Hot Current Topics of Descriptional Complexity

Martin Kutrib[1(✉)], Nelma Moreira[2], Giovanni Pighizzini[3], and Rogério Reis[2]

[1] Institut für Informatik, Universität Giessen, Arndtstr. 2, 35392 Giessen, Germany
kutrib@informatik.uni-giessen.de
[2] CMUP and DCC, Faculdade de Ciências da Universidade do Porto,
Rua do Campo Alegre, 4169-007 Porto, Portugal
{nelma.moreira,rogerio.reis}@fc.up.pt
[3] Dipartimento di Informatica, Università degli Studi di Milano, via Celoria 18,
20133 Milano, Italy
pighizzini@di.unimi.it

Preamble

Descriptional complexity has historically been a multidisciplinary area of study, with contributions from automata theory, computational complexity, cryptography, information theory, probability, statistics, pattern recognition, machine learning, computational learning theory, computer vision, neural networks, formal languages and other fields. Some basic questions are: How succinctly can a descriptional system represent objects (for example, encoded as formal languages) in comparison with other descriptional systems? What is the maximal size trade-off when changing from one system to another, and can it be achieved?

Since the late nineties the scope of the IFIP Working Group 1.02 encompasses all aspects of descriptional complexity, both in theory and application. The formal orientation suggested its establishment under the head of the IFIP Technical Committee TC1 on Foundations of Computer Science.

The topics of the working group include but are not limited to descriptional complexity of formal systems and structures, various measures of descriptional complexity of automata, grammars, languages and of related systems, trade-offs between descriptional complexity and mode of operation, circuit complexity of Boolean functions and related measures, succinctness of description of (finite) objects, descriptional complexity in resource bounded or structure bounded environments, structural complexity, descriptional complexity of formal systems for applications (for example, software reliability, software and hardware testing, modeling of natural languages), descriptional complexity aspects of nature motivated (bio-inspired) architectures and unconventional models of computing.

Furthermore, the Working Group tries to promote interaction and the exchange of information across traditional discipline boundaries and to provide a point of contact for all researchers in all disciplines interested in descriptional complexity and its applications.

Published by Springer Nature Switzerland AG 2021
M. Goedicke et al. (Eds.): Advancing Research in Information and Communication Technology,
IFIP AICT 600, pp. 3–28, 2021. https://doi.org/10.1007/978-3-030-81701-5_1

Here, we first address the basic ideas and concepts of descriptional complexity from a general abstract perspective. Then we select some recent trends in the area, discuss problems, results, and open questions. In particular, we address operational state complexity, that is, the size impact of decomposing formal systems, a bridge between descriptional and computational complexity, where the size of two-way finite automata is related to the L versus NL problem, the descriptional complexity of so-called limited automata which are Turing machines with rewriting restrictions, and look at parameterized nondeterminism in finite automata (that may change their mind). Then we are interested in the question to what extent the descriptional capacity can be boosted by transductions. Finally, we turn to enumerations and average complexity. The results presented are not proved but we merely draw attention to the overall picture and some of the main ideas involved.

Our tour on the subjects covers some current hot topics in the field of descriptional complexity. It obviously lacks completeness and it reflects our personal view of what constitute some of the most interesting links to descriptional complexity theory. In truth there is much more to the field than can be summarized here.

1 Descriptional Complexity: Idea and Basic Concepts

Since the early days of theoretical computer science the relative succinctness of different representations of (sets of) objects by formal systems have been a subject of intensive research. An obvious choice to encode the objects is by strings over a finite number of different symbols. Then a set of objects is a set of strings. To move closer to a machinery that can be used for the studies, a string is called a word and a set of words is said to be a formal language. A formal language can be described by several means, for example, by automata, grammars, rewriting systems, equation systems, etc. In general, such a descriptional system is a set of finite descriptors for languages. Core questions of descriptional complexity are "How succinctly (related to a size complexity measure) can a system represent a formal language in comparison with other systems?" and "What is the maximum trade-off when the representation is changed from one descriptional system to another, and can this maximum be achieved?"

Descriptional complexity has historically been a multidisciplinary area of study, with contributions from very different areas of computer science such as, for example, automata and formal language theory, computational complexity, cryptography, information theory, etc. The approach to analyze the size of systems as opposed to the computational power seems to originate from Stearns [87] who studied the relative succinctness of regular languages represented by deterministic finite automata and deterministic pushdown automata. In the classification of automata, grammars, and related (formal) systems it turned out that the gain in economy of description heavily depends on the considered systems. For instance, it is well known that nondeterministic finite automata (NFA) can be converted into equivalent deterministic finite automata (DFA) of at most

exponential size. The underlying construction is probably one of the best-known results on descriptional complexity: by this so-called powerset construction, each state of the DFA is associated with a subset of NFA states [81]. Moreover, the construction turned out to be optimal. 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 [64,68,70]. For deterministic pushdown automata accepting a regular language, we know that they can be converted into equivalent finite automata of at most doubly-exponential size [88]. In the levels of exponentiation this bound is tight. In [68] a double exponential lower bound has been obtained. The precise bound is still an open problem. In contrast, if we replace "deterministic pushdown automata" with "nondeterministic pushdown automata" then we are faced with the phenomenon of non-recursive trade-offs. That is, the maximum size blow-up cannot be bounded by any recursive function. In other words, when the size trade-off from one descriptional system to another is non-recursive, one can choose an arbitrarily large computable function f but the gain in economy of description eventually exceeds f when changing from the former system to the latter. Essentially, this means that the gain in economy of description can be arbitrary and, thus, the achievable benefit in description length is of arbitrary size. This cornerstone of descriptional complexity theory originates from the seminal paper by Meyer and Fischer [68]. Non-recursive trade-offs usually sprout at the wayside of the crossroads of (un)decidability, and in many cases proving such trade-offs apparently requires ingenuity and careful constructions.

Nowadays, descriptional complexity has become a large and widespread area. Classical main branches are, for example, mutual simulations, state complexity of operations, whose systematic study was initiated in [92], magic numbers, a research field initiated in [48], the size impact of adding resources to a system, determinization of nondeterministic finite automata accepting subregular languages [8], transition complexity of NFA [23,33,46,47,63], and non-recursive trade-offs, and many others. Further results and references can be found, for example, in the surveys [32,41,42,57].

In order to be more precise, we now turn to present and discuss the very basics of descriptional complexity.

We denote the set of nonnegative integers by $\mathbb{N}$. Let Σ^* denote the set of all words over a finite alphabet Σ. The *empty word* is denoted by λ, and we set $\Sigma^+ = \Sigma^* - \{\lambda\}$. For the *reversal of a word* w we write w^R and for its *length* we write $|w|$. We use $\subseteq$ for *inclusions* and $\subset$ for *strict inclusions*. In general, the family of all languages accepted by a device of some type X is denoted by $\mathscr{L}(X)$.

Next, we formalize the intuitive notion of a representation or description of formal languages. We say that a *descriptional system* $\mathcal{S}$ is a set of encodings of items where each item $D \in \mathcal{S}$ *represents* or *describes* a formal language $L(D)$, and the underlying alphabet $\mathrm{alph}(D)$ over which D represents a language can be read off from D. In the following, we call the items *descriptors* and identify the encodings of some language representation with the representation itself. The *family of languages represented* (or *described*) by $\mathcal{S}$ is $\mathscr{L}(\mathcal{S}) = \{ L(D) \mid D \in \mathcal{S} \}$.

For every language L, the set $\mathcal{S}(L) = \{ D \in \mathcal{S} \mid L(D) = L \}$ is the set of its descriptors in $\mathcal{S}$.

For example, deterministic finite automata can be encoded over some fixed alphabet such that their input alphabets can be extracted from the encodings. The set of these encodings is a descriptional system $\mathcal{S}$, and $\mathscr{L}(\mathcal{S})$ is the family of regular languages.

A *(size) complexity measure* for a descriptional system $\mathcal{S}$ is a total, computable mapping $c : \mathcal{S} \to \mathbb{N}$. From the viewpoint that a descriptional system is a set of encoding strings, the lengths of the strings are a natural measure for the size. We denote it by length. In fact, we will use length to obtain a rough classification of different complexity measures. We distinguish between measures that (with respect to the size of the underlying alphabet) are related with length by a computable function and measures that are not. If there is a total computable function $g : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ such that, for all $D \in \mathcal{S}$, $\mathsf{length}(D) \leq g(c(D), |\mathrm{alph}(D)|)$, then c is said to be an *s-measure* (a *size* measure). Since for any coding alphabet there are only finitely many descriptors having at most length $g(c(D), |\mathrm{alph}(D)|)$, over the same alphabet there are only finitely many descriptors in $\mathcal{S}$ having the same size as D. If, in addition, for any alphabet Σ, the set of descriptors in $\mathcal{S}$ describing languages over Σ is recursively enumerable in order of increasing size, then c is said to be an *sn-measure*.

For example, further size complexity measures for nondeterministic finite automata are the *number of states* (state) and the *number of transition* (trans). Clearly, length, state, and trans are sn-measures for finite automata.

Whenever we consider the relative succinctness of two descriptional systems $\mathcal{S}_1$ and $\mathcal{S}_2$, we assume the intersection $\mathscr{L}(\mathcal{S}_1) \cap \mathscr{L}(\mathcal{S}_2)$ to be non-empty. Let $\mathcal{S}_1$ and $\mathcal{S}_2$ be descriptional systems with complexity measures c_1 and c_2, respectively. A total function $f : \mathbb{N} \to \mathbb{N}$, is said to be a *lower bound* for the increase in complexity when changing from a descriptor in $\mathcal{S}_1$ to an equivalent descriptor in $\mathcal{S}_2$, if for infinitely many $D_1 \in \mathcal{S}_1$ with $L(D_1) \in \mathscr{L}(\mathcal{S}_2)$ there exists a *minimal* $D_2 \in \mathcal{S}_2(L(D_1))$ such that $c_2(D_2) \geq f(c_1(D_1))$. A total function $f : \mathbb{N} \to \mathbb{N}$ is an *upper bound* for the increase in complexity when changing from a descriptor in $\mathcal{S}_1$ to an equivalent descriptor in $\mathcal{S}_2$, if for all $D_1 \in \mathcal{S}_1$ with $L(D_1) \in \mathscr{L}(\mathcal{S}_2)$, there exists a $D_2 \in \mathcal{S}_2(L(D_1))$ such that $c_2(D_2) \leq f(c_1(D_1))$. If there is no recursive, that is, computable function serving as upper bound, the *trade-off is said to be non-recursive*.

2 Operational State Complexity

The *(deterministic) state complexity* of a regular language L is the number of states of its minimal complete deterministic finite automaton, and is denoted by $sc(L)$. This is the most well-studied descriptional measure for regular languages, but one may as well consider the minimal number of states or the minimal number of transitions of nondeterministic finite automata, $nsc(L)$ and $tsc(L)$, or even consider the same measures but for incomplete DFAs. Another usual measure is the size of the syntactic monoid (syntactic complexity).

While representational complexity considers the changes of size caused by conversions between different models, the operational complexity considers the size of the model of a language resulting from an operation performed on one or more languages. As an example for the first case we have the determinization: given an n-state NFA for a regular language, the equivalent DFA has at most 2^n states. This, established in 1957 by Rabin and Scott [81], is considered the first result in state complexity.

The *state complexity of an operation* (or *operational state complexity*) on regular languages is the worst-case state complexity of a language resulting from the operation, considered as a function of the state complexities of the operands.

Given a binary operation $\circ$, the $\circ$-*language operation state complexity problem* can be stated as follows:

- given an m-state DFA A_1 and an n-state DFA A_2;
- how many states are sufficient and necessary, in the worst case, to accept the language $L(A_1) \circ L(A_2)$ by a DFA?

This formulation can be generalized for operations with a different number of arguments, other kinds of automata and classes of languages.

An upper bound can be obtained by providing an algorithm that, given DFAs for the operands, constructs a DFA that accepts the resulting language. Most algorithms first construct an NFA and then apply to it the subset construction. The number of states of the resulting DFA is an upper bound for the state complexity of the referred operation.

To show that an upper bound is tight, for each operand a family of languages, indexed by their state complexity, must be given such that the resulting automata achieve that bound. We can call those families *witnesses*.

Table 1. State complexity and nondeterministic state complexity for basic operations on regular and finite languages. The state complexity of the operands is m and n, and k is the alphabet size.

	Regular		Finite	
	sc	nsc	sc	sc
$L_1 \cup L_2$	mn	$m+n+1$	$mn-(m+n)$	$m+n-2$
$L_1 \cap L_2$	mn	mn	$mn-3(m+n)+12$	$O(mn)$
$\overline{L}$	m	2^m	m	$\Theta(k^{\frac{m}{1+\log k}})$
$L_1 L_2$	$m2^n-2^{n-1}$	$m+n$	$(m-n+3)2^{n-2}$	$m+n-1$
$L^\star$	$2^{m-1}+2^{m-1}$	$m+1$	$2^{m-3}+2^{m-4}$	$m-1$
L^R	2^m	$m+1$	$O(k^{\frac{m}{1+\log k}})$	m

In 1994, Yu, Zhuang and Salomaa [93] studied the state complexity of basic regularity preserving operations such as concatenation, star, reversal, union, and intersection. More than two decades before, in 1970, Maslov [66] had already

presented some estimates for union, concatenation, and star, as well as for other regularity preserving operations such as cyclic shift or proportional removals.

In the last decades, a huge amount of results were obtained on this subject. Several lines of research have emerged. One was to consider other operations, such as the "shuffle", or combined operations, such as the "star of union" or the "star-complement-star". The state complexities of most of these combined operations are much lower than the mathematical composition of the state complexities of their component individual operations. Another line of research was to reduce the alphabet size of the witness languages. The range of state complexities that may result from an operation, as opposed to the worst-case value, was studied under the so called *magic number problem*. *Magic numbers* for a given operation (or conversion) are the ones that cannot occur as state complexity values [48]. Many subclasses of regular languages are of special interest, such as, finite, unary, or star-free. For those, and many other, the operational state complexity restricted to a given subclass was studied. Moreover many of the above problems were also considered for nondeterministic state complexity or transition complexity. A list of these recent results as well as details on the witnesses used can be found in the surveys by Gao, Moreira, Reis and Yu [27] and by Brzozowski [13]. Davies's PhD thesis [21] presents detailed proofs of the operational state complexity for some basic operations on regular languages, and provides an excellent introduction to the subject. As an illustration, in Table 1 we present the results for some basic operations on regular and finite languages, both for deterministic and nondeterministic state complexities. From these results it is evident that the complexity of determinization plays a fundamental role in the operational complexity. Given an m-state NFA for a finite language over an alphabet of size k, the equivalent DFA has at most $\Theta(k^{\frac{m}{1+\log k}})$ states, and this bound is tight [83]. Campeanu, Culik and Salomaa [14], presented the first formal study of operational state complexity on finite languages. The state complexity of basic operations on NFA (both for regular and finite languages) were studied by Holzer and Kutrib [40].

A great effort was spent to find witnesses with minimal alphabet size. Many different witnesses were found for the various operations. Symbolic manipulation software is in general used to help to establish witness candidates that after can be formally proved to attain the maximal bounds. However, the reason why a given witness would work for several complexity bounds is not well understood. In 2012, Brzozowski identified a family of languages that witnesses the state complexity of all basic operations on regular languages [12]. Figure 1 presents the minimal DFA for the language family. More importantly, he established conditions for a family of languages to attain those bounds and, in this sense, to be *the most complex regular languages*. A fundamental condition was that, for a language with state complexity m, the size of its syntactic monoid should be m^m, which is the tight upper bound.

This research triggered the development of an algebraic approach to operational descriptional complexity. Given a complete DFA A, each letter corresponds to a transformation on the set of states. One says that the letter acts on the set

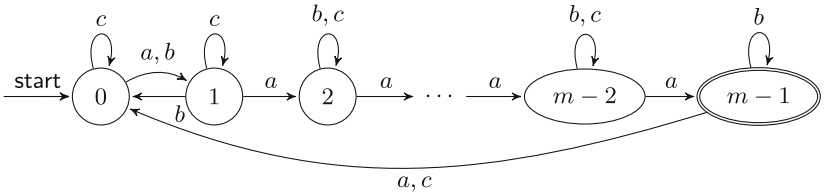

Fig. 1. Brzozowski's "universal" witness.

of states. This notion can be extended to words, and the set of transformations induced by all words, with composition, is a monoid, the transition monoid of A. The syntactic complexity of a regular language is the size of the transition monoid of its minimal DFA (also called the *syntactic monoid*). For instance, in Fig. 1 the letter a performs a cyclic permutation, b a transposition $(0, 1)$ (i.e. interchanges those states and all other are fixed points), and c sends $m-1$ to 0, keeping unaltered the other states. It can be shown that every transformation of $\{0, \ldots, m-1\}$ can be represented as the action of a word over $\{a, b, c\}$ [12]. The operational syntactic complexity on regular and subregular languages was extensively studied mainly by Brzozowski and co-authors.

The use of algebraic characterizations in operational state complexity is now a hot topic of research. Bell, Brzozowski, Moreira, and Reis [4] considered the following question: for which pairs of languages (L_m, L'_n) (with state complexities m and n, respectively) does $L_m \circ L'_n$ reach the maximal state complexity mn for every proper binary Boolean operation $\circ$? A sufficient condition (excluding known counterexamples) is that the transition monoids contain the symmetric groups S_m and S_n, respectively. Davies refined those conditions proving that in general (except restricted cases) it is sufficient that the transition monoids contain 2-transitive groups [20,21].

It is known that a witness with a maximal transition monoid is guaranteed to maximize the state complexity of reversal. This is not the case for other operations. In general it is difficult to know which transformations one needs to associate to each letter on an operand DFA to ensure that the resulting DFA from the operation is, in some sense, maximal. If the alphabet size is not an issue, one can use the one-letter-per-action (OLPA) technique: i.e. the witnesses have one letter for each possible transformation. Sakoda and Sipser used this technique for studying the state complexity of the conversions between one-way and two-way finite automata [82] (see Sect. 3). Although some authors used this technique sparsely in past, only recently it was formalized by Davies [19,21] and Caron, Hamel-De le Court, Luque and Patrou [15]. Its power, limitations, and which consequences it can have in this topic remains open. Anyhow, deepening the connection between combinatorial and algebraic methods seems fruitful.

3 When Descriptional Complexity Meets Computational Complexity

Usually, when we refer to finite automata, we implicitly assume that we are considering *one-way finite automata*, namely automata that can read the input only from left to right.

The extension to the *two-way* case, namely to automata which are able to move the input head in both directions, is an interesting area, with challenging descriptional complexity problems and important connections with classical computational complexity. Actually, the investigation of complexity questions for one-way and two-way finite automata can be carried out as a part of the area of computational complexity, with classes of problems, reductions, complete problems and, of course, open questions, as emphasized in [49] by Christos Kapoutsis who introduced the name *Minicomplexity* for this area.

Here, we present a short outline on two-way finite automata and on the connections with computational complexity.

First of all, we have to mention that it is well known that the capability of moving the head in both directions does not increase the computational power of finite automata. Two-way finite automata, in both deterministic and nondeterministic versions (2DFA and 2NFA, for short), still characterize the class of regular languages. This result was independently proved in 1959 by Rabin and Scott and by Shepherdson, by showing constructions transforming 2NFAs into equivalent DFAs [81,85]. (An alternative transformation was obtained by Vardi [89].) The increment in the number of states in such transformations is exponential. We will now present a simple example showing that such an increasing cannot be avoided. Before doing that, we mention that in two-way automata we assume that the input is surrounded by two special symbols called the *left* and the *right endmarker*.

For each integer $n > 0$, let us consider the language

$$I_n = (a+b)^* a (a+b)^{n-1} \ .$$

It is not difficult to see that I_n is recognized by an NFA with $n+1$ states, while any DFA accepting it requires 2^n states in order to remember, at each step of the computation, the suffix of length n of the input string so far inspected (a formal proof can be done with standard distinguishability arguments). We can easily construct a 2DFA which recognizes I_n by locating the right endmarker and then moving the head to the left to check if the nth symbol from the right is an a. This can be implemented using $n+2$ states.

A more sophisticated example with similar properties is

$$L_n = (a+b)^* a (a+b)^{n-1} a (a+b)^* \ ,$$

where we ask that strings contain a pair of a's with $n-1$ symbols in between. In this case, a 2DFA can check a string w by moving the head from the first symbol of w to the right, up to reach a cell containing a. Then, it moves n more positions to the right. If the reached cell contains another a, then the automaton

accepts. Otherwise, it moves the head $n-1$ cell to the left, reaching the cell to the right of the first a, and it repeats by using the same approach, looking for a cell containing a while moving to the right. Even for L_n we can obtain an NFA and a 2DFA with $O(n)$ states while each DFA requires a number of states exponential in n.[1]

One natural question arising after seeing these two examples is whether or not two-way motion can be used to remove the nondeterminism, without significantly increasing the size of the description. This problem was posed by Sakoda and Sipser in 1978 [82] and, actually, it was formulated by the two following questions:

1. *For every 2NFA M, is there an equivalent 2DFA with only polynomially more states than M?*
2. *For every NFA M, is there an equivalent 2DFA with only polynomially more states than M?*

Sakoda and Sipser conjectured that both questions have negative answers. To support such a conjecture, they presented a complete analogy with the P versus NP problem.

Two years later, Sipser proved exponential separations, under the conditions that the simulating 2DFAs are *sweeping*, namely they can reverse the movement of the input head only at the endmarkers [86]. However, Berman and Micali showed that this does not solve the general problem [5,69].

For several years not so much work has been done around these questions, until the first years of this millennium, when new investigations on two-way automata have been carried out and several new results, solving some special cases, have been obtained. However, it seems that we are still far from a solution for the general case.

Besides the result on sweeping 2DFAs, similar separations have been obtained for the simulation of NFAs and 2NFAs by restricted kinds of 2DFAs, as *oblivious* 2DFAs [45,60], *rotating* automata [54], *few reversal* automata [51]. Exponential separations between these kinds of devices and unrestricted 2DFAs have been also proved, showing that these results do not solve the general problem. For a more detailed overview and further references see [75].

Some results providing polynomial simulations of restricted forms of 2NFAs by (unrestricted) 2DFAs have been obtained by considering the unary case (i.e., the input alphabet contains only one-letter, so this gives a restriction on the class

[1] The 2DFA we described for I_n makes use of the endmarkers, while the 2DFA for L_n does not use them. Actually, we could adapt the technique used to recognize L_n in order to obtain a 2DFA with $O(n)$ states accepting I_n, without using the endmarkers. The main difference is that the 2DFA so obtained may need to reverse the direction of its head many times, while the 2DFA we described for I_n makes only one reversal. It seems quite natural to have endmarkers in two-way automata. However, in some works they are presented without endmarkers. This does not change the computational power. The example of I_n shows some differences when we consider size and number of reversals. In general, it has been proved that two-way automata can have different properties with or without endmarkers [90].

of accepted languages) [29], and the case of *outer nondeterministic automata*, which can make nondeterministic choices only when the head visits one of the endmarkers [28].[2]

Recently, a different approach was proposed, trying to obtain polynomial simulations of 2NFAs by some extensions of 2DFAs. In [34] it is presented a polynomial simulation by single-tape deterministic Turing machines working in linear time (by a result by Hennie [38], these machines are no more powerful than finite automata). It will be interesting to continue this approach by considering, for instance, simulations by *deterministic* 1*-limited automata*. These devices are discussed in the next section.

As already mentioned, Sakoda and Sipser presented an analogy between the above Questions 1. and 2. and the P versus NP question. In particular they defined a notion of reducibility and presented complete problems for the two questions.

Starting from an initial result by Berman and Lingas [6], strong connections have been discovered between the question of Sakoda and Sipser and the open question L versus NL (classes of languages accepted in logarithmic space by deterministic and nondeterministic Turing machines, respectively).

In [30], by considering the unary case, it was proved that $\mathsf{L} = \mathsf{NL}$ would imply a state polynomial simulation of unary 2NFAs by 2DFAs. So, showing that the simulation of 2NFAs by 2DFAs is not polynomial, already in the restricted case of a unary alphabet, would separate L and NL. We point out that at the moment the best known simulation in the unary case is superpolynomial, but subexponential [29]. This result and that of Berman and Lingas have been generalized by Kapoutisis in [52] and further generalized in [55], by considering the class $\mathsf{L/poly}$ of languages accepted by deterministic logspace bounded machines that can access a *polynomial advice*. It was proved the following:

Theorem 1 ([55])**.** $\mathsf{L/poly} \supseteq \mathsf{NL}$ *if and only if there is a polynomial simulation of* unary *2NFAs by 2DFAs.*

Actually, further characterizations of $\mathsf{L/poly} \supseteq \mathsf{NL}$ have been presented in [55]. Among them, it was shown that $\mathsf{L/poly} \supseteq \mathsf{NL}$ is equivalent to the existence of 2DFAs of size polynomial in h which are able to check accessibility in unary encoded h vertex graphs and to check two-way liveness in h-tall, h-column graphs. Hence, such versions of accessibility and liveness problems are complete for the above Question 1. in the unary case. It was conjectured that these statements are false.

These results show important connections between descriptional complexity of automata and classical computational complexity. As we mentioned at the beginning of the section, following the approach of *minicomplexity*, the investigation on the complexity of finite automata can be carried out in the wider area of

[2] Concerning the unary case, the state cost of simulation of *one-way* nondeterministic automata by 2DFAs has been proved to be quadratic [17]. This gives a positive answer to the second question in the unary case.

computational complexity, using the same methods (reductions, complete problems, etc.). We recommend the works [50,53] to appreciate the minicomplexity approach.

4 Turing Machines with Rewriting Restrictions

It is well known that each class of languages in the Chomsky hierarchy has a corresponding family of recognizing devices. By considering for the top level, i.e., Type 0 languages, Turing machines with a read-only input tape and a separate worktape, we easily obtain a hierarchy of machines by restricting the space used on the worktape to be linear in the input length for context-sensitive languages, by accessing it as a pushdown store in the case of context-free languages[3], and by removing the worktape for regular languages.

Having the same computational power of multi-tape machines, single-tape Turing machines are sufficient to characterize the class of Type 0 languages. Furthermore, restricting these devices to use only the portion of the tape which initially contains the input, we obtain *linear bounded* automata, which characterize context-sensitive languages. However, by still considering pushdown automata for the family of context-free languages, we do not obtain a hierarchy of machines.

A less-known characterization of the class of context-free languages in terms of machines was obtained by Hibbard in 1967, in terms of *limited automata*, a class of single-tape nondeterministic Turing machines satisfying the following rewriting restriction: fixed an integer $d \geq 0$, a d-limited automaton (d-LA, for short) can rewrite the contents of each tape cell *only in the first d visits* [39]. Without loss of generality, d-LAs can be restricted to use only the portion of the tape which initially contains the input.

Hibbard proved that for each fixed $d \geq 2$, the class of languages accepted by d-LAs coincides with the class of context-free languages. Since d-LAs are a restriction of linear bounded automata and, clearly, they are extensions of finite automata, using 2-LAs for the class of context-free languages we obtain a single-tape machine hierarchy corresponding to the Chomsky hierarchy.

To give a sake of the way used by limited automata to operate, we describe a simple strategy that can be implemented by a 2-LA in order to recognize the language consisting of all sequences of balanced brackets. An input sequence of brackets can be inspected starting from the leftmost symbol and moving to the right, until reaching the first closing bracket. If the sequence is balanced, then the corresponding opening bracket is necessarily the last bracket before it, that must be of the same type. If so, these two brackets can be overwritten by a special symbol, and the same procedure can be repeated by moving from the position of the just overwritten opening bracket to locate the first closing bracket, overwriting it, then checking if the last bracket before it is of the same type, overwriting it, and so on. When no more closing brackets are left in the

[3] It can be easily seen that each context free-language can be accepted by a pushdown automaton which uses an amount of pushdown store which is linear in the input length.

sequence, even none opening bracket should be left. In this case the original sequence was balanced and the machine can accept; in all other cases the machine rejects. If the input sequence is written on a Turing machine tape, one bracket per cell, and the computation starts, as usual, with the head scanning the cell containing the leftmost input symbol, then each cell containing a closing bracket is overwritten only when the head visits it for the first time. Thereafter, the head is moved back to the left to search the corresponding opening bracket, which was already visited one time and which is overwritten when the head visits it for the second time. After these *active visits*, a cell can be visited further many times, but it cannot be overwritten, so it is "frozen". Hence, each tape cell is overwritten at most in the first 2 visits.

In the last years, limited automata have been investigated in a series of papers, with the main focus on their descriptional complexity (for a recent overview see [76]). The costs of the simulations between pushdown automata and 2-LAs have been stated in [78]. These results have been extended to d-LAs, with $d > 2$ in [61]. In [78] it was also proved that *deterministic* 2-LAs (D-2-LAs) are equivalent to deterministic pushdown automata. As proved in [39], the class of languages accepted by deterministic limited automata becomes larger by increasing the number of possible rewritings, i.e., by increasing the value of d we obtain a proper infinite hierarchy of languages accepted by D-d-LAs. However, this hierarchy does not cover all the class of context-free languages.

We now focus on the subfamily of *1-limited automata*, i.e., single-tape machines that can rewrite the contents of any tape cell only in the first visit. This model characterizes regular languages [91], so they are equivalent to finite automata. The cost of the conversions of nondeterministic and deterministic 1-LAs into equivalent finite automata have been studied in [77] and they are summarized in Fig. 2, with some other costs. Some comments on these results:

- The double exponential simulation (a) of 1-LAs by one-way DFAs is obtained by combining the Shepherdson construction for simulating 2NFAs by DFAs with the classical subset construction. The double exponential gap, which cannot be reduced, is due to a double role of the nondeterminism: when a cell is visited for the first time, a symbol is written on it according to a nondeterministic decision; in the next visits to the same cell, the available nondeterministic choices also depend on the symbol which is written in it, namely the symbol written during the first visit.
- The arrow (b) represents the Sakoda and Sipser question, for which an exponential upper bound but only a polynomial lower bound are known.
- Arrow (c) presents a similar question for limited automata, translated of one exponentiation level: by (a) we know that the elimination of the nondeterminism from 1-LAs costs at most a double exponential in size, however the best known lower bound is exponential. It could be interesting to know if there are strict relationships between these two questions, e.g., if an exponential gap from 2NFAs to 2DFAs would imply a double exponential gap from 1-LAs to D-1-LAs and vice versa.

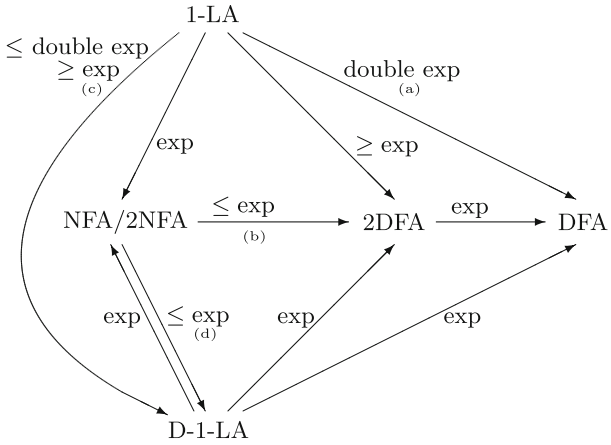

Fig. 2. Costs of some conversions between different kinds of limited automata and finite automata.

– The exponential upper bound in (d) derives from (b). An exponential lower bound will close the Sakoda and Sipser question. So the study of the conversion in (d) can be seen as a "relaxed" version of the Sakoda and Sipser question where nondeterminism is removed by allowing to rewrite tape cells in the first visit.
We point out that in [34] it was recently obtained a polynomial blowup from 2NFAs to single-tape deterministic machines working in linear time. Since with a polynomial increase in the size, each 1-LA can be converted into an equivalent one working in linear time, also preserving determinism [35], proving that the simulation (d) costs polynomial would improve that result.

In Sect. 3 we have already mentioned the *unary case* and its relevance in the connection with the Sakoda and Sipser question. Several results related to limited automata in the unary case have been obtained in [61,62]. Further results concerning unary 1-LA are presented in [79].

5 Automata that May Change Their Mind

The concept of nondeterministic machines was introduced in the seminal paper of Rabin and Scott [81] on finite automata and their decision problems from 1959. Over the years, nondeterminism turned out to be a very fruitful concept in different areas of computer science like, for example, computability theory, complexity theory, automata theory, formal language theory, etc., to mention only a few.

For finite automata it is folklore that deterministic finite automata (DFA) are as powerful as nondeterministic finite automata (NFA) from a computational capacity point of view. However, from a descriptional complexity point of view

NFAs can offer exponential savings in size compared with DFAs. That is reason enough to consider nondeterminism as a resource of the underlying model and to quantify its usage to some extend. A lot of results on such quantifications are subsumed under the name *limited nondeterminism* in the literature, see, for example, [32] for a survey.

Being interested in the power of the amount of nondeterminism with respect to computations and conciseness it has been newly interpreted in terms of *one-time nondeterministic automata* [44] and its generalization *mind-changing automata* [43]. The idea of mind-changing automata is that at the outset the automaton is partially deterministic. In this way, defined transitions constitute situations for which the automaton already has an opinion (on how to proceed), while undefined transitions constitute situations for which the automaton is still irresolute. Whenever the automaton encounters a situation for which it is irresolute, it can form its opinion by choosing an appropriate transition out of a set of transitions. The chosen transition is then added to the transition function. Finally, whenever the automaton is in a situation for which a transition is defined, it can change its mind and interchange the transition by an alternative matching transition from the set of available transitions. Now, the total number of mind changes is considered as a limited resource.

We illustrate the notion by the following example.

Example 2. Consider the mind-changing finite automaton (MCFA) M depicted in Fig. 3, where the transitions of the initial transition function δ_0 are drawn with solid arrows and that of the initial set of alternative transitions T_0 are depicted with dashed arrows. So, we have $\delta_0(1,a) = \delta_0(1,b) = 1$, $\delta_0(2,a) = \delta_0(2,b) = 3$, and $T_0 = \{(1,a,2)\}$. The *language accepted* by M with up to $k \geq 0$ mind changes is denoted by $L_k(M)$.

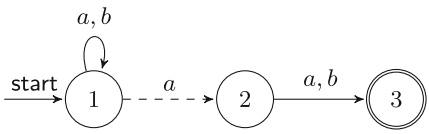

Fig. 3. The MCFA M of Example 2.

Obviously, $L_0(M) = \emptyset$, since the automaton can never change any transition and thus the sole accepting state 3 cannot be reached.

Whenever M decides to make a mind-changing step, that is, exchanging the original transition $(1,a,1)$ by $(1,a,2)$ from T_0, then the sole accepting state 3 can be reached from 1 *via* state 2 by reading either aa or ab. Let us see how this works on input $w = baab$. To this end let $\delta' = (\delta_0 \cup \{(1,a,2)\}) \setminus \{(1,a,1)\}$ and $T' = (T_0 \cup \{(1,a,1)\}) \setminus \{(1,a,2)\}$. Then an accepting computation on input w is

$$(1, baab, \delta_0, T_0) \vdash (1, aab, \delta_0, T_0) \vdash (1, ab, \delta_0, T_0) \vdash (2, b, \delta', T') \vdash (3, \lambda, \delta', T'),$$

where the sole mind-change appeared at the next to last computation step. Yet there is another computation on w which is not accepting, since the mind-change appeared too early and the computation blocks. This non-accepting computation is

$$(1, baab, \delta_0, T_0) \vdash (1, aab, \delta_0, T_0) \vdash (2, ab, \delta', T') \vdash (3, b, \delta', T').$$

It is worth mentioning, that although the underlying automata induced by δ_0 and δ' are both deterministic, there are more than one computation on M, due to the mind-changes. By our example it is not hard to see that

$$L_1(M) = \{\, w \in \{a, b\}^* \mid \text{the next to last letter of } w \text{ is an } a \,\}$$

and moreover $L_k(M) = L_1(M)$, for $k \geq 1$.

In case we consider the MCFA M' with initial transition function δ' and initial set of alternative transitions T', then one observes that

$$L_0(M') = L_1(M') = b^*a(a + b)$$

and $L_k(M') = \{\, w \in \{a, b\}^* \mid \text{the next to last letter of } w \text{ is an } a \,\}$, for $k \geq 2$. ■

Intuitively, it is clear that the family of languages accepted by MCFAs coincides with the regular languages. Although the concept of mind changes does do not improve the computational power of ordinary finite automata, the question for the descriptional complexity of such devices arises. It turned out that the upper bound on the costs for the simulations of an MCFA M by a DFA depends on the *nondeterministic degree* $d(M)$ of M that is defined as

$$d(M) = \prod_{\substack{(q,a) \in Q \times \Sigma \\ |\delta_0(q,a) \cup \{\, p \mid (q,a,p) \in T_0 \,\}| \neq 0}} |\delta_0(q, a) \cup \{\, p \mid (q, a, p) \in T_0 \,\}|,$$

where Q is the state set and Σ is the input alphabet.

Theorem 3. *Let M be an n-state MCFA. Then $(k+1) \cdot n \cdot d(M) + 1$ states are sufficient for an NFA to accept the language $L_k(M)$, for every $k \geq 0$.*

From Theorem 3 and the powerset construction on NFAs an upper bound for the simulation by DFAs follows. Note that the DFAs are partial. At least one state can be saved in the exponent.

Corollary 4. *Let $k \geq 0$ be a constant and M be an n-state MCFA with input alphabet Σ. Then, $2^{(k+1) \cdot n \cdot d(M)} + 1$ states are sufficient for a DFA to accept the language $L_k(M)$.*

How about the lower bounds for the simulations? In this connection a special case, that is, an MCFA M that may change its mind only once ($L_1(M)$) on a single transition ($|T_0|$), has been studied in more detail. Even in this special case the MCFA can be more succinct than nondeterministic finite automata.

We consider complete MCFAs, where an MCFA M is said to be *complete* if the underlying initial DFA M' is complete, that is, $|\delta_0(q, a)| = 1$, for every state q and every input symbol a. For $k = 0$ it is obvious that $L_0(M) = L(M')$. Thus, in this case we do not save states when comparing MCFAs and DFAs.

Theorem 5. *Let M be an n-state* complete *MCFA having $|T_0| = 1$. Then $2^{n+\log n-1}$ states are sufficient and necessary in the worst case for a DFA to accept the language $L_1(M)$.*

An MCFA that witnesses the matching lower bound of Theorem 5 is depicted in Fig. 4.

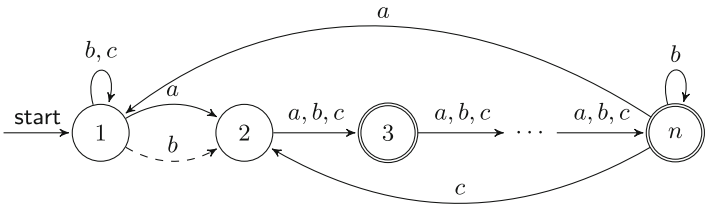

Fig. 4. The n-state complete MCFA M with a singleton set of alternative transitions T_0 witnessing the lower bound of Theorem 5. The transitions from δ_0 are drawn with solid arrows and that of T_0 are depicted with dashed arrows.

The lower bound of Theorem 5 implies that any nondeterministic finite automaton accepting these languages requires at least $n+\log n-1$ states. In other words, there is a sequence of regular languages $(R_n)_{n\geq 3}$ accepted by an n-state complete mind-changing finite automaton with a single alternative transition with at most one mind-change such that any nondeterministic finite automaton accepting R_n requires at least $n + \log n - 1$ states.

At this point one may ask, whether it is possible to generalize Theorem 5 to $k \geq 1$ mind-changing moves in general? Since T_0 is required to be a singleton set, the constraint $k \geq 1$ means that during the computation one can alternate up to k times between two transitions during the computation. Indeed, it is not hard to see that the upper bound construction given in the proof of Theorem 5 for the case $k = 1$ generalizes to arbitrary fixed k with $k \geq 1$.

Theorem 6. *Let M be an n-state* complete *MCFA having $|T_0| = 1$. Then, for $k \geq 2$, $2^{\log(k+1)\cdot(n-1)+\log n}$ states are sufficient for a DFA to accept the language $L_k(M)$.*

What about the lower bound? Here the situation is much more involved compared to the lower bound construction. At the moment, we can only provide the non-matching lower bound from Theorem 5 for $k \geq 2$, which is somehow trivial. Nevertheless, to improve the upper or the lower bound for the conversion under consideration is left as an open problem.

Finally, it is mentioned that also mind-changing pushdown automata have been considered [43]. While for any (constant) number of mind-changes MCFAs characterize the regular languages, the situation for pushdown automata changes drastically. In fact, for mind-changing pushdown automata, there is an infinite proper hierarchy depending on the number of mind-changes strictly in between

the deterministic and arbitrary context-free languages. From the descriptional complexity point of view there are non-recursive trade-offs between all hierarchy levels.

6 Boosting the Descriptional Capacity by Transductions

Finite-state transducers are finite automata with an output and they have been studied at least since 1950s. A typical application of finite-state transducers is, for example, the lexical analysis of computer programs or XML documents. Here, the correct formatting of the input is verified, comments are removed, the correct spelling of the commands is checked, and the sequence of input symbols is translated into a list of tokens. The output produced is subsequently processed by a pushdown automaton that realizes the syntactic analysis. Another example is the use of *cascading* finite-state transducers. Here, one has a finite number of transducers $T_1, T_2, \ldots, T_n$, where the output of T_i is the input for the next transducer T_{i+1}. Such cascades of finite-state transducers have been used, for example, in [26] to extract information from natural language texts. Another example is the Krohn-Rhodes decomposition theorem which shows that every regular language is representable as the cascade of several finite-state transducers, each one having a "simple" algebraic structure [31,36]. Finally, it is shown in [18] that cascades of deterministic pushdown transducers lead to a proper infinite hierarchy in between the deterministic context-free and the deterministic context-sensitive languages with respect to the number of transducers involved. All the examples of cascading transductions so far addressed have in common that the subsequently applied transducers are, at least in principle, different. Another point of view is taken in [7,65], where subsequently applied *identical* transducers are studied. Such *iterated* finite-state transducers are considered as language generating devices starting with some symbol in the initial state of the transducer, iteratively applying in multiple sweeps the transducer to the output produced so far, and eventually halting in an accepting state of the transducer after a last sweep. These iterated finite-state transducers are quite powerful since their nondeterministic version can generate non-recursive languages with only three states. Even in the deterministic case, one state suffices to generate the class of D0L Lindenmayer languages and two states are sufficient to generate languages which are neither context-free nor in 0L. It is worth remarking that an essential feature in these models is that the underlying finite-state transducer is *not length-preserving*. In contrast to all these examples and other works in the literature (see, for example, [74]), where the subsequently applied transducers are in principle different and not necessarily length-preserving, the model of *iterated uniform finite-state transducers* introduced in [58,59] as language accepting devices requires that the same transducer is applied in every sweep and that the transduction is length-preserving.

So, an iterated uniform finite-state transducer is basically a finite-state transducer which processes the input in multiple passes (also sweeps). In the first pass, it reads the input word followed by an endmarker and emits an output word.

In the following passes, it reads the output word of the previous pass and emits a new output word. The number of passes taken, the *sweep complexity*, is given as a function of the length of the input. The transducers to be iterated are length-preserving finite-state transducers, also known as Mealy machines [67].

It is known that at least a logarithmic number of sweeps is necessary to accept non-regular languages. Under a natural constructibility condition it is shown that there exists a proper infinite hierarchy of accepted language families depending on the sweep complexity beyond the logarithm, both in the deterministic (IUFST) and nondeterministic (NIUFST) case [59]. Also, nondeterminism is separated from determinism for all the hierarchy levels.

From the descriptional complexity viewpoint a constant bound on the sweep complexity is of particular interest. So, the succinctness dependent on the constant $k \geq 1$ and in comparison with more traditional models of finite-state automata has been studied in more detail.

Example 7. For $k \geq 1$, the language $E_k = \{a,b\}^* b \{a,b\}^{k-1}$, whose words are characterized by having the letter b at the kth position from the right, is considered. It is well known that any deterministic finite automaton (DFA) needs at least 2^k states to accept E_k.

A k-sweep IUFST T (k-IUFST) can accept E_k with three states only. The basic idea of T's processing is to shift the input word symbol by symbol to the right, whereby an a is inserted at the first position and the last symbol is removed (this takes two states). In this way, in the first $k-1$ sweeps the input is shifted $k-1$ positions to the right. In a final sweep, it is sufficient to check whether the last symbol is a b. The number of the current sweep can be stored as index of the endmarker. ■

The example shows that iterated transductions may lead to a drastic decrease of the number of states for accepting regular languages. It may suggest that the descriptional power of k-IUFSTs always outperforms that of DFAs. However, some languages are particularly size-demanding that even iterated transduction cannot reduce the number of states for their recognition. Witnesses for this fact are the unary regular languages $L_p = \{ a^{m \cdot p} \mid m \geq 0 \}$ where p is a prime number. The state-graph of the minimal DFA accepting L_p consists of a simple directed cycle of p states, beginning from and ending into a designated state which is both the initial and the unique accepting state. In fact, this elementary automaton is actually the best we can provide for L_p, regardless the (classical) computational paradigm we may want to adopt: for example, it is known that p states are necessary for accepting L_p on DFAs, NFAs, two-way DFAs, and two-way NFAs. Even on k-IUFSTs the language L_p needs at least p states to be accepted.

Theorem 8. *Let $k \geq 1$. Then p states are necessary and sufficient for a k-IUFST to accept L_p.*

To study the descriptional power of k-sweep iterated uniform finite-state transducer versus classical finite automata, it is helpful to consider first the sweep reduction and determinization.

Theorem 9. *Let $n, k > 0$ be integers. Every n-state k-NIUFST (resp., k-IUFST) can be converted to an equivalent $2n^i$-state $\lceil \frac{k}{i} \rceil$-NIUFST (resp., $\lceil \frac{k}{i} \rceil$-IUFST).*

The sweep reduction can directly be used to transform constant sweep bounded IUFST and NIUFSTs into equivalent DFAs and NFAs.

Corollary 10. *Let $n, k > 0$ be integers. Every n-state k-NIUFST (resp., k-IUFST) can be converted to an equivalent NFA (resp., DFA) with at most $2n^k$ states.*

The result presented in Theorem 9 turned out to be almost optimal in the sense that there are languages witnessing that n^k states are necessary.

Let us discuss in more detail the possibility of trading states for input sweeps and vice versa. Concerning the relation between the necessary number of states and the number of sweeps, we have the following situation: Theorem 8 shows that there are languages for which additional sweeps do not help to decrease the number of states at all. By Corollary 10, any n-state k-IUFST can be converted into an equivalent DFA with at most $2n^k$ states. Conversely, clearly we cannot reduce the number of states below two or three (for non-trivial languages) by increasing the number of sweeps. So, there is an upper bound for the number of sweeps that may help. In other words, for any regular language L we have a fixed sweep range from 1 to some k_L in which we can trade states for sweeps and vice versa.

Theorem 11. *Let $k \geq 2$ and $n \geq 3$ be integers, and T be an n-state k-IUFST such that the minimal DFA for $L(T)$ has $2n^k$ states. Then any $(k-1)$-IUFST for $L(T)$ must have at least $\lceil n^{k/k-1} \rceil$ states.*

Finally, removing the nondeterminism and the sweeps at the same time means to convert a k-NIUFST to a DFA. Conversion to an NFA with subsequent determinization gives the upper bound of 2^{2n^k} states. A lower bound for this size blow-up is derived by considering the witness language

$$E_{n,k} = \{\ vbw \mid v, w \in \{a, b\}^*,\ |w| = c \cdot n^k \text{ for } c > 0\}$$

for any $n, k > 1$.

Theorem 12. *For any integers $m > 1$ and $k > 0$, an m-state k-NIUFST can be converted to an equivalent 2^{2n^k}-state DFA. There is an m-state k-NIUFST which cannot be converted to an equivalent DFA with less than $2^{(m-1)^k}$ states.*

Finally, it should be mentioned that for k-IUFST the commonly considered decision problems have the same computational complexity as for deterministic finite automata, that is, they are NL-complete. When the bound of the sweep complexity is beyond logarithm, typical decision problems become non-semidecidable.

Several possible lines of research on iterated transducers may be tackled. First of all, it would be natural to consider the same decision problems as in the deterministic case as well as the size cost of implementing language operations for

nondeterministic transducers. It would also be interesting to study more general forms of iterated transduction where, for example, different transductions can be performed at different sweeps, or where further information, apart from the output, can be passed on from one sweep to the next.

7 Enumerations and Average Complexity

Descriptional complexity, similarly to what happens in computational complexity, is almost always considered for its worst-case. However, in most cases, this worst-case complexity is only achieved for sets of inputs of very small significance. For practical applications, the average-case analysis, where input data is assumed to follow a given probability distribution, can provide much more accurate prediction of the needed computational resources.

The study of complexity results on average can be performed through experimentation, for which well behaved random generators for the computational models, and thus rigorous enumerative descriptions of their classes, are needed. Enumerative formulae and asymptotic estimates for different kinds of finite automata were presented in 1960s and 1970s. For a survey, we refer the reader to Domaratzki [22]. Concerning uniform random generation, first Nicaud [71] presented a random generator for binary accessible nonisomorphic DFAs that was extended to arbitrary alphabets by Champarnaud and Paranthoën [16]. Almeida, Moreira and Reis [1] based on a string representation, gave another enumeration and random generator for the same class of automata, that avoided any ulterior rejection phase. Bassino and Nicaud [3] improving their previous work gave asymptotic estimates on the number of objects of this class of a given size, as well as, presented a random generator based on Boltzmann samplers. Almeida, Moreira and Reis [2] presented a canonical form for minimal acyclic DFAs which allows its exact enumeration, and that was later extended to non-minimal (trim) acyclic DFAs. An exact enumerative formula based on that representation, and using generalized parking functions, was published by Priez [80]. A random generator for accessible acyclic DFAs was developed by Felice and Nicaud [24], but that is feasible only for small sized automata. On this subject, a survey by Nicaud [73] can be consulted for further details. In the case of NFAs the situation is much more challenging. Testing if two NFAs are isomorphic is an hard problem and thus feasible uniform random generators are not expected to be found. Moreover, with high probability, a uniform random NFA is universal, so other distributions should be considered. Restricted subclasses of NFAs adequate for random generation were studied by Héam and Joly [37] and by Ferreira, Moreira and Reis [25].

For regular expressions random generation is easily obtained from any unambiguous context-free grammar that generated them. All random generators listed here aim an uniform distribution of the objects generated. This distribution, although "naturally" chosen as unbiased, was subject of a study of its expressivity by Koechlin, Nicaud and Rotondo [56] showing that, at least for some aspects of the behavior of the regular expressions other distributions should

be considered. In particular, those authors studied absorbing patterns in regular expressions with respect to language equivalence. What is implied by their results is that if one uniformly random generates regular expressions one cannot expect to obtain, with a reasonable probability, regular languages outside a constant set of languages. This means that a core set of languages have so many regular expressions representatives that the remaining languages very scarcely appear. Both the experimental studies and the analytic combinatorics studies, mentioned below, aim to the estimation of the relative descriptional complexity of different models as combinatorial objects by themselves, disregarding language equivalence.

Alternative methods to obtain average results in descriptional complexity can be used in order to avoid the experimentation. Because Kolmogorov incompressible objects have a behavior that it is, by definition, indistinguishable from the average, its study should give rise to average complexity results in a very elegant and succinct manner. Nevertheless, and although canonical string representations exist for some of the models and the successful application of these ideas in computational complexity, no results can be here listed using such technique.

An elegant alternative is the framework of analytic combinatorics [84], by relating combinatorial objects to algebraic properties of complex analytic generating functions. In particular, the behavior of these functions around their dominant singularities gives access to the asymptotic form of their (power) series coefficients. In recent years, the average size of different NFA constructions from regular expressions, using the framework of analytic combinatorics was studied. Nicaud [72] showed that, for a uniform distribution, the position automaton has asymptotically and on average linear size w.r.t the size of the expressions. Several other constructions for regular expressions and other kind of expressions were also considered [9,11]. In those studies, whenever explicit expressions for the generating functions were obtained, it was possible to estimate asymptotic average values using relatively standard analytic methods. However, in general, one does not deal with just one combinatorial class, but with an infinite family of combinatorial classes indexed by the size of the alphabet. This raises problems that do not appear in standard combinatorial classes, such as graphs or trees. Moreover, in many cases, having explicit expressions for the generating functions is unmanageable. Then one needs to use generating functions implicitly defined by algebraic curves, and develop a method to extract the required information for the asymptotic estimates. This method allowed to find, for the combinatorial classes considered, the behavior of the generating function without knowing beforehand the explicit value of its singularity. As an example it was possible to study the average behavior of regular expressions in star normal form [10]. The average results obtained so far on the descriptional complexity of conversions of regular expressions to other models have revealed that asymptotic complexities in the worst case when linear are halved on the average case and square-rooted in the other cases (see [11] for a recent list of average case results).

All these studies using the framework of analytic combinatorics are based on the combinatorial class of regular expressions (or some subset of those). It would

be very interesting to see if the same approach could be applied to combinatorial classes of other models, like DFAs, and, in particular, if the goal of characterizing the average behavior of determinization of NFAs is attainable with this technique.

References

1. Almeida, M., Moreira, N., Reis, R.: Enumeration and generation with a string automata representation. Theoret. Comput. Sci. **387**(2), 93–102 (2007)
2. Almeida, M., Moreira, N., Reis, R.: Exact generation of minimal acyclic deterministic finite automata. Int. J. Found. Comput. Sci. **19**(4), 751–765 (2008)
3. Bassino, F., Nicaud, C.: Enumeration and random generation of accessible automata. Theoret. Comput. Sci. **381**(1–3), 86–104 (2007)
4. Bell, J., Brzozowski, J., Moreira, N., Reis, R.: Symmetric groups and quotient complexity of Boolean operations. In: Esparza, J., Fraigniaud, P., Husfeldt, T., Koutsoupias, E. (eds.) ICALP 2014. LNCS, vol. 8573, pp. 1–12. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-43951-7_1
5. Berman, P.: A note on sweeping automata. In: de Bakker, J.W., van Leeuwen, J. (eds.) Proceedings 7th ICALP 1980. LNCS, vol. 85, pp. 91–97. Springer (1980). https://doi.org/10.1007/3-540-10003-2_62
6. Berman, P., Lingas, A.: On the complexity of regular languages in terms of finite automata. Technical Report 304, Polish Academy of Sciences (1977)
7. Bordihn, H., Fernau, H., Holzer, M., Manca, V., Martin-Vide, C.: Iterated sequential transducers as language generating devices. Theoret. Comput. Sci. **369**, 67–81 (2006)
8. Bordihn, H., Holzer, M., Kutrib, M.: Determinization of finite automata accepting subregular languages. Theoret. Comput. Sci. **410**, 3209–3222 (2009)
9. Broda, S., Machiavelo, A., Moreira, N., Reis, R.: Average size of automata constructions from regular expressions. Bull. EATCS **116**, 167–192 (2015)
10. Broda, S., Machiavelo, A., Moreira, N., Reis, R.: On average behaviour of regular expressions in strong star normal form. Int. J. Found. Comput. Sci. **30**(6–7), 899–920 (2019)
11. Broda, S., Machiavelo, A., Moreira, N., Reis, R.: Analytic combinatorics and descriptional complexity of regular languages on average. ACM SIGACT News **51**(1), 38–56 (2020)
12. Brzozowski, J.: In search of most complex regular languages. Int. J. Found. Comput. Sci. **24**(6), 691–708 (2013)
13. Brzozowski, J.A.: Towards a theory of complexity of regular languages. J. Autom. Lang. Comb. **23**(1–3), 67–101 (2018)
14. Câmpeanu, C., Culik, K., Salomaa, K., Yu, S.: State complexity of basic operations on finite languages. In: Boldt, O., Jürgensen, H. (eds.) WIA 1999. LNCS, vol. 2214, pp. 60–70. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45526-4_6
15. Caron, P., le Court, E.H., Luque, J., Patrou, B.: New tools for state complexity. Discret. Math. Theor. Comput. Sci. 22(1) (2020)
16. Champarnaud, J.M., Paranthoën, T.: Random generation of DFAs. Theoret. Comput. Sci. **330**(2), 221–235 (2005)
17. Chrobak, M.: Finite automata and unary languages. Theoret. Comput. Sci. **47**(3), 149–158 (1986)

18. Citrini, C., Crespi-Reghizzi, S., Mandrioli, D.: On deterministic multi-pass analysis. SIAM J. Comput. **15**, 668–693 (1986)
19. Davies, S.: A general approach to state complexity of operations: formalization and limitations. In: Hoshi, M., Seki, S. (eds.) DLT 2018. LNCS, vol. 11088, pp. 256–268. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-98654-8_21
20. Davies, S.: Primitivity, uniform minimality, and state complexity of Boolean operations. Theory Comput. Syst. **62**(8), 1952–2005 (2018)
21. Davies, S.: Algebraic Approaches to State Complexity of Regular Operations. Ph.D. thesis, University of Waterloo, Ontario, Canada (2019)
22. Domaratzki, M.: Enumeration of formal languages. Bull. EATCS **89**, 113–133 (2006)
23. Domaratzki, M., Salomaa, K.: Lower bounds for the transition complexity of NFAs. In: Královič, R., Urzyczyn, P. (eds.) MFCS 2006. LNCS, vol. 4162, pp. 315–326. Springer, Heidelberg (2006). https://doi.org/10.1007/11821069_28
24. De Felice, S., Nicaud, C.: Random generation of deterministic acyclic automata using the recursive method. In: Bulatov, A.A., Shur, A.M. (eds.) CSR 2013. LNCS, vol. 7913, pp. 88–99. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38536-0_8
25. Ferreira, M., Moreira, N., Reis, R.: Forward injective finite automata: exact and random generation of nonisomorphic NFAs. In: Konstantinidis, S., Pighizzini, G. (eds.) DCFS 2018. LNCS, vol. 10952, pp. 88–100. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-94631-3_8
26. Friburger, N., Maurel, D.: Finite-state transducer cascades to extract named entities in texts. Theoret. Comput. Sci. **313**, 93–104 (2004)
27. Gao, Y., Moreira, N., Reis, R., Yu, S.: A survey on operational state complexity. J. Autom. Lang. Comb. **21**(4), 251–310 (2017)
28. Geffert, V., Guillon, B., Pighizzini, G.: Two-way automata making choices only at the endmarkers. Inf. Comput. **239**, 71–86 (2014)
29. Geffert, V., Mereghetti, C., Pighizzini, G.: Converting two-way nondeterministic unary automata into simpler automata. Theoret. Comput. Sci. **295**, 189–203 (2003)
30. Geffert, V., Pighizzini, G.: Two-way unary automata versus logarithmic space. Inf. Comput. **209**(7), 1016–1025 (2011)
31. Ginzburg, A.: Algebraic Theory of Automata. Academic Press, Cambridge (1968)
32. Goldstine, J., et al.: Descriptional complexity of machines with limited resources. J. UCS **8**, 193–234 (2002)
33. Gruber, H., Holzer, M.: On the average state and transition complexity of finite languages. Theoret. Comput. Sci. **387**, 155–166 (2007)
34. Guillon, B., Pighizzini, G., Prigioniero, L., Průša, D.: Two-way automata and one-tape machines. In: Hoshi, M., Seki, S. (eds.) DLT 2018. LNCS, vol. 11088, pp. 366–378. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-98654-8_30
35. Guillon, B., Prigioniero, L.: Linear-time limited automata. Theoret. Comput. Sci. **798**, 95–108 (2019)
36. Hartmanis, J., Stearns, R.E.: Algebraic Structure Theory of Sequential Machines. Prentice-Hall, Hoboken (1966)
37. Héam, P.-C., Joly, J.-L.: On the uniform random generation of non deterministic automata up to isomorphism. In: Drewes, F. (ed.) CIAA 2015. LNCS, vol. 9223, pp. 140–152. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-22360-5_12
38. Hennie, F.C.: One-tape, off-line Turing machine computations. Inform. Control **8**(6), 553–578 (1965)
39. Hibbard, T.N.: A generalization of context-free determinism. Inform. Control **11**(1/2), 196–238 (1967)

40. Holzer, M., Kutrib, M.: State complexity of basic operations on nondeterministic finite automata. In: Champarnaud, J.-M., Maurel, D. (eds.) CIAA 2002. LNCS, vol. 2608, pp. 148–157. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-44977-9_14
41. Holzer, M., Kutrib, M.: Descriptional complexity - an introductory survey. In: Martin-Vide, C. (ed.) Scientific Applications of Language Methods, pp. 1–58. Imperial College Press (2010)
42. Holzer, M., Kutrib, M.: Descriptional and computational complexity of finite automata - a survey. Inform. Comput. **209**, 456–470 (2011)
43. Holzer, M., Kutrib, M.: Automata that may change their mind. In: Freund, R., Hospodár, M., Jirásková, G., Pighizzini, G. (eds.) Non-Classical Models of Automata and Applications (NCMA 2018). books@ocg.at, vol. 332, pp. 83–98. Austrian Computer Society, Vienna (2018)
44. Holzer, M., Kutrib, M.: One-time nondeterministic computations. Int. J. Found. Comput. Sci. **30**, 1069–1089 (2019)
45. Hromkovič, J., Schnitger, G.: Nondeterminism versus determinism for two-way finite automata: generalizations of Sipser's separation. In: Baeten, J.C.M., Lenstra, J.K., Parrow, J., Woeginger, G.J. (eds.) ICALP 2003. LNCS, vol. 2719, pp. 439–451. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-45061-0_36
46. Hromkovič, J., Schnitger, G.: NFAs with and without ϵ-transitions. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 385–396. Springer, Heidelberg (2005). https://doi.org/10.1007/11523468_32
47. Hromkovič, J., Seibert, S., Wilke, T.: Translating regular expressions into small ϵ-free nondeterministic finite automata. J. Comput. System Sci. **62**, 565–588 (2001)
48. Iwama, K., Kambayashi, Y., Takaki, K.: Tight bounds on the number of states of DFAs that are equivalent to n-state NFAs. Theoret. Comput. Sci. **237**, 485–494 (2000)
49. Kapoutsis, C.A.: Minicomplexity. In: Kutrib, M., Moreira, N., Reis, R. (eds.) DCFS 2012. LNCS, vol. 7386, pp. 20–42. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31623-4_2
50. Kapoutsis, C.A.: Minicomplexity. J. Autom. Lang. Comb. **17**(2–4), 205–224 (2012)
51. Kapoutsis, C.A.: Nondeterminism is essential in small two-way finite automata with few reversals. Inf. Comput. **222**, 208–227 (2013)
52. Kapoutsis, C.A.: Two-way automata versus logarithmic space. Theory Comput. Syst. **55**(2), 421–447 (2014)
53. Kapoutsis, C.A.: Minicomplexity - some motivation, some history, and some structure (invited talk extended abstract). In: Catania, B., Královic, R., Nawrocki, J.R., Pighizzini, G. (eds.) SOFSEM 2019. LNCS, vol. 11376, pp. 28–38. Springer (2019)
54. Kapoutsis, C.A., Královic, R., Mömke, T.: Size complexity of rotating and sweeping automata. J. Comput. Syst. Sci. **78**(2), 537–558 (2012)
55. Kapoutsis, C.A., Pighizzini, G.: Two-way automata characterizations of L/poly versus NL. Theory Comput. Syst. **56**(4), 662–685 (2015)
56. Koechlin, F., Nicaud, C., Rotondo, P.: Uniform random expressions lack expressivity. In: Rossmanith, P., Heggernes, P., Katoen, J. (eds.) MFCS 2019. LIPIcs, vol. 138, pp. 51:1–51:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2019)
57. Kutrib, M.: The phenomenon of non-recursive trade-offs. Int. J. Found. Comput. Sci. **16**, 957–973 (2005)

58. Kutrib, M., Malcher, A., Mereghetti, C., Palano, B.: Descriptional complexity of iterated uniform finite-state transducers. In: Hospodár, M., Jirásková, G., Konstantinidis, S. (eds.) DCFS 2019. LNCS, vol. 11612, pp. 223–234. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-23247-4_17
59. Kutrib, M., Malcher, A., Mereghetti, C., Palano, B.: Deterministic and nondeterministic iterated uniform finite-state transducers: computational and descriptional power. In: Anselmo, M., Della Vedova, G., Manea, F., Pauly, A. (eds.) CiE 2020. LNCS, vol. 12098, pp. 87–99. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-51466-2_8
60. Kutrib, M., Malcher, A., Pighizzini, G.: Oblivious two-way finite automata: decidability and complexity. Inf. Comput. **237**, 294–302 (2014)
61. Kutrib, M., Pighizzini, G., Wendlandt, M.: Descriptional complexity of limited automata. Inf. Comput. **259**(2), 259–276 (2018)
62. Kutrib, M., Wendlandt, M.: On simulation cost of unary limited automata. In: Shallit, J., Okhotin, A. (eds.) DCFS 2015. LNCS, vol. 9118, pp. 153–164. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-19225-3_13
63. Lifshits, Y.: A lower bound on the size of ϵ-free NFA corresponding to a regular expression. Inform. Process. Lett. **85**(6), 293–299 (2003)
64. 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 (1966), 328–335
65. Manca, V.: On the generative power of iterated transductions. In: Ito, M., Păun, G., Yu, S. (eds.) Words, Semigroups, and Transductions - Festschrift in Honor of Gabriel Thierrin, pp. 315–327. World Scientific (2001)
66. Maslov, A.N.: Estimates of the number of states of finite automata. Dokllady Akademii Nauk SSSR **194**, 1266–1268 (1970). (in Russian). English translation in Soviet Mathematics Doklady, 11, 1373–1375 (1970)
67. Mealy, G.H.: A method for synthesizing sequential circuits. Bell Syst. Tech. J. **34**, 1045–1079 (1955)
68. 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 (1971)
69. Micali, S.: Two-way deterministic finite automata are exponentially more succinct than sweeping automata. Inf. Process. Lett. **12**(2), 103–105 (1981)
70. 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**(10), 1211–1214 (1971)
71. Nicaud, C.: Étude du comportement en moyenne des automates finis et des langages rationnels. Ph.D. thesis, Université de Paris 7 (2000)
72. Nicaud, C.: On the average size of Glushkov's automata. In: Dediu, A.H., Ionescu, A.M., Martín-Vide, C. (eds.) LATA 2009. LNCS, vol. 5457, pp. 626–637. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-00982-2_53
73. Nicaud, C.: Random deterministic automata. In: Csuhaj-Varjú, E., Dietzfelbinger, M., Ésik, Z. (eds.) MFCS 2014. LNCS, vol. 8634, pp. 5–23. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44522-8_2
74. Pierce, A.: Decision Problems on Iterated Length-Preserving Transducers. Bachelor's thesis, SCS Carnegie Mellon University, Pittsburgh (2011)
75. Pighizzini, G.: Two-way finite automata: old and recent results. Fundam. Inform. **126**(2–3), 225–246 (2013)

76. Pighizzini, G.: Limited automata: properties, complexity and variants. In: Hospodár, M., Jirásková, G., Konstantinidis, S. (eds.) DCFS 2019. LNCS, vol. 11612, pp. 57–73. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-23247-4_4
77. Pighizzini, G., Pisoni, A.: Limited automata and regular languages. Int. J. Found. Comput. Sci. **25**(7), 897–916 (2014)
78. Pighizzini, G., Pisoni, A.: Limited automata and context-free languages. Fundam. Inform. **136**(1–2), 157–176 (2015)
79. Pighizzini, G., Prigioniero, L.: Limited automata and unary languages. Inf. Comput. **266**, 60–74 (2019)
80. Priez, J.B.: Enumeration of minimal acyclic automata via generalized parking functions. In: FPSAC 2015. DMTCS, January 2015
81. Rabin, M.O., Scott, D.: Finite automata and their decision problems. IBM J. Res. Dev. **3**, 114–125 (1959)
82. Sakoda, W.J., Sipser, M.: Nondeterminism and the size of two way finite automata. In: Lipton, R.J., Burkhard, W.A., Savitch, W.J., Friedman, E.P., Aho, A.V. (eds.) Proceedings of the 10th Annual ACM Symposium on Theory of Computing, 1–3 May 1978, San Diego, California, USA, pp. 275–286. ACM (1978)
83. Salomaa, K., Yu, S.: NFA to DFA transformation for finite languages over arbitrary alphabets. J. Autom. Lang. Comb. **2**(3), 177–186 (1997)
84. Sedgewick, R., Flajolet, P.: Analysis of Algorithms. Addision-Wesley, Boston (1996)
85. Shepherdson, J.C.: The reduction of two-way automata to one-way automata. IBM J. Res. Dev. **3**(2), 198–200 (1959)
86. Sipser, M.: Lower bounds on the size of sweeping automata. J. Comput. Syst. Sci. **21**(2), 195–202 (1980)
87. Stearns, R.E.: A regularity test for pushdown machines. Inform. Control **11**, 323–340 (1967)
88. Valiant, L.G.: Regularity and related problems for deterministic pushdown automata. J. ACM **22**, 1–10 (1975)
89. Vardi, M.Y.: A note on the reduction of two-way automata to one-way automata. Inf. Process. Lett. **30**(5), 261–264 (1989)
90. Vardi, M.Y.: Endmarkers can make a difference. Inf. Process. Lett. **35**(3), 145–148 (1990)
91. Wagner, K.W., Wechsung, G.: Computational Complexity. D. Reidel Publishing Company, Dordrecht (1986)
92. Yu, S.: State complexity of regular languages. J. Autom. Lang. Comb. **6**, 221–234 (2001)
93. Yu, S., Zhuang, Q., Salomaa, K.: The state complexities of some basic operations on regular languages. Theoret. Comput. Sci. **125**(2), 315–328 (1994)