Descriptional Complexity of (Un)ambiguous Finite State Machines and Pushdown Automata

Markus Holzer and Martin Kutrib

Institut für Informatik, Universität Giessen,
Arndtstr. 2, 35392 Giessen, Germany
{holzer,kutrib}@informatik.uni-giessen.de

Abstract. Unambiguity and its generalization to quantified ambiguity are important concepts in, e.g., automata and complexity theory. Basically, an unambiguous machine has at most one accepting computation path for each accepted word. While unambiguous pushdown automata induce a language family strictly in between the deterministic and general context-free languages, unambiguous finite automata capture the regular languages, that is, they are equally powerful as deterministic and nondeterministic finite automata. However, their descriptional capacity is significantly different. In the present paper, we summarize and discuss developments relevant to (un)ambiguous finite automata and pushdown automata problems from the descriptional complexity point of view. We do not prove these results but we merely draw attention to the big picture and some of the main ideas involved.

1 Introduction

Finite automata are traditionally classified into deterministic (DFA), nondeterministic (NFA), and unambiguous (UFA) machines, and it is well known that all these devices are equally powerful and capture the family of regular languages. Here an NFA is *unambiguous* if for every word in the language there is at most one accepting computation. Clearly, any DFA is a UFA and every UFA is an NFA. The more complicated part of this equality is to show that every NFA or UFA can be simulated by a DFA without changing the accepted language. The construction is normally given by the *powerset construction* [56], but this simulation can also be interpreted as a reachability analysis on the configuration graph induced by the NFA with a device without nondeterminism. Here the "search algorithm" keeps track of all possible configurations (here states) the NFA may reach simultaneously by reading some word (see Figure 1).

Thus, given some n-state NFA one can always construct a language equivalent DFA with at most 2^n states [56], and therefore NFAs can offer exponential savings in space compared with DFAs. In fact, later independently in [49,51,52] it was shown that this exponential upper bound is best possible, i.e., for every n there is an n-state NFA which cannot be simulated by any DFA with strictly less than 2^n states. Exactly the same bound is reached when simulating UFAs

A. Kučera and I. Potapov (Eds.): RP 2010, LNCS 6227, pp. 1–23, 2010.

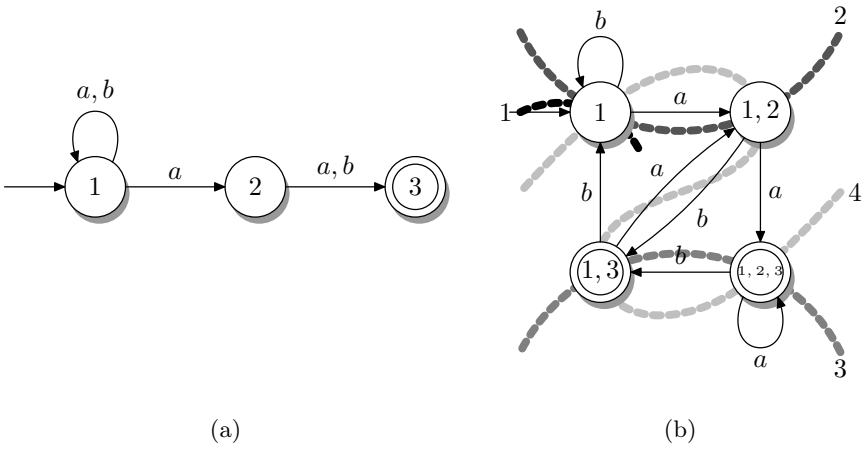

Fig. 1. Powerset construction: (a) NFA and (b) equivalent DFA with initial state {1}, visualizing the "search front" by numbers from 1 to 4 of an incremental powerset construction for the sub-automaton reachable from the initial state—in the drawing the curly brackets are omitted. The first search front 1 consist of the state {1}, the second of the states $\{1, 2\}$ and {1}, the third of $\{1, 2, 3\}$ and $\{1, 3\}$, and finally the fourth of all depicted states. Observe, that the NFA in (a) is actually also a UFA.

by DFAs [45,48], while for the simulation of NFAs by UFAs the upper bound drops to $2^n - 1$, which was also shown to be tight in [47].

These results are only three examples from a vast of different simulation results of various devices from automata theory that can be found in the literature; for some further readings we refer to, e.g., [24,25,26]. We tour a fragment of the literature summarizing simulation results of variants of NFAs and pushdown automata. In particular we pay special attention to unambiguous finite automata and pushdown machines, because we think that unambiguity deserves more attention since it is a valuable and important concept that appears in a lot of sub-fields of theoretical computer science such as, e.g., automata and formal language theory, complexity theory, etc. A size lower bound on a simulation can be interpreted as a succinctness gain, when changing from one description to the other description. In most cases tight bounds for the simulations (in order of magnitude) are obtained, but there are also situations, where the gain in succinctness in a simulation cannot be bounded by any recursive function. This latter phenomenon known as *non-recursive trade-off* appears when pushdown automata become involved. This is not entirely true in general, because for certain restrictions on pushdown machines such as, e.g., deterministic pushdown automata (DPDA) accepting regular languages only, or pushdown machines accepting unary languages, the simulation becomes recursive again. Moreover, we also draw a picture of the relations between higher degrees of ambiguity and nondeterminism. We do not prove these results but we merely draw attention to the big picture and some of the main ideas involved.

The paper is organized as follows: in the next section we introduce the necessary notations. Then in Section 3 we focus on finite automata presenting simulation results and complexity bounds on the minimization problem. The concept of higher degrees of ambiguity and nondeterminism is then discussed for finite automata in Section 4. Finally, similar questions on simulations, and the relation of ambiguity and nondeterminism are addressed for pushdown automata in Section 5.

2 Definitions

In connection with formal languages, strings are called *words*. Let Σ^* denote the set of all words over a finite alphabet Σ. The *empty word* is denoted by λ, and we set $\Sigma^+ = \Sigma^* \setminus \{\lambda\}$. For the *length of a word* w we write $|w|$; in particular, the length of the empty word is zero, i.e., $|\lambda| = 0$. A *formal language* L is a subset of Σ^*. One of the easiest devices in formal language theory are finite automata, which are defined as follows:

A *nondeterministic finite automaton* (NFA) is a quintuple $A = (Q, \Sigma, \delta, q_0, F)$, where Q is the finite set of *states*, Σ is the finite set of *input symbols*, $q_0 \in Q$ is the *initial state*, $F \subseteq Q$ is the set of *accepting states*, and $\delta : Q \times \Sigma \to 2^Q$ is the *transition function*, where 2^Q refers to the *powerset* of the set Q. The *language accepted* by the NFA A is defined as

$$L(A) = \{\, w \in \Sigma^* \mid \delta(q_0, w) \cap F \neq \emptyset \,\},$$

where the transition function δ is recursively extended to $\delta : Q \times \Sigma^* \to 2^Q$ by $\delta(q, \lambda) = \{q\}$ and $\delta(q, aw) = \bigcup_{p \in \delta(q,a)} \delta(p, w)$. A finite automaton is said to be *minimal* if its number of states is minimal with respect to the accepted language.

Special kinds of NFAs are *deterministic* and *unambiguous* finite automata. Let $A = (Q, \Sigma, \delta, q_0, F)$ be a finite automaton. Then A is *deterministic* (DFA) if $|\delta(q, a)| = 1$, for all states $q \in Q$ and letters $a \in \Sigma$. In this case we simply write $\delta(q, a) = p$ instead of $\delta(q, a) = \{p\}$ assuming that the transition function is a mapping $\delta : Q \times \Sigma \to Q$. Moreover the NFA A is *unambiguous* (UFA) if for every word $w \in L(A)$ there is at most one accepting computation path, i.e., the sequence of states seen during the accepting computation on the given word is unique. Clearly, every DFA is a UFA. Any DFA is *complete*, that is, the transition function is total, whereas it may be a partial function for NFAs and UFAs in the sense that the transition function of nondeterministic machines may map to the empty set. So, a *sink* or *dead state* is counted for DFAs, since they are always complete, whereas it is not counted for NFAs and UFAs, since these devices are not necessarily complete. For further details we refer to [29].

A natural generalization of finite automata are pushdown machines. A *nondeterministic pushdown automaton* (NPDA) is a 7-tuple $A = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, where Q is the finite set of *states*, Σ is the finite set of *input symbols*, Γ is the finite *stack alphabet*, $q_0 \in Q$ is the *initial state*, $Z_0 \in \Gamma$ is the *bottom of stack symbol* which initially appears on the pushdown store, $F \subseteq Q$ is the set of *accepting states*, and *transition function* δ maps $Q \times (\Sigma \cup \{\lambda\}) \times \Gamma$ to finite subsets

of $Q \times \Gamma^*$. An NPDA A is in *configuration* $c = (q, w, \gamma)$ if A is in state $q \in Q$ with $w \in \Sigma^*$ as remaining input, and $\gamma \in \Gamma^*$ on the pushdown store, the rightmost symbol of γ being the top symbol on the pushdown. We write

$$c = (q, aw, \gamma Z) \vdash_A (p, w, \gamma\beta),$$

if $(p, \beta) \in \delta(q, a, Z)$, for $a \in \Sigma \cup \{\lambda\}$, $w \in \Sigma^*$, $\gamma, \beta \in \Gamma^*$, and $Z \in \Gamma$. As usual, the reflexive transitive closure of $\vdash_A$ is denoted by $\vdash_A^*$, and the subscript A will be dropped from $\vdash_A$ and $\vdash_A^*$ whenever the meaning remains clear. The *language accepted* by A *with empty pushdown* is defined by

$$L(A) = \{\, w \in \Sigma^* \mid (q_0, w, Z_0) \vdash^* (q, \lambda, \lambda), \text{ for some } q \in Q \,\}.$$

Equivalently, the *language accepted* by A *with final state* is defined by

$$L_f(A) = \{\, w \in \Sigma^* \mid (q_0, w, Z_0) \vdash^* (q, \lambda, \gamma), \text{ for some } q \in F \text{ and } \gamma \in \Gamma^* \,\}.$$

As in the case of finite automata one can define deterministic (DPDA) and unambiguous pushdown automata (UPDA) in a straightforward way (cf. [19]). By definition every DPDA is a UPDA. NPDAs characterize the family of context-free languages defined by context-free grammars. This characterization carries over to UPDAs in the sense that the family of unambiguous context-free languages, generated by unambiguous context-free grammars, is equal to the family of languages accepted by UPDAs. Finally, the family of deterministic context-free languages are simply defined to be all languages accepted with final state by DPDAs (or equivalently by LR(k) context-free grammars). These three types of devices induce a strict hierarchy of language families [19].

3 (Un)ambiguous Finite Automata

Since regular languages have many representations in the world of finite automata, it is natural to investigate the succinctness of their representation by different types of automata in order to optimize the space requirements. Here we measure the costs of representations in terms of the states of a minimal automaton accepting a language. More precisely, the *simulation problem* is defined as follows:

- Given two classes of finite automata C_1 and C_2, how many states are sufficient and necessary in the worst case to simulate n-state automata from C_1 by automata from C_2?

In particular, we are interested in simulations between DFAs, UFAs, and NFAs. In order to compare these simulations the following relation from the literature, see, e.g., [57] is of use: if the transformation from an automaton from C_1 to an equivalent automaton from C_2 is polynomially bounded, i.e., there is a polynomial p such that for any n-state automaton from C_1 one finds an equivalent automaton from C_2 with at most $p(n)$ states, then we write $C_1 \leq_p C_2$.

In this case we consider this transformation to be cheap. In case $C_1 \leq_p C_2$ but $C_2 \not\leq_p C_1$, we say that C_1 is *(polynomially) separated* from C_2 and abbreviate this by $C_1 <_p C_2$. In other words, while the transformation of an automaton from C_1 into an equivalent C_2 automaton is cheap the converse transformation is expensive and exceeds any polynomial bound. Note that the non-polynomial bound for the transformation from right to left is in most cases not explicitly specified. In what follows we will see that in most cases one obtains separation results for most combinations of classes of finite automata.

3.1 Simulations

It is well known that to any NFA one can always construct an equivalent DFA [56]. This so-called *powerset construction*, where each state of the DFA is associated with a subset of NFA states, turned out to be optimal in general. That is, the bound on the number of states necessary for the construction is tight in the sense that for an arbitrary n there is always some n-state NFA which cannot be simulated by any DFA with strictly less than 2^n states [49,51,52]. So, NFAs can offer exponential savings in the number of states compared with DFAs. This gives rise to DFA $<_p$ NFA or more precisely to the following theorem.

Theorem 1 (NFA by DFA Simulation). *Let $n \geq 1$ and A be an n-state NFA. Then 2^n states are sufficient and necessary in the worst case for a DFA to accept $L(A)$.*

The situation for UFAs is similar. A first result on UFAs was shown in [61] proving a $2^{\Omega(\sqrt{n})}$ lower bound on the trade-off between NFAs and UFAs and between UFAs and DFAs. Hence DFA $<_p$ UFA and UFA $<_p$ NFA. Since DFAs are also unambiguous both results led a large gap between the lower and the upper bound of $2^n - 1$ (as for the case of NFAs in general, the dead state can be saved for UFAs). Later the lower bound for the transformation of an NFA to an equivalent UFA was improved to $2^{\Omega(n)}$ in [65] and finally, for every n, an n-state NFA was exhibited in [47]—see Figure 2—whose smallest equivalent UFA cannot do better in the number of states than the smallest equivalent DFA besides the aforementioned saving of the dead state. Hence a $2^n - 1$ tight bound

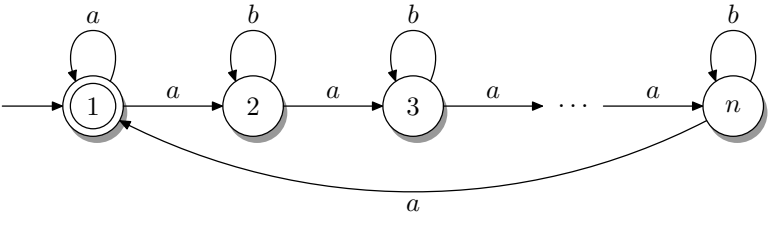

Fig. 2. Leung's NFA A_n with n states, for $n \geq 2$, accepting a language for which any equivalent UFA needs at least $2^n - 1$ states

on the trade-off between NFAs and UFAs was established. For the remaining UFA by DFA simulation it was mentioned in [65] that the well-known language $L_n = (a+b)^*a(a+b)^{n-1}$, that is the set of words over the alphabet $\{a,b\}$ whose nth last letter is an a, may serve as a witness for a 2^{n-1} lower bound (see Figure 3). Then this transformation problem was solved in [45] in a similar vain as the NFA to DFA transformation (providing UFAs with several initial states) obtaining a tight bound of 2^n in the exact number of states. Later this problem was reconsidered in [48] giving UFAs with a single initial state that reach the maximal trade-off when transformed into equivalent DFAs. Again exponential state savings in both cases are possible. These results are summarized as follows:

Theorem 2 (NFA by UFA and UFA by DFA Simulation). *Let $n \geq 1$ and A be an 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 $L(A)$.*

Proving lower bounds for NFAs is complicated in general. Several authors have introduced methods for proving such lower bounds; see, e.g., the fooling set technique [10], the extended fooling set technique [1,31], and the biclique edge cover technique [17]. Although the bounds provided by these techniques are not always tight and, in fact, can be arbitrarily worse compared to the nondeterministic state complexity, they give good results in many cases. For UFAs a lower bound method was already given in [61], which is based on a rank argument on certain matrices, which was further elaborated in, e.g., [47,48]. This method reads as follows—note that the rank technique shares the deficit of the previously mentioned lower bound techniques for NFAs that the provided bounds may not be tight in general:

Theorem 3 (Rank Method for UFAs Lower Bounds). *Let $L \subseteq \Sigma^*$ be a regular language and $\{ (x_i, y_i) \mid x_i, y_i \in \Sigma^* \text{ with } 1 \leq i \leq n \}$ a finite set of pairs of strings. Consider the $n \times n$ matrix $M = (m_{ij})$ over the field of characteristic 2 defined by $m_{ij} = 1$, if $x_i y_j \in L$, and $m_{ij} = 0$, otherwise. Then any UFA accepting L has at least the rank of M number of states.*

For the particular case of unary regular languages the situation is significantly different. The general problem of evaluating the costs of unary automata simulations was raised in [63], and has led to emphasize some relevant differences with the general case. For state complexity issues of unary finite automata *Landau's function*

$$F(n) = \max\{ \operatorname{lcm}(x_1, \cdots, x_k) \mid x_1, \ldots, x_k \geq 1 \text{ and } x_1 + \cdots + x_k = n \},$$

which gives the maximal order of the cyclic subgroups of the symmetric group on n elements, plays a crucial role. Here, lcm denotes the least common multiple. Since F depends on the irregular distribution of the prime numbers, we cannot expect to express $F(n)$ explicitly by n. In [43,44] the asymptotic growth rate

$$\lim_{n\to\infty} (\ln F(n)/\sqrt{n \cdot \ln n}) = 1$$

was determined, which for our purposes implies the (sufficient) rough estimate $F(n) \in e^{\Theta(\sqrt{n \cdot \ln n})}$. The following asymptotic tight bound on the unary NFA by DFA simulation was presented in [5,6]. Its proof is based on a normalform (Chrobak normalform) for unary NFAs introduced in [5]. Each n-state unary NFA can be replaced by an equivalent $O(n^2)$-state NFA consisting of an initial deterministic tail and some disjoint deterministic loops, where the automaton makes only a single nondeterministic decision after passing through the initial tail, which chooses one of the loops.

Theorem 4 (Unary NFA by DFA Simulation). *Let $n \geq 1$ 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 $L(A)$.*

Surprisingly the corresponding simulation questions on unary languages involving UFAs was investigated only recently in [53]. Based on a refined transformation, presented in [34], of UFAs into Chrobak normalform without increasing the number of states, the precise number of states for converting a UFA accepting a unary language into an equivalent DFA is determined by a more complicated variant of Landau's function, which is defined as

$$\begin{aligned}\tilde{F}(n) = \max\{\, \operatorname{lcm}(x_1, \ldots, x_k) \mid x_1, \ldots, x_k \geq 1, \text{ and } x_1 + \cdots + x_k = n \text{ and} \\ \exists f_1, \ldots f_k \text{ with } 0 \leq f_i \leq x_i - 1 \text{ such that} \\ \forall i, j \text{ with } i \neq j \text{ we have } f_i \neq f_j \bmod \gcd(x_i, x_j)\,\}.\end{aligned}$$

Here the additional condition compared to Landau's function is the criterion that forces a unary NFA in Chrobak normalform to be unambiguous. By involved calculations the function $\tilde{F}(n)$ is asymptotically estimated by $e^{\Theta(\sqrt[3]{n \cdot \ln^2 n})}$, which gives the following result on unary UFA by DFA simulation.

Theorem 5 (Unary UFA by DFA Simulation). *Let $n \geq 1$ and A be an n-state UFA 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 $L(A)$.*

What concerns the simulation of unary NFAs by UFAs? In fact, in [53] it was shown that one cannot do asymptotically better than in the unary NFAs to DFAs transformation. This nicely contrasts the results on the NFA by UFA and UFA by DFA simulation in general, given in Theorem 2, where in both cases an exponentially tight bound is reported. Here in the unary case easy calculations show that the UFAs to DFAs transformation is asymptotically better than the NFAs to UFAs conversion since $\tilde{F}(n) \in o(F(n))$, which intuitively means that UFAs are somehow "closer" to DFAs than NFAs.

Theorem 6 (Unary NFA by UFA Simulation). *Let $n \geq 1$ 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 UFA to accept $L(A)$.*

Very often UFAs are compared to a slight extension of DFAs, namely multiple-entry DFAs (MDFAs), which were defined in [8,68]. Here the sole guess appears at the beginning of the computation, that is, by choosing one out of k initial states. So, the nondeterminism is limited in its amount and in the situation at which it appears—it is worth mentioning that MDFAs are a special case of ambiguous finite automata, which are discussed in Section 4 in detail. Converting an MDFA with k initial states into a DFA by the powerset construction shows immediately that any reachable state contains at most k states of the MDFA. This gives an upper bound for the conversion. In [27] it has been shown that this upper bound is tight resulting in DFA $<_p$ MDFA or more precisely:

Theorem 7 (MDFA by DFA Simulation). *Let $n, k \geq 1$ with $k \leq n$ and A be an n-state MDFA with k entry states. Then $\sum_{i=1}^{k} \binom{n}{i}$ states are sufficient and necessary in the worst case for a DFA to accept $L(A)$.*

So, for $k = 1$ we obtain DFAs while for $k = n$ we are concerned with the special case that needs $2^n - 1$ states. Interestingly, NFAs can be exponentially concise over MDFAs. The following lower bound has been derived in [36].

Theorem 8 (NFA to MDFA Simulation). *Let $n \geq 1$ and A be an n-state NFA. Then $\Omega(2^n)$ states are necessary in the worst case for an MDFA to accept $L(A)$.*

For the trade-off between MDFAs and UFAs a tight bound in the exact number of states was shown in [46,48]. This nicely fits into the known upper and lower bound results presented earlier, and shows that even very limited use of nondeterminism can induce a dramatic increase in the number of states.

Theorem 9 (MDFA by UFA Simulation). *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 $L(A)$.*

Recently a variant of UFAs, so called *structurally unambiguous* finite automata were introduced and investigated in [46]. 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. Observe, that compared to the original definition of unambiguity the computations need *not* be accepting. Thus, unambiguity is a semantic concept, while structurally unambiguity is a syntactic one, which is independent on the choice of the set of final states. However, if there is only one final state, that is, $|F| = 1$, then a SUFA is also a UFA, but in general SUFAs may differ from UFAs. But what can be said about the relation of SUFAs to DFAs, UFAs, and MDFAs, in general? First of all every DFA is a SUFA and it is easy to see that this two automata classes are separated from each other, i.e, DFA $<_p$ SUFA, since the automaton depicted in Figure 3 is structurally unambiguous, and any equivalent DFA needs at least an exponential number of states. But we can do slightly better as we will see below. Moreover, every MDFA can be transformed

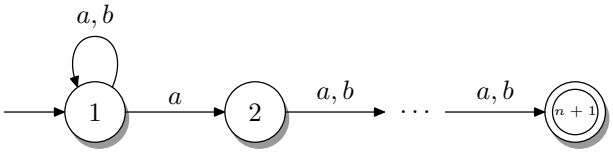

Fig. 3. UFA which is also a SUFA A_n with $(n+1)$-states accepting the set of all words having a letter a at the nth position, i.e., $L(A_n) = (a+b)^*a(a+b)^{n-1}$, which any equivalent DFA needs at least 2^{n-1} states

into an equivalent $O(n^2)$-state SUFA by simply making at most n copies of the MDFA and introducing a new initial state that is appropriately connected to these copies. It is easy to see that the constructed automaton is a SUFA, hence DFA $<_p$ SUFA and MDFA $\leq_p$ SUFA; recall that DFA $<_p$ MDFA holds true. The missing separations of MDFAs and SUFAs and of SUFAs and UFAs can be found in [46], and were proven by a single witness language only. In fact, the results presented there give tight bounds for the simulations in the exact number of states, and read as follows:

Theorem 10 (SUFA by DFA, UFA, or MDFA Simulation). *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 $L(A)$.*

3.2 Minimization Problems

We continue with some comments on a problem closely related to finding good lower bounds on the automata simulations for certain types of devices, namely the minimization problem. The study of the minimization problem for finite automata dates back to the early beginnings of automata theory—for further reading we refer to [35] and references therein. The decision version of the minimization problem, for short the *NFA-to-NFA minimization problem*, is defined as follows: given a NFA A and a natural number k in binary, that is, an encoding $\langle A, k\rangle$, is there an equivalent k-state NFA? This notation naturally generalizes to other types of finite automata, for example, the DFA-to-NFA minimization problem. It is well known that for a given n-state DFA one can efficiently compute an equivalent minimal automaton in $O(n \log n)$ time [28]. More precisely, the DFA-to-DFA minimization problem is complete for NL, even for DFAs without inaccessible states [4]. This is contrary to the nondeterministic case since the minimization problem for NFAs is known to be computationally hard [35]. The PSPACE-hardness result for NFAs was shown by a reduction from the union universality problem to the NFA-to-NFA minimization problem. For some further problems related to minimization we refer also to [17].

In order to better understand the very nature of nondeterminism one may ask for minimization problems for restricted types of finite automata such as, e.g., UFAs. Already in [35] it was shown that for UFAs some minimization problems

remain intractable. To be more precise, the UFA-to-UFA and the DFA-to-UFA minimization problems are NP-complete. Later in [50] it was shown that the minimization of finite automata equipped with a very small amount of nondeterminism is already computationally hard. To this end, a reduction from the NP-complete minimal inferred DFA problem [11,35] to the the minimization problems for MDFAs with a fixed number of initial states as well as for NFAs with fixed finite branching has been shown. Prior to this, the MDFA-to-DFA minimization problem in general was proven to be PSPACE-complete in [27]. Here the minimal inferred DFA problem [11] is defined as follows: given a finite alphabet Σ, two finite subsets $S, T \subseteq \Sigma^*$, and an integer k, is there a k-state DFA that accepts a language L such that $S \subseteq L$ and $T \subseteq \Sigma^* \setminus L$? Such an automaton can be seen as a consistent "implementation" of the sets S and T. Recently, the picture was completed in [2] by getting much closer to the tractability frontier for NFAs minimization. Interestingly it turned out, that unambiguity plays an important role in this characterization. There a class of NFAs is identified, the so called δ-nondeterministic finite automata (δNFA), such that the minimization problem for any class of finite automata that contains δNFAs is NP-hard, even if the input is given as a DFA. Here the class of δNFAs contains all NFAs A with the following properties: (1) the automaton A is a UFA, (2) the maximal product of the degrees of nondeterminism over the states in a possible computation is at most 2, and (3) there is at most on state q and a letter a such that the degree of nondeterminism of q and a is 2. It is worth mentioning that for every n-state δNFA there is an equivalent DFA with at most $O(n^2)$ states.

4 Quantified Ambiguity

The concept of unambiguity implied devices whose mode of operation is somehow in between determinism and nondeterminism. On the one hand, UFAs can be seen as DFAs that are allowed to guess in rejecting computations. On the other hand, UFAs are NFAs that are not allowed to guess in accepting computations. A natural generalization is to relax the condition that forbids guessing in accepting computations to allow a certain amount of guessing in accepting computations. This idea and a formalization of what is a certain amount brings us to the concept of *quantified ambiguity* [57].

For an NFA A, we define the ambiguity of a word w, denoted by $\mathsf{amb}_A(w)$, to be the number of different *accepting* computations of w. Note that a word w is in the language $L(A)$ if and only if the ambiguity of w is not zero. The ambiguity function $\mathsf{amb}_A : \mathbb{N} \to \mathbb{N}$ is defined such that $\mathsf{amb}_A(n)$ is the maximum of the ambiguities of words that are of length n or less. Here $\mathbb{N}$ refers to the set of natural numbers. Observe, that amb_A is nondecreasing by definition. This definition fits that for UFAs previously given, because an NFA A is unambiguous if the ambiguity of any word is either zero or one; in the latter case we may also say that A is a 1-*ambiguous* NFA. Moreover, automaton A is called *finitely* (*polynomial*, *exponential*, respectively) ambiguous if amb_A is bounded by a constant (polynomial, exponential, respectively) function f such that $\mathsf{amb}_A(n) \leq f(n)$,

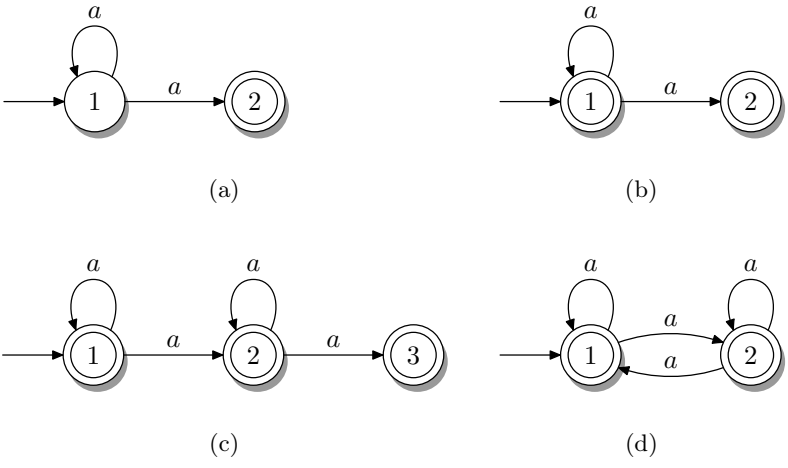

Fig. 4. NFAs A with different degrees of ambiguity: (a) UFA, (b) FNA with amb_A is constant 2, (c) PNA with amb_A is even linear, and (d) ENA. These automata drawings also nicely illustrate the structural characterizations of (strictly) polynomial and (strictly) exponential ambiguity presented in Theorem 11.

for every $n \in \mathbb{N}$—see Figure 5 for (unary) NFAs with different degrees of ambiguity. We abbreviate finitely (polynomial, exponential, respectively) automata by FNA (PNA, ENA, respectively). It is easy to see for any NFA A we have $\mathsf{amb}_A(n) \leq |Q|^n$, where Q is the state set of A, i.e., every NFA is exponential ambiguous. We mention in passing that in [4] necessary and sufficient structural conditions on NFAs were utilized to distinguish between exponential, polynomial, bounded, and k-bounded ambiguity, and it was shown that these *ambiguity problems*, i.e., determining whether the degree of ambiguity of a given NFA is exponential, polynomial, constantly bounded, k-bounded, where k is a fixed integer, or unambiguous are all NL-complete. These structural characterizations read as follows [33,57,71] (cf. Figure 4) 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 11 (Structural Characterization of NFAs Ambiguities). *Let A be an NFA with state set Q and input alphabet Σ, in which all states are useful.*[1] *Then we have the following structural characterizations of (strictly) finitely, polynomially, and exponentially ambiguities on finite automata:*

1. *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. *Automaton A is strictly polynomially ambiguous if and only if A is* not *exponentially ambiguous and there exists different states $p, q \in Q$ and a*

[1] A state q is *useful* if it is reachable from the initial state and one can reach at least one final state from q.

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. *Automaton A is finitely ambiguous if and only if A is neither strictly exponentially nor strictly polynomially ambiguous.*

Next we consider the relation between the types of ambiguity to the relative succinctness in the number of states for NFAs. This is still a vivid area of research, even after more than 30 years since one of the first results on UFAs simulations appeared in [62].

4.1 Ambiguity and the Succinctness of Representation

Once quantified ambiguity has come into play it is interesting to explore how several structural and computational parameters relate to the degree of ambiguity. Is a certain parameter independent of the ambiguity? If not, what are the precise relations? This subsection is devoted to discuss the connections between the degree of ambiguity and the succinctness of the representation, that is, the necessary number of states.

Recall that by definition

$$\text{DFA} \leq_p \text{UFA} \leq_p \text{FNA} \leq_p \text{PNA} \leq_p \text{ENA} =_p \text{NFA},$$

where $C_1 =_p C_2$ is a short hand notation for $C_1 \leq_p C_2$ and $C_2 \leq_p C_1$. The following separation results are known (not listing the already reported results on the relations between DFAs, UFAs, and NFAs): DFA $<_p$ UFA [45,48,61,65], UFA $<_p$ FNA [48,57,61], PNA $<_p$ NFA [32,47], and the intermediate relation between FNAs and PNAs turned out to be very complicated to separate. Several attempts to prove it failed, see, e.g., [32,57], until recently, where this long standing open problem was solved in the affirmative, resulting in FNA $<_p$ PNA [30]. These separation results can be summarized as follows:

Theorem 12 (Simulations of NFAs with Different Ambiguities). *The following separation results on NFA with different degrees of ambiguity are known:*

1. *For every $n \geq 1$, there is an n-state NFA A (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 true when changing FNA to UFA and UFA to DFA.*

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

Next we investigate the question whether there is a relation between ambiguity and the amount of nondeterminism used during the computation.

4.2 Ambiguity and the Amount of Nondeterminism

Nondeterminism has started to be seen as an additional limited resource at the disposal of time or space bounded computations in [7,37]. The concept of limited nondeterminism in finite automata is more generally studied in [12,38]. In the latter reference a bound on the number of nondeterministic steps allowed during a computation as well as on the maximal number of choices for every nondeterministic step is imposed. Since in a certain sense the degree of ambiguity restricts nondeterministic computations, it is suspenseful to explore the question whether there is a relation between the degree of ambiguity and the degree of nondeterminism.

Here, the nondeterminism is measured dynamically by counting the number of guesses an automaton has to make [13]. More precisely, for an NFA $A = (Q, \Sigma, \delta, q_0, F)$, the *amount of guessing* of a single move $\delta(q, a)$, for $q \in Q$ and $a \in \Sigma$, is defined to be $\log_2(|\delta(q, a)|)$. This concept is extended additively to computations by adding the amounts of the single steps. Then for each $w \in L(A)$ the amount of guessing, referred to $\mathsf{guess}_A(w)$, is the minimum over all accepting computations on w, and the guessing function guess_A is defined such that $\mathsf{guess}_A(n)$ is the maximum of the amounts of guessing of words in $L(A)$ that are of length n or less. Note that, in general, $\mathsf{guess}_A(n)$ is not an integer. If the NFA branches to at most two states in every step, then guess_A simply counts the number of nondeterministic steps. Moreover, guess_A counts a branch to 2^k successor states as equal to k branches to two successor states.

Concerning the relation between the amount of nondeterminism and the degree of ambiguity, it is illustrated in [13] that finite automata A with constant or linear nondeterminism, that is, guess_A is a constant or linear function, can be of all types UFA, FNA, PNA, and ENA. So, no prediction can be made about the degree of ambiguity. Figure 4 shows the four types of ambiguity in question for NFAs with a *linear* amount of nondeterminism, and Figure 5 depicts examples for NFAs with a *constant* amount of nondeterminism. The surprising result obtained in [13] revealed that the situation is different for the intermediate level of nondeterminism. The subtle relation between the two concepts is that an automaton with a non-constant but sublinear guessing function must have an infinite degree of ambiguity. Furthermore, it is shown that for each $k \geq 1$ there is, in fact, an NFA A with $\mathsf{guess}_A(n) \in \Theta(n^{1/k})$. The key result of [13] is the following trade-off lemma.

Lemma 13. *If A is an n-state NFA and $w \in L(A)$ is such that there is no word $v \in L(A)$ with $|v| < |w|$ and $\mathsf{guess}_A(v) \geq \mathsf{guess}_A(w)$, then*

$$\frac{n^{\mathsf{amb}_A(w)}(\mathsf{amb}_A(w)\,\mathsf{guess}_A(w) + 1)}{|w|} > 2^{-n}.$$

For a given NFA A the number 2^{-n} is a positive constant. So, the lemma can be interpreted such that an input that requires few nondeterminism at the same time causes a high degree of ambiguity. The next theorem [13] is an immediate consequence of the trade-off lemma.

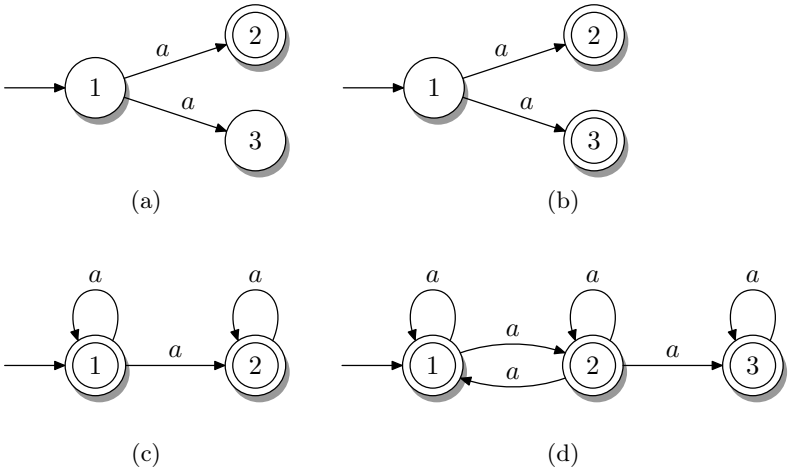

Fig. 5. NFAs A with a *constant* amount of nondeterminism and different degrees of ambiguity: (a) UFA, (b) FNA with amb_A is constant 2, (c) PNA with amb_A is linear, and (d) ENA

Theorem 14. *Every NFA with a non-constant but sublinear guessing function has an infinite degree of ambiguity.*

4.3 Finite Ambiguity and the Structure of Finite Automata

The characterizations of finite automata with different degrees of ambiguity already discussed in Theorem 11 are with respect to the inner structure of the finite automata. From this point of view the relation between the number of states and the amount of ambiguity are worth studying. Since the example given in Figure 4(d) shows that there is 2-state NFA with exponential ambiguity, the question for upper and lower bounds on the *finite* degree of ambiguity of n-state nondeterministic finite automata is of particular interest.

In connection with the questions asking for the decidability of the finiteness of a finitely generated monoid of matrices with entries in the natural numbers $\mathbb{N}$, (or in a larger semiring), and for an algorithm which computes the degree of an NFA, rough upper bounds have been derived in the late seventies. In three papers it was independently obtained that the degree of ambiguity of a finitely ambiguous n-state NFA with input alphabet Σ is at most (1) $n^{2^{3^{3n^2+1}}}$, (2) $n^{f(n,|\Sigma|)}$, where f is a recursive function, and (3) $2^n n^{2n} 2^{4n^3}$. A systematic study of this problem started in [71], where the next theorem has been shown.

Theorem 15. *Let A be an n-state FNA. Then amb_A is at most $5^{n/2}n^n$.*

By reduction, Theorem 15 can be generalized to NFAs with λ-moves. The upper bound has slightly be improved in the context of formal power series. In [39] it was decreased to $2^{1+k_2 n}n^n$, where $k_2 < 0.7956$. In order to compare the results note that $5^{n/2}n^n = 2^{k_1 n}$, where $k_1 \approx 1.161$.

A so-called *chain-NFA* has a certain inner structure. Roughly speaking, the transition graph representing a chain-NFA consists of strongly connected components (SCC), say $Q_1, Q_2, \ldots, Q_k$ acting as chain links, such that there is a single initial state in Q_1, a single accepting state in Q_k, and exactly one transition from Q_i to Q_{i+1}, for $1 \leq i \leq k-1$. A chain-NFA can be seen as a sequence of modules which have to be passed through in a one-way fashion. The upper bound for chain-NFAs is much lower than for arbitrary NFAs [71]:

Theorem 16. *Let A be an n-state chain FNA. Then amb_A is at most n^n.*

Now we turn also to lower bounds. In order to capture classes C of NFAs defined with respect to their inner structures more generally, let $C_f \subseteq C$ denote the subclass including exactly all FNAs from C, and set

$$\mathsf{amb}_C(n,m) = \max\{\, \mathsf{amb}_A(m) \mid A \in C_f \text{ and } A \text{ has } n \text{ states} \,\}.$$

By Theorem 15 $\mathsf{amb}_C(n,m)$ is at most $5^{n/2}n^n$. In Table 1 we summarize the results for several subclasses of NFAs. The proofs can be found in [70,71]. Notably, for several subclasses upper and lower bounds are tight (in the order of magnitude $2^{\Theta(n)}$).

5 (Un)ambiguous Pushdown Automata

In this section we consider the descriptional capacity of unambiguous as well as finitely ambiguous pushdown automata. In particular, the relative succinctness of those machines among each other and to finite automata are discussed. It turned out that the situation is completely different compared to the previously presented finite automata simulations. First, these pushdown devices induce a strict hierarchy of languages [19], which was not the case for finite automata. Moreover, and even more importantly, we will come across a qualitatively new

Table 1. Upper and lower bounds on the ambiguities for several subclasses of NFAs in relation to the number of states n

Automata class C	Ambiguity amb_A	
	lower bound	upper bound
NFAs	$2^{1.0221 \cdot n}$, for $n = 0 \bmod 64$	$2^{1.161 \cdot n} n^n$
chain-NFAs	$2^{1.0221 \cdot n}$, for $n = 0 \bmod 64$	n^n
chain-NFAs with 2 SCCs	2^{n-2}, for $n \geq 2$	2^{n-1}
NFAs for finite languages	$\binom{n}{\lfloor (n+1)/2 \rfloor}$	$\binom{n}{\lfloor (n+1)/2 \rfloor}$
NFAs for unary languages	2^{n-1}	2^{n-1}

phenomenon first observed in [51], the so-called *non-recursive trade-offs*. That is, there is no recursive function bounding the succinctness gap (for non-trivial simulations). Before we report on results we briefly have to discuss some issues on measuring the size of pushdown automata.

Measuring the size of a pushdown automaton by its number of states, as is done for finite automata, is clearly ineligible. It is well known that every pushdown automaton can effectively be converted into an equivalent one having just one sole state [29]. But, in general, one has to pay with an increase in the number of stack symbols, and determinism or unambiguity is not preserved. For DPDAs accepting by empty pushdown, the computational capacity is known to increase strictly with the number of states [19]. So, measuring the size of a (deterministic) pushdown automaton by its number of stack symbols is also too crude. In fact, it is also possible to reduce the number of stack symbols if one pays with an increase in the number of states. The precise relations between states and stack symbols have been shown in [15] and [16]. So, the number of states as well as the number of stack symbols have to be considered to measure the size of a pushdown automaton. But even their product is still insufficient. For example, for all integers $n \geq 1$ the language $L_n = (a^n)^*$ can be accepted by a pushdown automaton with two states and two stack symbols that, in one move, is able to push n symbols onto the stack. So, in addition, we have to take into account the lengths of the right-hand sides of the transition rules which can get long when a pushdown automaton pushes lots of symbols during single transitions. Therefore, the size of a pushdown automaton $A = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ is measured as $|Q| \cdot |\Sigma| \cdot |\Gamma| \cdot h$, where h is the length of the longest word pushed in a single transition.

5.1 Simulations

Here we present some fundamental results in connection with the representation of *regular languages by pushdown automata*. In [64] the decidability of regularity for DPDAs has been shown by a deep proof. This effective procedure revealed the following upper bound for the trade-off in descriptional complexity when DPDAs accepting regular languages are converted into DFAs. Given a DPDA with $n > 1$ states and $t > 1$ stack symbols that accepts a regular language. Then the number of states which is sufficient for an equivalent DFA is bounded by an expression of the order $t^{n^{n^n}}$. Later this triple exponential upper bound has been improved by one level of exponentiation in [66].

Theorem 17 (DPDA by DFA Simulation). *Let A be a deterministic pushdown automaton with n states, t stack symbols, and h is the length of the longest word pushed in a single transition. If $L(A)$ is regular then $2^{2^{O(n^2 \log n + \log t + \log h)}}$ states are sufficient for a DFA to accept the language $L(A)$.*

In the levels of exponentiation this bound is tight, since the following double exponential lower bound has been obtained in [51]. It is open whether the precise lower bound or the precise upper bound can be improved in order to obtain matching bounds.

Theorem 18 (DPDA by DFA Simulation). *Let $n \geq 1$. Then there is a language L_n accepted by a deterministic pushdown automaton of size $O(n^3)$, and each equivalent DFA has at least 2^{2^n} states.*

It is clear that these bounds on the simulation by DFAs implicitly imply also bounds for NFAs. While we deal with finite automata simulations of NPDAs in the next subsection, to our knowledge the remaining simulation of UPDAs accepting regular languages by finite automata is an open problem in the sense that (1) it is not known whether regularity for UPDAs is decidable and (2) whether the trade-off between UPDAs and finite automata is non-recursive or not. Note, that the succinctness result for UPDAs and finite automata corresponds to the decidability result of regularity for UPDAs as follows: assume that $f(n)$ is an upper bound on the number of states for the simulation of an n-size UPDA by a finite automaton. To decide whether a UPDA A accepts a regular language, just enumerate all finite automata with fewer states than $f(n)$ and check equivalence with A, which is decidable by [58].

Finally let us mention that the situation on simulations involving pushdown automata is different again, to the general case, when pushdown automata accepting unary languages are considered. This is somehow comparable to the case of finite automata. It is well known that every unary context-free language is regular [9]. From the viewpoint of descriptional complexity, unary DPDAs and NPDAs have been investigated in, e.g., [54,55]. For further results concerning simulations involving pushdown automata we refer to the comprehensive survey [26].

5.2 Non-recursive Trade-Offs

Now we present some pushdown automata simulations that cause non-recursive trade-offs. To keep the presentation simple, we introduce only the necessary terminology, a more general and elaborately treatment is given in [26]. Assume that we have two classes of automata C_1 and C_2 which induce a non-empty intersection on the corresponding languages accepted by these types of devices. Then a *non-recursive trade-off* between C_1 and C_2, means that there is an infinite family of languages accepted by both device types, such that when changing from the C_1-descriptors to equivalent minimal C_2-descriptors, the blow-up in *size* cannot be bounded by any recursive function. A cornerstone of descriptional complexity is the result of Meyer and Fischer [51] who showed for the first time a non-recursive trade-off. It appears between context-free grammars or equivalently NPDAs and finite automata. In Theorem 17 we have seen that the trade-off between DPDAs and finite automata is recursive, where the proof relies on the fact that regularity is decidable for DPDAs. This goes hand in hand with the undecidability of regularity for NPDAs

Theorem 19 (NPDA by FA Simulation). *The trade-off between NPDAs and finite automata is non-recursive.*

Even the computational power of DPDAs is not enough, similarly, the non-recursive trade-off between NPDAs and DPDAs follows (cf. [61]):

Theorem 20 (NPDA by DPDA Simulation). *The trade-off between NPDAs and DPDAs is non-recursive.*

Most proofs of non-recursive trade-offs that appear in the literature basically rely on one of two different proof schemes. One fundamental technique is due to Hartmanis [21]. In [22] a generalization is developed that relates semi-decidability to trade-offs and a slightly generalized and unified form of this technique can be found in [40]. When applying these techniques very often non-semi-decidable properties of Turing machines are utilized by encoding complex Turing machine computations in small automata [20]. To this end encodings of *(in)valid computations of Turing machines* and variants thereof are considered. Nevertheless, even simpler proof schemes only using (full) TRIO-closure properties of the underlying formal language families and semi-decidability of certain decision problems were recently developed and applied in [18]. Recall that a formal language family is called a *TRIO* (*full TRIO*, respectively) if it is closed under λ-free morphism (general morphism, respectively), inverse morphism, and intersection with regular languages [29].

Here we are particularly interested in the remaining trade-offs caused by the devices "in between" NPDAs and FAs, that is in the trade-offs between NPDAs and UPDAs as well as between UPDAs and DPDAs. The next result from [67] compares DPDAs and unambiguous context-free grammars or equivalently UPDAs. It exploits the following crucial result on the size of DPDAs. If for some DPDA A with state set Q and set of stack symbols Γ the string w is the shortest string such that wa and wb are accepted, then there is a positive constant k such that $|Q| \cdot |\Gamma| \geq (\log |w|)^k$. This can be used to show the next non-recursive trade-off:

Theorem 21 (UPDA by DPDA Simulation). *The trade-off between UPDAs and DPDAs is non-recursive.*

The remaining trade-off between NPDAs and UPDAs has been shown in [62].

Theorem 22 (NPDA by UPDA Simulation). *The trade-off between NPDAs and UPDAs is non-recursive.*

5.3 Bounded Ambiguity and Bounded Nondeterminism

The quantitative study of nondeterminism in context-free languages originates from [69], where two measures for the amount of nondeterminism in pushdown automata are proposed. By bounding the number of nondeterministic steps dependent on the length of the input, a hierarchy of three classes is obtained. A modification of the measure can be found in [59]. The second measure depends on the depth of the directed acyclic graph that represents a given pushdown automaton. The corresponding proof of an infinite nondeterministic hierarchy of properly included classes is completed in [60]. The so-called *branching* as measure of nondeterminism in connection with pushdown automata, introduced for finite automata [12], is studied in [14,23]. In [14] lower bounds for the minimum

amount of nondeterminism to accept certain context-free languages are established. Pushdown automata with limited nondeterminism are investigated in [41] from the viewpoint of context-dependent nondeterminism, and in [42] from the viewpoint of regulated nondeterminism.

Considering a computation of a pushdown automaton we call a single step nondeterministic if the automaton has more than one choice for its move. The *branching* of the step is defined to be the number of choices. The branching of a computation is the product of the branchings of all steps of the computation. In order to be more precise, let A be a NPDA, then we define the branching branch_A as follows:

1. The *branching of a configuration* c is $\mathsf{branch}_A(c) = |\{\, c' \mid c \vdash c' \,\}|$.
2. A sequence of configurations (computation) $C = c_0 \vdash c_1 \vdash \cdots \vdash c_k$ has branching
$$\prod_{i=0}^{k-1} \mathsf{branch}_A(c_i).$$
3. For words $w \in L(A)$ we define the branching as
$$\mathsf{branch}_A(w) = \min\{\, \mathsf{branch}_A(C) \mid C \text{ is an accepting computation on } w \,\}.$$
4. Finally, let the *branching of* A be $\mathsf{branch}_A = \sup\{\, \mathsf{branch}_A(w) \mid w \in L(A) \,\}$.

In an NPDA A whose branching is bounded by a constant k all computations can be cut off when the branching exceeds k without changing the accepted language. So, the branching of an NPDA tells us up to which width the computation tree of some input word has to be examined until an accepting computation is found. The branching of a word is not necessarily equal to the number of computations on this word. Moreover, there is no relation between the ambiguity and the branching of an NPDA, since there are NPDAs A with $\mathsf{amb}_A = 1$ and $\mathsf{branch}_A = \infty$ as well as with $\mathsf{amb}_A = \infty$ and $\mathsf{branch}_A = 1$.

In [3] and [23] a two-dimensional infinite hierarchy dependent on the finite degree of ambiguity and the finite branching in between the deterministic and nondeterministic context-free languages is obtained. Denote the classes of pushdown automata with ambiguity and branching bounded by a constant k by $\mathrm{NPDA}(\mathsf{amb} \leq k)$ and $\mathrm{NPDA}(\mathsf{branch} \leq k)$. If both resources are bounded at the same time, we write $\mathrm{NPDA}(\mathsf{amb} \leq k, \mathsf{branch} \leq k')$. Clearly, for every k, we have the inclusions[2]
$$\mathrm{NPDA}(\mathsf{amb} \leq k) \subseteq \mathrm{NPDA}(\mathsf{amb} \leq k+1)$$
and
$$\mathrm{NPDA}(\mathsf{branch} \leq k) \subseteq \mathrm{NPDA}(\mathsf{branch} \leq k+1).$$
In [23] it is proven that $\mathrm{NPDA}(\mathsf{branch} \leq k) \subseteq \mathrm{NPDA}(\mathsf{amb} \leq k)$. Moreover, it is shown that all of these inclusions are proper. This means that every time the

[2] In abuse of notation the three inclusions yet to come are meant w.r.t. the languages generated by the corresponding pushdown automata with limited ambiguity and/or branching.

allowed amount of ambiguity or the allowed amount of nondeterminism in an NPDA is increased by just one, a more powerful device is obtained. Intuitively, the corresponding language families are close together. Nevertheless, there are non-recursive trade-offs between levels of the hierarchy. In [3] the next theorem has been shown.

Theorem 23 (NPDA with Bounded Ambiguity Simulation). *Let $k \geq 1$. Then the trade-off between NPDA($\mathsf{amb} \leq k+1$) and NPDA($\mathsf{amb} \leq k$) is non-recursive.*

The result has been generalized in [23]:

Theorem 24 (NPDA with Bounded Ambiguity and Bounded Branching by NPDA with Bounded Ambiguity Simulation). *Let $k \geq 1$. Then the trade-offs between*

1. *NPDA($\mathsf{amb} \leq k+1, \mathsf{branch} \leq k+1$) and NPDA($\mathsf{amb} \leq k$), and*
2. *NPDA($\mathsf{amb} \leq 1, \mathsf{branch} \leq k+1$) and NPDA($\mathsf{branch} \leq k$)*

are non-recursive.

Finally, in addition to the non-recursive trade-offs a nontrivial recursive trade-off is shown in [23].

Theorem 25 (NPDA with Bounded Branching by NPDA with Bounded Ambiguity and Branching Simulation). *Let $k \geq 1$ and A be an n-size NPDA belonging to NPDA($\mathsf{branch} \leq k$). Then $2^{O(n)}$ states are sufficient for an NPDA from NPDA($\mathsf{amb} \leq k, \mathsf{branch} \leq k$) to accept $L(A)$.*

References

1. Birget, J.C.: Intersection and union of regular languages and state complexity. Inform. Process. Lett. 43, 185–190 (1992)
2. Björklund, H., Martens, W.: The tractability frontier for NFA minimization. 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)
3. Borchardt, I.: Nonrecursive tradeoffs between context-free grammars with different constant ambiguity. Diploma thesis, Universität Frankfurt (1992) (in German)
4. Cho, S., Huynh, D.T.: The parallel complexity of finite-state automata problems. Inform. Comput. 97, 1–22 (1992)
5. Chrobak, M.: Finite automata and unary languages. Theoret. Comput. Sci. 47, 149–158 (1986)
6. Chrobak, M.: Errata to "finite automata and unary languages". Theoret. Comput. Sci. 302, 497–498 (2003)
7. Fischer, P.C., Kintala, C.M.R.: Real-time computations with restricted nondeterminism. Math. Systems Theory 12, 219–231 (1979)
8. Gill, A., Kou, L.T.: Multiple-entry finite automata. J. Comput. System Sci. 9, 1–19 (1974)

9. Ginsburg, S., Rice, H.G.: Two families of languages related to ALGOL. J. ACM 9(3), 350–371 (1962)
10. Glaister, I., Shallit, J.: A lower bound technique for the size of nondeterministic finite automata. Inform. Process. Lett. 59, 75–77 (1996)
11. Gold, E.M.: Complexity of automaton identification from given data. Inform. Control 37, 302–320 (1978)
12. Goldstine, J., Kintala, C.M.R., Wotschke, D.: On measuring nondeterminism in regular languages. Inform. Comput. 86, 179–194 (1990)
13. Goldstine, J., Leung, H., Wotschke, D.: On the relation between ambiguity and nondeterminism in finite automata. Inform. Comput. 100, 261–270 (1992)
14. Goldstine, J., Leung, H., Wotschke, D.: Measuring nondeterminism in pushdown automata. J. Comput. System Sci. 71, 440–466 (2005)
15. Goldstine, J., Price, J.K., Wotschke, D.: On reducing the number of states in a PDA. Math. Systems Theory 15, 315–321 (1982)
16. Goldstine, J., Price, J.K., Wotschke, D.: On reducing the number of stack symbols in a PDA. Math. Systems Theory 26, 313–326 (1993)
17. Gruber, H., Holzer, M.: Finding lower bounds for nondeterministic state complexity is hard (extended abstract). In: Ibarra, O.H., Dang, Z. (eds.) DLT 2006. LNCS, vol. 4036, pp. 363–374. Springer, Heidelberg (2006)
18. Gruber, H., Holzer, M., Kutrib, M.: On measuring non-recursive trade-offs. In: Descriptional Complexity of Formal Systems (DCFS 2009), pp. 187–198. Otto-von-Guericke-Universität Magdeburg (2009)
19. Harrison, M.A.: Introduction to Formal Language Theory. Addison-Wesley, Reading (1978)
20. Hartmanis, J.: Context-free languages and Turing machine computations. In: Proc. Symposia in Applied Mathematics, vol. 19, pp. 42–51 (1967)
21. Hartmanis, J.: On the succinctness of different representations of languages. SIAM J. Comput. 9, 114–120 (1980)
22. Hartmanis, J.: On Gödel speed-up and succinctness of language representations. Theoret. Comput. Sci. 26, 335–342 (1983)
23. Herzog, C.: Pushdown automata with bounded nondeterminism and bounded ambiguity. Theoret. Comput. Sci. 181, 141–157 (1997)
24. Holzer, M., Kutrib, M.: Descriptional and computational complexity of finite automata. In: Dediu, A.H., Ionescu, A.M., Martín-Vide, C. (eds.) LATA 2009. LNCS, vol. 5457, pp. 23–42. Springer, Heidelberg (2009)
25. Holzer, M., Kutrib, M.: Nondeterministic finite automata - Recent results on the descriptional and computational complexity. Int. J. Found. Comput. Sci. 20, 563–580 (2009)
26. Holzer, M., Kutrib, M.: Descriptional complexity – An introductory survey. In: Scientific Applications of Language Methods. Imperial College Press, London (to appear, 2010)
27. Holzer, M., Salomaa, K., Yu, S.: On the state complexity of k-entry deterministic finite automata. J. Autom., Lang. Comb. 6, 453–466 (2001)
28. Hopcroft, J.E.: An $n \log n$ algorithm for minimizing the state in a finite automaton. In: The Theory of Machines and Computations, pp. 189–196. Academic Press, London (1971)
29. Hopcroft, J.E., Ullman, J.D.: Introduction to Automata Theory, Languages, and Computation. Addison-Wesley, Reading (1979)
30. Hromkovič, J., Schnitger, G.: Ambiguity and communication. In: Theoretical Aspects of Computer Science (STACS 2009), Dagstuhl, Germany. LIPICS, vol. 3, pp. 107–118 (2009)

31. Hromkovič, J.: Communication Complexity and Parallel Computing. Springer, Heidelberg (1997)
32. Hromkovič, J., Seibert, S., Karhumäki, J., Klauck, H., Schnitger, G.: Communication complexity method for measuring nondeterminism in finite automata. Inform. Comput. 172, 202–217 (2002)
33. Ibarra, O.H., Ravikumar, B.: On sparseness, ambiguity and other decision problems for acceptors and transducers. In: Monien, B., Vidal-Naquet, G. (eds.) STACS 1986. LNCS, vol. 210, pp. 171–179. Springer, Heidelberg (1986)
34. Jiang, T., McDowell, E., Ravikumar, B.: The structure and complexity of minimal NFA's over a unary alphabet. Int. J. Found. Comput. Sci. 2, 163–182 (1991)
35. Jiang, T., Ravikumar, B.: Minimal NFA problems are hard. SIAM J. Comput. 22, 1117–1141 (1993)
36. Kappes, M.: Descriptional complexity of deterministic finite automata with multiple initial states. J. Autom., Lang. Comb. 5, 269–278 (2000)
37. Kintala, C.M.R.: Computations with a Restricted Number of Nondeterministic Steps. PhD thesis, Pennsylvania State University (1977)
38. Kintala, C.M.R., Wotschke, D.: Amounts of nondeterminism in finite automata. Acta Inform. 13, 199–204 (1980)
39. Kuich, W.: Finite automata and ambiguity. Report 253 of the IIG, Technische Universität Graz (1988)
40. Kutrib, M.: The phenomenon of non-recursive trade-offs. Int. J. Found. Comput. Sci. 16, 957–973 (2005)
41. Kutrib, M., Malcher, A.: Context-dependent nondeterminism for pushdown automata. Theoret. Comput. Sci. 376, 101–111 (2007)
42. Kutrib, M., Malcher, A., Werlein, L.: Regulated nondeterminism in pushdown automata. Theoret. Comput. Sci. 410, 3447–3460 (2009)
43. Landau, E.: Über die Maximalordnung der Permutationen gegebenen Grades. Archiv der Math. und Phys. 3, 92–103 (1903)
44. Landau, E.: Handbuch der Lehre von der Verteilung der Primzahlen. Teubner (1909)
45. Leiss, E.L.: Succinct representation of regular languages by Boolean automata. Theoret. Comput. Sci. 13, 323–330 (1981)
46. Leung, H.: Structurally unambiguous finite automata. In: Ibarra, O.H., Yen, H.-C. (eds.) CIAA 2006. LNCS, vol. 4094, pp. 198–207. Springer, Heidelberg (2006)
47. Leung, H.: Separating exponentially ambiguous finite automata from polynomially ambiguous finite automata. SIAM J. Comput. 27, 1073–1082 (1998)
48. Leung, H.: Descriptional complexity of NFA of different ambiguity. Int. J. Found. Comput. Sci. 16, 975–984 (2005)
49. Lupanov, O.B.: A comparison of two types of finite sources. Problemy Kybernetiki 9, 321–326 (1963) (in Russian); German translation: Über den Vergleich zweier Typen endlicher Quellen. Probleme der Kybernetik 6, 328–335 (1966)
50. Malcher, A.: Minimizing finite automata is computationally hard. Theoret. Comput. Sci. 327, 375–390 (2004)
51. Meyer, A.R., Fischer, M.J.: Economy of description by automata, grammars, and formal systems. In: Symposium on Switching and Automata Theory (SWAT 1971), pp. 188–191. IEEE, Los Alamitos (1971)
52. Moore, F.R.: On the bounds for state-set size in the proofs of equivalence between deterministic, nondeterministic, and two-way finite automata. IEEE Trans. Comput. 20, 1211–1214 (1971)
53. Okhotin, A.: A study of unambiguous finite automata over a one-letter alphabet. TUCS Technical Report No 951, Turku Centre for Computer Science (2009)

54. Pighizzini, G.: Deterministic pushdown automata and unary languages. In: Ibarra, O.H., Ravikumar, B. (eds.) CIAA 2008. LNCS, vol. 5148, pp. 232–241. Springer, Heidelberg (2008)
55. Pighizzini, G., Shallit, J., Wang, M.W.: Unary context-free grammars and pushdown automata, descriptional complexity and auxiliary space lower bounds. J. Comput. System Sci. 65, 393–414 (2002)
56. Rabin, M.O., Scott, D.: Finite automata and their decision problems. IBM J. Res. Dev. 3, 114–125 (1959)
57. Ravikumar, B., Ibarra, O.H.: Relating the type of ambiguity of finite automata to the succinctness of their representation. SIAM J. Comput. 18, 1263–1282 (1989)
58. Salomaa, A., Soittola, M.: Automata-theoretic Aspects of Formal Power Series. Springer, Heidelberg (1978)
59. Salomaa, K., Yu, S.: Limited nondeterminism for pushdown automata. Bull. EATCS 50, 186–193 (1993)
60. Salomaa, K., Yu, S.: Measures of nondeterminism for pushdown automata. J. Comput. System Sci. 49, 362–374 (1994)
61. Schmidt, E.M.: Succinctness of Dscriptions of Context-Free, Regular and Finite Languages. PhD thesis, Cornell University, Ithaca, NY (1978)
62. Schmidt, E.M., Szymanski, T.G.: Succinctness of descriptions of unambiguous context-free languages. SIAM J. Comput. 6, 547–553 (1977)
63. Sipser, M.: Lower bounds on the size of sweeping automata. J. Comput. System Sci. 21, 195–202 (1980)
64. Stearns, R.E.: A regularity test for pushdown machines. Inform. Control 11, 323–340 (1967)
65. Stearns, R.E., Hunt III, H.B.: On the equivalence and containment problems for unambiguous regular expressions, regular grammars, and finite automata. SIAM J. Comput. 14, 598–611 (1985)
66. Valiant, L.G.: Regularity and related problems for deterministic pushdown automata. J. ACM 22, 1–10 (1975)
67. Valiant, L.G.: A note on the succinctness of descriptions of deterministic languages. Inform. Control 32, 139–145 (1976)
68. Veloso, P.A.S., Gill, A.: Some remarks on multiple-entry finite automata. J. Comput. System Sci. 18, 304–306 (1979)
69. Vermeir, D., Savitch, W.: On the amount of nondeterminism in pushdown automata. Fund. Inform. 4, 401–418 (1981)
70. Weber, A.: Über die Mehrdeutigkeit und Wertigkeit von endlichen Automaten und Transducern. Dissertation, Institut für Informatik, Johann Wolfgang Goethe-Universität Frankfurt am Main (1987) (in German)
71. Weber, A., Seidl, H.: On the degree of ambiguity of finite automata. Theoret. Comput. Sci. 88, 325–349 (1991)