Available online at www.sciencedirect.com
SCIENCE DIRECT®
Theoretical Computer Science 330 (2005) 287–298
Theoretical Computer Science
www.elsevier.com/locate/tcs

State complexity of some operations on binary regular languages

Galina Jirásková[1]

Mathematical Institute, Slovak Academy of Sciences, Grešákova 6, 040 01 Košice, Slovakia

Received 17 November 2003; received in revised form 8 April 2004; accepted 21 April 2004

Abstract

We investigate the state complexity of some operations on binary regular languages. In particular, we consider the concatenation of languages represented by deterministic finite automata, and the reversal and complementation of languages represented by nondeterministic finite automata. We prove that the upper bounds on the state complexity of these operations, which were known to be tight for larger alphabets, are tight also for binary alphabets.
© 2004 Elsevier B.V. All rights reserved.

Keywords: State complexity; Regular language operations; Binary languages

1. Introduction

Regular languages and finite automata are one of the oldest topics in computer science. They have been extensively studied since the 1950s. Nevertheless, some important problems concerning finite automata are still open. For instance, we recall the question how many states are sufficient and necessary for two-way deterministic finite automata to simulate two-way nondeterministic finite automata; the problem is closely related to the famous open question whether or not DLOGSPACE equals NLOGSPACE [1,25].

Recently, there has been renewed interest in regular languages and finite automata (see [15,31] for a discussion). Some aspects of this area are now intensively investigated. One such aspect is the state complexity of regular languages and their operations.

E-mail address: jiraskov@saske.sk (G. Jirásková).

[1] Research supported by the VEGA Grant no. 2/3164/23.

0304-3975/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2004.04.011

The state complexity of a regular language is the number of states of its minimal deterministic finite automaton (DFA). The nondeterministic state complexity of a regular language is the number of states of a minimal nondeterministic finite automaton (NFA) accepting the language. If we speak about the state complexity of an operation on regular languages, we ask how many states are sufficient and necessary in the worst case to accept the language resulting from the operation.

Some early results concerning the state complexity of regular languages can be found in [19–21]. The state complexity of some operations on regular languages was investigated in [2,3,18]. Yu et al. [27] were the first to systematically study the complexity of regular language operations. Their paper was followed by several articles investigating the state complexity of finite languages operations and unary languages operations [5,22,23]. The nondeterministic state complexity of regular languages operations was studied by Holzer and Kutrib in [10–13]. Further results on this topic are presented in [6,7] and state-of-the-art surveys for DFAs can be found in [29,30].

In this paper, we investigate the state complexity of some operations on binary regular languages and provide answers to some problems which have been open for the binary case. In particular, we consider the concatenation of DFA languages, and the reversal and complementation of NFA languages.

For the concatenation of DFA languages, the worst case $m2^n - 2^{n-1}$ was given by an m-state DFA language and an n-state DFA language over a three-letter alphabet in [27]. We show that the worst case can be reached by the concatenation of two binary DFA languages.

The reversal of any n-state NFA language can be accepted by an $(n + 1)$-state NFA and this upper bound was shown to be tight for a three-letter alphabet by Holzer and Kutrib [13]. We give a binary n -state NFA language reaching the upper bound on the reversal.

To accept the complement of any n-state NFA language 2^n states suffice since we can simply convert a given NFA to an equivalent DFA and then exchange accepting and rejecting states. Birget [3] claimed that the upper bound is tight for a three-letter alphabet but later corrected this to a four-letter alphabet [4]. We prove that the upper bound is also tight for a binary alphabet by presenting a binary n-state NFA language such that any NFA accepting its complement needs at least 2^n states.

To prove the result for concatenation we show that a deterministic finite automaton is minimal. We obtain the lower bound on reversal using a counting argument. To obtain the lower bound on complementation we use a fooling-set lower-bound technique known from communication complexity theory [14], cf. also [2,3,9].

The paper consists of six sections, including this introduction, and an appendix. The next section contains basic definitions and notations used throughout the paper. In Section 3 we present our result for concatenation. Section 4 deals with the reversal operation. In Section 5 we investigate the concatenation operation. The last section contains concluding remarks and open problems. In the appendix, we give some omitted proofs.

2. Preliminaries

In this section, we recall some basic definitions and notations. For further details, the reader may refer to [26,28].

Let Σ be an alphabet and Σ^* the set of all strings over the alphabet Σ including the empty string ε. The length of a string $w \in \Sigma^*$ is denoted by $|w|$. The power-set of a set A is denoted by 2^A.

A *deterministic finite automaton* (DFA) is a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$, where Q is a finite set of states, Σ is a finite input alphabet, $\delta : Q \times \Sigma \to Q$ is the transition function, $q_0 \in Q$ is the initial state, and $F \subseteq Q$ is the set of accepting states. In this paper, all DFAs are assumed to be complete, i.e., the next state $\delta(q, a)$ is defined for any q in Q and any a in Σ. The transition function δ is extended to a function from $Q \times \Sigma^*$ to Q in the natural way. A string w in Σ^* is accepted by the DFA M if the state $\delta(q_0, w)$ is an accepting state of M.

A *nondeterministic finite automaton* (NFA) is a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$, where Q, Σ, q_0, and F are as above, and $\delta : Q \times \Sigma \to 2^Q$ is the transition function which can be naturally extended 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 M.

The *language accepted by* a finite automaton M, denoted $L(M)$, is the set of all strings accepted by M. Two automata are said to be *equivalent* if they accept the same language.

Any nondeterministic finite automaton $M = (Q, \Sigma, \delta, q_0, F)$ can be converted to an equivalent deterministic finite automaton $M' = (Q', \Sigma, \delta', q_0', F')$ using an algorithm known as the "subset construction" [24] in the following way. Every state of the DFA M' is a subset of Q. The initial state of M' is $\{q_0\}$. A state $R \subseteq Q$ is an accepting state of the DFA M' if it contains an accepting state of the NFA M. The transition function $\delta' : Q' \times \Sigma \to Q'$ is defined by

$$\delta'(R, a) = \bigcup_{r \in R} \delta(r, a).$$

3. Concatenation

We start our investigation with the concatenation operation. The state complexity of the concatenation of regular languages represented by deterministic finite automata was studied by Yu et al. [27]. They showed that $m2^n - 2^{n-1}$ states are sufficient for a DFA to accept the concatenation of an m-state DFA language and an n-state DFA language. In the case of $n = 1$, the upper bound m was shown to be tight for a unary alphabet. In the case of $m = 1$ and $n \geqslant 2$, the worst case $2^n - 2^{n-1}$ was given by the concatenation of two binary languages. Otherwise the upper bound $m2^n - 2^{n-1}$ was proved to be tight for a three-letter alphabet. The next theorem shows that the upper bound can be reached by the concatenation of two binary languages.

Theorem 1. *For any integers $m \geqslant 2$ and $n \geqslant 2$, there exist a binary DFA A of m-states and a binary DFA B of n-states such that any DFA accepting the language $L(A)L(B)$ needs at least $m2^n - 2^{n-1}$ states.*

Proof. Let m and n be arbitrary but fixed integers such that $m \geqslant 2$ and $n \geqslant 2$. Let $d = (m - n + 1) \bmod (m - 1)$ and let $\Sigma = \{a, b\}$.

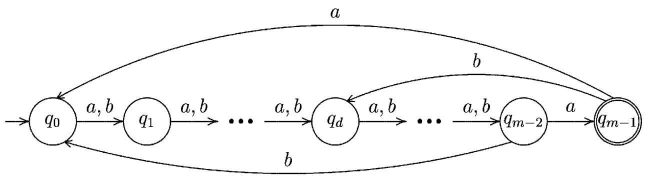

Fig. 1. The deterministic finite automaton A; $d = (m - n + 1) \bmod (m - 1)$.

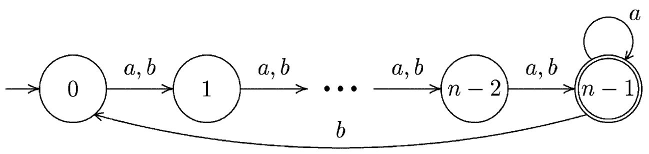

Fig. 2. The deterministic finite automaton B.

Define an m-state DFA $A = (Q_A, \Sigma, \delta_A, q_0, F_A)$, where $Q_A = \{q_0, q_1, \ldots, q_{m-1}\}$, $F_A = \{q_{m-1}\}$, and for any $i \in \{0, 1, \ldots, m-1\}$,

$$\delta_A(q_i, X) = \begin{cases} q_j,\ j = (i+1) \bmod m & \text{if } X = a, \\ q_{i+1} & \text{if } i \leqslant m-3 \text{ and } X = b, \\ q_0 & \text{if } i = m-2 \text{ and } X = b, \\ q_d,\ d = (m-n+1) \bmod (m-1) & \text{if } i = m-1 \text{ and } X = b. \end{cases}$$

Define an n-state DFA $B = (Q_B, \Sigma, \delta_B, 0, F_B)$, where $Q_B = \{0, 1, \ldots, n-1\}$, $F_B = \{n-1\}$, and for any $i \in \{0, 1, \ldots, n-1\}$,

$$\delta_B(i, X) = \begin{cases} i+1 & \text{if } i \leqslant n-2 \text{ and } X = a, \\ n-1 & \text{if } i = n-1 \text{ and } X = a, \\ (i+1) \bmod n & \text{if } X = b. \end{cases}$$

The DFA A and B are shown in Figs. 1 and 2, respectively.

We first describe an NFA accepting the language $L(A)L(B)$, then we construct an equivalent DFA and show that it has at least $m2^n - 2^{n-1}$ states no two of which are equivalent.

Let $C = (Q, \Sigma, \delta, q_0, F)$, where $Q = Q_A \cup Q_B$, $F = \{n-1\}$, and for any $q \in Q$ and any $X \in \Sigma$,

$$\delta(q, X) = \begin{cases} \{\delta_A(q, X)\} & \text{if } q \in Q_A - \{q_{m-1}\}, \\ \{\delta_A(q, X), 1\} & \text{if } q = q_{m-1}, \\ \{\delta_B(q, X)\} & \text{if } q \in Q_B, \end{cases}$$

see Fig. 3. Clearly, the NFA C accepts the language $L(A)L(B)$.

Let $C' = (Q', \Sigma, \delta', \{q_0\}, F')$ be the DFA obtained from the NFA C by the subset construction. Let $\mathcal{R}$ be the following system of sets:

$$\mathcal{R} = \{\ \{q_i\} \cup S \mid 0 \leqslant i \leqslant m-2 \text{ and } S \subseteq Q_B\} \cup \{\ \{q_{m-1}\} \cup S \mid S \subseteq Q_B - \{0\}\ \},$$

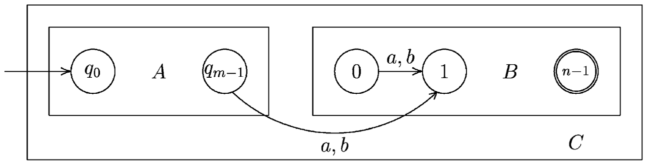

Fig. 3. The nondeterministic finite automaton C.

i.e., any set in $\mathcal{R}$ consists of exactly one state of Q_A and some states of Q_B, and if a set in $\mathcal{R}$ contains state q_{m-1}, then it does not contain state 0. There are $m2^n - 2^{n-1}$ sets in $\mathcal{R}$. To prove the theorem it is sufficient to show that (I) any set in $\mathcal{R}$ is a reachable state of the DFA C' and (II) no two different states in $\mathcal{R}$ are equivalent.

We prove (I) by induction on the size of sets. The singletons $\{q_0\}, \{q_1\}, \ldots, \{q_{m-1}\}$ are reachable since $\{q_i\} = \delta'(\{q_0\}, a^i)$ for $i = 1, 2, \ldots, m-1$. Let $1 \leqslant k \leqslant n$ and assume that any set in $\mathcal{R}$ of size k is a reachable state of the DFA C'. Using this assumption we prove that any set $\{q_i, j_1, j_2, \ldots, j_k\}$, where $0 \leqslant j_1 < j_2 < \cdots < j_k \leqslant n-1$ if $0 \leqslant i \leqslant m-2$, and $1 \leqslant j_1 < j_2 < \cdots < j_k \leqslant n-1$ if $i = m-1$, is a reachable state of the DFA C'. There are three cases:

(i) $i \leqslant m-2$ and $j_k = n-1$. We prove this case by induction on i.
For $i = 0$ we have

$$\{q_0, j_1, j_2, \ldots, j_{k-1}, n-1\} = \delta'(\{q_{m-1}, j_1+1, j_2+1, \ldots, j_{k-1}+1\}, b^{n-1}),$$

where the latter set of size k is reachable by induction on k since $j_1 + 1 \geqslant 1$.
Let $0 \leqslant i \leqslant m-3$ and assume that any set $\{q_i, j_1, j_2, \ldots, j_{k-1}, n-1\}$ is reachable. Since for $j_1 \geqslant 1$ we have

$$\{q_{i+1}, j_1, j_2, \ldots, j_{k-1}, n-1\} = \delta'(\{q_i, j_1-1, j_2-1, \ldots, j_{k-1}-1, n-1\}, a),$$

and for $j_1 = 0$ we have

$$\{q_{i+1}, j_1, j_2, \ldots, j_{k-1}, n-1\} = \delta'(\{q_i, j_2-1, \ldots, j_{k-1}-1, n-2, n-1\}, b),$$

we are ready in this case.

(ii) $i \leqslant m-2$ and $j_k < n-1$. Let $t = (i - j_1 - 1) \bmod (m-1)$. Then we have

$$\{q_i, j_1, j_2, \ldots, j_k\} = \delta'(\{q_t, j_2 - j_1 - 1, \ldots, j_k - j_1 - 1, n-1\}, b^{j_1+1}),$$

where the latter set is considered in case (i).

(iii) $i = m-1$ and $j_1 \geqslant 1$. Then we have

$$\{q_{m-1}, j_1, j_2, \ldots, j_k\} = \delta'(\{q_{m-2}, j_1-1, j_2-1, \ldots, j_k-1\}, a),$$

where the latter set is considered in case (ii).

To prove (II) let $\{q_i\} \cup S$ and $\{q_l\} \cup T$ be two different states in $\mathcal{R}$ with $0 \leqslant i \leqslant l \leqslant m-1$. There are two cases:

(i) $i < l$. Then the string $a^{m-1-i}b^{n-1}$ is accepted by the DFA C' starting in state $\{q_i\} \cup S$ but it is not accepted starting in state $\{q_l\} \cup T$.

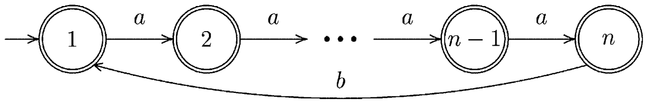

Fig. 4. The nondeterministic finite automaton D.

(ii) $i = l$. Without loss of generality, there is a state j in Q_B such that $j \in S$ and $j \notin T$ (note that $j \geqslant 1$ if $i = l = m - 1$). Then the string b^{n-1-j} is accepted by the DFA C' starting in state $\{q_i\} \cup S$ but it is not accepted starting in state $\{q_l\} \cup T$.

Thus our proof is complete. □

The concatenation of two languages represented by nondeterministic finite automata was investigated by Holzer and Kutrib [13]. They showed that $m + n$ states are sufficient for an NFA to accept the concatenation of an m-state NFA language and an n-state NFA language, and they proved that the upper bound is tight for a binary alphabet.

4. Reversal

In this section, we deal with the reversal operation. It is known that the reversal of any n-state DFA language can be accepted by a DFA of 2^n states and the worst case can be reached by the reversal of a binary DFA language [18]. The upper bound on the size of an NFA accepting the reversal of an n-state NFA language is known to be $n + 1$ and Holzer and Kutrib [13] proved that the upper bound is tight for a three-letter alphabet. The next theorem shows that the upper bound $n + 1$ can be reached by the reversal of a binary n-state NFA language. To obtain the result we use a counting argument. Since the reversal of any 1-state NFA language is the same language, we assume that $n \geqslant 2$.

Theorem 2. *For any integer $n \geqslant 2$, there exists a binary NFA D of n states such that any NFA accepting the reversal of the language $L(D)$ needs at least $n + 1$ states.*

Proof. Let n be arbitrary but fixed integer such that $n \geqslant 2$. Let $\Sigma = \{a, b\}$.

Define an n-state NFA $D = (Q_D, \Sigma, \delta_D, 1, F_D)$, where $Q_D = \{1, 2, \ldots, n\}$, $F_D = Q_D$, and for any $i \in Q_D$ and any $X \in \Sigma$,

$$\delta_D(i, X) = \begin{cases} \{i+1\} & \text{if } i < n \text{ and } X = a, \\ \{1\} & \text{if } i = n \text{ and } X = b, \\ \emptyset & \text{otherwise.} \end{cases}$$

The NFA D is shown in Fig. 4. We prove that any NFA accepting the reversal of the language $L(D)$ needs at least $n + 1$ states.

Let $N = (Q, \Sigma, \delta, q_0, F)$ be any NFA accepting the reversal of the language $L(D)$. Since the NFA N accepts the empty string, the initial state q_0 must be an accepting state.

Next, the NFA N accepts the string ba^{n-1}. Therefore, a sequence of states $q_1, q_2, \dots, q_n$ must exist in Q such that

$$q_1 \in \delta(q_0, b), \quad q_i \in \delta(q_{i-1}, a) \text{ for } i = 2, 3, \dots, n, \text{ and } q_n \in F.$$

Next, the NFA N does not accept any string ba^i, where $0 \leqslant i \leqslant n-2$. It follows that the states $q_1, q_2, \dots, q_{n-1}$ must be rejecting and pairwise distinct states in Q. Since the NFA N accepts the string a but does not accept the string ba^n, the initial state q_0 must be different from the accepting state q_n. Hence the NFA N has at least two accepting and at least $n-1$ rejecting states which proves the theorem. □

5. Complementation

We now turn our attention to the complementation operation. In contrast to the previous two operations, complementation is an efficient operation for DFAs since to accept the complement we can simply exchange accepting and rejecting states. On the other hand, complementation of NFAs is an expensive task. The upper bound on the size of an NFA accepting the complement of an n-state NFA language is 2^n and it is known to be tight. Sakoda and Sipser [25] gave an example of languages over a growing alphabet size reaching the upper bound. Birget claimed the result for a three-letter alphabet [3] but later corrected this to a four-letter alphabet [4]. Ellul [8] gave binary $O(n)$-state witness languages. Holzer and Kutrib [13] proved the lower bound 2^{n-2} for a binary n-state NFA language.

In this section, we show that the upper bound 2^n on the complementation of NFA languages is tight likewise for a binary alphabet. We describe a binary n-state NFA language such that any NFA accepting its complement needs at least 2^n states. To obtain a lower bound on the size of the NFA we use a fooling-set lower-bound technique known from communication complexity theory [14]. Although lower bounds based on fooling sets may sometimes be exponentially smaller than the true bounds [16,17], for some regular languages the lower bounds are tight [2,3,9]. In this section, the technique helps us to obtain tight lower bounds.

After defining a fooling set, we give the lemma from [2] describing a fooling-set lower-bound technique. For the sake of completeness, we repeat its proof in the Appendix.

Definition 3. A set of pairs of strings $\{(x_i, y_i) \mid i = 1, 2, \dots, n\}$ is said to be a fooling-set for a regular language L if for any i and j in $\{1, 2, \dots, n\}$, (1) $x_i y_i \in L$, and (2) if $i \neq j$ then $x_i y_j \notin L$ or $x_j y_i \notin L$.

Lemma 4 (*Birget [2]*). *Let a set of pairs $\{(x_i, y_i) \mid i = 1, 2, \dots, n\}$ be a fooling set for a regular language L. Then any NFA accepting the language L needs at least n states.*

We can now prove the following result.

Theorem 5. *For any positive integer n, there exists a binary NFA M of n states such that any NFA accepting the complement of the language $L(M)$ needs at least 2^n states.*

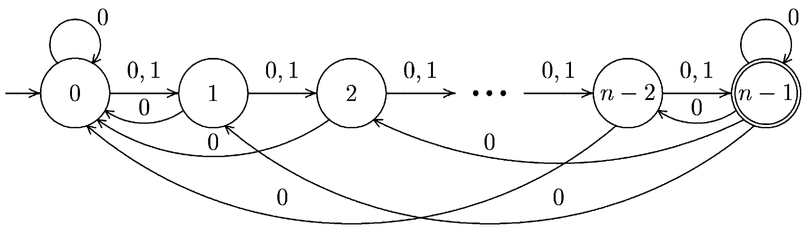

Fig. 5. The nondeterministic finite automaton M.

Proof. Let n be an arbitrary but fixed positive integer. Let $\Sigma = \{0, 1\}$.

Define an n-state NFA $M = (Q, \Sigma, \delta, 0, F)$, where $Q = \{0, 1, \ldots, n-1\}$, $F = \{n-1\}$, and for any $i \in Q$ and $X \in \Sigma$,

$$\delta\{i, X\} = \begin{cases} \{i+1\} & \text{if } i < n-1 \text{ and } X = 1, \\ \emptyset & \text{if } i = n-1 \text{ and } X = 1, \\ \{0, i+1\} & \text{if } i < n-1 \text{ and } X = 0, \\ \{1, 2, \ldots, n-1\} & \text{if } i = n-1 \text{ and } X = 0, \end{cases}$$

i.e., for any state $i \in Q$ except $n-1$, the NFA M goes from i to $i+1$ on input 1, and to $i+1$ or to the initial state on input 0, and M goes from state $n-1$ to any state except the initial state on input 0. The NFA M is shown in Fig. 5.

Let L be the language accepted by the NFA M. Denote by S, the set $\Sigma^{\leqslant n-1} \cup \{1^n\}$ containing the string 1^n and all strings over the alphabet Σ of length at most $n-1$. Our goal is to construct a fooling set for the language L^c of size 2^n. We will do this in the following way. For any string x in S, we define a string y_x such that $xy_x \in L^c$ and, moreover, for any two different strings x and z in S, either $xy_z \notin L^c$ or $zy_x \notin L^c$.

Let x be a string in S. Then the set $\delta(0, x)$ contains all states of the NFA M that are reachable from the initial state 0 after reading the string x.

If $\delta(0, x) = Q$, let $y_x = 1^n$.

If $\delta(0, x) = \{0, 1, \ldots, n-2\}$, let $y_x = \varepsilon$.

Otherwise define the string y_x as follows. Let $y_x = y_1 y_2 \cdots y_{n-1-k}$, where k is *the least* number in Q such that $k \notin \delta(0, x)$, and for any $j \in \{1, 2, \ldots, n-1-k\}$,

$$y_j = \begin{cases} 1, & \text{if } n-j \in \delta(0, x), \\ 0, & \text{if } n-j \notin \delta(0, x), \end{cases}$$

i.e., the jth symbol of y_x equals 1 if state $n-j$ in $\{k+1, k+2, \ldots, n-1\}$ belongs to the set $\delta(0, x)$, and equals 0 otherwise.

The following claims are proved in the Appendix.

Claim 6. *For any string x in S and any state p in Q,*
(a) *if $p \in \delta(0, x)$ then $n-1 \notin \delta(p, y_x)$,*
(b) *if $p \notin \delta(0, x)$ then $n-1 \in \delta(p, y_x)$.*

Claim 7. *For any two different strings x and z in S, the set $\delta(0, x)$ is different from the set $\delta(0, z)$.*

We are now going to prove that the set of pairs $\{(x, y_x) \mid x \in S\}$ is a fooling set for the language L^c. We need to show that for any x and z in S, (1) $xy_x \in L^c$, and (2) if $x \neq z$, then $xy_z \notin L^c$ or $zy_x \notin L^c$.

To prove (1) let x be any string in S. By Claim 6(a), state $n-1$ cannot be reached from any state in the set $\delta(0, x)$ after reading the string y_x. Since state $n-1$ is the only accepting state of the NFA M, it follows that the string xy_x is not accepted by M. Hence $xy_x \in L^c$.

To prove (2) let x and z be two different strings in S. By Claim 7, without loss of generality, there is a state q in Q such that $q \in \delta(0, x)$ and $q \notin \delta(0, z)$. By Claim 6(b), state $n-1$ can be reached from the state q after reading the string y_z. It follows that the string xy_z is accepted by the NFA M. Hence $xy_z \notin L^c$.

We have shown that the set of pairs $\{(x, y_x) \mid x \in S\}$ is a fooling set for the language L^c. By Lemma 4, any NFA for the language L^c needs at least 2^n states. □

6. Conclusions

In this paper, we have obtained several results concerning the state complexity of some operations on binary regular languages. We proved that some upper bounds which were known to be tight for larger alphabets are tight likewise for binary alphabets. We presented an m-state DFA A, an n-state DFA B, an n-state NFA D, and an n-state NFA M, all with a binary input alphabet, such that:

- any DFA accepting $L(A)L(B)$ needs at least $m2^n - 2^{n-1}$ states;
- any NFA accepting the reversal of $L(D)$ needs at least $n+1$ states;
- any NFA accepting the complement of $L(M)$ needs at least 2^n states.

Note however, that the upper bound on the concatenation of DFA languages is, in fact, $m2^n - k2^{n-1}$, where k is the number of accepting states in the m-state DFA [27, Theorem 2]. It would be interesting to find languages reaching this bound for any k with $1 < k < m$.

Acknowledgements

I would like to thank Jožko Jirásek for his help with the computational verification of some conjectures. I am also very grateful to an anonymous referee for his corrections and suggestions.

Appendix

Proof of Lemma 4. Let $M = (Q, \Sigma, \delta, q_0, F)$ be any NFA accepting the language L. Since $x_i y_i \in L$, there must be a state p_i in Q such that

$$p_i \in \delta(q_0, x_i) \quad \text{and} \quad \delta(p_i, y_i) \cap F \neq \emptyset.$$

Assume that a fixed choice of p_i has been made for any i in $\{1, 2, \dots, n\}$. We prove that $p_i \neq p_j$ for $i \neq j$. Suppose by contradiction that $p_i = p_j$ for some $i \neq j$. Then the NFA M accepts both strings $x_i y_j$ and $x_j y_i$ which contradicts the assumption that the set $\{(x_i, y_i) \mid 1 \leqslant i \leqslant n\}$ is a fooling set for the language L. Hence the NFA M has at least n states. □

Proof of Claim 6. Let x be any string in $\{0, 1\}^{\leqslant n-1} \cup \{1^n\}$.

If $\delta(0, x) = Q$, then $y_x = 1^n$. Since $\delta(q, 1^n) = \emptyset$ for any $q \in Q$, the claim holds in this case.

If $\delta(0, x) = \{0, 1, \dots, n-2\}$, then $y_x = \varepsilon$. Since $\delta(q, \varepsilon) = \{q\}$ for any $q \in Q$, the claim holds in this case as well.

Otherwise $y_x = y_1 y_2 \cdots y_{n-1-k}$ where $k \notin \delta(0, x)$, $\{0, 1, \dots, k-1\} \subseteq \delta(0, x)$, and for any $j \in \{1, 2, \dots, n-1-k\}$, $y_j = 1$ if $n-j \in \delta(0, x)$ and $y_j = 0$ if $n - j \notin \delta(0, x)$.

To prove (a) let p be any state in Q such that $p \in \delta(0, x)$. There are two cases:

(i) $p \leqslant k-1$. Then state $n-1$ cannot be reached from state p after reading the string y_x of length less than $n-k$. Hence $n-1 \notin \delta(p, y_x)$.

(ii) $p \geqslant k+1$. Then $y_{n-p} = 1$, and so $y_x = u1v$ for some strings u and v such that

$$|u| = n - p - 1 \quad \text{and} \quad |v| = p - 1 - k.$$

It follows that the set $\delta(p, u)$ is a subset of $\{0, 1, \dots, n-p-2\} \cup \{n-1\}$. Therefore $\delta(p, u1) \subseteq \{1, 2, \dots, n-p-1\}$, and so state $n-1$ cannot be reached from the set $\delta(p, u1)$ after reading the string v of length less than p. Thus $n-1 \notin \delta(0, u1v)$.

To prove (b) let p be any state in Q such that $p \notin \delta(0, x)$. There are two cases:

(i) $p = k$. Then state $n-1$ can be reached from state k after reading the string y_x of length $n-1-k$, so $n-1 \in \delta(p, y_x)$.

(ii) $p \geqslant k+1$. Then $y_{n-p} = 0$, and so $y_x = u0v$ for some strings u and v such that

$$|u| = n - p - 1 \quad \text{and} \quad |v| = p - 1 - k.$$

It follows that $n-1 \in \delta(p, u)$ and $n-1 \in \delta(n+k-p, v)$. Since $n+k-p \geqslant 1$, state $n+k-p$ belongs to the set $\delta(n-1, 0)$. Hence $n-1 \in \delta(p, u0v)$ which completes the proof. □

Proof of Claim 7. Let x and z be two different strings in $\{0, 1\}^{\leqslant n-1} \cup \{1^n\}$. Let $l_x = |x|$ and $l_z = |z|$. Without loss of generality, assume that $l_x \leqslant l_z$. There are three cases:

(i) $0 \leqslant l_x \leqslant n-1$ and $z = 1^n$. Then $\delta(0, z) = \emptyset$ and $l_x \in \delta(0, x)$.

(ii) $0 \leqslant l_x < l_z \leqslant n-1$. Then $l_z \in \delta(0, z)$ and $\delta(0, x) \subseteq \{0, 1, \dots, l_x\}$.

(iii) $0 \leqslant l_x = l_z \leqslant n-1$. Then, without loss of generality, $x = u0v$ and $z = u1w$ for some strings u, v, w such that $0 \leqslant |v| = |w| \leqslant n-2$. Set $l = |v|$. We show that $l \in \delta(0, u0v)$ and $l \notin \delta(0, u1w)$. Since the string u has length at most $n-2-l$,

$$\delta(0, u) \subseteq \{0, 1, \dots, n-2-l\}.$$

Therefore $0 \in \delta(0, u0)$, and so state l can be reached from the set $\delta(0, u0)$ after reading the string v of length l. Thus $l \in \delta(0, u0v)$.

Furthermore, $\delta(0, u1) \subseteq \{1, 2, \ldots, n-1-l\}$. Since the string w has length l, we have $\delta(0, u1w) \subseteq \{0, 1, \ldots, l-1\} \cup \{l+1, l+2, \ldots, n-1\}$. Hence $l \notin \delta(0, u1w)$ and the proof is complete. $\square$

References

[1] P. Berman, A. Lingas, On the complexity of regular languages in terms of finite automata, Techn. Report 304, Polish Academy of Sciences, 1977.

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

[3] J.C. Birget, Partial orders on words, minimal elements of regular languages, and state complexity, Theoret. Comput. Sci. 119 (1993) 267–291.

[4] J.C. Birget, ERRATUM: partial orders on words, minimal elements of regular languages, and state complexity, 2002. Available at: `http://clam.rutgers.edu/~birget/papers.html`.

[5] C. Câmpeanu, K. Culik II, K. Salomaa, S. Yu, State complexity of basic operations on finite languages, in: O. Boldt, H. Jürgensen (Eds.), Proc. Fourth Internat. Workshop on Implementing Automata (WIA'99), Lecture Notes in Computer Science, Vol. 2214, Springer, Heidelberg, 2001, pp. 60–70.

[6] C. Câmpeanu, K. Salomaa, S. Yu, Tight lower bound for the state complexity of shuffle of regular languages, J. Automat. Lang. Comb. 7 (2002) 303–310.

[7] M. Domaratzki, State complexity and proportional removals, J. Automat. Lang. Comb. 7 (2002) 455–468.

[8] K. Ellul, Descriptional complexity measures of regular languages, Master's thesis, University of Waterloo, 2002.

[9] I. Glaister, J. Shallit, A lower bound technique for the size of nondeterministic finite automata, Inform. Process. Lett. 59 (1996) 75–77.

[10] M. Holzer, M. Kutrib, Nondeterministic descriptional complexity of regular languages, Internat. J. Found. Comput. Sci. (to appear). Preprint: IFIG Research Report 0205, University of Giessen, 2002. Available at: `http://www.informatik.uni-giessen.de/reports/Report0205.ps.gz`.

[11] M. Holzer, M. Kutrib, Unary language operations and their nondeterministic state complexity, in: M. Ito, M. Toyama (Eds.) Developments in Language Theory (DLT 2002), Lecture Notes in Computer Science, Vol. 2450, Springer, Heidelberg, 2003, pp. 162–172.

[12] M. Holzer, M. Kutrib, State complexity of basic operations on nondeterministic finite automata, in: J.M. Champarnaud, D. Maurel (Eds.), Implementation and Application of Automata (CIAA 2002), Lecture Notes in Computer Science, Vol. 2608, Springer, Heidelberg, 2003, pp. 148–157.

[13] M. Holzer, K. Salomaa, S. Yu, On the state complexity of k-entry deterministic finite automata, J. Automat. Lang. Comb. 6 (2001) 453–466.

[14] J. Hromkovič, Communication Complexity and Parallel Computing, Springer, Berlin, Heidelberg, 1997.

[15] J. Hromkovič, Descriptional complexity of finite automata: concepts and open problems, J. Automat. Lang. Comb. 7 (2002) 519–531.

[16] J. Hromkovič, S. Seibert, J. Karhumäki, H. Klauck, G. Schnitger, Communication complexity method for measuring nondeterminism in finite automata, Inform. and Comput. 172 (2002) 202–217.

[17] G. Jirásková, Note on minimal automata and uniform communication protocols, in: C. Martin-Vide, V. Mitrana (Eds.), Grammars and Automata for String Processing: From Mathematics and Computer Science to Biology, and Back, Taylor and Francis, London, 2003, pp. 163–170.

[18] E. Leiss, Succinct representation of regular languages by boolean automata, Theoret. Comput. Sci. 13 (1981) 323–330.

[19] O.B. Lupanov, A comparison of two types of finite automata, Problemy Kibernetiki (9) (1963) 321–326 (in Russian).

[20] A.R. Meyer, M.J. Fischer, Economy of description by automata, grammars and formal systems, in: Proc. 12th Annual Symposium on Switching and Automata Theory, 1971, pp. 188–191.

[21] F.R. Moore, On the bounds for state-set size in the proofs of equivalence between deterministic, nondeterministic, and two-way finite automata, IEEE Trans. Comput. 20 (1971) 1211–1214.

[22] G. Pighizzini, Unary language concatenation and its state complexity, in: S. Yu, A. Pun (Eds.), Implementation and Application of Automata: Fifth Internat. Conference, CIAA 2000, Lecture Notes in Computer Science, Vol. 2088, Springer, Heidelberg, 2001, pp. 252–262.
[23] G. Pighizzini, J. Shallit, Unary language operations, state complexity and Jacobsthal's function, Internat. J. Found. Comput. Sci. 13 (2002) 145–159.
[24] M. Rabin, D. Scott, Finite automata and their decision problems, IBM Res. Develop. 3 (1959) 114–129.
[25] W.J. Sakoda, M. Sipser, Nondeterminism and the size of two-way finite automata, in: Proc. 10th Ann. ACM Symp. on Theory of Computing, 1978, pp. 275–286.
[26] M. Sipser, Introduction to the Theory of Computation, PWS Publishing Company, Boston, 1997.
[27] S. Yu, Regular languages, in: G. Rozenberg, A. Salomaa (Eds.), Handbook of Formal Languages, Vol. 1, Springer, Berlin, New York, pp. 41–110 (Chapter 2).
[28] S. Yu, Q. Zhuang, K. Salomaa, The state complexity of some basic operations on regular languages, Theoret. Comput. Sci. 125 (1994) 315–328.
[29] S. Yu, State complexity of regular languages, J. Automat. Lang. Comb. 6 (2001) 221–234.
[30] S. Yu, A renaissance of automata theory, Bull. Eur. Assoc. Theoret. Comput. Sci. EATCS 72 (2000) 270–272.
[31] S. Yu, State complexity of finite and infinite regular languages, Bull. Eur. Assoc. Theoret. Comput. Sci. EATCS 76 (2000) 270–272.