Reversal of regular languages and state complexity

Juraj Šebej

Institute of Computer Science, Faculty of Science, P. J. Šafárik University
Jesenná 5, 04001 Košice, Slovakia
juraj.sebej@gmail.com

Abstract. *We study the state complexity of languages that can be obtained as reversals of regular languages represented by deterministic finite automata. We show that the state complexity of the reversal of a regular language with state complexity n is between $\log n$ and 2^n. We first prove that the upper bound is tight in the ternary case. Then we present binary languages reaching this upper bound on the reversal. We also obtain some other partial results in the binary case.*

1 Introduction

Regular languages and finite automata are the oldest and the simplest topics in computer science. They have been investigated since the 1950s. Despite their simplicity, some problems are still open. Probably the most challenging is the question of how many states are sufficient and necessary for two-way deterministic automata to simulate two-way nondeterministic automata which is connected to the well-known DLOGSPACE vs. NLOGSPACE problem.

Motivating by applications of regular languages in software engineering, programming languages, and other areas in computer science, as well as by their importance in theory, this class of languages is intensively studied in recent years; for the discussion, we refer the reader to [8, 23]. Various areas in this field are now deeply and intensively examined. One of such areas is descriptional complexity which studies the cost of description of languages represented by different formal systems such as deterministic and nondeterministic finite automata, two-way automata, regular expressions, or grammars.

Rabin and Scott in 1959 [18] described an algorithm for the conversion of nondeterministic finite automata into deterministic automata known as the subset construction. The algorithm shows that every n-state nondeterministic automaton can be simulating by at most 2^n state deterministic automaton. In 1963, Lupanov [16] proved the optimality of this construction by describing a ternary and even a binary regular language accepted by an n-state nondeterministic automaton that requires exactly 2^n deterministic states.

Maslov in 1970 [13] considered the state complexity of union, product, and Kleene star. He gave binary worst-case examples for these three operations, however he did not present any proofs. Birget in his works [1, 2] examined intersection and union of several languages, and also the question of the size of nondeterministic automaton for complements. The systematic study of the state complexity of operations on regular languages began in the paper by Yu, Zhuang, and Salomaa [24]. This work was followed by papers studying state complexity of operations on unary languages [17] and on finite languages [3], complexity of proportional removals [5], and shuffle in [4].

Another stream of research is the study of so called "magic" numbers, where not only worst-case complexities are important, but also all values that can be obtained as a corresponding complexity are considered. The problem was stated by Japanese authors Iwama, Kambayashi, and Takaki [9] who asked what values can be obtained as the size of the minimal deterministic automaton equivalent to a given n-state nondeterministic automaton. The values that cannot be obtained in such a way are called "magic" numbers in [10]. The following research showed that there are no magic numbers in the ternary case [12], while a lot of them exist in the unary case [6]. The binary case is still open.

Similar results for the size of nondeterministic automata for complements can be found in [19], for the union and intersection in [7], and for the reversal and star in [11]. In all cases, the whole range of complexities can be obtained, however while in the case of union and intersection the used alphabet is fixed, in the case of reversal and star, the alphabet grows exponentially with n.

In this paper, we continue the study of the state complexity of reversals of regular languages. In 1966, Mirkin [14] pointed out that Lupanov's ternary worst-case example is a reversal of a deterministic automaton, which proves that the complexity of the reversal of a language accepted by a ternary n-state deterministic automaton is 2^n. The binary language with more than one accepting state reaching this upper bound has been given in 1983 by Leiss [15]. In 2004, the paper [20] claimed a binary worst-case example with a single accepting state. Unfortunately, the result does not hold: in the case of $n = 8$, the number of reachable states in the subset automaton for the re-

versal is 252 instead of 256. Since the result has been used in the literature several times, our first aim is to present a correct example, and a correct proof. We start with an observation that all states in the subset automaton corresponding to the nfa that is obtained as a reversal of a minimal dfa are pairwise inequivalent. We show that the state complexity of the reversal of an n-state dfa language is between $\log n$ and 2^n, and present a ternary worst-case example with a very simple proof of reachability of all subsets. In a much more difficult way, we prove that the upper bound 2^n is tight also in the binary case. Our witness automaton has a single accepting state, and is uniformly defined for all integers n. Therefore, it can be used in all cases where the incorrect result from [20] was used. We next find binary n-state deterministic automata that need $n+1$ or $n+2$ deterministic states for their reversals. Finally, we present binary 1-, 2-, and 3-state automata that reach all particular values from $\log n$ to 2^n as the state complexity of their reversals.

2 Preliminaries

This section gives some basic definitions, notations, and preliminary results used throughout the paper. For further details, we refer to [21, 22].

Let Σ be a finite alphabet and Σ^* the set of all strings over the alphabet Σ including the empty string ε. The length of a string w is denoted by $|w|$. A language is any subset of Σ^*. We denote the cardinality of a finite set A by $|A|$ and its power-set by 2^A.

A *deterministic finite automaton* (dfa) is a 5-tuple $M = (Q, \Sigma, \delta, s, F)$, where Q is a finite set of states, Σ is a finite input alphabet, δ is the transition function that maps $Q \times \Sigma$ to Q, s is the starting state, $s \in Q$, and F is the set of accepting states, $F \subseteq Q$. In this paper, all dfa's are assumed to be complete, that is, the next state $\delta(q, a)$ is defined for every state q in Q and every symbol a in Σ. The transition function δ is generalized to a function from $Q \times \Sigma^*$ to Q in a natural way. A string w in Σ^* is accepted by the dfa M if the state $\delta(s, w)$ is an accepting state of the dfa M. The *language accepted by* the dfa M, is the set $L(M) = \{w \in \Sigma^* \mid \delta(s, w) \in F\}$.

A *nondeterministic finite automaton* (nfa) is a 5-tuple $M = (Q, \Sigma, \delta, S, F)$, where Q, Σ, S and F are defined identically as for a dfa, S is the set of starting states, and δ is now the nondeterministic transition function that maps $Q \times \Sigma$ to 2^Q. The transition function can be naturally generalized to the domain $Q \times \Sigma^*$. A string w in Σ^* is accepted by the nfa M if the set $\delta(q_0, w)$ contains an accepting state of the nfa M. The *language accepted by* the nfa M is $L(M) = \{w \in \Sigma^* \mid \delta(S, w) \cap F \neq \emptyset\}$.

Two automata are *equivalent* if they recognize the same language. A dfa (an nfa) M is called *minimal* if every dfa (every nfa, respectively) that is equivalent to M has at least as many states as M. It is well-known that a dfa $M = (Q, \Sigma, \delta, s, F)$ is minimal if all its states are reachable from the starting state and no two its different states are equivalent (states p and q are equivalent if for all strings w in Σ^*, the state $\delta(p, w)$ is accepting if and only if the state $\delta(q, w)$ is accepting). Every regular language has a unique minimal dfa, up to the naming of states. However, the same result does not hold for nfa's.

The *state complexity* of a regular language is the number of states in its minimal dfa. A regular language with deterministic state complexity n is called an n-state dfa language.

Every nfa $M = (Q, \Sigma, \delta, S, F)$ can be transformed to an equivalent deterministic finite automaton $M' = (2^Q, \Sigma, \delta', s', F')$ thanks to an algorithm known as the "subset construction" in the following way. Every state of the dfa M' is a subset of the state set Q. The starting state of the dfa M' is the set S. The transition function δ' is defined by $\delta'(R, a) = \bigcup_{r \in R} \delta(r, a)$ for every state R in 2^Q and every symbol a in Σ. A state R in 2^Q is an accepting state of the dfa M' if it contains at least one accepting state of the nfa M. We call the dfa M' the subset automaton corresponding to the nfa M. The subset automaton M' need not be minimal since some states may be unreachable or equivalent.

We next give the definitions and some preliminary results concerning the reversal operation.

Definition 1. *The reversal w^R of a string w is defined as follows: $\varepsilon^R = \varepsilon$ and if $w = a_1 a_1 \cdots a_n$ with $a_i \in \Sigma$, then $w^R = a_n a_{n-1} \cdots a_2 a_1$. The reversal of a language L is the language $L^R = \{w^R \mid w \in L\}$.*

The reversal of a dfa $A = (Q, \Sigma, \delta, s, F)$ is the nfa A^R obtained from A by reversing all transitions and by swapping the role of starting and accepting states, that is $A^R = \big(Q, \Sigma, \delta^R, F, \{s\}\big)$, where $\delta^R(q, a) = \{p \in Q : \ \delta(p, a) = q\}$.

Proposition 1. *The reversal of a dfa A recognizes the language $L(A)^R$.*

Proof. We prove that a string w is in $L(A)^R$ if and only if the string w is accepted by the nfa A^R.

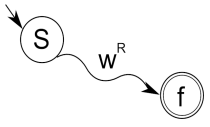

Fig. 1. The string w^R is accepted by the dfa A.

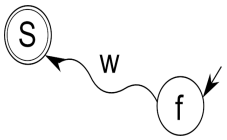

Fig. 2. The string w is accepted by the nfa A^R.

If w is in $L(A)^R$, then w^R is in $L(A)$, and so the starting state s goes to an accepting state f in F by w^R. It follows that the starting state f of the nfa A^R goes to the accepting state s of A^R by w, and so w is accepted by A^R.

Next, if a string w is accepted by the nfa A^R, then there is a starting state f in F that goes to the accepting state s of A^R by w. It turns out, that in the dfa A, the starting state s goes to an accepting state f by w^R. Thus the string w^R is in the language $L(A)$, and so the string w is in the language $L^R(A)$. □

Since a language is regular if and only if it is recognized by a dfa or, equivalently, by an nfa, we get the following result.

Corollary 1. *The reversal of every regular language is a regular language.*

After the construction of nfa for the reversal of a regular language we can use the subset construction to get a dfa for the reversal. This gives the following bounds on the size of the dfa.

Theorem 1. *Let L be a regular language accepted by a minimal n-state dfa. Then the minimal dfa for the language L^R has at most 2^n and at least $\lceil log_2 n \rceil$ states.*

Proof. Let A be an n-state dfa for a language L. The reversal A^R of the dfa A is an n-state nfa for the language L^R. After applying the subset construction to this nfa A^R, we get at most 2^n-state dfa for the language L^R. Now since $(L^R)^R = L$, the lower bound is $\lceil \log n \rceil$. □

We now prove quite interesting result that in the subset automaton corresponding to the reversal of a minimal dfa, all states are pairwise inequivalent. This means that we need not prove inequivalence of states troughtout the paper.

Lemma 1. *Let for each state q of an nfa there exists a string w_q such that w_q is accepted by the nfa from state q, but is not accepted from any other state. Then in the corresponding subset automaton, all states are pairwise inequivalent.*

Proof. Let $M = (Q, \Sigma, \delta, S, F)$ be an nfa, and let for each state q in Q, w_q be a string that is accepted by M only from state q. Let S and T be two different subsets in the subset automaton corresponding to the nfa M. Then, without loss of generality,there exists a state q in Q such that $q \in S$ and $q \notin T$. It follows that the string w_q is accepted by the subset automaton from state S but not from state T. Thus the states S and T are inequivalent. □

Theorem 2. *All states in the subset automaton corresponding to the reversal of a minimal dfa are pairwise inequivalent.*

Proof. Let us show that every nfa obtained as the reversal of a minimal dfa satisfies the condition in Lemma 1. Let q be a state of the nfa. Since state q is reachable in the given dfa, there exists a string x such that the starting state of the dfa goes to state q by x, as illustrated in Fig. 3.

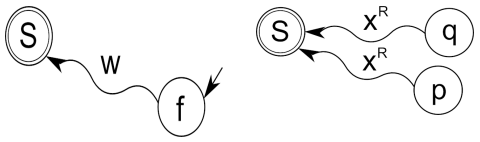

Fig. 3. State q is reachable in the dfa A (left); $p \neq q$ in the nfa A^R (right) implies two distinct conputations of the dfa on the string x.

This means that the string x^R is accepted by the nfa from state q, see Fig. 3. We now prove that the string x^R is not accepted by the nfa from any other state. Assume for contradiction that the string x^R is accepted by the nfa from a state p with $p \neq q$. It turns out that the starting state of the dfa might go by the string x to state q as well as to state p, which is a contradiction since in the dfa we woud have two different computations on the string x. Hence the nfa satisfies the condition of Lemma 1, and so all states in the corresponding subset automaton are pairwise inquivalent. □

3 Ternary alphabet

We start with the upper bound 2^n in the ternary case. The next theorem presents a ternary worst-case example for the reversal with a very simple proof of reachability of all subsets.

Theorem 3. *For every integer n with $n \geq 3$, there exists an n-state dfa A over a three-letter alphabet such that the minimal dfa for the reversal of the language $L(A)$ has 2^n states.*

Proof. Let A be the minimal n-state dfa shown in Fig. 4. Construct an nfa for the reversal of the language $L(A)$ from the dfa A by reversing all transitions,

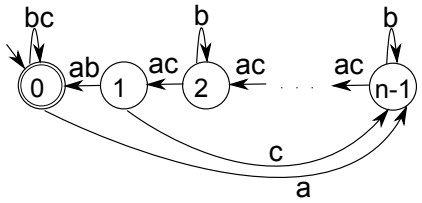

Fig. 4. The ternary dfa A reaching the bound 2^n.

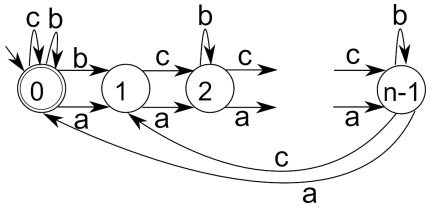

Fig. 5. The nfa for the reversal of the language $L(A)$.

and exchanging the starting and accepting states. The nfa is shown in Fig. 5.

Let us show that the corresponding subset automaton has 2^n reachable and pairwise inequivalent states. We first show that every set containing state 0 is reachable. The proof is by induction on the size of sets. The basis, $|S| = 1$, holds true because state 0 is the starting state of the subset automaton. Assume that every set of size k, $1 \leq k \leq n-1$, containing state 0 is reachable. Let $S = \{0, i_1, i_2, ..., i_k\}$ with $1 \leq i_1 < i_2 < \cdots < i_k \leq n-1$ be a set of size $k+1$. Consider the set $S' = \{0, i_2 - i_1 + 1, ..., i_k - i_1 + 1\}$. The set S' is of size k and contains state 0, and so is reachable by the induction hypothesis. The set S' goes to the set S by bc^{i_1-1} since S' goes to $\{0, 1, i_2 - i_1 + 1, \ldots, i_k - i_1 + 1\}$ by b, and then to S by c^{i_1-1}. It turns out that the set S is reachable.

We next prove the reachability of sets without state 0. Let $S = \{i_1, i_2, ..., i_k\}$ with $1 \leq i_1 < i_2 < \cdots < i_k \leq n-1$. Then the set S is reached from the set $\{0, i_2 - i_1, \ldots, i_k - i_1\}$, containing state 0, by a^{i_1}. Finally, the empty set is reached from the set $\{1\}$ by b. This completes the proof since the inequivalence follows from Theorem 2. □

4 Binary alphabet and upper bound

The authors of the paper [20] present a binary n-state dfa and claim that its reversal requires 2^n deterministic states. Unfortunately, the example does not work: in the case of $n = 8$, the resulting dfa has 252 reachable states instead of 256. The next theorem describes correct binary n-state witness dfa's with a single accepting state, uniformly defined for every n with $n \geq 2$.

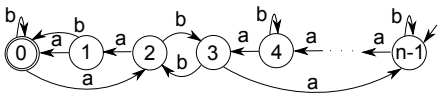

Fig. 6. The binary dfa A reaching the bound 2^n.

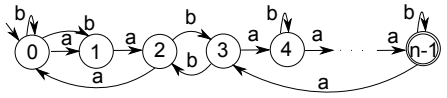

Fig. 7. The nfa A^R for the binary language $L(A)^R$.

Theorem 4. *For every integer n with $n \geq 2$, there exists an n-state dfa A over a two-letter alphabet such that the minimal dfa for the reversal of the language $L(A)$ has 2^n states.*

Proof. Let us consider a binary n-state dfa A in Fig. 6 with states $0, 1, \ldots, n-1$, where $n \geq 4$, state n is the starting state and state 0 is the sole accepting state. For all $i = 4, 5 \ldots, n-1$, state i goes to state $i-1$ by symbol a, and to itself by symbol b. State 3 goes to state $n-1$ by symbol a, and to state 2 by b. State 2 goes to state 1 by a, and to state 3 by b. State 1 goes to state 0 by both symbols a and b. State 0 goes to state 2 by a, and to itself by b. In the case of $n = 2$ or $n = 3$, there are some small changes in the structure of the automaton. If $n = 2$, then state 0 goes to state 1 by symbol a. If $n = 3$, then state 2 goes to itself by symbol b.

In these two cases, we reverse the dfa A, and after the determinisation of the reversal, we get a four-state minimal dfa if $n = 2$ in Fig. 12, and an eight-state minimal dfa if $n = 3$ in Fig. 17.

Now let $n \geq 4$. Construct an nfa for the reversal of the language $L(A)$ by exchanging the starting and accepting states, and by reversing all transitions in the dfa A, see Fig. 7. We are going to show that the corresponding subset automaton has 2^n reachable states. To make the proof more understandable, we call the set of states $\{0, 1, 2\}$ the first part, and the set of states $\{3, 4, \ldots, n-1\}$ the second second part of the nfa.

We will consider two cases:
1. $n = 3k + 1$ or $n = 3k + 2$,
2. $n = 3k$,
where k is a positive integer.

1. If $n = 3k + 1$ or $n = 3k + 2$, then the number of states in the first part is three, while the number of states in the second part is $3(k-1) + 1$ or $3(k-1)+2$. Thus these two numbers are are relativily prime. First, the set $\{0, 1\}$ is reached from the starting set $\{0\}$ by symbol b. Now we demonstrate how to add a new state ℓ to a set $\{0, 1\} \cup S$, where S is a subset of the second part with $\ell \notin S$, to get a set $\{0, 1\} \cup S \cup \{\ell\}$. By symbol a, we can rotate states in both parts.

Consider the set $\{0,1,\ell\}$. Since the sizes of the two parts are relatively primes, there exists an integer x such that the set $\{0,1,\ell\}$ goes to the set $\{0,2,3\}$ by the string a^x. Apply the string a^x to the set $\{0,1\}\cup S$, and then apply symbol b. We get the set $\{0,1,3\}\cup S'$, where S' is a rotation of the set S by the string a^x. And now, again, there exists an integer y such that the set $\{0,1,3\}\cup S'$ goes to the set $\{0,1\}\cup S\cup\{\ell\}$ by a^y. So, in this way, we can reach every set $\{0,1\}\cup S$. Let us show how to get every subset of states in first part without changing the second part. Every set $\{0,1\}\cup S$ goes to the set $\{1,2\}\cup S$ as well as to the set $\{0,2\}\cup S$ by an appropriate numbers of a's. Every set $\{1,2\}\cup S$ goes to the set $\{2\}\cup S$ by bb, and then to $\{0\}\cup S$ and $\{1\}\cup S$ by an appropriate numbers of a's. Every set $\{0,2\}\cup S$ goes to the set $\{0,1,2\}\cup S$ by bb. Finally, every set $\{1\}\cup S$ goes to the set $\emptyset\cup S$ by bb. This completes the proof of reachability if $n=3k+1$ or $n=3k+2$.

2. If $n=3k$, we can split the states of the nfa into triples, the first part is a triple 0, and the second part consists of triples $1,2,\ldots,k-1$. We first reach the set $\{0,1,2\}$ from the starting set $\{0\}$ by the string $baabb$. Let us show how to set a triple in the second part without changing the other triples. We use the automaton B shown in Fig. 8. In automaton B, every set is reachable from the set $\{0,1,2\}$. Assume we want to set the ℓ-th triple with $2\le\ell\le k-1$, and let us denote the states of this triple by ℓ_0,ℓ_1,ℓ_2. We choose which configaration for this triple we want obtain, and show that we set this configuration with the 0-th triple set to $\{0,1,2\}$. When finally setting the first triple, we also show how to set it with an arbitrary configuration in the 0-th triple. So, first let $\ell\ge 2$. A configuration in this triple is given by a subset S of $\{3,4,5\}$. We first count the numbers of a's in the string on a path from $\{0,1,2\}$ to $\{0,1,2\}\cup S$ in the dfa B, and denote it by $a_\#$. Now consider some starting strings:

$a^{s_0}=a^{3.(k-1-\ell+1)}$,
$a^{s_1}=a^{3.(k-1-\ell+1)-1}$,
$a^{s_2}=a^{3.(k-1-\ell+1)-2}$;

different starting strings are needed because the number of a's must be a multiply of 3 in the end.

Next we move the ℓ-th triple to the place of first triple by one the of starting strings a^{s_0},a^{s_1},a^{s_2}: if $a_\#\pmod 3=0$ we use a^{s_0} so we get ℓ_0,ℓ_1,ℓ_2 at the place of the first triple, if $a_\#\pmod 3=1$ we use a^{s_1} so we get ℓ_1,ℓ_2,X at place of first triple, if $a_\#\pmod 3=2$ we use a^{s_2} so we get ℓ_2,X,X at place of first triple where X is a state from some other triple, thus we cannot modify X.

Next we proceed by the string w and count the number of a's. If the starting string was a^{s_0}, after the 1st, 4th, 7th, ... symbol a, we apply a rotation a^{rot} where a $a^{rot}=a^{3.(k-2)}$, so that we do not modify the other triples. Similarly, if the starting string was a^{s_1}, we apply the rotation a^{rot} after the 2nd, 5th, 8th, ... symbol a. Finally, if the starting string was a^{s_2}, we apply the rotation after the 3rd, 6th, 9th, ... symbol a.

Now we have set the ℓ-th triple, but still have to move the triple to its place ℓ: we just need to apply the string $a^{3(\ell-1)}$ (a back string).

So the complete string consists of one of the strating strings, a new route string, and a back string. Thus in this way, we can set the 0-th triple $\{0,1,2\}$ with all triples except for the first triple. We set the first triple in a similar way, but now we use paths from $\{0,1,2\}$ to every state in the dfa B. That means that all subsets are reachable. This completes the proof of reachability for $n=3k$. □

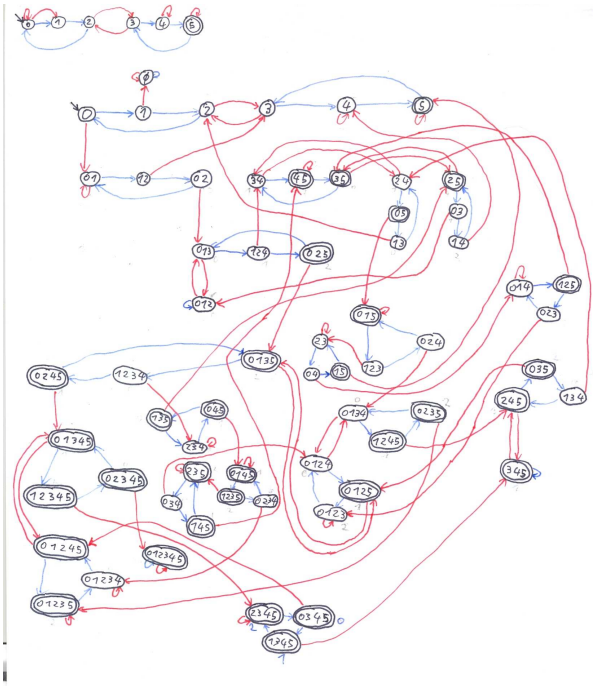

Fig. 8. The nfa A^R from Proof of Theorem 4 for $n=6$ (left up corner), and dfa for $L(A)^R$, the main part of the picture. Red lines correspond to the transitions by symbol b, and blue lines to the transitions by symbol a.

5 Unary alphabet

We now show that we cannot reduce the size of the alphabet to one symbol.

Theorem 5. *The minimal dfa for the reversal of every unary n-state dfa language has n states.*

Proof. Every string w in a unary language L consists only of symbols, for example, a. Therefore, $w=w^R$, and so $L=L^R$. That means that the reversal of the language L has also complexity n. □

6 Binary automata with one, two, and three states

In this section, we examine the reversals of regular languages that can be accepted by one-, two- and three-state dfa's. We first observe that the reversal of a one-state dfa language is the same language. It turns out that that the reversal of no two-state dfa language can be accepted by a one-state dfa, and so in this case, the lower bound $\log 2$ cannot be reached. On the other hand, we show that all other possible values, that is, 2, 3, and 4, can be obtained as the size of the minimal dfa for the reversal of a two-state binary dfa language. We next prove that all values from 2 to 8 can be reached as the number of states in the minimal dfa recognizing the reversal of a binary language represented by a three-state deterministic finite automaton.

Theorem 6. *The reversal of every one-state dfa language is a one-state dfa language.*

Proof. Let us prove the theorem by inspecting all one-state automata. We only have two possibilities shown in Fig. 9. If the state is accepting, then the automaton accepts all strings. If the state is rejecting, the automaton does not accept any string. In both cases, the reversal is the same language, and so is accepted by the same one-state dfa. □

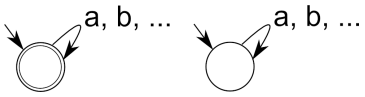

Fig. 9. The one-state dfa accepting all strings (left), and the one-state dfa accepting no strings (right).

Theorem 7. *For each α with $2 \leq \alpha \leq 4$, there exists a two-state binary dfa A such that the minimal dfa for the reversal of the language $L(A)$ has exactly α states.*

Proof. The corresponding automata for $\alpha = 2, 3, 4$ are shonw in Fig. 10, Fig. 11, and Fig. 12, respectively. The figures show a two-state dfa, its reversal, and the reachable states in the corresponding subset automaton. By Theorem 2, the subset automata are minimal.

Theorem 8. *For each α with $2 \leq \alpha \leq 8$, there exists a three-state binary dfa A such that the minimal dfa for the reversal of the language $L(A)$ has exactly α states.*

Proof. Similarly as in the previous proof, we show the appropriate three-state binary automata for $\alpha = 2, 3, 4, 5, 6, 7, 8$ in Figures 13, 14, 15, 16, 17.

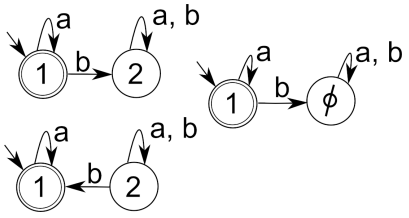

Fig. 10. The dfa A (top left), the reversal of A (bottom left), the subset automaton for the reversal; $\alpha = 2$.

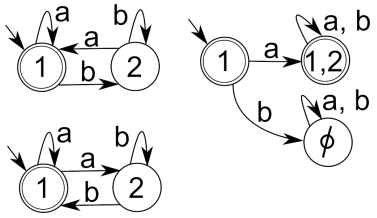

Fig. 11. The dfa A, the reversal of A, the subset automaton for the reversal; $\alpha = 3$.

7 Binary alphabet

In this section, we describe n-state dfa's whose reversals need exactly $n+1$ and $n+2$ deterministic states. Notice that by Theorem 5, the reversal of an n-state unary language needs exactly n-states.

Theorem 9. *For every integer n with $n \geq 2$, there exists an n-state dfa A over a two-letter alphabet such that the minimal dfa for the reversal of the language $L(A)$ has $n+1$ states.*

Proof. Let $n \geq 2$. Consider the n-state dfa A shown in Fig. 18 with states $1, 2, \ldots, n$, of which 1 is the starting state and also the sole accepting state. For all $i = 1, 2, \ldots, n-1$ state i goes by symbol a to state $i+1$, and state n goes by symbol a to itself. For all $i = 2, 3, \ldots, n$ state i goes by symbol b to state $i-1$, and state 1 goes by b to itself. The dfa A is minimal since for two states i, j with $i < i$, the string b^{i-1} is accepted from state i but not from state j.

Construct an nfa for the reversal of the language $L(A)$ by swapping the starting and accepting states, and by reversing all transitions in A. Let us show that the corresponding subset automaton has $n+1$ reachable states . The set $\{1\}$ is reachable because it is the starting state in the subset automaton. The set $\{1\}$ goes to the empty set by symbol a, and to the set $\{1, 2\}$ by symbol b. Every set $\{1, 2, \ldots, i\}$ with $2 \leq i \leq n-1$ goes to the set $\{1, 2, \ldots, i-1\}$ by a, and to the set $\{1, 2, \ldots, i+1\}$ by b. The set $\{1, 2, \ldots, n\}$ goes to itself by a and b. Thus the sets $\{1\}$, $\{1, 2\}$, ..., $\{1, 2, \ldots, n\}$, and the empty set are reachable, while no other set is reachable. It follows that the minimal dfa for the reversal of $L(A)$ has $n+1$ sets. □

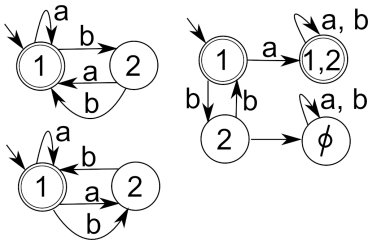

Fig. 12. The dfa A, the reversal of A, the subset automaton for the reversal; $\alpha = 4$. □

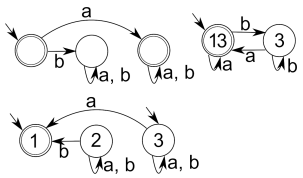

Fig. 13. The dfa A (top left), the reversal of A (bottom left), the subset automaton for the reversal; $\alpha = 2$.

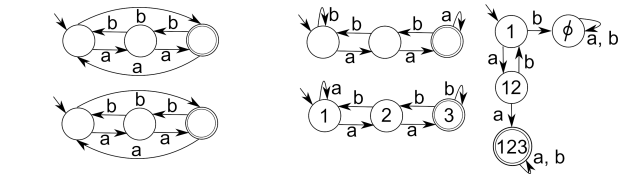

Fig. 14. The dfa A, the reversal of A, the subset automaton for the reversal; $\alpha = 3, 4$.

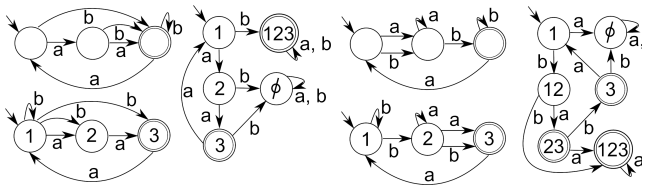

Fig. 15. The dfa A, the reversal of A, the subset automaton for the reversal; $\alpha = 5, 6$.

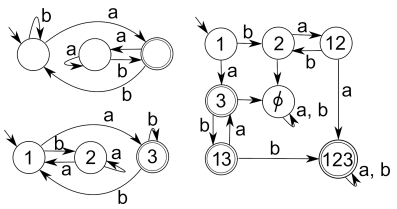

Fig. 16. The dfa A, the reversal of A, the subset automaton for the reversal; $\alpha = 7$.

Theorem 10. *For every integer n with $n \geq 2$, there exists an n-state dfa A over a two-letter alphabet such that the minimal dfa for the reversal of the language $L(A)$ has $n + 2$ states.*

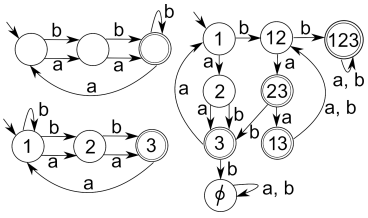

Fig. 17. The dfa A, the reversal of A, the subset automaton for the reversal; $\alpha = 8$. □

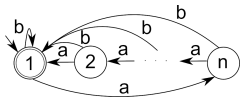

Fig. 18. The n-state binary dfa requiring $(n+1)$-state dfa for the reversal.

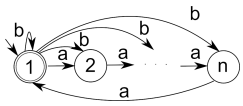

Fig. 19. The n-state binary dfa requiring $(n+2)$-state dfa for the reversal.

Proof. Let $n \geq 2$. Consider the n-state dfa A shown in Fig. 19 with states $1, 2, \ldots, n$, of which 1 is the starting and the sole accepting state. Each state i goes to state $i - 1$ by symbol a, except for state 1 that goes to state n by symbol a. Each state i goes to state 1 by symbol b. The dfa A is minimal since for each state i, the string a^{i-1} is accepted only from state i.

Construct an nfa for the reversal of the language $L(A)$ by swapping the starting and accepting states, and by reversing all transitions in A. Let us show that the corresponding subset automaton has $n+2$ reachable states. The set $\{1\}$ is the starting state in the suset automaton. For all $i = 1, 2, \ldots, n-1$, the set $\{i\}$ goes to set $\{i+1\}$ by symbol a, and the set $\{n\}$ goes to the set $\{1\}$ by symbol a. The set $\{1\}$ goes to set $\{1, 2, \ldots, n\}$ by symbol b. Each set $\{i\}$ with $i \geq 2$ goes to the empty set by symbol b. The set $\{1, 2, \ldots, n\}$ goes to itself by symbols a and b. So the sets $\{1\}$, $\{2\}$, …, $\{n\}$, $\{1, 2, \ldots, n\}$, and the empty set are reachable, while no other set is reachable. □

8 Conclusions

We studied the state complexity of languages that can be obtained as reversals of regular languages represented by deterministic finite automata. We showed that the state complexity of the reversal of a regular language with state complexity n is between $\log n$ and 2^n. We gave a simple proof of a fact that the upper bound is tight in the ternary case. Then we presented binary languages reaching this upper bound on

the reversal. Our witness deterministic automata have a single accepting state, which can be used in some results in the literature instead of an incorrect example in [20]. We also obtained some other partial results in the binary case for one-, two-, and three-state automata. We described automata, the reversal of which has state complexity n, $n+1$, and $n+2$. In future, we want to do statistics of reachable complexities for the reversal of all automata up to five states. We also want to find automata, with other complexities then n, $n+1$, $n+2$, and 2^n, and try to answer the question whether all values from $\log n$ to 2^n can be reached, or whether there are some "magic numbers" for the reversal.

References

1. J.-C. Birget: *Intersection and union of regular languages and state complexity.* Inform. Process. Lett. **43**, 1992, 185–190.
2. J.-C. Birget: *Partial orders on words, minimal elements of regular languages, and state complexity.* Theoret. Comput. Sci. **119**, 1993, 267–291.
3. C. Câmpeanu, K. Culik II, K. Salomaa, S. Yu: *State complexity of basic operations on finite languages.* In: WIA'99, LNCS, vol. 2214, 2001, 60–70.
4. C. Câmpeanu, K. Salomaa, S. Yu: *Tight lower bound for the state complexity of shuffle of regular languages.* J. Autom. Lang. Comb. **7**, 2002, 303–310.
5. M. Domaratzki: *State complexity and proportional removals.* J. Autom. Lang. Comb. **7**, 2002, 455–468.
6. V. Geffert: *Magic numbers in the state hierarchy of finite automata.* In: Královič, R., Urzyczyn, P. (eds.) MFCS 2006. LNCS, vol. 4162, 2006, 412–423.
7. M. Hricko: *Finite automata, regular languages, and state complexity.* Master's Thesis. P.J. Šafárik University in Košice, Slovakia, 2005.
8. J. Hromkovič: *Descriptional complexity of finite automata: Concepts and open problems.* J. Autom. Lang. Comb. **7**, 2002, 519–531.
9. K. Iwama, Y. Kambayashi K. Takaki: *Tight bounds on the number of states of DFAs that are equivalent to n-state NFAs.* Theoret. Comput. Sci. **237**, 2000, 485–494.
10. K. Iwama, A. Matsuura, M. Paterson: *A family of NFAs which need $2^n - \alpha$ deterministic states.* Theoret. Comput. Sci. **301**, 2003, 451–462.
11. G. Jirásková: *On the state complexity of complements, stars, and reversals of regular languages.* In: Ito M., Toyama M. (eds.) DLT 2008. LNCS, vol. 5257, Springer, 2008, 431–442.
12. G. Jirásková: *Magic numbers and ternary alphabet.* In: Diekert, V. (ed.) DLT 2009, LNCS, vol. 5583, Springer, Heidelberg, 2009, 300–311.
13. A.N. Maslov: *Estimates of the number of states of finite automata.* Soviet Math. Dokl. **11**, 1970, 1373–1375.
14. B.G. Mirkin: *On dual automata.* Kibernetika (Kiev) **2**, 1966, 7–10, (in Russian). English translation: Cybernetics **2**, 1966, 6–9.
15. E. Leiss: *Succinct representation of regular languages by Boolean automata.* Theoret. Comput. Sci. **13**, 1981, 323–330.
16. U.I. Lupanov: *A comparison of two types of finite automata.* Problemy Kibernetiki **9**, 1963, 321–326.
17. G. Pighizzini, J. Shallit: *Unary language operations, state complexity and Jacobsthal's function.* Internat. J. Found. Comput. Sci. **13**, 2002, 145–159.
18. M. Rabin, D. Scott: *Finite automata and their decision problems.* IBM Res. Develop. **3**, 1959, 114–129.
19. A. Szabari: *Regular languages and descriptional complexity.* PhD. Thesis, in preparation.
20. A. Salomaa, D. Wood, S. Yu: *On the state complexity of reversals of regular languages.* Theoret. Comput. Sci. **320**, 2004, 315–329.
21. M. Sipser: *Introduction to the theory of computation.* PWS Publishing Company, Boston, 1997.
22. S. Yu: *Chapter 2: Regular languages.* In: Rozenberg, G., Salomaa, A. (eds.) Handbook of Formal Languages - Vol. I, Springer, Heidelberg, 1997, 41–110.
23. S. Yu: *A renaissance of automata theory?* Bull. Eur. Assoc. Theor. Comput. Sci. **72**, 2000, 270–272.
24. S. Yu, Q. Zhuang, K. Salomaa: *The state complexity of some basic operations on regular languages.* Theoret. Comput. Sci. **125**, 1994, 315–328.