Descriptional Complexity of Finite State Automata

Ľubomíra Ištoňová*
Departement of Computer Science
Faculty of Science
Pavol Jozef Šafárik University in Košice
Jesenná 5, 040 01 Košice, Slovakia
istonova@atknet.sk

Abstract

This thesis is devoted to the descriptional complexity of finite state automata. We focus mainly on the transition complexity of one-way nondeterministic automata produced originally from regular expressions and on the state complexity of two-way finite automata equipped with an additional "pebble" movable along the input tape.

We consider conversions of regular expressions into realtime finite state automata, in which the length of each ε-path is bounded by a fixed constant k. For $k = 2$, i.e., for automata in which no ε-path is longer than 2, we show that the conversion of a regular expression into such automaton produces only $O(n)$ states, $O(n)$ ε-free edges, and $O(n \log n)$ ε-edges. We also show how to transform these 2-realtime machines into 1-realtime automata, where each ε-path degenerates into a single ε-edge, still with only $O(n \log n)$ edges. These results break the known lower bound $\Omega(n \log^2 n / \log \log n)$, holding for 0-realtime automata, i.e., for automata with no ε-transitions.

We also study the relation between the standard two-way automata and more powerful devices, namely, one-pebble two-way finite automata. Similarly as in the case of the classical two-way machines, it is not known whether there exists a polynomial trade-off, in the number of states, between the nondeterministic and deterministic pebble two-way automata. However, we show that these two machine models are not independent: if there exists a polynomial trade-off for the classical two-way automata, then there must also exist a polynomial trade-off for the pebble two-way automata. Thus, we have an upward collapse (or a downward separation) from the classical two-way automata to more powerful pebble automata, still staying within the class of regular languages. The same upward collapse holds for complementation of nondeterministic two-way machines. These results are obtained by showing that each pebble machine can be, by using suitable inputs, simulated by a classical two-way automaton (and vice versa), with only a linear number of states, despite the existing exponential blow-up between the classical and pebble two-way machines.

Categories and Subject Descriptors

F.4.3 [**Theory of Computation**]: Mathematical Logic and Formal Languages—*Formal languages*; F.1.1 [**Theory of Computation**]: Computation by abstract devices—*Models of computation*

Keywords

descriptional complexity, finite state automata, regular languages

1. Introduction

This thesis concerns the descriptional complexity of finite state automata. More precisely, we examine two rather independent problems in the field of descriptional complexity of one-way and two-way finite automata.

The analysis of different tools describing formal languages not only with respect to their expressive power, but taking also into account descriptional complexity of these tools, is a classical topic of the formal language theory. Descriptional complexity studies the efficiency while describing objects. Its measure is the description length, represented by the number of states (state complexity), transitions (transition complexity), alphabet size. It also investigates the "inherent" complexity of tools, e.g., upper and lower bounds on the number of states or transitions.

The main topics on which the descriptional complexity is concentrating its attention, could be summarized in the following questions:

- What is the economics of a model (tool) representing a formal language, that is how efficiently can a model describe a formal language besides other models?
- What is the price of a conversion from one tool to another? What are the corresponding lower and upper bounds and can they be attained?

The same questions can be posed when applying operations on models. We shall focus mainly on the second question.

*Recommended by thesis supervisor:
Prof. Viliam Geffert
Defended at Faculty of Science, Pavol Jozef Šafárik University in Košice on September 9, 2010.

Ištoňová, Ľ. Descriptional Complexity of Finite State Automata. Information Sciences and Technologies Bulletin of the ACM Slovakia, Vol. 2, No. 2 (2010) 1-7

The notion of an algorithmic "machine" has appeared for the first time in the 1930's, when Turing defined an abstract model, later called Turing machine. During the next twenty years, several scientists studied simple versions of present finite automata. For example, in the early 1940's, McMulloch and Pitts [49] used finite state machines to model neuron nets. Over the same period, Huffman, Mealy and Moore, independently from each other, introduced the "today's" one-way deterministic finite automata (1DFAs) in [37], [51] and [53], respectively. Intensive study of 1DFAs at that time (sometimes under the name of sequential machines) established many basic properties, including, for example, the Kleene's definition of regular languages [44]:

DEFINITION 1.1. *A language $L \subseteq \Sigma^*$ is* regular *if and only if it can be described by a regular expression α, i.e., $L = L(\alpha)$.*

Thereupon, in 1959, Rabin and Scott [55] suggested the concept of the one-way nondeterministic finite automaton (1NFA) and also proved its equivalence to the one-way deterministic machine, defining the well-known subset construction. They also gave another equivalent definition of regular languages:

DEFINITION 1.2. *[55] A language $L \subseteq \Sigma^*$ is* regular *if and only if it can be accepted by a deterministic or nondeterministic finite automaton M. i.e. $L = L(M)$.*

A rapid development of formal language and automata theory continued also during the 1960's and 1970's. There were many papers published in this area. On behalf of all let us mention at least: examination of relations between different formalisms describing regular languages [50], issue of the determinism versus nondeterminism question in finite automata [62], exploring nondeterminism in two-way finite automata [58], and the birth of other models, while investigating regular languages, augmented by some new and special properties, such are e.g., sweeping automata [64], or pebble automata [6, 56]. We cannot omit one of the first books on finite automata, written by Marcus in 1964 [48].

But automata theory was not only a theoretical branch of the computer science. Besides, it has had quite many applications in practice, for example lexical analysis [1, 2], text editing and pattern matching [45].

Later, at the turn of 1970's and 1980's, it was supposed that all interesting facts about finite automata are known except for a few isolated and very hard problems. It seemed that not much further work could be done on regular languages [68]. However, the contrary is the case. Mainly in the last twenty years, many new and fascinating results were published, for example on state complexity of operations on regular languages [40, 41, 59, 69, 30], minimalization of nondeterministic finite automata [39], trade-offs between diferent formalisms (1DFA, 1NFA, regular expression, two-way automata, pebble automata...), see for example [21, 22, 35, 36, 34, 42, 13], nondeterminism versus determinism problem [42, 43, 34], etc.

Even nowadays there is an important progress in research areas of formal language and automata theory. It seems that the theory of automata and languages experiences a revival these days.

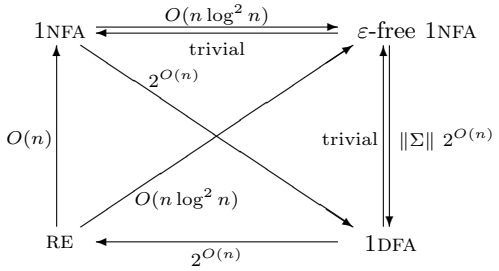

Figure 1: Relations between different models representing regular languages. 1NFA - one-way nondeterministic finite automata including ε-transitions, ε-free 1NFA - one-way nondeterministic finite automata without any ε-transitions, RE - regular expressions, 1DFA - one-way deterministic finite automata.

2. Thesis objectives

The objective of the presented dissertation is to investigate the descriptional complexity of finite state automata, primary focusing on:

- the transition complexity of one-way nondeterministic automata working in the "real" time
- the relation between the state complexities of classical two-way automata and two-way automata equipped with a "pebble" as a marker movable along the input tape.

3. Conversion of regular expressions to realtime automata

Still, not all relations among descriptional complexity of different kinds of formalisms are known, even when restricting to devices describing regular languages only. Despite the simplicity of regular languages, some important problems concerning them are still open. Regular expressions and finite automata without ε-transitions are two of the most popular formalisms for regular languages. The size of an automaton is measured by the number of its transitions, while the size of a regular expression is the number of occurrences of alphabet symbols in it.

It is known that a conversion of a nondeterministic ε-free finite automaton into an equivalent regular expression may cause an exponential blow-up [16]. For the converse direction, it was conjectured that the best conversion is an $O(n^2)$ conversion [63]. Only quite recently [35, 36], it was proved that, for each regular expression of size n, there exists an equivalent ε-free automaton with only $O(n \log^2 n)$ transitions. After that, it has been shown that if we consider regular languages over a binary input alphabet, the above upper bound can be reduced to $O(n \log n)$ transitions [21]. This result was also generalized to the case of regular languages over a fixed alphabet with at most s input symbols: $O(s\, n \log n)$ edges are sufficient here. We have summarized the mentioned results in a demonstrative picture, showing the relations between different formalisms representing regular languages (Figure 1).

The upper bounds, which were presented above, are complemented in [46] by the lower bound $\Omega(n \log^2 n / \log\log n)$.

Comparing $O(n \log^2 n)$ with $\Omega(n \log^2 n / \log\log n)$, that is, the upper and lower bounds, it seems at the first glance that no significant improvement is possible.

However, if we allow ε-transitions, the standard conversion of a regular expression into an equivalent automaton does not produce more than $O(n)$ transitions. The crucial drawback of such automaton is that, due to cycles consisting of ε-edges only, there may exist computation paths of unbounded length.

Therefore, it is quite natural to ask what we have to add to an ε-free automaton to break the lower bound $\Omega(n \log^2 n / \log\log n)$, and still have an automaton with a linear computation time, in the length of the input. This condition can be even more restrictive, by requiring the automaton to execute only a constant number of steps in between reading any two consecutive symbols from the input.

We have named such machines, with the length of all ε-paths bounded by a constant, as "realtime" automata. Realtime devices play an important role in practice. Here we shall concentrate on the simplest realtime devices, without any auxiliary memory, which are one-way finite state automata.

A *k-realtime automaton* M, where $k \geq 0$ is an integer constant, is a nondeterministic finite state automaton in which the length of each ε-path is bounded by k. Therefore, a k-realtime automaton processes the input in "real time," with a constant number of executed steps in between reading any two consecutive symbols from the input. For this reasons, an automaton is called *realtime*, if it is k-realtime for some constant k.

We investigate the properties of the first few levels. As a special case, the automaton is 2-realtime, if each ε-path is of length at most 2. In other words, the machine cannot change its state, without reading an input symbol, more than two times in a row, and hence an input of length ℓ must be processed in at most $3\ell + 2$ steps. Similar, but even more restrictive, properties are typical for 1-realtime automata, where each ε-path degenerates into a single ε-edge.

It should be clear that for $k = 0$ we get exactly the class of ε-free automata.

In this work, we shall show that already for $k = 2$, the conversion of a regular expression produces only $O(n \log n)$ transitions. More precisely:

THEOREM 3.1. *For each regular expression of size $n \geq 1$, there exists an equivalent nondeterministic 2-realtime automaton with at most $4n+2$ states, n ε-free transitions, and $n \cdot (\frac{2}{\log(3/2)} \cdot \log n + 0.744)$ ε-transitions.*

The basic idea of the translation consists of several steps. First, the given regular expression α is converted into the normal form, also with unifying the unary nodes in the tree representation of α with their sons. Thus, we get a special form of a binary tree. Then, the regular expression α is translated into a 1NFA with at most $2n$ states and n ε-free-transitions, according to the following theorem and using rewriting rules described in the dissertation work:

THEOREM 3.2. *[21] Each regular expression α of size $n \geq 1$ can be replaced by en equivalent nondeterministic automaton M with at most $2n$ states and n ε-free transitions, such that:*

(a) For each subexpression β in α, corresponding to a subtree below some node in the tree representation of α, there exists a subautomaton M_β in M, which is a subgraph in the graph representation of M.

(b) For each β', a subexpression of β corresponding to some subtree with the top node located in the subtree for β, $M_{\beta'}$ is a subautomaton of M_β, i.e., a subgraph nested in the subgraph for M_β.

(c) M_β has a single entry point, a state $\mathrm{en}_\beta \in Q$, and a single exit point, a state $\mathrm{ex}_\beta \in Q$, with $\mathrm{en}_\beta \neq \mathrm{ex}_\beta$, such that a string $a_1 \dots a_k \in \Sigma^*$ is in $L(\beta)$ if and only if there exists a path of edges connecting, within the subgraph for M_β, the state en_β with ex_β and labeled by $a_1 \dots a_k$.

(d) Any path going into the subgraph for M_β from the surrounding environment must pass through the state en_β. Further, M_β has no edges ending in en_β.

(e) Any path leaving the subgraph M_β to the surrounding environment must pass through the state ex_β. Further, there are no edges from the surrounding environment ending in ex_β.

The resulting automaton still contains some ε-transitions, and is *not* 2-realtime. Finally, in order to eliminate the lengths of the ε-path to at most 2, we use special sets of incoming and outcoming boundary states.

We also show an easy transformation of these 2-realtime machines into 1-realtime automata, where each ε-path degenerates into a single ε-edge. The basic idea is to replace a path consisting of an ε-free edge followed by an ε-transition by a single ε-free transition. The original ε-edge is removed.

The number of transitions in the resulting automaton is still below $O(n \log n)$.

THEOREM 3.3. *For each regular expression of size $n \geq 2$, there exists an equivalent nondeterministic 1-realtime automaton with at most $4n + 2$ states, $n \cdot (\frac{1}{\log(3/2)} \cdot \log n - 0.128)$ ε-free transitions, and $n \cdot (\frac{1}{\log(3/2)} \cdot \log n + 1.872)$ ε-transitions.*

4. Translation from classical two-way automata to pebble two-way automata

The relation between determinism and nondeterminism is extensively studied. It is one of the key topics in theoretical computer science. The most famous is the $\mathrm{P} \stackrel{?}{=} \mathrm{NP}$ question, but the oldest problem of this kind is

$$\mathrm{DSPACE}(n) \stackrel{?}{=} \mathrm{NSPACE}(n).$$

Similarly, we do not know whether

$$\mathrm{DSPACE}(\log n) \stackrel{?}{=} \mathrm{NSPACE}(\log n).$$

However, a positive answer for the $O(\log n)$ space would imply the positive answer for the $O(n)$ space, and hence

the answers to these two questions are not independent. Analogically, a collapse of nondeterminism with determinism for the $O(\log\log n)$ space would imply the same collapse for the $O(\log n)$ space. (For a survey and bibliography about such translations, see e.g., [20, 67].) Analogous upward translations can be derived for time complexity classes as well.

At first glance, the problem has been resolved for finite state automata. Even a two-way nondeterministic finite automaton (2NFA, for short) and hence any simpler device as well (e.g., its deterministic version, 2DFA) can recognize a regular language only. Thus, 2NFAs can be converted into deterministic one-way automata. However, the problem reappears, if we take into account the size of these automata, measured in the number of states (i.e., estimating the state complexity of these automata).

On one hand, we know that eliminating nondeterminism in one-way n-state automata does not cost more than 2^n states (by the classical subset construction), and that there exist witness regular languages for which exactly 2^n states are indeed required. (For examples of such languages, see [47, 54, 59].)

On the other hand, we know very little about eliminating nondeterminism in the two-way case: it was conjectured by Sakoda and Sipser [58] that there must exist an exponential blow-up for the conversion of 2NFAs into 2DFAs. Nevertheless, the best known lower bound is $\Omega(n^2)$ [14], while the best conversion uses about 2^{n^2} states (converting actually into deterministic one-way machines). Thus, it is not clear whether there exists a polynomial trade-off. The problem has been attacked several times by proving exponential lower bounds for restricted versions of 2DFAs: Sipser [64] — for sweeping machines (changing the direction of the input head movement at the endmarkers only); Hromkovič and Schnitger [34] — for oblivious machines (moving the input head along the same trajectory on all inputs of the same length); Kapoutsis [42] — a computability separation for "moles" (seeing only a part of the input symbol thus traveling "in a network of tunnels" along the input). For machines accepting unary languages, a subexponential upper bound $2^{O(\log^2 n)}$ has been obtained by Geffert, Mereghetti and Pighizzini [23].

It was even observed [58] that there exists a family of regular languages $\{B_n : n \geq 1\}$ which is *complete* for the two-way automata, playing the same role as, e.g., the satisfiability of boolean formulas for the P $\stackrel{?}{=}$ NP question or the reachability in graphs for $\text{DSPACE}(\log n) \stackrel{?}{=} \text{NSPACE}(\log n)$: the trade-off between the 2NFAs and 2DFAs is polynomial if and only if it is polynomial for B_n, i.e., if and only if B_n can be accepted by a 2DFA with a polynomial number of states. (For 2NFAs, n states are enough to accept B_n.)

We have summarized the known upper bounds for conversions between different types of finite state automata on Figure 2.

In the absence of a solution for the general case, it is quite natural to ask whether some properties of the two-way automata cannot be translated into more powerful machines or language classes, in perfect analogy with the corresponding results for the upward translation established for the classical space and time complexity classes. So far, the only result of this kind [4] is that if an exponential trade-off between 2NFAs and 2DFAs could be obtained already by using a subset of the original language that consists of polynomially long strings, then $\text{DSPACE}(\log n) \neq \text{NSPACE}(\log n)$.

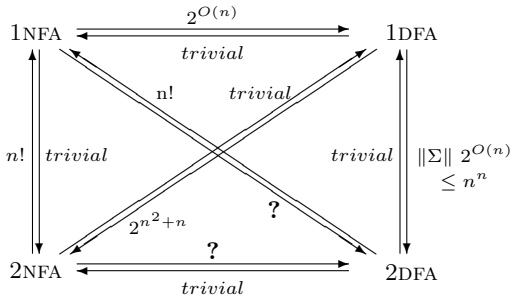

Figure 2: Relations between different models of one-way finite state automata and two-way finite state automata.

In the same spirit as in [4], we shall study the relation between the standard two-way automata and more powerful devices, namely, two-way nondeterministic and deterministic finite automata equipped with a single additional "pebble" (one-pebble machines), movable along the input tape (pebble-2NFA, pebble-2DFA, respectively). The two-way one-pebble machines were first introduced in 1967 by Blum and Hewitt [6]. Later, several scientists were studying variations of devices augmented with one or more pebbles (in particular Turing machines [13, 67], and robots exploring graphs [3]).

Formally, a *one-pebble two-way nondeterministic finite automaton* (pebble-2NFA, for short) is $A = (Q, \Sigma, \delta, q_I, F)$, where Q is the finite set of states, Σ is the finite input alphabet, $\vdash, \dashv \notin \Sigma$ are two special symbols, called the left and the right endmarker, respectively, and $q_I \in Q$ is the initial state. The transition function δ is of the form $\delta : Q \times (\Sigma \cup \Sigma^\bullet \cup \{\vdash, \dashv, \vdash^\bullet, \dashv^\bullet\}) \to 2^{Q\times\{-1,0,+1,-1^\bullet,+1^\bullet\}}$. The presence of the pebble on the current input tape symbol $a \in \Sigma \cup \{\vdash, \dashv\}$ is indicated by using the symbol $a^\bullet \in \Sigma^\bullet \cup \{\vdash^\bullet, \dashv^\bullet\}$, while the new input head movements $-1^\bullet, +1^\bullet$ are introduced to move the pebble. More precisely, a classical transition in the form $\delta(q, a) \ni (q', d)$, with $a \in \Sigma\cup\{\vdash, \dashv\}$ and $d \in \{-1, 0, +1\}$, is applicable only if the pebble in not placed on the current input tape symbol (change the current state from q to q' and move the input head in the direction d), while $\delta(q, a^\bullet) \ni (q', d)$ can be executed only if the pebble is placed on $a \in \Sigma \cup \{\vdash, \dashv\}$ at the moment (move the input head in the direction d, but leave the pebble in its original position). Finally, a transition $\delta(q, a^\bullet) \ni (q', d^\bullet)$, with $d^\bullet \in \{-1^\bullet, +1^\bullet\}$, moves also the pebble in the same direction d, together with the input head. Transitions in the form $\delta(q, a) \ni (q', d^\bullet)$ are meaningless, and hence not allowed.

The machine A starts its computation in the initial state q_I with both the input head and the pebble placed on the left endmarker, and accepts by reaching, anywhere along the input tape, a final state $q \in F$. Similarly, the final position of the pebble is irrelevant for acceptance.

A one-pebble two-way *deterministic* finite state automaton (pebble-2DFA) is defined in the usual way.

Despite the fact that such single pebble can be used to mark some input tape position, even a nondeterministic one-pebble machine cannot accept a nonregular language [13, 67]. However, measured in the number of states, the pebble machines are much more powerful. Converting a pebble-2DFA to a classical 2NFA may require an exponential blow-up, i.e., the loss of the pebble cannot be compensated economically by gaining nondeterminism:

THEOREM 4.1. *For each sufficiently large n, there exists a finite unary language L_n that can be accepted by a pebble-2DFA with n states, but for which each 2NFA requires at least $2^{\Omega(\sqrt{n \cdot \log n})}$ states.*

Similarly as in the case of the classical two-way machines, we do not know whether there exists a polynomial trade-off between the pebble-2NFAs and pebble-2DFAs. However, we show that these two machine models are related:

THEOREM 4.2. *If, for some function $f(n)$, each 2NFA with n states can be replaced by an equivalent 2DFA with at most $f(n)$ states (no pebbles), then each pebble-2NFA with m states can be replaced by an equivalent pebble-2DFA having no more than $5 \cdot f(3m)$ states.*

In particular, if $f(n) \leq O(n^k)$, that is, if there exists a polynomial transformation from nondeterministic to deterministic classical two-way automata, then there must also exist a polynomial transformation, with the same degree of the polynomial, from nondeterministic to deterministic two-way automata equipped with a pebble, since $5 \cdot (3m)^k = (5 \cdot 3^k) \cdot m^k \leq O(m^k)$.

Thus, we have an upward collapse (and a downward separation) between the classical two-way automata and the much more powerful pebble model, but staying still within the class of regular languages.

A similar upward collapse holds for the trade-off between a two-way nondeterministic automaton accepting a language L and a machine for the complement of L:

THEOREM 4.3. *If, for some function $f(n)$, each 2NFA with n states can be replaced by a 2NFA with at most $f(n)$ states recognizing the complement of the original language (no pebbles), then each pebble-2NFA with m states can be replaced by a pebble-2NFA with no more than $5f(3m)$ states recognizing the complement.*

In particular, if $f(n) \leq O(n^k)$, that is, if there exists a polynomial transformation for complementing nondeterministic classical two-way automata, then there must also exist a polynomial transformation, with the same degree of the polynomial, for complementing nondeterministic two-way automata equipped with a pebble.

COROLLARY 4.4. *For each pebble-2DFA with m states, there exists a pebble-2DFA with at most $60 \cdot m$ states recognizing the complement of the original language.*

So far, the problem of the polynomial trade-off is open for both these models. These results are obtained by showing that each pebble-2NFA (or pebble-2DFA) can be, by using suitable inputs, simulated by a classical 2NFA (or 2DFA, respectively) with only a linear number of states, despite the fact, that the blow-up between the classical and pebble two-way automata is exponential. (See Theorem 4.1 above.)

The same holds for the corresponding conversions in the opposite direction, from the classical machines to pebble machines.

5. Concluding remarks

In this thesis, we have studied the descriptional complexity of finite state automata. More precisely, our interest has been focused on two research topics: we have examined the transition (and state) complexity of the translation of regular expressions into one-way finite automata, and the state complexity while eliminating nondeterminism within two-way pebble automata.

We have shown a conversion of regular expressions into nondeterministic 2-realtime automata, using at most $4n+2$ states, n ε-free transitions and $n \cdot (\frac{2}{\log(3/2)} \cdot \log n + 0.744)$ ε-edges. We have also shown a construction of a 1-realtime automaton with at most $4n+2$ states, $n \cdot (\frac{1}{\log(3/2)} \cdot \log n - 0.128)$ ε-free transitions and $n \cdot (\frac{1}{\log(3/2)} \cdot \log n + 1.872)$ ε-transitions. Alternatively, there exists also another construction of a 1-realtime automaton, with a smaller number of states and ε-transitions, but with a slightly increased number of ε-free transitions. In all above cases, the total number of all edges is bounded by $O(n \log n)$.

How many transitions are needed for a general case of k-realtime automata, where $k > 2$, is still an open problem. It might be possible that, by using ε-paths longer than 2, we may potentially save some transitions. This is related to the open question of whether there exists a fixed constant k such that, for each n, the resulting k-realtime automaton will have only $O(n)$ transitions. The best (rather trivial) construction of a realtime automaton with a linear number of transitions, known to the authors, can be obtained by using the automaton of Theorem 3.2 as a starting point, in which we eliminate all cycles composed of ε-edges by unifying all states in such a cycle into a single state. However, this results in an $(n-1)$-realtime automaton, that is, in $k = n - 1$, which is not a fixed constant.

In Table 1, we give a survey on complexity of different types automata resulted from conversions of regular expressions.

Afterwards, we have discussed the conversion of pebble-2NFAs to pebble-2DFAs from the state complexity perspective. Already in 1978, it was conjectured by Sakoda and Sipser that there must exist an exponential blow-up, in the number of states, for the transformation of the classical 2NFAs into 2DFAs. Nevertheless, this problem is still open. We have shown, by Theorem 4.2 above, that such blow-up could possibly be derived by proving an exponential gap between pebble-2NFAs and pebble-2DFAs. Even showing a less impressive lower bound for the pebble-2NFA versus pebble-2DFA trade-off, say, $\Omega(n^k)$ with some $k \geq 3$, would imply the same lower bound $\Omega(n^k)$ for the classical 2NFA versus 2DFA conversion. (To the best of authors' knowledge, the highest lower bound obtained so

Type of automata	States	ε-free edges	ε-edges
ε-free automata (0-realtime)	$O(n)$	$O(n \log^2 n)$	0
1 realtime automata	$O(n)$	$O(n \log n)$	$O(n \log n)$
2 realtime automata	$O(n)$	$O(n)$	$O(n \log n)$
k realtime automata $k > 2$	?	?	?
ε-automata	$O(n)$	$O(n)$	$O(n)$

Table 1: Conversion of regular expressions into 1NFA. The best known upper bounds for different types of automata.

far is $\Omega(n^2)$ [14].) Since a pebble automaton is a different computational model, the argument might use some different witness languages.

Similarly, by Theorem 4.3, proving an exponential gap for the complementation of the pebble-2NFAs would imply the same exponential gap for the complementation of the classical 2NFAs. This, in turn, would imply the exponential gap for the trade-off between 2NFAs and 2DFAs, and also between pebble-2NFAs and pebble-2DFAs, since the complementation for the deterministic two-way machines is linear (namely, $4{\cdot}n$ states for 2DFAs, by [24], and at most $60 \cdot n$ states for pebble-2DFAs, by Corollary 4.4 above).

The most natural related open problem is whether the translation results presented in Theorems 4.2 and 4.3 cannot be generalized to two-way automata equipped with more than one pebble placed on the input tape. More precisely, we do not know whether a polynomial trade-off between nondeterministic and deterministic k-pebble two-way automata implies the polynomial trade-off for automata equipped with $k+1$ pebbles. The same question can be asked about complementation of multi-pebble 2NFAs. (The argument might be quite difficult, since such machines can accept non-regular languages: as an example, already with 2 pebbles we can easily recognize $L = \{a^n b^n c^n : n \geq 0\}$, which is known not to be context-free.) Nevertheless, the answers to these questions might bring a deep insight into the world of $O(\log n)$ space bounded computations, since the multi-pebble 2NFAs and 2DFAs correspond to the complexity classes NSPACE($\log n$) and DSPACE($\log n$), respectively (see Section 3.2 in [67]). For example, we know that NSPACE($\log n$) is closed under complement, by the inductive counting [38, 66], but the inductive counting technique increases the number of pebbles.

Similarly, we need translation(s) among other computational models that are weak enough to stay within the class of regular languages but strong enough to provide, in some cases, a more succinct representation than the classical models. (As an example, we do not know too much about complexity of two-way automata equipped with a pushdown of a constant height [22].)

References

[1] A. V. Aho and J. D. Ullman. The Theory of parsing, translation, and compiling, Vol. I. *Prentice-Hall, Englewood Cliffs*, 1972.

[2] A. V. Aho, R. Sethi, and J. D. Ullman. Compilers - Principles, techniques, and tools. *Addison-Wesley, Reading*, 1986.

[3] M. A. Bender, A. Fernández, D. Ron, and A. Sahai. The power of a pebble: Exploring and mapping directed graphs. *Information and Computation*, Vol. 176, Issue 1–21, 2002.

[4] J. Berman and A. Lingas. On the complexity of regular languages in terms of finite automata. Tech. Rep. 304, Polish Academy of Sciences, 1977.

[5] J. C. Birget. Intersection and union of regular languages and state complexity. *Inform. Process. Lett.*, 43, 185–190, 1992.

[6] M. Blum nad C .Hewitt. Automata on a 2-dimensional tape. *Art. Int. Memo*, 135,1967.

[7] R. Book, S. Even, S. Greibach, and G. Ott. Ambiguity in graphs and expressions. *IEEE Trans. Comput.*, C-20, 149–153, 1971.

[8] C. Boyer. *A History of Mathematics*. John Wiley & Sons, 1968.

[9] R. Brzozowski. Derivatives of regular expressions. *Journal of ACM.*, 11, 481–494, 1964.

[10] A. Brüggemann+Klein. Regular expressions into finite automata. *Theoretical Computer Science*, 120, 197–213, 1993.

[11] J. M. Champarnaud, D. Zaidi. From C-continuations to new quadratic algorithms for automaton synthesis. *Int. Journal of Algebra and Computation*, 11, 707–735, 2001.

[12] R. Chang, J. Hartmanis, and D. Ranjan. Space bounded computations: Review and new separation results. *Theoret. Comput. Sci.*, 80, 289–302, 1991.

[13] J. H. Chang, O. H. Ibarra, M. A. Palis, and B. Ravikumar. On pebble automata. *Theoret. Comput. Sci.*, 44, 111–21, 1986.

[14] M. Chrobak. Finite automata and unary languages. *Theoret. Comput. Sci.*, 47, 149–58, 1986. (Corrigendum: *ibid.*, 302, 497–98, 2003).

[15] P. Ďuriš, J. Hromkovič, S. Jukna, M. Sauerhoff, G. Schnitger. On multipartition communication complexity. *In: Proc. STACS '01. Lecture Notes in Computer Science*, Springer 2001, 206–217, 2010. (Corrigendum: *ibid.*, 302, 497–98, 2003).

[16] A Ehrenfeucht, P. Zieger. Complexity measures for regular expressions. *Journal of Computer and System Sciences*, 12, 134–146, 1976.

[17] W. Ellison and F. Ellison. *Prime Numbers*. John Wiley & Sons, 1985.

[18] J. Engelfriet and H. J. Hoogeboom. Tree-walking pebble automata. In *Jewels are forever*, (J. Karhumäki et al., eds.), Springer Verlag 1999, 72–83.

[19] V. Geffert. Nondeterministic computations in sublogarithmic space and space constructibility. *SIAM J. Comput.*, 20, 484–98, 1991.

[20] V. Geffert. Bridging across the $\log(n)$ space frontier. *Inform. & Comput.*, 142, 127–58, 1998.

[21] V. Geffert. Translation of binary regular expressions into nondeterministic ε-free automata with $O(n \log n)$ transitions. *Journal of Computer and System Sciences*, 67, 451-472, 2003.

[22] V. Geffert, C. Mereghetti, and B. Palano. More concise representation of regular languages by automata and regular expressions. In *Proc. Develop. Lang. Theory*, *Lect. Notes Comput. Sci.*, 5257, pp. 359–70. Springer-Verlag, 2008.

[23] V. Geffert, C. Mereghetti, and G. Pighizzini. Converting two-way nondeterministic unary automata into simpler automata. *Theoret. Comput. Sci.*, 295, 189–203, 2003.

[24] V. Geffert, C. Mereghetti, and G. Pighizzini. Complementing two-way finite automata. *Inform. & Comput.*, 205, 1173–87, 2007.

[25] V. Geffert, C. Mereghetti, and G. Pighizzini. One pebble versus $\log n$ bits. *In Proceedings of the workshop Non-Classical Models of Automata and Applications (NCMA 2009)*, 121–133, 2009.

[26] S. Ginsburg. *Algebraic and Automata-Theoretic Properties of Formal Languages*. North-Holland, 1975.

[27] V. M. Glushkov. The abstract theory of automata. *Russian Math. Surveys*, 16, 1–53, 1961. Translation from Usp. Math. Naut. 16, 3-41, 1961, by J. M. Jackson,

[28] Chr. Hagenah and A. Muscholl. Computing ε-free 1NFA from Regular Expressions in $O(n \log^2 n)$ Time. *RAIRO Theoretic Informatics and Application*, 34, 257–277, 2000.

[29] J. Hartmanis, P. M. Lewis II, and R. E. Stearns. Memory bounds

for the recognition of context free and context sensitive languages. In *IEEE Conf. Record on Switching Circuit Theory and Logical Design*, pp. 191–202, 1965.

[30] M. Holzer and M. Kutrib. State complexity of basic operations on nondeterministic finite automata. In: J. M. Champarnaud, D. Maruel (Eds.). *Implementation and Application of automata (CIAA 2002)*, LNCS 2608, Springer-Verlag, Heidelberg, 148–157, 2003.

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

[32] J. E. Hopcroft, R. Motwani, J. D. Ullman. Introduction to Automata Theory, Languages, and Computation. *Addison-Wesley*, 2001

[33] J. Hromkovič. Descriptional complexity of finite automata: concepts and open problems. *J. Autom. Lang. Comb.*, 7, 519–531, Springer Berlin, 1997.

[34] J. Hromkovič and G. Schnitger. Nondeterminism versus determinism for two-way nondeterministic automata: Generalizations of Sipser's separation. In *Proc. Internat. Colloq. Automata, Languages, & Programming*, *Lect. Notes Comput. Sci.*, 2719, pp. 439–51. Springer-Verlag, 2003.

[35] J. Hromkovič, S Seibert, T. Wilke. Translating regular expressions into small ε-free nondeterministic automata. *Proceedings of the Symposium on Theoretical Aspects of Computers Science, Lecture Notes in Computer Science*, Vol. 1200, 55–66, Springer Berlin, 1997.

[36] J. Hromkovič, S Seibert, T. Wilke. Translating regular expressions into small ε-free nondeterministic finite automata. *Journal of Computer and System Science*, 62, 565–588, 2001.

[37] D. A. Huffman. The synthesis of sequential swithing circuits. *J. Franklin Inst.*, No. 3-4, 161–190,257–303, 1954.

[38] N. Immerman. Nondeterministic space is closed under complementation. *SIAM J. Comput.*, 17, 935–38, 1988.

[39] T. Jiang and B. Ravikumar. Minimal NFA problems are hard. *Proceedings of 18th ICALP*, LNCS 510,Springer-Verlag, 629–640, 1991.

[40] J. Jirásek, G. Jirásková, A. Szabari. State complexity of concatenation and complementation of regular languages. *Pre-Proc. 9th Conference of Implementation and Application of Automata*, Kingston, Ontario, Canada, 132–141, 2004.

[41] G. Jirásková. State complexity of some operations on regular languages. In E. Csuhaj-varjú, C. Kintala, D. Wotschke, Gy. Vaszil (Eds.). *Proc. 5th workshop Descriptional Complexity of formal Systems*, MTA SZTAKI, Budapest,114–125, 2003.

[42] Ch. A. Kapoutsis. Deterministic moles cannot solve liveness. *J. Automat. Lang. Combin.*, 12, 215–35, 2007.

[43] Ch. A. Kapoutsis. Size complexity of two-way finite automata. *Proc.* 13*th International Conference , Developments in Language Theory, Lecture Notes in Computer Science*, Vol. 5583, 47–66, 2009.

[44] S. C. Kleene. Representation of events in nerve nets and finite automata. *In C .E. Shannon and J. McCarthy, Automata Studies, Princeton Univ. Press*, 3–42, 1956.

[45] D. E. Knuth, J. H. Morris, and V. R. Pratt. Fast pattern matching in strings. *SIAM Journal on Computing*, vol. 6, no. 2, 323–350,1977.

[46] Y. Lifshits. A Lower Bound on the Size of ε-free NFA Corresponding to a Regular Expression, Manuscript, St. Petersburg State University, 2002.

[47] O. B. Lupanov. Über den Vergleich zweier Typen endlicher Quellen. *Probleme der Kybernetik*, 6, 329–35, 1966. (Akademie-Verlag, Berlin, in German).

[48] S. Marcus. Gramatici si automate finite (Finite Grammars and Automata). *Ed Academiei (the Publishing House of the Romanian Academy)*, Bucharest, 1943.

[49] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bull. Math. Biophysics* , 5, 115–133, 1943.

[50] R. F. McNaughton and H. Yamada. Regular expressions and state graphs for automata. *IRE Trans. Electron. Comput.*, EC-9(1), 39–47, 1960.

[51] G. M. Mealy. A method for synthesizing sequential circuits. *Bell System Technical Journal*,34, No 5,1045–1079, 1955.

[52] C. Mereghetti and G. Pighizzini. Optimal simulations between unary automata. *SIAM J. Comput.*, 30, 1976–92, 2001.

[53] E. F. Moore. Gedanken-experiments on Sequential Machines. *Automata Studies, Annals of Mathematical Studies*,34, 129Ű153,Princeton University Press, 1956.

[54] F. Moore. On the bounds for state-set size in the proofs of equivalence between deterministic, nondeterministic, and two-way finite automata. *IEEE Trans. Comput.*, C-20, 1211–14, 1971.

[55] M. O. Rabin and D. Scott. Finite automata and their decision problems. *IBM J. Research and Development*, 3:2, 115–125, 1959.

[56] R. Ritchie and F. Springsteel. Language recognition by marking automata. *Infor. and Control*, 20, 313–330, 1972.

[57] A. Rosenfeld. Picture Languages. *Academic Press, New York*, 1979.

[58] W. Sakoda and M. Sipser. Nondeterminism and the size of two-way finite automata. In *Proc. ACM Symp. Theory of Comput.*, pp. 275–86, 1978.

[59] A. Salomaa, D. Wood, and S. Yu. On the state complexity of reversals of regular languages. *Theoret. Comput. Sci.*, 320, 315–29, 2004.

[60] K. Salomaa. Descriptional complexity of nondeterministic finite automata. *DLT 2007, LNCS*, 4588, 31–35, 2007.

[61] S. Seibert. Efficient transformations from regular expressions to finite automata. *Preprint*, 2004.

[62] J. I. Seiferas. Manuscript communicated to Michael sipser. 1973.

[63] S. Sippu, E. Soisalon-Soininen. Parsing Theory, Vol. I: Languages and Parsing. *EATCS Monographs of Theoretical Computer Science, Vol. 15, Springer, Berlin*, 1988.

[64] M. Sipser. Lower bounds on the size of sweeping automata. In *Proc. ACM Symp. Theory of Comput.*, pp. 360–64, 1979.

[65] M. Sipser. Halting space bounded computations. *Theoret. Comput. Sci.*, 10, 335–38, 1980.

[66] R. Szelepcsényi. The method of forced enumeration for nondeterministic automata. *Acta Inform.*, 26, 279–84, 1988.

[67] A. Szepietowski. *Turing Machines with Sublogarithmic Space*, *Lect. Notes Comput. Sci.*, vol. 843. Springer-Verlag, 1994.

[68] S. Yu. Regular languages. *In Handbook of formal languages,edited by G. Rozenberg and A. Salomaa*, Springer, 41–110, 1998.

[69] S. Yu. State complexity: recent results nad open problems. *Fundamenta Informaticae* 64, 471–480, 2005.

[70] S. Yu, Q. Zhuang, and K. Salomaa. The state complexity of some basic operations on regular languages. *Theoretical Computer Science*,125, 315-328, 1994.

Papers by the Author

V. Geffert, Ľ. Ištoňová. Translation from Classical Two-Way Automata to Pebble Two-Way Automata. Submitted to *RAIRO - Theoretical Informatics and Applications - Informatique Théorique et Applications*.

V. Geffert, Ľ. Ištoňová. Translation from Classical Two-Way Automata to Pebble Two-Way Automata. *EPTCS (Electronic Proceedings in Theoretical Computer Science)*, Vol.3, pages 137-146, 2009. ISSN 2075-2180, doi:10.4204/EPTCS 3.13

V. Geffert, Ľ. Ištoňová. Translation from Classical Two-Way Automata to Pebble Two-Way Automata. In *Proc. of DCFS'09 (Descriptional Complexity of Formal Systems, 11th Internat. Workshop)*, pages 175–186, Published by IFIP and Univ.Magdeburg, 2009.

V. Geffert, Ľ. Ištoňová. Conversion of regular expressions into realtime automata. In *RAIRO - Theoretical Informatics and Applications - Informatique Théorique et Applications, EDP Science*, Vol 40, pages 611–629. Springer Boston, 2006.

Ľ. Ištoňová. Conversion of regular expressions into realtime automata. In Proc. Information Technologies - Applications and Theory, *ITAT 2005, workshop on Theory and Practice of Information Technologies*, pages 65–74., 2005.