore, in the deterministic case we have the answer from the Myhill-Nerode theorem [18], but in the nondeterministic case we don't have a clear solution. We only have some techniques that can work in some cases, but not always. These techniques are the fooling set [9, 15], the extended fooling set [3, 20, 15] and the biclique edge cover technique [13, 15]. It seems impossible to have a clear way of finding a minimal nondeterministic finite automaton, since there can be multiple non-isomorphic minimum finite machines for the same regular language. Additionally, the problem of finding a minimal state nondeterministic automaton equivalent to a given deterministic automaton is PSPACE-hard [25].

In order to study more closely the trade off of nondeterminism against state complexity, it seems appropriate to treat nondeterminism as just one more resource by quantifying it [29]. This quantification of nondeterminism varies among authors. In [11], the authors suggest the spectrum of a regular language. This way of quantifying nondeterminism is by multiplying the number of nondeterministic choices that a machine does as it reads a word. The authors of [23] consider various measures for quantifying nondeterminism. They give the "advice" of a word, which is the number of nondeterministic choices during reading this word. The leaf-size of a word is the number of leaves in the computation tree of the automaton and the given word. The ambiguity of a word is the number of accepting leaves in the computation tree of the automaton and the given word.

In this depth paper we survey the above work. The paper is divided mainly into two sections. In the first section we deal with important results, in our opinion, in nondeterministic state complexity and we give more emphasis in the techniques that we have until now for giving lower bounds in terms of state complexity of nondeterministic finite automata. In the other section we consider different measures of quantifying nondeterminism in finite automata and we give a review of them. Additionally, we consider open problems and di-

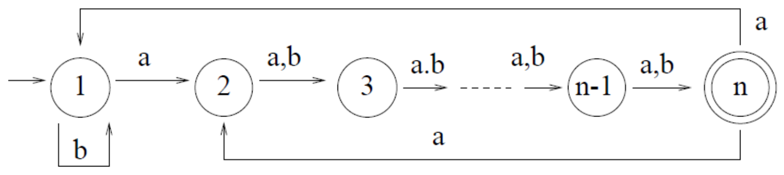

Figure 1: An NFA from [40].

rections for future work. These open problems deal, in particular, with comparing different ways of measuring nondeterminism from the point of view of descriptional complexity. Also, the state complexity of language operations for limited nondeterminism has not been investigated. In other words, it is an interesting problem to give upper and lower bounds for the state complexity of a language resulting from a regularity preserving operation as a function of the state complexity of the argument languages.

2 Lower Bound Techniques on Nondeterministic Finite Automata

In this section of the paper we review some main points, in our opinion, of the basic results on state complexity of nondeterministic finite automata. Moreover, we divide this section into two subsections. In the first subsection we discuss the relation between the the number of states of a DFA with an equivalent NFA. Moreover, we give some examples which make the upper bound tight as shown in bibliography. We also discuss the algorithmic problem of finding, for a given automaton A, an equivalent automaton B with a minimal number of states. This is called the minimization problem. We also mention results for other types of finite automata, as an example we mention results on multiple entry deterministic finite automata. In the second subsection we review techniques which give lower bounds for the number of states for NFAs.

2.1 Basic Results on State Complexity of Nondeterministic Finite Automata

Nondeterministic finite automata (NFAs) were introduced in [44], where their equivalence to deterministic finite automata (DFAs) was shown. This equivalence is established with the very well known subset construction [18]. The subset construction transforms an NFA with n states into a DFA with 2^n states. It is well known that this bound can be tight [36, 39, 40]. Some examples are in Figure 1, in Figure 2, and in Figure 3.

For the particular case of unary languages the same question was raised in [49]. The question is what is the relation between the state complexity of NFAs and DFAs in the case that we have a unary alphabet. Chrobak in [6, 7] presents the asymptotic tight bound.

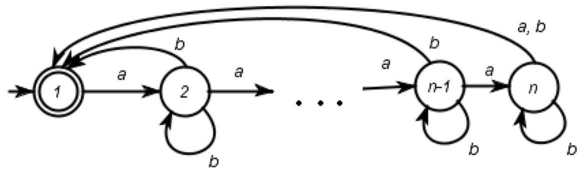

Figure 2: Another NFA from [39].

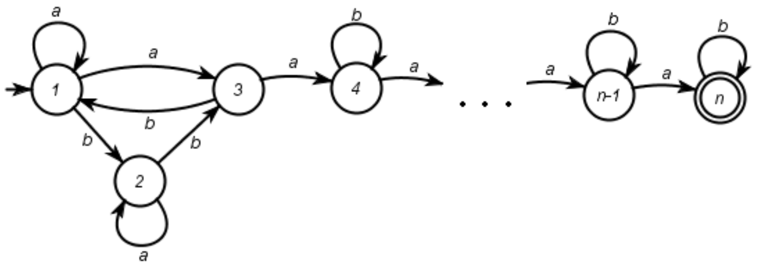

Figure 3: Another NFA from [36].

The bound is that for $n \geq 1$ and A be an n-state NFA accepting a unary language, then $e\Theta(\sqrt{n \cdot \ln n})$ states are sufficient and necessary in the worst case for a DFA to accept $L(A)$.

Another relative question about state complexity of nondeterministic finite automata is given an automaton can we find an equivalent automaton which is minimal from the state complexity point of view. The answer to the same question for deterministic finite automata is given by the Myhill-Nerode relation and it is an easy problem. Every DFA with n states can be converted to a minimal equivalent DFA in $O(n \log n)$ time [17]. Moreover, in the case of deterministic finite automata this minimal automaton is unique. But, this situation is not equally simple in the case of nondeterministic finite automata. Let us see at the Figure 4, from [1], there are two NFAs which both recognize the language $\{ab, ac, bc, ba, ca, cb\}$. Although, these automata recognize the same language, there is no isomorphism between them. Moreover, the problem of finding a minimal state nondeterministic automaton equivalent to a given deterministic automaton is PSPACE-hard [25].

Furthermore, minimizing NFAs with very strong restrictions on the amount of nondeterminism remains NP-complete [4, 5]. Methods for quantifying the amount of nondeterminism will be discussed in section 3.

Given all the above, we understand that it is useful to have techniques which give us lower bounds in the case of nondeterministic finite automata. In other words, given a regular language following some properties, can we prove that every NFA recognizing this language has at least a specific number of states? We will deal with this question in the following subsection. But, before we end this subsection let us mention that there are questions that we could ask here.

We also mention some results concerning the size blow up occurring in restricted variants

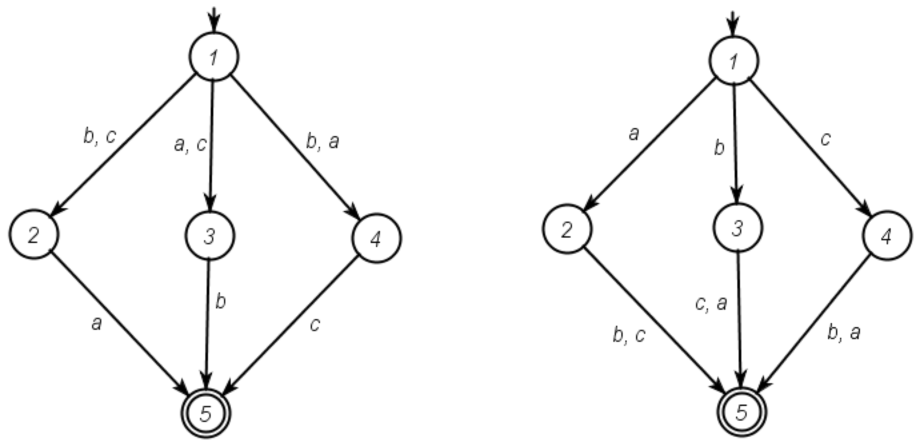

Figure 4: Two NFAs accepting the same language but there is no isomorphism between them.

of NFAs. Firstly, we concentrate in the question of what is the relation of the number of states between DFAs and NFAs when we deal only with finite languages. This question has been considered in [38, 46, 47]. In [38] Mandl considers the case of finite languages over a two letter alphabet, where he showed that for each n-state NFA accepting a finite language there exists an equivalent DFA which has $O(2^{n/2})$ states; more specifically, no more than $2^{\frac{n}{2}+1}-1$ states if n is even and $3 \cdot 2^{\lfloor \frac{n}{2} \rfloor}-1$ states if n is odd. In [47], they show that $O(2^{n/2})$ is the worst-case optimal upper bound on the number of states of a DFA that is equivalent to an n-state NFA accepting a finite language over an arbitrary k-letter alphabet, where they generalize the previous result on binary alphabet. They show that, for any n-state NFA accepting a finite language over an arbitrary k-letter alphabet, $n, k > 1$, there is an equivalent DFA of $O(k^{n/(\log k+l)})$ states, and they show that this bound is optimal in the worst case.

Finally, we mention results on another case of restricted variants of NFAs. Here, we consider deterministic finite automata with multiple initial states (MDFA) [8, 51, 28]. An k-entry deterministic finite automaton is a deterministic finite automaton with the difference that it has k initial states. In other words there is a nondeterministic step, which is to choose the initial state. In [16], they show that for every k-entry DFA M, with n states, there is a DFA M' with at most

$$\sum_{i=1}^{k} \binom{n}{i}$$

states such that $L(M) = L(M')$. In the same paper they show that this bound can be tight. The automaton which makes it tight is the following. Let $k \leq n$, consider the k-entry DFA $M_{l,n}$ with state set $Q = \{1, \ldots, n\}$, where 1 up to k are the initial states, and state n is the only final state. Let a and b be two distinct input symbols. Set $\delta(i, a) = i + 1$ for $1 \leq i < n$ and $\delta(n, a) = 1$, whereas $\delta(i, b) = i$ if $1 \leq i < n$, and $\delta(n, b) = 1$. In Figure 5 there is the automaton $M_{k,n}$ for $k = 3$ and $n = 4$.

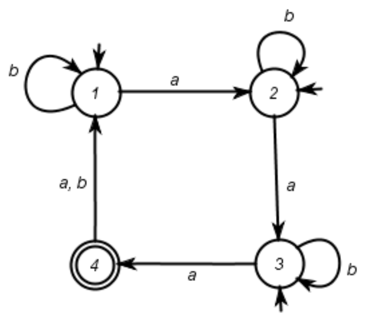

Figure 5: The 3-entry deterministic finite automaton $M_{3,4}$.

2.2 Lower Bound Techniques for Nondeterministic Finite Automata

As we have already mentioned in the previous subsection, the problem of minimizing a DFA is easy. However, it is not the same case when it comes to NFAs. We have already seen in Figure 4 that there are more than one state minimal NFAs recognizing the same language. Also, we have already seen that finding a minimal NFA from a DFA is PSPACE-hard [25]. Then, it is useful to look at techniques which tell us how many states at least an NFA for a given regular language requires. In other words, given a regular language we want to know what is the lower limit for any NFA which recognize this language.

In this direction there are three techniques. These techniques are the fooling set [9, 15], the extended fooling set [3, 20, 15] and the biclique edge cover technique [13, 15]. Each of them are an extension of the previous one.

The first two techniques are simple to explain and to use. The biclique edge cover technique is more complicated. All of these techniques use a set of pairs of strings. When there is a set with n such pairs which they have some properties, which depend on the regular language L, then we get that any NFA recognizing L has at least n states. Let us see the theorems which give us these results.

Theorem 1 (Fooling Set Technique). *Let $L \subseteq \Sigma^*$ be a regular language, and suppose there exists a set of pairs $P = \{(x_i, y_i) \mid 1 \leq i \leq n\}$ such that:*

(a) $x_i y_i \in L$ for $1 \leq i \leq n$;

(b) $x_i y_j \notin L$ for $1 \leq i, j \leq n$ and $i \neq j$.

Then any NFA accepting L has at least n states.

Before we go to the next technique let us give some examples to help us understand this technique better. The following examples are from [9].

Example 1. *Let us have the language $L_k = \{0^i 1^i 2^i \mid 0 \leq i < k\}$, and let us take the set of string pairs $P = \{(0^i 1^j, 1^{i-j} 2^i) \mid 0 \leq j \leq i < k\}$. Here we notice that for every $(x, w) \in P$*

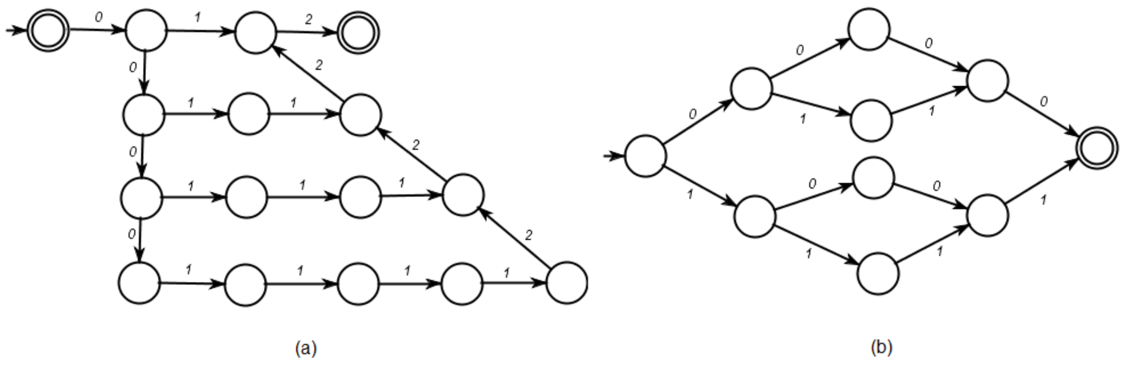

Figure 6: Two NFAs where the fooling set technique works well.

the word xw is in L_k. Moreover, if $(x, w) \in P$ and $(x', w') \in P$ for some $x \neq x'$ and $w \neq w'$, then the word xw' is not in L_k and neither the word $x'w$. The set of string pairs P has $k(k+1)/2$ elements, then from the fooling set technique theorem we have that for every NFA recognizing L_k has at least $k(k+1)/2$ states. In fact, L_k can be recognized by an NFA with $k(k+1)/2+1$ states. In Figure 6(a) we give an NFA for L_k for $k = 5$.

Example 2. *Let us have the language of palindromes of length k over a binary alphabet $A_k = \{w \in \{0, l\}^k \mid w = w^R\}$.*[1] *Let us take the set of string pairs $P = \{(x, 0^{k-2|x|}x^R) \mid |x| \leq k/2\} \cup \{(x0^{k-2|x|}, x^R) \mid |x| \leq (k-1)/2\}$. Then, the smallest NFA accepting A_k has at least $2^{\lfloor k/2 \rfloor+1} + 2^{\lfloor (k+1)/2 \rfloor} - 2$ states. In fact, this bound is tight, as we can construct an NFA with the given number of states that accepts A_k. In Figure 6(b) we give an NFA for A_k for $k = 4$.*

Of course there are cases where the fooling set technique does not work well. In [9] they give the language $H_m = \overline{(0^m)^+}$ where there is no set of string pairs that has the properties of Theorem 1 and also has more than 2 elements. Moreover, they argue that every NFA for H_m has at least $\log(m+1)$ states.

Theorem 2 (Extended Fooling Set Technique)**.** *Let $L \subseteq \Sigma^*$ be a regular language, and suppose there exists a set of pairs $P = \{(x_i, y_i) \mid 1 \leq i \leq n\}$ such that:*

(a) $x_i y_i \in L$ for $1 \leq i \leq n$;

(b) $x_i y_j \notin L$ or $x_j y_i \notin L$ for $1 \leq i, j \leq n$ and $i \neq j$.

Then any NFA accepting L has at least n states.

Now, let us give some examples to makes us understand better the extending fooling set technique. In the second example [13] we compare the two previous techniques and we show that the extending fooling set technique can be arbitrary bigger that the fooling set technique. More specific, we give a family of languages that the biggest fooling set for them can have size at most 3 but there are extending fooling sets with size $n+2$, for $n \geq 3$.

[1]The word w^R denotes the reverse of the word w.

Example 3. *Let L be the regular language recognized be the automaton in Figure 3. We notice that the set of string pairs $P = \{(bba^i, a^{n-3-i}) \mid 0 \leq i \leq n-3\} \cup \{(\epsilon, b^2a^{n-3}), (b, ba^{n-3})\}$ is an extending fooling set for L. The set P has n elements, since we already have an automaton with n states we know that this set in a maximal extending fooling set and that the technique works well in this case. We can also notice that the set P is a fooling set as well.*

Example 4. *Let $\Sigma = \{a_i \mid 1 \leq i \leq n\}$. Consider the finite language $L_n = \{a_ia_j \mid 1 \leq i \leq j \leq n\}$. It is easy to see that $S_n = \{(a_i, a_i) \mid 1 \leq i \leq n\} \cup \{(\epsilon, a_1a_1), (a_1a_1, \epsilon)\}$ is an extended fooling set for L_n of size $n+2$. Then the analysis that any fooling set for L_n has size at most 3 goes as follows: (i) First one observes, that any fooling set can only contain at most one pair of the form (ϵ, w) and (w, ϵ). Then (ii) no two pairs of the form (a_i, a_j) and (a_k, a_l) with $1 \leq i \leq j \leq n$ and $1 \leq k \leq l \leq n$ can be members of any fooling set for L_n. Without loss of generality assume that $i \leq k$, then $a_ia_l \in L_n$, which contradicts the fooling set properties. Hence, any fooling set for L_n can have at most 3 elements. This bound is tight, which is seen by the fooling set $S = \{(\epsilon, a_1a_1), (a_1a_1, \epsilon), (a_1, a_1)\}$ for the language L_n. Moreover, the bound of the extending fooling set in tight since there is a DFA with $n+2$ states which recognizes L_n.*

Here we also notice that there are cases where the extending fooling set technique does not work well either. In [13] they give the language $H_m = \overline{(0^m)^+}$ where there is no set of string pairs that has the properties of Theorem 2 and also has more than 3 elements. Moreover, they argue that every NFA for H_m has at least $\log(m+1)$ states.

From the previous two theorems we have the fooling set technique and the extended fooling set technique, respectively. The last technique needs some more explanations before we can state the corresponding theorem. We have to remind ourselves some graph theoretic notions.

A bipartite graph is a 3-tuple $G = (X, Y, E)$, where X and Y are the (not necessarily finite) sets of vertices, and $E \subseteq X \times Y$ is the set of edges. A bipartite graph $H = (X', Y', E')$ is a subgraph of G, if $X' \subseteq X$, $Y' \subseteq Y$, and $E' \subseteq E$. The subgraph H' is *induced* if $E' = (X' \times Y') \cap E$. Given a set of edges E', the subgraph induced by E' with respect to E is the smallest induced subgraph containing all edges in E'.

Now in [13] they relate pairs of strings with bipartite graphs. This relation happens quite natural. Let us have any sets $X, Y \subseteq \Sigma^*$ and any language $L \subseteq \Sigma^*$, then we associate them with the bipartite graph $G = (X, Y, E_L)$ where the set of edges E_L is defined by $(x, y) \in E_L$ if $xy \in L$, for every $x \in X$ and $y \in Y$.

For the lower bound technique to come we need the notion of a biclique edge cover for bipartite graphs. Let $G = (X, Y, E)$ be a bipartite graph. A set $C = \{H_1, H_2, \ldots\}$ of non-empty bipartite subgraphs of G is an edge cover of G, if every edge in G is present in at least one subgraph. An edge cover C of the bipartite graph G is a biclique edge cover if every subgraph in C is a biclique, where a biclique is a bipartite graph $H = (X, Y, E)$ satisfying $E = X \times Y$. The bipartite dimension of G is denoted $d(G)$ and is defined to be the size of the smallest biclique edge cover of G if it exists and is infinite otherwise. Then the biclique edge cover technique reads as follows:

Theorem 3 (Biclique Edge Cover Technique)**.** *Let $L \subseteq \Sigma^*$ be a regular language and suppose there exists a bipartite graph $G = (X, Y, E_L)$ with $X, Y \subseteq \Sigma^*$ (not necessarily finite) for the language L. Then any non-deterministic finite automaton accepting L has at least as many states as the bipartite dimension of G, i.e., $nsc(L) \geq d(G)$.*

Corollary 1. *Let $L \subseteq \Sigma^*$ be a regular language. Then the bipartite graph $G = (\Sigma^*, \Sigma^*, E_L)$ has finite bipartite dimension.*

Similar with the previous lower bound techniques we are giving an example to help us understand better the concept of biclique edge cover technique. This example [13] will be about the language $H_m = \overline{(0^m)^+}$ where we have seen before that the other techniques do not work. Now by having that the biclique edge cover technique is more powerfull than the fooling set technique or the extended fooling set technique [13], the following example will also illustrate that the biclique edge cover technique is strictly more powerful than the other two techniques.

Before we go to the next example let us recall the notion of an induced matching. A matching in a graph is a set of edges without any common vertices. An induced matching M of a graph $G = (V, E)$ is a set of edges $M \subseteq E$ such that M is a matching and no two edges of M are joined by an edge of G.

Example 5. *Let us have the language $H_m = \overline{(0^m)^+}$ and the bipartite graph $G_{H_m} = (X, Y, E)$, where $X = \{0^k \mid 1 \leq k \leq m\}$, $Y = \{0^{m-k} \mid 0 \leq k \leq m-1\}$, and we define $(0^i, 0^j) \in E$ if $i + j \neq 0 \mod m$. We denote by $\overline{G_{H_m}} = (X, Y, (X \times Y)/E)$ the complement of G_{H_m} with respect to the edge set E.*

We observe that $\overline{G}_{H_m}$ is an induced matching with m edges. The bipartite dimension of graphs with the property that their complement with respect to the edge set is an induced matching with m edges was determined in [2]. It was shown that it is equal to h, where h is the smallest integer such that $m \leq \binom{h}{\lfloor \frac{h}{2} \rfloor}$. Hence, we have that the biclique edge cover technique tells us that any NFA for H_m has at least h states. Moreover, h increases as m increases, then h cannot be bounded from any constant as it is in the case of extending fooling set.

In the discussion above we compared the fooling set and extending fooling set techniques with the state complexity of NFAs. We saw that there are cases where the bounds, which these two techniques give us, are bounded by a constant but the size of the corresponding NFAs increases. Now we would expect the comparison between bipartite dimension and nondeterministic state complexity. The following theorem implies that the lower bounds given by the biclique edge cover technique for an infinite sequence of regular languages cannot be upper bounded by a constant.

Theorem 4. *Let $L \subseteq S^*$ be a regular language and the bipartite graph $G = (\Sigma^*, \Sigma^*, E_L)$. Then $2^{d(G)}$ is greater or equal to the deterministic state complexity of L, i.e., $2^{d(G)} \geq sc(L)$.*

From [13] we have that finding the properties that each of these techniques requires in order to give lower bounds are difficult, even if these techniques some times do not give tight bounds compared with the real ones. Here we have to transform these techniques into problems, which is done by the natural way. The problems are given a deterministic finite automaton A and a natural number k in binary, i.e., encoding $< A, k >$, is there a corresponding set S for the language $L(A)$ of size at least k? Then we have the following results. The fooling set problem is NP-hard and contained in PSPACE. The extended fooling set problem and the biclique edge cover problem are PSPACE-complete.

2.3 Communication Complexity and Finite Automata

Yao [54] introduced communication complexity which is a very well studied sub-area of complexity theory. It is closely related to other complexity measures of fundamental computational models (e.g., Boolean circuits, VLSI circuits, branching programs, Turing machines, etc.). Here we take a glimpse of its connection to the model of finite automata.

Alice and Bob want to calculate a Boolean function. Neither Alice or Bob have the whole input. Alice has the half one and Bob knows the other half. The only thing that they can do is to send messages to each other until they have enough information for calculating the function. At each time they can send only one bit, 0 or 1, through the communication channel which are using. What we are interested in is the amount of information that is necessary for them to send before they can know the output of the Boolean function. This required amount of information is the communication complexity of the Boolean function.

Before we give a more formal definition let us give an example, hopefully to illustrate better this concept. Let's give to Bob the number i, and let's give to Alice the number j, both of the numbers are in binary. Now, the function they want to calculate is $f(i, j) = i + j \mod 2$. It is sufficient for Bob to sent just the last bit of the binary number i and Alice will be able to calculate the result of $i + j \mod 2$. In this case, it is easy to see that the communication complexity of the function f is just one.

Let $M = \{0, 1, 2, \ldots, m - 1\}$, $N = \{0, 1, 2, \ldots, n - 1\}$, and $f : M \times N \to \{0, 1\}$. Let $i \in M$, $j \in N$ the numbers known to Alice and Bod respectively. At each time Alice first sends a bit $a_k \in \{0, 1\}$ to Bob and Bob respond with a bit $b_k \in \{0, 1\}$. Cooperatively, they try determine the value of $f(i, j)$. Alice starts by sending a_1 to Bob, Bob knowing j and a_1 calculates b_1 and he sends it to Alice. Alice now knows i and b_1, then she calculates and sends a_2. The calculation continues similarly until one of them calculates the answer. Precisely, an algorithm P specifies the Boolean functions $\{h_k(i, u_1, u_2, \ldots, u_{k-1}), l_k(j, v_1, v_2, \ldots, v_k) \mid k = 0, 1, 2, \ldots\}$ and the bits $a_k = h_k(i, b_1, b_2, \ldots, b_{k-1})$, $b_k = l_k(j, a_1, a_2, \ldots, a_k)$. Moreover the algorithm P is also called a protocol. The cost $\alpha(P)$ is defined to be the maximum number of bits exchanged for any $i \in M, j \in N$. The two-way communication complexity of the function f is defined as:

$$cc(f) = \min\{\alpha(P) \mid \text{the protocol } P \text{ computes } f\}$$

Let us give another example to illustrate the previous concept[54]. Let $M = \{0, 1, 2, 3, 4\}$, $N = \{0, 1, 2\}$, and the Boolean function f which is given from the following table:

(i,j)	0	1	2
0	0	1	0
1	0	0	0
2	1	1	1
3	0	0	1
4	1	1	0

Let the protocol P be the following instructions. If i is 1 or 2, Alice would know the outcome of $f(i,j)$ and she halts. Now, she splits the rest possible inputs into two groups, $\{0,3\}$ and $\{4\}$. She sends 0 if $i=0$ or $i=3$ and she sends 1 if $i=4$. When Bob receives $a_1=1$ from Alice, he already knows the output $f(i,j)$. When he receives $a_1=0$ from Alice, he cannot guess the output yet. Then, Bob has three possible inputs, $0,1,2$, if $j=0$ he knows the output and he stops, if $j=1$ he sends $b_1=1$, and in the last case where $j=2$ he sends $b_1=0$. It is easy to see that when $(i,j)=(4,1)$, it takes just two messages for Alice and Bob to calculate the output, they send $a_1=1$ and Bob send the solution to Alice. When $(i,j)=(3,2)$, Alice sends first a 0, Bob replies with 0, and Alice send back the output, which is 1. Note that the cost of the protocol P $\alpha(P$ is 3 as we need 3 messages when $i=3$ and $j=2$.

We notice from the previous example that we can represent the initial problem in the form of a Boolean matrix, also called communication matrix, and as we proceed with the calculation we restrict on smaller and smaller submatrices. Further study of the very interesting area of communication complexity goes outside of the purpose of this paper. For the reader interested in further study of communication complexity we suggest [20, 30, 31].

Without getting into more details we also mention about nondeterministic communication complexity and its relation to some combinatorial properties of $M(f)$. A nondeterministic protocol P computing a finite function $f: U \times V \to \{0,1\}$ consists of two nondeterministic parties A and B that have a nondeterministic choice from a finite number of messages for every input argument. For any input $(a,b) \in U \times V$, we say that P computes 1 (or that P accepts (a,b)) if there exists a computation of P on (a,b) that ends with the result 1. So, P computes 0 for an input (a,b) if all computations of P on (a,b) end with the result 0. The nondeterministic communication complexity of P, denoted $ncc(P)$, is the maximum of the communication complexities of all accepting computations of P. The nondeterministic communication complexity of f is the following:

$$ncc(f) = \ min\{ncc(P) \mid P \text{ is a nondeterministic protocol computing } f\}$$

Note here that the last two techniques, the communication complexity technique and the biclique edge cover technique, are different representations of the same concept. The communication complexity approach is in the language of Boolean matrices and the biclique edge cover approach is in the language of graphs.

Now, we can establish a connection between communication complexity and finite automata as follows. We represent regular languages as communication matrices of infinite size. For an alphabet Σ, the communication matrix of a language L is the infinite Boolean matrix $M(L) = \{\alpha_{x,y}\}$ for $x \in \Sigma^*$, $y \in \Sigma^*$, where $\alpha_{x,y} = 1$ if and only if $xy \in L$. Since every regular language has a finite index, form Myhill-Nerode theorem, the number of different rows of $M(L)$ is finite. Then, we can use the techniques from communication complexity area. We have results on this direction by [19, 20] which say that for any n-state automaton A recognizing the language L, if A is a DFA we have that $n \geq 2^{cc(L)}$ and if A is an NFA then $n \geq 2^{ncc(L)}$.

We conclude this section by considering other directions for further research. An obvious question is if we can find other techniques for giving lowers bounds for the state complexity of NFAs. As we know there are only the techniques which were discussed above. Another question is to find other similar techniques when we have NFAs with limited resources. In this direction in [43], we give a lower bound technique for NFAs with limited tree width. Moreover, in the same direction Hromkovic et. al. in [23] combine communication complexity to give lower bounds lower bounds with limited ambiguity. We will discuss quantifying nondeterminism in more detail in the next section.

3 Quantifying Nondeterminism in Finite Automata

In the rest of this paper we consider quantifying nondeterminism in finite automata as this subject appears in the literature. We have already seen in Subsection 2.1 some examples of exponential blow up, from the state complexity point of view, as we transform some NFAs into DFAs. A natural question here is how quick is this logarithmic saving in terms of "allowing" more nondeterminism in finite automata. The previous question doesn't make a lot of sence before we explain how do we restrict the nondeterminism used by finite automata.

We can view nondeterminism as another resource that finite automata can use. In this point of view, we can quantify nondeterminism and measure it. There are many possible ways to do that. In this part of the paper we discuss different ways that have already appeared in the literature. For each of these ways of measuring nondeterminism we have a different subsection which also contains results on this measurement.

The terminology used in the literature for finite automata employing limited nondeterminism is not well established and the notations vary from paper to paper. Below we use our own names and we attempt to make the notation, arguably, more consistent. We always give references for the results and, when necessary, explain when the original paper used significantly different terminology.

3.1 Branching and Guessing

In this subsection we consider the nondeterminisitc measures of branching and guessing. These two measures appear in [11]. The only papers where the branching and guessing measures are studied are [11],[28], [34], and [10].

Before we can define the measures of branching and guessing, we should define what is a move of an automaton, moreover what is a computation. A move μ of an automaton $M = (Q, \Sigma, \delta, q_0, F)$ is a triple $(p, a, q) \in Q \times \Sigma \times Q$ where $q \in \delta(p, a)$. A computation of an automaton M is a finite sequence of moves which has some specific properties. Let us have the sequence of moves $\mu_1 \mu_2 \ldots \mu_k$ for a natural number k, also let the move μ_i be the triple (p_i, a_i, q_i). If for every $1 \leq i \leq k-1$ we have that $q_i = p_{i+1}$ and p_1 is the initial state then we call the sequence of moves $\mu_1, \mu_2, \ldots, \mu_k$ a computation of the automaton M for the word $w = a_1 a_2 \ldots a_k$. If additionally the state q_k is a final state then we call the sequence of moves $\mu_1, \mu_2, \ldots, \mu_k$ an accepting computation of M for w.

Definition 1. *For an automaton $A = (Q, \Sigma, \delta, q_0, F)$ the* branching *$brh_A(\mu)$ and* guessing *$gs_A(\mu)$ of a move $\mu = (p, a, q)$ are defined to be:*

$$brh_A(\mu) = \#\delta(p,a) \text{ and } gs_A(\mu) = \log_2(\#\delta(p,a)).$$

Where $\#S$ denotes the cardinality of a set S.

Branching and guessing are extended to computations $\mu_1 \mu_2 \ldots \mu_t$, for $t \geq 0$, by setting:

$$brh_A(\mu_1 \ldots \mu_t) = brh_A(\mu_1) \times \ldots \times brh_A(\mu_t)$$

and

$$gs_A(\mu_1 \ldots \mu_t) = gs_A(\mu_1) + \ldots + gs_A(\mu_t)$$

For a word x in the language $L(A)$ defined by A, we define the branching $brh_A(x)$ and guessing $gs_A(x)$ of the word x on A, when A is clear from the context we omit it from the notation:

$$brh_A(x) = \min\{brh_A(\mu_1 \ldots \mu_t) \mid \mu_1 \ldots \mu_t \text{ is an accepting computation of } A \text{ and reads the word } x\}$$

and

$$gs_A(x) = \min\{gs_A(\mu_1 \ldots \mu_t) \mid \mu_1 \ldots \mu_t \text{ is an accepting computation of } A \text{ and reads the word } x\}$$

It is clear that for a deterministic automaton A and for any word x in $L(A)$ the branching $brh_A(x)$ will be always 1 and the guessing $gs_A(x)$ will be always 0.

Now, we define the branching and guessing of an automaton A to be the largest branching and guessing among all the words accepted by the automaton.

Definition 2. *If the language $L(A)$ accepted by the finite automaton A is empty, let $brh_A = 1$ and $gs_A = 0$. Otherwise, let*

$$brh_A = \sup\{brh_A(x) \mid x \in L(A)\}$$

and

$$gs_A = \sup\{gs_A(x) \mid x \in L(A)\}$$

The measures of branching brh_A and guessing gs_A is one way of quantifying the amount of non-determinism that A requires to recognize $L(A)$. Whenever either of them is finite, we have that $brh = 2^{gs}$. So, we will consider the branching brh and the guessing gs as one measure and we will refer to only one of them. We will choose the one that is more convenient for us, in each case.

Since we intend to study the trade-off between the amount of non-determinism and the size of a finite automaton, we make the following definition.

Definition 3. *Let $|A|$ be the number of states in the finite automaton A. Denote $scbrh_i(L) = \min\{|A| \mid A$ is a finite automaton for L with $brh_A \leq i\}$. We also call the spectrum $scbrh(L)$ of a regular language L to be the infinite sequence:*

$$scbrh(L) = (scbrh_1(L), scbrh_2(L), \ldots, scbrh_i(L), \ldots; scbrh_\infty(L))$$

Note that $scbrh_1 \geq scbrh_2 \geq \ldots \geq scbrh_\infty \geq 1$ for every spectrum. Moreover, $scbrh_\infty(L)$ is the number of states in a minimal-state non-deterministic finite automaton for L. Finally, $scbrh_1(L)$ is the number of states in a minimal-state incomplete deterministic finite automaton for L. Moreover, with similar way we define $scgs_i(L)$ for guessing.

Moreover in [11] they consider about computability issues of this measure. They show that the branching brh_A of an automaton A is computable, as well as the spectrum of a regular language is computable.

In [34], Leung gave an algorithm for finding if a given NFA has finite nondeterminism. He also showed that the corresponding decision problem is PSPACE-complete. The algorithm and his computational complexity result are too complicated for the purpose of this paper and we avoid them here.

In [11] also continue by showing other interesting properties of the branching measure. In the next lemma they show that there are languages where by allowing finite number of nondeterminism does not produce any saving in the number of states.

Lemma 1. *Let L be a regular language and \$ a new symbol. Then for some possitive integers k and n.*

$$scbrh((\$L\$)^*) = (k, k, k, \ldots; n)$$

As a result of the previous lemma we have that there exist regular languages for which any NFA with a finite branching cannot be smaller than a DFA. Moreover, there are languages where the exponential blow up that we have by transforming an NFA to a DFA remains the same even if we transform the NFA to another NFA but with finite amount of nondeterminism.

Theorem 5. *For every regular language L,*

$$scbrh(L) \leq (2^{n-1}, 2^{n-1}, \ldots, 2^{n-1}, \ldots; n),$$

where $n = scbrh_\infty(L)$. Moreover, the upper bound can be tight. For each $n \geq 1$, there is a regular language L_n with

$$scbrh(L_n) = (2^{n-1}, 2^{n-1}, \ldots, 2^{n-1}, \ldots; n).$$

From Lemma 1 we have that minimal NFAs with finite branching can have the same size with the minimal equivalent DFAs. From Theorem 5 we have that this case can be happen even when there is an exponential blow up in the number of states comparing DFAs and NFAs. Here, we have to notice that the bound of the Theorem 5 doesn't reach the general exponential bound of 2^n of comparisons NFAs and DFAs but half of it 2^{n-1}. Intuitively, this difference is because when we compare DFAs and NFAs with we use all the possible sets that come up from the power set construction, in the later case we have all the sets which contain a specific state. This specific state is responsible for separating the languages that involve in the star operation, as we do in Lemma 1. It turns out to be difficult to have a result like Theorem 5 without having this separating state. This result is difficult to do without separating the languages that involve in the star operation, and similar is difficult to separate these languages without having a separating state.

Finely, in [28] Kappes consider comparing an NFA with finite branching with deterministic finite automata with multiple initial states (MDFA). In other words, he consider the case where the whole nondeterminism of the automata to be only in the beginning. His result is that for each NFA M with n states and finite branching $\beta_M = k$ there is an MDFA M' with $k \cdot n + 2$ states and $L(M) = L(M')$.[2]

3.2 Advice

We have already seen one way of measuring the amount of non-determinism on a given automaton. Now, we continue by giving three other ways of measuring non-determinism [21, 23]. The first one is based on the number of nondeterministic choices that we have to make during a computation. The other two measures are similar and they are based on the computation tree of a word, on a given automaton.

The first measure of these papers are called advice. Advice measures the maximum number of nondeterministic choices in all possible computations on a given word. In other words, for a path C the advice of C is the number of states in C which have more than one successor. Now, let us define the advice measure.

Definition 4. *For every NFA A we measure the degree of non-determinism as follows. Let Σ denote the alphabet of A. For every input $x \in \Sigma^*$ and for every computation C of A on x we define $adv(C)$ as the number of non-deterministic choices during the computation C. Then,*

$$adv_A(x) = \max\{adv(C) \mid C \text{ is a computation of } A \text{ on } x\}$$

$$adv_A(n) = \max\{adv_A(x) \mid x \in \Sigma^n\}$$

$$adv_A = \sup\{adv_A(x) \mid x \in L(A)\}$$

These measures haven't been studied much. There are some results on this measure but they are comparisons with other measures. Then, we will discuss more about this measure in the subesection 4 where we study the relation between different measures.

Kintala and Wotschke [29] studied a somewhat related measure. They still count the number of nondeterministic choices that we make as we read a word, the difference now is that we count these nondeterministic choices only in accepting paths, i.e. the path ends up in a final state after reading the input word, and among all these accepting paths we take the one which gives us the smallest number of nondeterministic choices. We call this

[2]This result appears in Kappes' paper with the formula $k \cdot n + 1$. But we need an extra state since we consider NFAs with only one initial states, and he considered the model where there are multiple initial states.

measure M-advice, from Mandl who first used this measure [38], and we denote it by *mdl*. For completeness, we define $mdl_A(C) = adv_A(C)$ for an accepting computation C and $mdl_A(x) = \min\{mdl(C) \mid C \text{ is an accepting computation of } A \text{ on } x\}$.

Kintala and Wotschke give the following language $R'_k = \{x1y \mid x, y \in \{0,1\}^*, |x| \leq k-1, |y| = k\}$ and they prove the following theorem which gives us a lower bound for the relation between deterministic finite automata and finite automata with finite M-advice. Moreover, we should note that for every $k \geq 1$, R'_k can be accepted by a $(2k+1)$-state nondeterministic finite automaton.

There are automata where a limitation in M-advice doesn't change much the succinct of general NFA, and the exponential blow up in deterministic case still occurs. An example of this is the language R'_k, where it can be accepted by $(4k - 3 \cdot \log k)$-state NFA with M-advice at most $\log k$. Moreover any deterministic finite automaton for R'_k must have at least 2^{k+1} states. We have to notice that it reached the tight bound $O(2^{n/2})$ of the case of finite languages [46, 47].

Moreover in the same paper, Kintala and Wotschke [29] give the language $L_{h,k} = \{x1y \mid x, y \in \{0,1\}^*, |x| \leq k-1, |y| = k, \text{ and } x \text{ has at most } h \text{ 1's in it}\}$. They prove that every deterministic finite automaton accepting $L_{h,k}$ must have at least $\sum_{i=0}^{h} \binom{k}{i}$ distinct states. Moreover there is a $O(k^2)$-state nondeterministic finite automaton accepting $L_{h,k}$ with M-advice at most $\log h$.

Another interesting result that they have in the same paper is that there is a hierarchy between the amount of M-advice in NFAs. For any two given function $g_1(n) < g_2(n) < \log n$ there are NFAs with M-advice at most $g_2(n)$ which are more succinctness than any NFA with M-advice at most $g_1(n)$. They come to this result by showing that for any given function $g(n) < \log n$, there is a class of n-state nondeterministic finite automata with at most $g(n)$ M-advice, and the equivalent minimal deterministic finite automaton has at least $\sum_{i=0}^{2g(O(\sqrt{n}))} \binom{O(\sqrt{n})}{i}$ states, for big enough n. Which intuitively the previous tells us that the bigger function that we have, but smaller than $\log n$, the more succinctness in the number of states we can have.

3.3 Tree Width

The other two ways of measuring non-determinism are based on the computation tree of a word. In this subsection we consider the case of the so called tree width measure.

Here, we should mention what is the computation tree. The computation tree of the automaton A on input x, denoted by $T_{A,x}$, is defined as follows. The root of $T_{A,x}$ is labelled by the initial configuration of A on x. If a node u of $T_{A,x}$ is labelled by configuration K and $K_1, \ldots, K_m$ are the configurations such that $K \vdash K_i$, $i = 1, \ldots, m$, then u has m children that are labeled, respectively, by the configurations $K_1 \ldots, K_m$.

We will call *tree width* of a word x for a given automaton A the number leaves of the computation tree of the word x of automaton A, i.e., $tw_A(x)$ is the number of leaves of $T_{A,x}$. We should mention here that the papers [21, 23], which use this measure, refer to it as leaf size. We can now define the following notation.

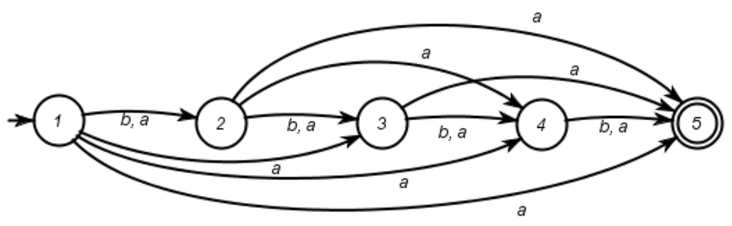

Figure 7: An NFA with 2^3 tree width.

Definition 5. *We define the tree width of A on a word x, denoted by $tw_A(x)$, to be the number of leaves of $T_{A,x}$. We also define the following:*

$$tw_A(n) = \max\{tw_A(x) \mid x \in \Sigma^n\}$$

$$tw_A = \sup\{tw_A(x) \mid x \in L(A)\}$$

In [21, 23], they give the following lemma where the investigate the growth of the tree width function. It is a variation of a result in [24].

Lemma 2. *For every NFA A with n states, either $tw_A(k) \leq (k \cdot n)^n$ or $tw_A(k) \geq 2^{\Omega(k)}$.*

Also in [23] they give the following theorem.

Theorem 6. *For every NFA A, $tw_A(k)$ is either bounded by a constant or in between linear and polynomial in k, or otherwise $2^{\Theta(k)}$.*

As it appears, this measure hasn't been extensively studied in the literature. In [43] we have studied the last measure, the tree width. After making the notice that a NFA with finite tree width does not have cycles containing a nondeterministic transition, we prove the following theorem.

Theorem 7. *If A is an NFA with n states and no cycle contains a nondeterministic transition, then*

$$tw_A \leq 2^{n-2}$$

A natural question that may appear to the reader is if there are examples where the inequivalence of the previous theorem becomes an equality. It turns out that there are such examples. Such example is the Example 6 where we also see an instance of it in Figure 7 for 5-states.

Example 6. *Let $n \geq 2$, $\Sigma = \{a, b\}$ and $A = (Q, \Sigma, \delta, 1, \{n\})$ where $Q = \{1, \ldots, n\}$ and we define for $i \in \{1, \ldots, n-1\}$, $\delta(i, a) = \{i+1, i+2, \ldots, n\}$, $\delta(i, b) = \{i+1\}$, $\delta(n, a) = \delta(n, b) = \emptyset$.*

We have already mentioned that a NFA with finite tree width does not have cycles containing a nondeterministic transition. It is easy to see that also for an NFA A which no cycle of A contains a nondeterministic transition has finite tree width. In that case, we can decide in polynomial time whether or not a given NFA has finite tree width.

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. In the same paper we show that this deterministic decomposition fully characterize NFAs with finite tree with. First, let us formulate the DFA decomposition.

We have already seen that the tree width can be exponential compared to the number of states for a finite automaton with finite tree width. In the case where we have small amount of tree width we compare the number of states between DFAs and NFAs with finite tree width and we see that the succinctness is polynomial. By noticing that after a nondeterministic transition we can not come back to the initial state when we have finite tree width, we show the following lemma.

Lemma 3. *Let L be a regular language where $sctw_k(L) = n$ for some $k \leq n-1$. Then*

$$\mathrm{sc}(L) \leq 1 + \sum_{j=1}^{k} \binom{n-1}{j}.$$

From the following theorem we get that the bound in the previous lemma is tight as we show in [43]. The proof is based on a construction similar to [16], which we have discussed in subsection 2.1.

Theorem 8. *For every $1 \leq k \leq n-1$ there exists an n-state NFA $A_{n,k}$ such that $tw(A_{n,k}) = k$ and $\mathrm{sc}(L(A_{n,k})) = 1 + \sum_{j=1}^{k} \binom{n-1}{j}$.*

We know that minimization is NP-hard for any class of finite automata that contains a restricted version of NFAs with tree width 2 [4, 5]. Thus, in order to establish lower bounds for the size of NFAs with finite tree width we need to rely on ad hoc methods inspired by the fooling set techniques that we have seen in the first section.

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 7.

Lemma 4. *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*

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

We end our study of the tree width measure by giving an example where we use the previous lemma to give lower bound on the number of states for a NFA with finite tree width.

Example 7. *Recall the unary language $H_m = \overline{(0^m)^+}$ we consider earlier. The language H_m has a separator set $\{\varepsilon, 0, 0^2, \ldots, 0^{m-1}, 0^m\}$. According to Lemma 4, if H_m has* tw(k)*-NFA with n states where $k \leq \frac{n}{2}$, we have the estimation $\binom{n}{k} \geq m+1$. This is exponentially better than the lower bound obtained for the size of (general) NFAs with the bipartite dimension method, remember Example 5. 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 [13, 15].*

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

3.4 Ambiguity

Our final measure of the amount of non-determinism is ambiguity. The ambiguity of a nondeterministic computation refers to the number of accepting paths in the computation tree. This measure is the most well studied in the literature, it appears in many papers [10, 12, 14, 21, 23, 33, 35, 36, 37, 41, 42, 45, 50, 52, 53]. The ambiguity of the word x for automaton A, denoted by $amb_A(x)$, is the number of all accepting leaves of the computation tree of the word x of automaton A. In other words, $amb_A(x)$ is the number of all accepting paths of A when it reads x. We define the concept of bounded ambiguity later in this subsection.

Here we split this subsection into two parts. In the first part, we study unambiguous finite automata (UFA), which are nondeterministic finite automata with ambiguity one. In other words, each accepted word has a unique accepting path. In the second part, we study bounded ambiguity which can be considered a generalization of unambiguous finite automata.

3.4.1 Unambiguous Finite Automata

In a survey of ambiguity in finite automata and pushdown automata Holzer and Kutrib [14] summarize some results of different papers [48, 50, 35, 32, 36] in the following theorem. This theorem compares NFAs with UFAs and UFAs with DFAs.

Theorem 9. *Let $n \geq 1$ and A be and n-state NFA. Then $2^n - 1$ states are sufficient and necessary in the worst case for a UFA to accept $L(A)$. If automaton A is a UFA, then 2^n states are sufficient and necessary in the worst case for a DFA to accept $L(A)$.*

The first property that we consider about UFAs is their relation with DFAs. We have already seen in Figure 3 that there is an NFA with n states that the minimal equivalent DFA needs 2^n states, or $2^n - 1$ states in the case where we don't count the 'dead' state.

The corresponding question of comparing the state complexity of UFAs with DFAs for unary alphabets was investigated recently by Okhotin in [41, 42]. He presents that if A is an n-state UFA, for $n \geq 1$, accepting a unary language, then $e^{\Theta(\sqrt[3]{n \cdot \ln^2 n})}$ states are sufficient and necessary in the worst case for a DFA to accept $L(A)$.

On the other hand if we want to simulate a unary NFA by a UFA in the same papers [41, 42] it was shown that one cannot do asymptotically better than in the unary NFAs to DFAs transformation, which we have seen in the previous section.

For the trade off between MDFAs and UFAs a tight bound in number of states was given in [37, 36]. Again, let $n \geq 1$ and A be an n-state MDFA. Then, $2^n - 1$ states are sufficient and necessary in the worst case for a UFA to accept $L(A)$.

We are closing this part of unambiguous finite automata by mentioning a similar concept which Leung investigated in [37]. A variation of UFAs is the so called structurally unambiguous finite automata. An NFA $A = (Q, \Sigma, \delta, q_0, F)$ is structurally unambiguous (SUFA) if for every word $w \in \Sigma^*$ and every state $q \in Q$ there is at most one computation from the initial state q_0 to state q reading word w. Note that compared to the original definition of unambiguity the computations need not be accepting/rejecting but to end with the same state. Thus, unambiguity is a semantic concept, while structural unambiguity is a syntactic one, which is independent of the choice of the set of final states. Notice that when in the set of final states we have only one state then SUFA is UFA as well. In the same paper Leung showed that a SUFA can be exponentially more succinct in the number of states than a UFA.

Now for the comparison between SUFA and DFA, UFA, or MDFA we have the following results. Let $n \geq 1$ and A be an n-state SUFA. Then 2^n states are sufficient and necessary in the worst case for a DFA to accept $L(A)$. Moreover, $2^n - 1$ states are sufficient and necessary in the worst case for a UFA or MDFA to accept $L(A)$ [37, 14].

3.4.2 NFAs with bounded ambiguity

A natural generalization of unambiguous finite automata is to relax the condition that for each accepted word we have only one accepted path but a certain number of them. In other words, for each accepting word we can have a maximum of a fixed number of accepting paths, this brings us to the concept of quantified ambiguity. Let us define now this concept more formally.

Definition 6. *We define the* ambiguity *of an automaton A with respect to a word x, denoted by $amb_A(x)$, to be the number of accepting leaves of $T_{A,x}$. Also, we define the following:*

$$amb_A(n) = \max\{amb_A(x) \mid x \in \Sigma^{\leq n}\}$$

$$amb_A = \sup\{amb_A(x) \mid x \in L(A)\}$$

As usual, we need a different notation for denoting the number of states which an automaton has with ambiguity less that a given number. Then, denote $scamb_i(L)$ the size of a smallest NFA for L having ambiguity at most i. More formally, we have the following.

$$scamb_i(L) = \min\{|A| \mid A \text{ is a finite automaton for } L \text{ with } amb_A \leq i\}$$

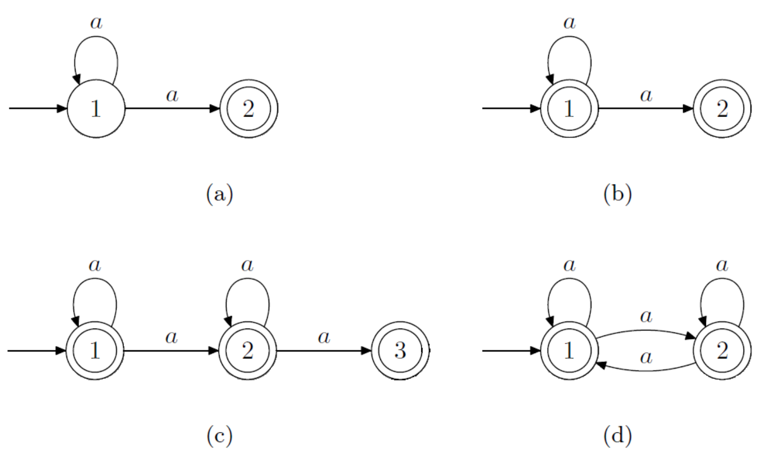

Figure 8: NFAs with different degrees of ambiguity: (a) UFA, (b) FNA with amb_A is constant 2, (c) PNA with linear amb_A, and (d) ENA.

Firstly, we bound the function $amb_A(n)$ by a function f, $amb_A(n) \leq f(n)$. We will call an automaton A finitely ambiguous if the function f is a constant, similar we call the automaton A polynomially or exponentially ambiguous if the function f is polynomial or exponential, respectively we denote them by FNA, PNA, and ENA. The first easy result is that for any NFA A we have that $amb_A(n) \leq |Q|^n$, where Q is the set of states of A [14]. This is because as we read any word of length n, in each step we have at most as many choices as the number of states. Hence, every NFA is exponentially ambiguous, i.e. an ENA. Figure 8 shows us some examples of NFAs with different ambiguity [14, 12].

In the papers [14, 24, 45, 53] they give the following theorem which is a structural characterization of NFAs ambiguities. Before we give the theorem we have to mention when an automaton is strictly ambiguous. An automaton is strictly ambiguous of a certain degree, if it is ambiguous of this degree, but not of any lower degree in the ambiguity hierarchy induced by the classes above.

Theorem 10. *Let A be an NFA with state set Q and input alphabet Σ, in which all states are useful. Then we have the following structural characterizations of finitely, polynomially, and exponentially ambiguity on finite automata:*

1. *An automaton A is strictly exponentially ambiguous if and only if there exists a state $q \in Q$ and a word $w \in \Sigma^+$ such that there is more than one computation from state q to q reading word w.*

2. *An automaton A is strictly polynomially ambiguous if and only if A is not exponentially ambiguous and there exist different states $p, q \in Q$ and a word $w \in \Sigma^*$ such that there are computations from state p to itself, from state p to q, and from state q to itself, all reading the same word w.*

3. *An automaton A is finitely ambiguous if and only if A is neither strictly exponentially nor strictly polynomially ambiguous.*

Of course an interesting problem would be the comparison of state complexities of automata of different degrees of ambiguity. This question fits nicely with the whole paper so far. Results on that direction will be summarized in the following theorem [14]. More specifically, the comparison of DFAs and UFAs was studied in [32, 36, 48, 50]. The comparison between UFAs and FNAs was studied in [36, 45, 48, 50]. The comparison of PNAs and NFAs was studied in [23, 35]. Finally, the comparison of FNA and PNA turned out to be a dificult problem, some attempts on giving an answer initially failed [23, 45] and a solution was given in [22].

Theorem 11. *The following separation results on NFA with different degrees of ambiguity are known:*

1. *For every $n \geq 1$, there is an n-state ENA A (an NFA having exponential ambiguity) such that any PNA accepting $L(A)$ has at least $2^n - 1$ states.*
2. *For every $k, r \geq 1$, there is a $k \cdot r^{O(1)}$-state NFA with ambiguity $O(n^k)$ such that any NFA accepting $L(A)$ has an exponential (in k and r) number of states, if ambiguity $o(n^k)$ or finite ambiguity is required.*
3. *For every $n \geq 1$, there is an n-state FNA A such that any UFA accepting $L(A)$ has at least $2^n - 1$ states. This also holds when changing FNA to UFA and UFA to DFA.*

The given bounds in the first and last results are known to be tight.

We continue by studying the relationship of the number of states and the degree of ambiguity. Already in Figure 8 we show that there is a 2-state finite automaton with exponential ambiguity. What happens in the case that we have finite ambiguity. This problem was first considered in [53] and the following result was established there.

Theorem 12. *Let A be an n-state FNA. Then amb_A is at most $5^{n/2}n^n$.*

We have already discussed earlier about MDFAs. Another relevant question arises, this is the relation between MDFAs and UFAs. We have already seen that a DFA can be exponentially larger than a UFA, see Figure 3, moreover a DFA sometimes is polynomially larger than a MDFA [16]. Then, someone would think that simulating a MDFA by a UFA can save us some number of states, but the opposite is true. From the next theorem [36] we have that an UFA can be exponentially larger than a MDFA. Leung defined a MDFA as appears in Figure 9, and he calls it M_n. In this MDFA, with n states and over a two letter alphabet, by reading one letter we always go to the next state modulo n, and by reading the other letter we always stay in the same state except from the first state that we go to the second. Moreover, all the states are initial states and states alternative are final.

Theorem 13. *The smallest UFA equivalent to the MDFA M_n has $2^n - 1$ states.*

Continuing with the next results we have to remember the connection of communication complexity with finite automata as we have briefly discussed is Subsection 2.3. By using communication matrices of regular languages Hromkovic et al. [21, 23] give the following lower bounds for the size of NFA with finite ambiguity.

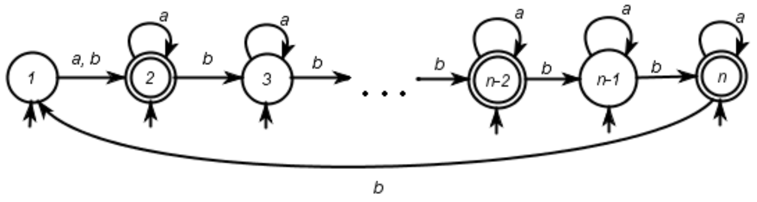

Figure 9: The MDFA M_n from [36].

Theorem 14. *For every regular $L \subseteq \Sigma^*$ and $k \in \mathbb{N}$*

a. $scamb_1(L) \geq rank_{\mathbb{Q}}(M(L))$

b. $scamb_k(L) \geq rank_{\mathbb{Q}}(M(L))^{1/k} - 1$

c. $scamb_k(L) \geq 2^{\sqrt{cc(L)/k}} - 2.$

Here we should mention that similar results were given earlier by Schmidt [48]. Now by using the previous lemma they show the following corollary which compares the sizes of NFAs with NFAs with finite ambiguity.

Corollary 2. *There is a language NID_m with the following properties:*

1. *The language NID_m can be recognized by an NFA A with ambiguity $O(m)$ and size $O(m)$*

2. *Any NFA with ambiguity k for NID_m has size at least $2^{m/k} - 1$, and in particular any UNFA for NID_m must have 2^m states*

3. *No NFA with ambiguity $o(m/\log m)$ for NID_m has polynomial size in m.*

4 Comparison of Different Measures of Nondeterminism and Future Work

By concluding the paper we discuss in this final section the relations between the different measures of nondeterminism. We discuss more on this relation in the first subsection. The second subsection is devoted in discussing open problems and general directions on further research in the topics that we have discussed here.

4.1 Comparison of the Different Measures

In the previous section we have seen different ways of measuring nondeterminism in finite automata. In this subsection we take a look into the relation that these measures have with each other. There are two ways that we can look at this problem. We can search the relation between the degrees of nondeterminism of the different measures in a given NFA. Such an example is that for any NFA A, the ambiguity of A is always less or equal to the tree width of A, i.e. $amb_A \leq tw_A$. The reader could be easily convinced for the previous statement by noticing that every accepting path of an automaton is also just a computational path of the same automaton. Another way of comparing two measures of nondeterminism is to relate the sizes of minimal NFAs where the nondeterminism is, respectively, bounded by the two measures. For example take the language $L_{p,q} = \{w \in a^* \mid p \text{ or } q \text{ divides } |w|\}$, and let p, q be primes. Then the minimal NFA with tree width 2 has $p + q + 1$ states and the same number of states has the minimal NFA with ambiguity 2. On the other hand, for all $n \geq 4$, there exists a UFA A with n states such that every equivalent NFA B with finite tree width has 2^{n-1} states [43].

We just saw the easy case of comparing the ambiguity and the tree width measures. Let's compare now the ambiguity with advice. In the case of comparing advice and ambiguity on the same automaton, it appears that, we can not have any equation. Remember the language $L_{p,q}$ that we just saw, and expand it to have all the strings $w \in a^*$ where their length are divided by one of the prime numbers $p_1, \ldots, p_k$, denote this new language $L_{p_1,\ldots,p_k}$. Let us have now the minimal NFA recognizing $L_{p_1,\ldots,p_k}$, this is the one which has an initial state and k disjointed cycles of length p_i for each prime p_i. This automaton has advice one and ambiguity k. On the other hand, remember the automaton from Example 6, this automaton has advice $n-1$ and it is unambiguous. From the same examples, we can see that comparing these two measures from the state complexity point of view is still difficult. In the case of the language $L_{p_1,\ldots,p_k}$ as we have already noticed $p_1 + \ldots + p_k + 1$ states are sufficient for an NFA recognizing it, but for unambiguous automata we need $p_1 \cdot \ldots \cdot p_k$. For the automaton from the Example 6 n states are sufficient but it seems that $2 \cdot (n-1)$ states are required in the case of advice one. Of course, it would be an interesting question to figure out the exact bounds concerning the relationship between these measures.

We mentioned earlier a similar measure with the measure advice, the M-advice. Since, M-advice counts the nondeterministic choices only from accepting paths it is immediate that M-advice is always smaller than ambiguity. M-advice chooses the accepting computational path with the smallest number of nondeterministic steps. Then, it is always smaller than advice. That means that M-advice is smaller than all other measures that are greater than advice. Finishing the discussion for M-advice we could also mention that M-advice is smaller than guessing as well. The guessing measure counts the number of bits needed to represent the number of nondeterministic choices on the 'best' accepting path on the given input. on the other hand, the M-advice counts just the total number of nondeterministic computation steps, again on the 'best' accepting path. Of course, the maximum of each case can be produced by different accepting paths, but still M-advice cannot be more than guessing.

Parenthetically, let us remind to the reader the direct relationship between guessing and branching. When at least one of them is finite, it holds that $brh_A = 2^{gs_A}$, for all NFA A. Hence, we can consider them as one measure and to use the one which best fits in the context.

Let us move now to our next comparison, the one between advice and tree width. It is easy to see that the advice of any NFA is always less or equal than the tree width. For every computation when we have a nondeterministic choice, and we count plus one for advice, we have at least one additional leaf in the computational tree since from that point we have at least two different computational branches. Moreover, as they notice in [21, 23], the tree width is bounded by the exponent of the advice measure, i.e. $tw_A(n) \leq 2^{O(adv_A(n))}$ for every NFA A and $n \in \mathbb{N}$. Notice here that the width, number of leaves, of a tree is less than the exponent of its depth multiplied by the maximum number of nondeterministic choices that we can have in one step. In the same paper, they also give some other results in this direction. They give the two following lemmata.

Lemma 5. *Let as have an NFA A with n states, then either*

a. $adv_A(k) \leq n$ *and* $tw_A(k) \leq n^n$ *or*

b. $adv_A(k) \geq k/n - 1$ *and* $tw_A(k) \geq k/n - 1$.

Lemma 6. *Every minimal NFA A, with n states satisfies the following equation for every* $x \in \Sigma^*$.

$$tw_A(x) \leq adv_A(|x| + n) \cdot |x| \cdot n \cdot +1$$

Here we should note that the previous lemma was stated in a more general form in the original papers [21, 23]. The lemma holds for every NFA with at most one, as they call, terminally rejecting state. Terminally rejecting state is a state which is not co-reachable by a final state. Clearly there is at most one terminally rejecting state in a minimal automaton.

Moreover, Hromkovic et. al. [21, 23] give another result which combines the measures of advice, ambiguity, and tree width. This result is for the case where the automaton is minimal, in other words there is no other equivalent NFA with a smaller number of states. They say that for minimal automata, the tree width is asymptotically at most the multiple of advice and ambiguity.

Theorem 15. *Every minimal NFA A satisfies the following:*

$$tw_A(n) \leq O(adv_A(n) \cdot amb_A(n))$$

Especially for any such NFA, $adv_A(n) = \Theta(tw_A(n))$.

Hromkovic et. al. [21, 23] also study the state complexity point of view of comparing the above measures. In the following theorems they show that there are cases where we have polynomial advice, exponential tree width , and polynomial size, while every equivalent NFA with polynomial tree width or ambiguity needs an exponential number of states. Moreover, they show that there are cases where as we degrease the ambiguity the number of states increases exponentially.

Theorem 16. *There is a family of languages* KL_m *such that* KL_m *can be recognized by an NFA with advice* $\Theta(n)$*, tree width* $2^{\Theta(n)}$*, and size poly*(m)*, while every NFA with polynomial tree width - ambiguity needs size at least* $2^{\Omega(m)}$ *to recognize* KL_m.

Theorem 17. *Let $KON_m = \{0,1\}^*0M_m0\{0,1\}^*$, where M_m contains all words in $\{0,1\}^*$ with a number of 1's that is divisible by m. KON_m can be recognized by an NFA A with $amb_A(n), tw_A(n) = \Theta(n)$, and size $m+2$, while any NFA with ambiguity k for KON_m needs at least $2^{(m-1)/k} - 2$ states.*

Goldstine et. al [12] study the relation between the measures of ambiguity and guessing. Firstly, they show that finite automata with constant or linear guessing can be of all types of ambiguous automata, UFA, FNA, PNA, and ENA. The interesting result of this paper is that in the case of a non-constant but sublinear guessing, the automaton must have an infinite degree of ambiguity. The key result of this paper is the following lemma which has as a consequence the following theorem.

Lemma 7. *If A is an n-state NFA and $x \in L(A)$ is such that there is no word $w \in L(A)$ with $|x| < |w|$ and $gs_A(x) \geq gs_A(w)$, then*

$$n^{amb_A(x)} \cdot (amb_A(x) \cdot gs_A(x) + 1) > 2^{-n} \cdot |x|$$

Theorem 18. *Every NFA with a non-constant but sublinear guessing function has an infinite degree of ambiguity.*

Continuing with comparing the guessing measure with the other measures we notice that $gs_A(x) \in O(adv_A(x))$, $gs_A(x) \leq tw_A(x)$, and $gs_A(x) = \Theta(mdl_A(x))$. For the relation $gs_A(x) \in O(adv_A(x))$ let c be the number of the maximum number of nondeterministic choices that automaton A can have in one step. Then, guessing is at most the product of c and advice, i.e. $gs_A(x) \leq c \cdot adv_A(x)$. The relation $gs_A(x) \leq tw_A(x)$ is trivial from the definition of the two measures. For the relation $gs_A(x) = \Theta(mdl_A(x))$, similar with the advice, it holds that $gs_A(x) \leq c \cdot mdl_A(x)$ and as we have already mentioned $mdl_A(x) \leq gs_A(x)$.

We notice here that some measures are more related than others. For the ones that are more related, it makes easier to establish relations between then. By noticing that, we can try to classify the different measures that we have seen so far. In this direction we could try to classify them as follows:

- Measures that are based on the best accepting computation.
  In this category there are the measures of guessing,branching, and M-advice.

- Measures that are based on the worst accepting computation.
  The advice measure belongs to this category.

- Measures that count only accepting computations.
  In this category falls the ambiguity measure.

- Measures that count all possible computation paths.
  The tree width measure belongs to this category.

Having this classification in mind we can see why some relations are easier to establish. Let as take the relation between tree width and ambiguity. Both of them measures number

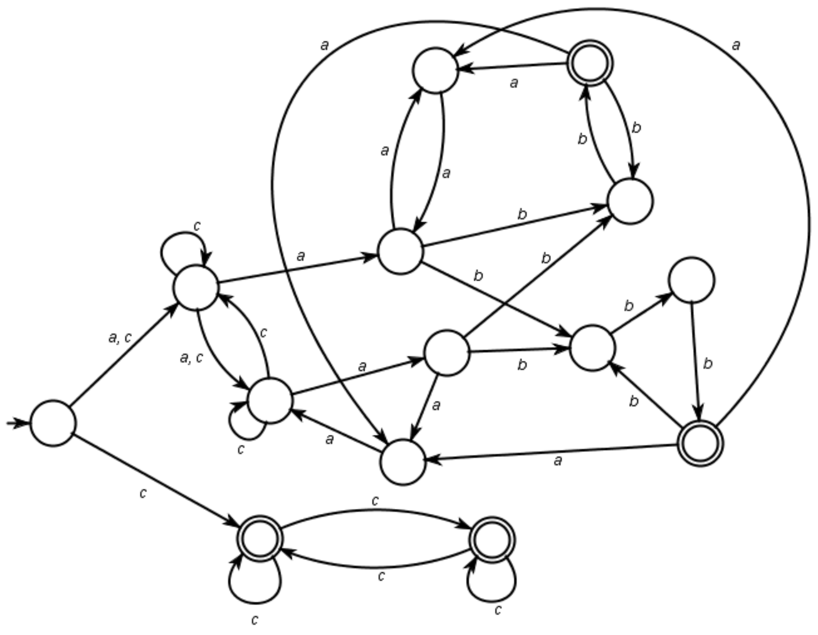

Figure 10: A nondeterministic finite automaton over $\{a, b, c\}$.

of paths in the computation tree. The difference between them is that the one measures all possible computations, on a given word. Even the ones that they don't reach an acceptance state or even the ones that the computation becomes blocked before reaching the end of the input. On the other hand, the other measure counts only the computations which reach an acceptance state.

Similarly, we can have an intuition of which measures are more difficult to relate. Let us show for example that measures from the first type, which are based on the best accepting computation, do not have a clear relation with measures from the fourth type, where measuring all possible nondeterministic paths. Let us choose the branching measure from the first type and the tree width measure from the fourth type. Moreover, let A be the automaton from the Figure 10. We are searching for strings x, y such that $tw_A(x) < brh_A(x)$ and $tw_A(y) > brh_A(y)$. Let us choose first the string $x = (a^2b^3)^n$, we notice that $tw_A(x) = 2 \cdot n + 1$ and $brh_A(x) = 4^n$. In other words, there is string x such that $tw_A(x) < brh_A(x)$. On the other hand, let us choose the string $y = c^n$, we notice that $tw_A(y) = 2^n$ and $brh_A(y) = 2^{n-1}$. Then, there is a string y such that $tw_A(y) > brh_A(y)$.

We just showed a case where two measures are difficult to relate. Then some interesting open problems can come from finding relationships between different measures. Such a question could be to find a clear relation for branching and tree width for minimal NFA. On the other hand, it could also hold that even in the case of minimal NFA we don't have a clear relation. We will discuss similar open problems in the next subsection.

4.2 Open problems and future work

At various points in the in the depth paper several open problems have been mentioned. While descriptional complexity of finite automata is a central research topic, relatively little work has been done on quantifying nondeterminism. In the following we briefly describe some open problems and future research directions dealing with limited nondeterminism.

A major research direction would be to compare the descriptional complexity of regular languages with respect to the different measures for limited nondeterminism. In the depth paper we have surveyed, to the best of our knowledge, most existing results in this direction. As was seen in the previous sections, the literature contains only some relatively scattered results and, in particular, the relationships between different measures are not properly understood. Different authors have introduced different ways of measuring nondeterminism, and besides the few results surveyed in section 3.5., very little is known about their relationships.

As observed in Section 4.1, for example, comparing the branching and the tree width measure is challenging even for a fixed NFA. In order to obtain a proper understanding of descriptional complexity, we need to consider regular languages instead of particular NFAs. The following types of questions would be challenging: for a regular language L establish upper and lower bounds for the size of a minimal NFA for L with branching k as a function of the size of a minimal NFA for L with tree width k', and vice versa.

Similar comparisons could be attempted between other pairs of measures for limited nondeterminism. Perhaps the simplest comparison would be between MDFAs with k initial states and NFAs of tree width k. Even in this seemingly very restricted case, the precise comparison of the descriptional complexity of the two models remain unknown.

Recently there has been very much interest in the state complexity of regularity preserving operations both for DFAs and NFAs [HK2010,Shal08,Y1997]. Naturally one can consider similar questions for NFAs with limited nondeterminism. This is a huge area of research and practically nothing is known about operational state complexity for limited nondeterminism. Okhotin [Okhotin12] has studied the operational state complexity of unambiguous NFAs on unary languages, but no results are known for regular languages over non unary alphabets. We have studied the state complexity of union and intersection for finite tree width NFAs in [PSA12], but, even in the case of finite tree width, no lower bounds are known for the state complexity of other basic operations such as complementation, concatenation of Kleene star.

As is known from [BM2008], determining the size of a minimal NFA with very limited nondeterminism is intractable. In particular, [BM2008] shows that minimization of δNFA's is NP-complete, where δNFAs are a subclass of tree width two automata. This means that for establishing lower bound results that are needed for a proper understanding of NFAs with limited nondeterminism we need some, more or less, ad hoc techniques to prove lower bounds. In the depth paper we have surveyed the fooling set, extended fooling set, biclique edge cover and communication complexity techniques that have been developed for general NFAs. As discussed in Section 2.3 the communication complexity technique also yielded useful lower bounds for NFAs with finite ambiguity. A major open problem is to find useful lower bound techniques for NFAs with limited nondeterminism with respect to the various nondeterminism measures. Naturally a general NFA lower bound applies also to limited nondeterminism NFAs, however, in order to yield useful descriptional complexity results the new techniques need to give stronger lower bounds, that is, the technique needs to somehow exploit the fact that the corresponding NFA has finite tree width, or finite branching etc.

In the depth paper we have restricted consideration to single tape one-way finite automata. Naturally notions of limited nondeterminism can be considered for the more general two-way automata or multitape finite automata and nothing is known about limited nondeterminism for such models. The descriptional complexity of various types of two-way deterministic and (general) nondeterministic automata has been studied in [26, 27], and the questions are significantly more challenging than for ordinary one-way automata.

References

[1] André Arnold, Anne Dicky, and Maurice Nivat. A note about minimal non-deterministic automata. *Bulletin of the EATCS*, 47:166–169, 1992.

[2] Sergei L. Bezrukov, Dalibor Froncek, Steven J. Rosenberg, and Petr Kovár. On biclique coverings. *Discrete Mathematics*, 308(2-3):319–323, 2008.

[3] Jean-Camille Birget. Intersection and union of regular languages and state complexity. *Inf. Process. Lett.*, 43(4):185–190, 1992.

[4] Henrik Björklund and Wim Martens. The tractability frontier for nfa minimization. In *ICALP (2)*, pages 27–38, 2008.

[5] Henrik Björklund and Wim Martens. The tractability frontier for nfa minimization. *J. Comput. Syst. Sci.*, 78(1):198–210, 2012.

[6] Marek Chrobak. Finite automata and unary languages. *Theor. Comput. Sci.*, 47(3):149–158, 1986.

[7] Marek Chrobak. Errata to 'finite automata and unary languages'. *Theor. Comput. Sci.*, 302:497–498, 2003.

[8] Arthur Gill and Lawrence T. Kou. Multiple-entry finite automata. *J. Comput. Syst. Sci.*, 9(1):1–19, 1974.

[9] Ian Glaister and Jeffrey Shallit. A lower bound technique for the size of nondeterministic finite automata. *Inf. Process. Lett.*, 59(2):75–77, 1996.

[10] Jonathan Goldstine, Martin Kappes, Chandra M. R. Kintala, Hing Leung, Andreas Malcher, and Detlef Wotschke. Descriptional complexity of machines with limited resources. *J. UCS*, 8(2):193–234, 2002.

[11] Jonathan Goldstine, Chandra M. R. Kintala, and Detlef Wotschke. On measuring nondeterminism in regular languages. *Inf. Comput.*, 86(2):179–194, 1990.

[12] Jonathan Goldstine, Hing Leung, and Detlef Wotschke. On the relation between ambiguity and nondeterminism in finite automata. *Inf. Comput.*, 100(2):261–270, 1992.

[13] Hermann Gruber and Markus Holzer. Finding lower bounds for nondeterministic state complexity is hard. In Oscar H. Ibarra and Zhe Dang, editors, *Developments in Language Theory*, volume 4036 of *Lecture Notes in Computer Science*, pages 363–374. Springer, 2006.

[14] Markus Holzer and Martin Kutrib. Descriptional complexity of (un)ambiguous finite state machines and pushdown automata. In *Reachability Problems, 4th International Workshop, RP*, pages 1–23, 2010.

[15] Markus Holzer and Martin Kutrib. Descriptional and computational complexity of finite automata - a survey. *Inf. Comput.*, 209(3):456–470, 2011.

[16] Markus Holzer, Kai Salomaa, and Sheng Yu. On the state complexity of k-entry deterministic finite automata. *Journal of Automata, Languages and Combinatorics*, 6(4):453–466, 2001.

[17] John E. Hopcroft. An n log n algorithm for minimizing the states in a finite automaton. In Z. Kohavi, editor, *The Theory of Machines and Computations*, pages 189–196. Academic Press, 1971.

[18] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages and Computation.* Addison-Wesley, 1979.

[19] Juraj Hromkovic. Relation between chomsky hierarchy and communication complexity hierarchy. *Acta Math. Univ. Com.*, 48/49:311–317, 1986.

[20] Juraj Hromkovic. *Communication complexity and parallel computing.* Texts in theoretical computer science. Springer, 1997.

[21] Juraj Hromkovic, Juhani Karhumäki, Hartmut Klauck, Georg Schnitger, and Sebastian Seibert. Measures of nondeterminism in finite automata. In Ugo Montanari, José D. P. Rolim, and Emo Welzl, editors, *ICALP*, volume 1853 of *Lecture Notes in Computer Science*, pages 199–210. Springer, 2000.

[22] Juraj Hromkovic and Georg Schnitger. Ambiguity and communication. In *STACS*, pages 553–564, 2009.

[23] Juraj Hromkovic, Sebastian Seibert, Juhani Karhumäki, Hartmut Klauck, and Georg Schnitger. Communication complexity method for measuring nondeterminism in finite automata. *Inf. Comput.*, 172(2):202–217, 2002.

[24] Oscar H. Ibarra and Bala Ravikumar. On sparseness, ambiguity and other decision problems for acceptors and transducers. In *STACS*, pages 171–179, 1986.

[25] Tao Jiang and Bala Ravikumar. Minimal nfa problems are hard. *SIAM J. Comput.*, 22(6):1117–1141, 1993.

[26] Christos A. Kapoutsis. Size complexity of two-way finite automata. In *Developments in Language Theory*, pages 47–66, 2009.

[27] Christos A. Kapoutsis, Richard Královic, and Tobias Mömke. Size complexity of rotating and sweeping automata. *J. Comput. Syst. Sci.*, 78(2):537–558, 2012.

[28] Martin Kappes. Descriptional complexity of deterministic finite automata with multiple initial states. *Journal of Automata, Languages and Combinatorics*, 5(3):269–278, 2000.

[29] Chandra M. R. Kintala and Detlef Wotschke. Amounts of nondeterminism in finite automata. *Acta Inf.*, 13:199–204, 1980.

[30] Eyal Kushilevitz. Communication complexity. *Advances in Computers*, 44:331–360, 1997.

[31] Eyal Kushilevitz and Noam Nisan. *Communication complexity.* Cambridge University Press, 1997.

[32] Ernst L. Leiss. Succint representation of regular languages by boolean automata. *Theor. Comput. Sci.*, 13:323–330, 1981.

[33] Hing Leung. Separating exponentially ambiguous nfa from polynomially ambiguous nfa. In *ISAAC*, pages 221–229, 1993.

[34] Hing Leung. On finite automata with limited nondeterminism. *Acta Inf.*, 35(7):595–624, 1998.

[35] Hing Leung. Separating exponentially ambiguous finite automata from polynomially ambiguous finite automata. *SIAM J. Comput.*, 27(4):1073–1082, 1998.

[36] Hing Leung. Descriptional complexity of nfa of different ambiguity. *Int. J. Found. Comput. Sci.*, 16(5):975–984, 2005.

[37] Hing Leung. Structurally unambiguous finite automata. In *CIAA*, pages 198–207, 2006.

[38] Robert Mandl. Precise bounds associated with the subset construction on various classes of nondeterministic finite automata. *Proceedings of the 7th Annual Princeton Conference on Information Sciences and Systems*, pages 263 – 267, 1973.

[39] Albert R. Meyer and Michael J. Fischer. Economy of description by automata, grammars, and formal systems. In *SWAT (FOCS)*, pages 188–191. IEEE Computer Society, 1971.

[40] F.R. Moore. On the bounds for state-set size in the proofs of equivalence between deterministic, nondeterministic, and two-way finite automata. *Computers, IEEE Transactions on*, C-20(10):1211 – 1214, oct. 1971.

[41] Alexander Okhotin. Unambiguous finite automata over a unary alphabet. In *MFCS*, pages 556–567, 2010.

[42] Alexander Okhotin. Unambiguous finite automata over a unary alphabet. *Inf. Comput.*, 212:15–36, 2012.

[43] Alexandros Palioudakis, Kai Salomaa, and Selim G. Akl. State complexity and limited nondeterminism. In *DCFS*, pages 252–265, 2012.

[44] Michael O. Rabin and Dana S. Scott. Finite automata and their decision problems. *IBM Journal of Research and Development*, 3(2):114–125, 1959.

[45] Bala Ravikumar and Oscar H. Ibarra. Relating the type of ambiguity of finite automata to the succinctness of their representation. *SIAM J. Comput.*, 18(6):1263–1282, 1989.

[46] Kai Salomaa and Sheng Yu. Nfa to dfa transformation for finite languages. In *Workshop on Implementing Automata*, pages 149–158, 1996.

[47] Kai Salomaa and Sheng Yu. Nfa to dfa transformation for finite languages over arbitrary alphabets. *Journal of Automata, Languages and Combinatorics*, 2(3):177–186, 1997.

[48] Erik Meineche Schmidt. Succinctness of description of context free, regular and unambiguous language. *Ph.D. thesis, Cornell University*, 1978.

[49] Michael Sipser. Lower bounds on the size of sweeping automata. *J. Comput. Syst. Sci.*, 21(2):195–202, 1980.

[50] Richard Edwin Stearns and Harry B. Hunt III. On the equivalence and containment problems for unambiguous regular expressions, regular grammars and finite automata. *SIAM J. Comput.*, 14(3):598–611, 1985.

[51] Paulo A. S. Veloso and Arthur Gill. Some remarks on multiple-entry finite automata. *J. Comput. Syst. Sci.*, 18(3):304–306, 1979.

[52] Andreas Weber. Distance automata having large finite distance or finite ambiguity. *Mathematical Systems Theory*, 26(2):169–185, 1993.

[53] Andreas Weber and Helmut Seidl. On the degree of ambiguity of finite automata. *Theor. Comput. Sci.*, 88(2):325–349, 1991.

[54] Andrew Chi-Chih Yao. Some complexity questions related to distributive computing (preliminary report). In *STOC*, pages 209–213, 1979.