The State Complexity of Permutations on Finite Languages over Binary Alphabets

Alexandros Palioudakis[1], Da-Jung Cho[1], Daniel Goč[2], Yo-Sub Han[1], Sang-Ki Ko[1], and Kai Salomaa[2(✉)]

[1] Department of Computer Science, Yonsei University, 50 Yonsei-Ro, Seodaemum-Gu, Seoul 120–749, Republic of Korea
{alex,dajung,emmous,narame7}@cs.yonsei.ac.kr
[2] School of Computing, Queen's University, Kingston, Ontario K7L 3N6, Canada
{goc,ksalomaa}@cs.queensu.ca

Abstract. We investigate the state complexity of the permutation operation over finite binary languages. We first give an upper bound of the state complexity of the permutation operation for a restricted case of these languages. We later present a general upper bound of the state complexity of permutation over finite binary languages, which is asymptotically the same as the previous case. Moreover, we show that there is a family of languages that the minimal DFA recognizing each of these languages needs at least as many states as the given upper bound for the restricted case. Furthermore, we investigate the state complexity of permutation by focusing on the structure of the minimal DFA.

Keywords: Finite automata · State complexity · Finite languages · Permutation · Parikh equivalence

1 Introduction

Finite automata are well studied in the theory of computation. McCulloch and Pitts [14] first introduced this model of computation. Following on these ideas Kleene [10] wrote the first paper on finite automata and regular expressions. Later, Rabin and Scott [18] first studied the nondeterministic version of finite automata, for which they received the Turing Award, the highest award in computer science.

Since then, much work has been done in the descriptional complexity of finite automata [12,13,15,16]. The descriptional complexity of finite automata is usually measured in the number of transitions or the number of states that a finite automaton requires in order to accept a given language. Most researchers have focused on the state complexity of finite automata [6,9].

A widely studied topic in the state complexity of finite automata is the state complexity of language operations. Yu et al. [20] studied the state complexity of some basic operations. Han and Salomaa [7] studied the state complexity of union and intersection of finite languages. Holzer and Kutrib [8]

J. Shallit and A. Okhotin (Eds.): DCFS 2015, LNCS 9118, pp. 220–230, 2015.
DOI: 10.1007/978-3-319-19225-3_19

studied the nondeterministic state complexity of some basic language operations. Câmpeanu et al. [1] studied the state complexity of basic language operation for finite languages. Domaratzki [2] studied the state complexity of proportional removals and recently, Goč et al. [5] studied the nondeterministic state complexity of proportional removals. For more information on the state complexity of language operations, the reader can consult the recent review by Gao et al. [4].

Here we investigate the operational state complexity of the permutation operation. The family of regular languages is not closed under permutation and, hence, in this paper we focus on finite languages. We first compute an upper bound of the state complexity of the permutation on a restricted case of regular languages over binary alphabets. We show an upper bound for the state complexity of permutation for general binary finite languages. We mention that the permutation operation is related to the Parikh mapping, which maps each string over n letters to an n-dimensional vector whose components give the number of occurrences of the letters in the string [11,17]. Ellul et al. [3] have given strong lower bounds for the size of NFAs or regular expressions recognizing permutations of symbols of a growing alphabet.

In Sect. 2, we briefly present definitions and notation used throughout the paper. In Sect. 3, we give the state complexity of permutation of binary languages recognized by DFAs that form a chain, and present a general upper bound of the state complexity of permutation of binary finite languages. In Sect. 4, we give an upper bound on the state complexity of permutation for languages that recognize strings with equal length. Moreover, we give lower bounds that are tight in the restricted cases and asymptotically tight in the general case.

2 Preliminaries

We assume that the reader is familiar with the basic definitions concerning finite automata [19,21] and descriptional complexity [6,9]. Here we just fix some notation needed in the following.

The set of strings over a finite alphabet Σ is Σ^*, the length of $w \in \Sigma^*$ is $|w|$ and ε is the empty string. Moreover, for a letter $a \in \Sigma$ and a string $w \in \Sigma^*$, we denote the numbers of occurrences of the letter a in the string w by $|w|_a$. The set of positive integers is denoted by $\mathbb{N}$. The cardinality of a finite set S is $\#S$.

A deterministic finite automaton (DFA) is a 5-tuple $A = (Q, \Sigma, \delta, q_0, F)$, where Q is a finite set of states, Σ is a finite alphabet, $\delta : Q \times \Sigma \to Q$ is the transition function (partial function), $q_0 \in Q$ is the start state and $F \subseteq Q$ is the set of accepting states. The function δ is extended in the usual way as a function $Q \times \Sigma^* \to Q$ and the language recognized by A consists of strings $w \in \Sigma^*$ such that $\delta(q_0, w) \in F$. By the *size of* A, we mean the number of states of A, $\text{size}(A) = \#Q$.

The minimal size of a DFA recognizing a regular language L is called the state complexity of L and denoted by $\text{sc}(L)$. Note that we allow DFAs to be incomplete and, consequently, the deterministic state complexity of L may differ by one from the definition using complete DFAs.

An important relation of languages is the Myhill-Nerode relation R_L of a language L. The relation R_L contains pairs of strings x and y if and only if for every $z \in \Sigma^*$ both strings $x \cdot z$ and $y \cdot z$ belong in L or both strings $x \cdot z$ and $y \cdot z$ do not belong in L. It is well known that the MyhillNerode relation of the language L has finite number of equivalence classes if and only if the language L is regular. Moreover, the unique minimal DFA for L has the same number of states as the number of equivalence classes of R_L. Hence, when we want to find a lower bound on the state complexity of a regular language L, it is sufficient to find a set of strings S such that, for every strings $w, w' \in S$, there is a string $u \in \Sigma^*$ with $w \cdot u \in L$ and $w' \cdot u \notin L$, or $w \cdot u \notin L$ and $w' \cdot u \in L$. Then, we have that $\mathrm{sc}(L) \geq \#S$.

We consider the state complexity of the operation of permutation on finite languages. We now define the permutation $\mathrm{per}(L)$ of a regular language L over the alphabet Σ as follows: A string w belongs in $\mathrm{per}(L)$ if and only if there is a string $u \in L$ such that the strings w and u have the same number of occurrences of every letter of Σ. Formally, we define

$$\mathrm{per}(L) = \{w \in \Sigma^* \mid (\exists u \in L)(\forall a \in \Sigma)(|u|_a = |w|_a)\}.$$

Remark that the family of regular languages is not closed under permutation. For example, for the language $(a \cdot b)^*$, the permutation of this language contains all the strings w such that $|w|_a = |w|_b$, which is not a regular language.

Given an alphabet $\Sigma = \{a_1, a_2, \ldots, a_k\}$, let $\Psi : \Sigma^* \to [\mathbb{N}_0]^k$ be a mapping defined by $\Psi(w) = (|w|_{a_1}, |w|_{a_2}, \ldots, |w|_{a_k})$. This function is called a *Parikh mapping* and $\Psi(w)$ is called the *Parikh vector* of w. The Parikh mapping is extended for a set of strings, with $\Psi : 2^{\Sigma^*} \to 2^{[\mathbb{N}_0]^k}$ be a mapping defined by $\Psi(L) = \{\Psi(w) \mid w \in L\}$. Two languages L_1, L_2 are Parikh equivalent, denoted by $L_1 \equiv_{\mathrm{Parikh}} L_2$, if $\Psi(L_1) = \Psi(L_2)$. Similarly, we say that two DFAs A, B are Parikh equivalent if $\Psi(L(A)) = \Psi(L(B))$ and denote it by $A \equiv_{\mathrm{Parikh}} B$.

3 Permutation Operation for Chain DFAs

We consider the problem of finding the state complexity of the permutation of a binary language L. We start with giving an upper bound for the restricted case of the language L where each string of L has length $\mathrm{sc}(L) - 1$.

Lemma 1. *Let n be a positive integer and $L \subseteq \{a, b\}^{n-1}$ be a finite language such that $\mathrm{sc}(L) = n$. Then, we have the following inequality for the state complexity of the permutation of L:*

$$\mathrm{sc}(\mathrm{per}(L)) \leq \frac{n^2 + n + 1}{3}.$$

Proof. Let A be the minimal DFA recognizing L over the alphabet $\{a, b\}$. Since we have that $L \subseteq \{a, b\}^{n-1}$, we know that each string w of L is of length $n - 1$. Hence, the DFA A forms a chain, that is, we can enumerate the states of A such that for all $1 \leq h \leq n - 1$ at least one of the following is true; $\delta(h, a) = h + 1$

or $\delta(h, b) = h + 1$. Additionally, each state h has only one target state $h + 1$ for $1 \leq h \leq n - 1$ and the transition function for h is one the the following three cases:

1. $\delta(h, a) = h + 1$ (a-transition)
2. $\delta(h, b) = h + 1$ (b-transition)
3. $\delta(h, a) = h + 1, \delta(h, b) = h + 1$ ($a\&b$-transition)

By the definition of the language $\mathrm{per}(L)$, we have all the possible permutations of the strings of L. The order of the different types of transitions (a, b, or $a\&b$) of A does not affect $\mathrm{per}(L)$. Hence, we can assume that, without loss of generality, we start with a-transitions, followed by b-transitions, followed by $a\&b$-transitions. From this assumption, we have that L is of the form $a^i b^j (a+b)^k$ for some non-negative integers i, j, k such that $i+j+k = n-1$. It is not difficult to construct a DFA with $(i+1)\cdot(j+1)+k\cdot j+k\cdot i+k$ states recognizing $\mathrm{per}(L)$. Since we search for an upper bound of the state complexity of $\mathrm{per}(L)$, we can find for which values of i, j, k the function $f(i, j, k) = (i+1)\cdot(j+1)+k\cdot j+k\cdot i+k$, and $i+j+k = n-1$, has a maximal value. It is easy to verify that f is maximized when i, j, k are $i = j = k = \frac{n-1}{3}$. Thus, for integer values of i, j, k

$$\max f(i, j, k) = \begin{cases} \frac{n^2+n+1}{3}, & \text{if} n \equiv 1 \pmod 3; \\ \frac{n^2+n}{3}, & \text{otherwise.} \end{cases}$$

□

In Lemma 1, we assume that every string w in L has a specific length—$|w| = \mathrm{sc}(L) - 1$. This restriction ensures that the states of the minimal DFA recognizing L form a chain. Now, we move one step further and show an upper bound of the state complexity of the permutation of L without such restrictions.

Lemma 2. *Let L be a binary finite language and $m = \max\{|w| \mid w \in L\}$ for some positive integer m. Then, we have*

$$\mathrm{sc}(\mathrm{per}(L)) \leq \frac{m^2 + m + 2}{2}.$$

Proof. We construct a DFA A that recognizes $\mathrm{per}(L)$ over the binary alphabet $\{a, b\}$. We keep track at each state of A what is the number of occurrences of a's and what is the number of occurrences of b's that we have already read. Then A has states of the form (i, j), for $0 \leq i, j \leq m$, where i (and j) keeps track of the occurrences of a's (and b's, respectively). Since m is the length of the longest string in L, we know that there is no computation path in A with more than $m + 1$ states. Thus, for all states (i, j) of A we have $i + j \leq m$. Moreover, it is immediate that all states (i, j) with $i + j = m$ are final and equivalent—we can merge them to one final state.

Now we counter the number of states. The total number of states of the resulting DFA is $1 + 2 + \cdots + m + 1$(the merged final state) $= \frac{m\cdot(m+1)}{2} + 1$. The final states of A are all states (i, j) such that there is a string $w \in L$ with $|w|_a = i$ and $|w|_b = j$. □

We notice that the maximum length of all the strings of the language L can be at most the state complexity of L minus one; in other words, we have $1+\max\{|w| \mid w \in L\} \leq \mathrm{sc}(L)$. By this observation and Lemma 2, we have the following corollary.

Corollary 1. *Let L be a binary finite language and $\mathrm{sc}(L) = n$ for some positive integer n. Then we have*

$$\mathrm{sc}(\mathrm{per}(L)) \leq \frac{n^2 - n + 2}{2}.$$

In the following theorem, we give a lower bound on the state complexity of the permutation of a language. This bound is asymptotically tight with the general upper bound in Corollary 1.

Theorem 1. *For any $n_0 \in \mathbb{N}$, there exists a regular language L with $\mathrm{sc}(L) = n$, for $n \geq n_0$, such that*

$$\mathrm{sc}(\mathrm{per}(L)) \geq \frac{n^2 + n + 1}{3}.$$

Proof. Let $n = 3k + 1 \geq n_0$, $k \in \mathbb{N}$ and $L_n = L(a^k b^k (a + b)^k)$. In Fig. 1, for $k = 3$ and $n = 10$, we see that L_n can be accepted by an incomplete DFA with n states.

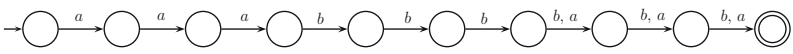

Fig. 1. The state minimal DFA recognizing the language L_{10}.

We prove a lower bound for the state complexity of $\mathrm{per}(L_n)$.

$$\mathrm{per}(L_n) = \{w \in \Sigma^{3 \cdot k} \mid |w|_a, |w|_b \geq k, n = 3 \cdot k + 1\}.$$

Let X and Y be the sets of strings as follows:

$$X = \{a^i b^j : 0 \leq i \leq 2k, 0 \leq j \leq k\} \text{ and } Y = \{a^i b^j : 0 \leq i < k, k < j \leq 2k\}.$$

We show that all strings of $X \cup Y$ are pairwise inequivalent with respect to the Myhill-Nerode congruence of $\mathrm{per}(L_n)$. Let $u = a^i b^j$ and $u' = a^{i'} b^{j'}$ be two arbitrary distinct strings from $X \cup Y$. We consider first the case where $|u| \neq |u'|$ and later we consider three separate cases $u, u' \in X$, $u, u' \in Y$, and, $u \in X$ and $u' \in Y$ (same case as $u' \in X$ and $u \in Y$):

1. We have that $|u| \neq |u'|$. It is straightforward to verify that u and u' are inequivalent since one can easily find a string z such that $uz \in \mathrm{per}(L_n)$ and $|u'z| \neq 3 \cdot k$—$u'z \notin \mathrm{per}(L_n)$.
2. We have $|u| = |u'|$ and $u, u' \in X$. Since $u \neq u'$, either $|u|_a < |u'|_a$ or $|u'|_a < |u|_a$. Without loss of generality, we assume that $|u|_a < |u'|_a$. For $z = a^{2 \cdot k - i} b^{k - j}$, we have $uz \in \mathrm{per}(L_n)$. However, for the string $u'z$, we have $|u'z|_a > 2 \cdot k$, which means, since $|uz| = |u'z| = 3 \cdot k$, that $|u'z|_b < k$ and $u'z \notin \mathrm{per}(L_n)$.

3. We have $|u| = |u'|$ and $u, u' \in Y$. Similar with the second case above, we assume that, without loss of generality, $|u|_b < |u'|_b$. For $z = a^{k-i}b^{2\cdot k-j}$, we have $uz \in \mathrm{per}(L_n)$. However, we have $|u'z|_b > 2 \cdot k$, which implies that $u'z \notin \mathrm{per}(L_n)$.
4. We have that $u \in X$ and $u' \in Y$ and $|u| = |u'|$. Since $u' \in Y$ and $u \in X$, we know that $|u'|_b > k$ and $|u|_b \leq k$. This implies that $|u|_a > |u'|_a$ because $|u| = |u'|$. Now for the string $z = a^{k-i}b^{2\cdot k-j}$, we have $uz \in \mathrm{per}(L_n)$. However, for the string $u'z$, we have that $|u'z|_a < k$ and, thus, $u'z \notin \mathrm{per}(L_n)$.

An example of the minimal DFA recognizing the language $\mathrm{per}(L_{10})$ is presented in Fig. 2.

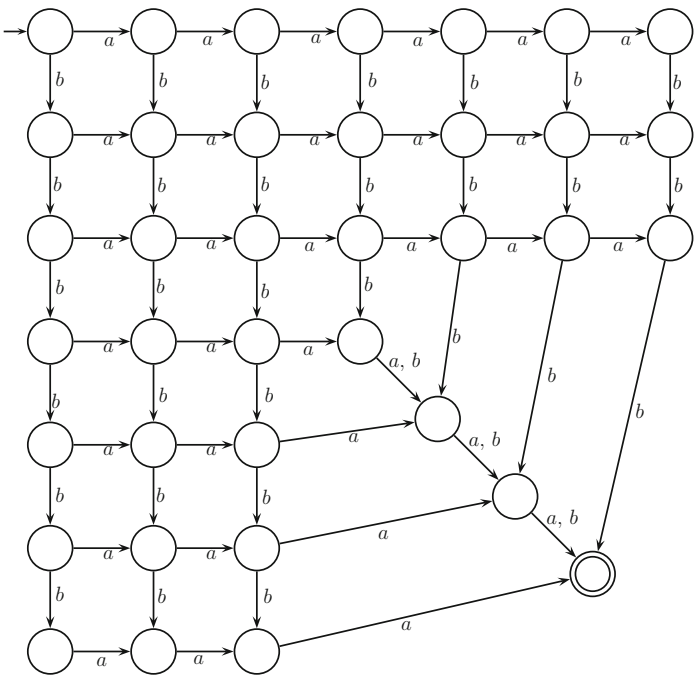

Fig. 2. The minimal DFA recognizing the language $\mathrm{per}(L_{10})$).

Hence, the number of states of the minimal DFA recognizing the language $\mathrm{per}(L_n)$ has at least $(2 \cdot k + 1) \cdot (k + 1) + k^2 = 3 \cdot k^2 + 3 \cdot k + 1$ states. We know that $n = 3 \cdot k + 1$ (namely, $k = \frac{n-1}{3}$) and, thus, the minimal DFA for $\mathrm{per}(L_n)$ has at least

$$3 \cdot \left(\frac{n-1}{3}\right)^2 + 3 \cdot \left(\frac{n-1}{3}\right) + 1 = \frac{n^2 - 2 \cdot n + 1}{3} + (n-1) + 1 = \frac{n^2 + n + 1}{3}$$

states. □

From the simple case studied in Lemma 1 and Theorem 1, we have the following corollary.

Corollary 2. *Let n be a positive integer and $L \subseteq \{a,b\}^{n-1}$ be a finite language such that* $\mathrm{sc}(L) = n$. *Then the state complexity of the permutation of L is bounded by the inequality,*

$$\mathrm{sc}(\mathrm{per}(L)) \leq \frac{n^2+n+1}{3}.$$

Moreover, sometimes $\frac{n^2+n+1}{3}$ states are necessary for the minimal DFA recognizing $\mathrm{per}(L)$.

4 Upper Bound for Sets of Equal Length Strings

We prove an upper bound for the state complexity of permutation of sets of equal length strings. The upper bound coincides with the lower bound from Theorem 1, which also uses sets of equal length strings.

We begin by introducing some terminology for DFAs that recognize sets of equal length strings. In the following, we consider a DFA $A = (Q, \Sigma, \delta, q_0, \{q_f\})$ recognizing a subset of Σ^ℓ, $\Sigma = \{a,b\}$. Without loss of generality A has one final state and has no useless states. The number of states of A is n.

The *level of a state* $q \in Q$ is the length of a string w such that $\delta(q_0, w) = q$. The level of a state is a unique integer in $\{0, 1, \ldots, \ell\}$. The set of level z states is $Q[z]$ for $0 \leq z \leq \ell$. We say that level z is *singular* if $|Q[z]| = 1$, $0 \leq z \leq \ell$. Levels 0 and ℓ are always singular. A *linear transition* is a transition between two singular levels. A linear transition can be labeled by a, b or $a\&b$. (A linear transition labeled by $a\&b$ is strictly speaking two transitions.) The number of linear transitions labeled by a (respectively, by b, $a\&b$) is denoted i_A (respectively, j_A, k_A).

The length of the nonlinear part of A is

$$h_A = \ell - (i_A + j_A + k_A). \tag{1}$$

Thus h_A denotes the number of pairs $(z, z+1)$, for $0 \leq z < \ell$, such that at least one of the levels z or $z+1$ is not singular.

Consider $0 \leq x \leq \ell$, $0 \leq y \leq \ell$, and $x+1 < y$, where levels x and y are singular and all levels strictly between x and y are non-singular. A *nonlinear block* $B_{x,y}$ of A between the levels x and y is a subautomaton of A consisting of all states of $\cup_{x \leq z \leq y} Q[z]$ and the transitions between them. The initial (respectively, final) state of the subautomaton is the state having level x (respectively, y). The length of the nonlinear block $B_{x,y}$ is $y - x$. The length of a block is always at least two.

Note that a nonlinear block begins and ends in a singular level and all levels between these are non-singular. In the following, nonlinear blocks are called simply *blocks.* Examples of blocks are illustrated in Fig. 3.

The sum of the lengths of the blocks of A equals to h_A. The estimation of the length of accepted strings ℓ in terms of the number of states n depends on the types of blocks that A has.

Assume that the total length h_A of the blocks of A is fixed. Then the maximal value of ℓ can be reached if all blocks have length two (and h_A is even). Note that a block of length two has always exactly 4 states. Thus, we have

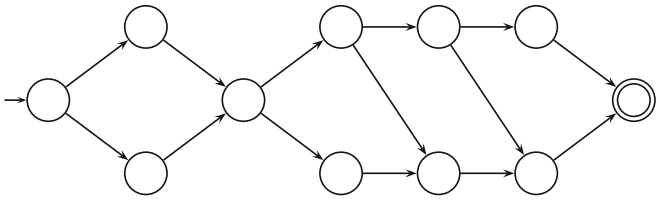

Fig. 3. A DFA with a block of length 2 and a block of length 4.

$$\ell \leq n - 1 - \frac{1}{2}h_A. \tag{2}$$

Example of the worst-case situation where $\ell = n-1-\frac{1}{2}h_A$ is illustrated in Fig. 4.

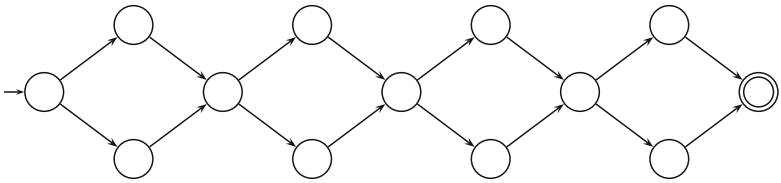

Fig. 4. $n = 13$ and $h_A = 8$, $\ell = 8$.

4.1 Estimate for DFAs Having Blocks of Length Two

We begin by providing an upper bound in the case where a DFA A includes blocks of length two and none of bigger length. As observed in the following subsection the same upper bound holds for arbitrary DFAs recognizing sets of equal length strings. The proof of the general case is based on similar ideas but is considerably more complicated. In this extended abstract we include the proof only for the case where the DFA has blocks of length at most two.

A block of length two that recognizes the language $\{aa, bb\}$ is called a *diamond* (see Fig. 5). There are a total of 9 different blocks of length two and it is easy to see that any block of length two that is not a diamond is "redundant" in the sense that it can be replaced by linear transitions and the modified DFA is Parikh-equivalent to A. This is stated in the following lemma.

Lemma 3. *Assume that A has a block of length two that is not a diamond. Then there exists a DFA A_1 having $n-1$ states such that $L(A_1) \equiv_{\text{Parikh}} L(A)$.*

Next we observe that if A has one or more diamonds, then without loss of generality A can be assumed to have no linear transitions with label $a\&b$.

Lemma 4. *Assume that A has $r \geq 1$ diamonds and $k_A \geq 1$. Then there exists a DFA A_2 with $n-r$ states such that $L(A_2) \equiv_{\text{Parikh}} L(A)$.*

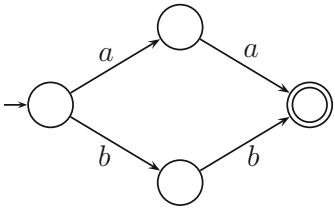

Fig. 5. A diamond.

Proof. This follows from the observation that when $k_A \geq 1$,

$$(aa + bb)^r (a + b)^{k_A} \equiv_{\text{Parikh}} (a + b)^{2r+k_A}.$$

□

By Lemmas 3 and 4, when computing an upper bound estimate for the state complexity of $\text{per}(L(A))$, in the case where A has blocks of length two, we can assume that all blocks of length two are all diamonds and, furthermore, that $k_A = 0$ (i.e., A has no linear transitions labeled with $a\&b$).

With the above assumptions combining with (1) and (2), we have

$$\frac{3}{2} \cdot h_A + i_A + j_A \leq n - 1.$$

We construct a DFA B recognizing $\text{per}(L(A))$. Note that it is sufficient for B to count a's up to $i_A + h_A$ and count b's up to $j_A + h_A$ with the further restriction that the sum of the counts is at most $i_A + j_A + h_A$. The states of B consist of pairs (x, y), where x is the a-count and y is the b-count. The states can be listed as follows:

- $(i_A + 1) \cdot (j_A + 1)$ pairs, where a-count is at most i_A and b-count is at most j_A.
- When a-count is $i_A + z$, for $1 \leq z \leq h_A$, b-count can be between 0 and $j_A + h_A - z$. This results in $\frac{1}{2} h_A (2j_A + h_A + 1)$ states. (The number of states comes from calculating, for some positive integers m and n, the cardinality of the following set $\{(i_0, j_0) \mid 1 \leq i_0 \leq m, 0 \leq j_0 \leq n + m - i\}$. After some calculations we conclude that the set has $\frac{1}{2} m(2n + m + 1)$ elements.)
- Additionally, for each b-count greater than j_A, we need to count up to i_A a's, which results in $h_A \cdot (i_A + 1)$ added states. (The situation where also the a-count is above i_A was included already in states listed above.)

In total, B has

$$i_A j_A + h_A i_A + h_A j_A + \frac{1}{2} h_A^2 + i_A + j_A + \frac{3}{2} h_A + 1$$

states.

This number is maximized whenever $i_A = \frac{2}{7}(n - 1)$, $j_A = \frac{2}{7}(n - 1)$, and $h_A = \frac{2}{7}(n-1)$ leading to a value of $\frac{2}{7}(n-1)^2 + n$ (in these cases $\frac{3}{2} \cdot h_A + i_A + j_A = n - 1$). (This maximization can be easily checked by mathematics software such as Maple.) This polynomial is bounded by $\frac{n^2+n+1}{3}$ and only reaches that bound in the trivial case where $i_A = 0$, $j_A = 0$, $h_A = 0$, and $n = 1$.

Above we have verified the following:

Proposition 1. *If A is a DFA with n states that recognizes a set of equal length strings over $\{a, b\}$ and the nonlinear part of A has only blocks of length two, then*

$$\mathrm{sc}(\mathrm{per}(L(A)) \leq \frac{n^2 + n + 1}{3}.$$

4.2 Estimate for General DFAs for Equal Length Languages

The result of the following theorem extends the result of Proposition 1 to all DFAs recognizing sets of equal length strings. Due to the limit on the number of pages the proof of Theorem 2 is omitted in this extended abstract.

Theorem 2. *Let A be an n-state DFA accepting a language $L \subseteq \Sigma^\ell$. Then there exists a DFA C accepting* $\mathrm{per}(L)$ *with no more than $\frac{n^2+n+1}{3}$ states.*

From Theorem 1, we already know that the upper bound of Theorem 2 can be reached.

Corollary 3. *For every $n_0 \in \mathbb{N}$, there is a positive integer ℓ and a regular language $L \subseteq \Sigma^\ell$, with* $\mathrm{sc}(L) = n$, *for $n \geq n_0$, such that every DFA accepting* $\mathrm{per}(L)$ *needs at least $\frac{n^2+n+1}{3}$ states.*

5 Conclusions

We have studied the deterministic state complexity of permutation of finite languages over binary alphabets. More specifically, we have presented asymptotically tight upper and lower bounds for the general case. We have also established the matching upper and lower bound on the restricted cases when the given language recognizes strings with equal length. Matching bounds of the general case remain open. Moreover, the state complexity of permutation over non-binary languages remains open, as well as, the nondeterministic state complexity of finite languages.

Acknowledgment. This research was supported by the Basic Science Research Program through NRF funded by MEST (2012R1A1A2044562), the International Cooperation Program managed by NRF of Korea (2014K2A1A2048512) and the Natural Sciences and Engineering Research Council of Canada Grant OGP0147224.

References

1. Câmpeanu, C., Culik, K., Salomaa, K., Yu, S.: State complexity of basic operations on finite languages. In: Boldt, O., Jürgensen, H. (eds.) WIA 1999. LNCS, vol. 2214, pp. 60–70. Springer, Heidelberg (2001)
2. Domaratzki, M.: State complexity of proportional removals. J. Autom. Lang. Comb. **7**(4), 455–468 (2002)
3. Ellul, K., Krawetz, B., Shallit, J., Wang, M.: Regular expressions: new results and open problems. J. Autom. Lang. Comb. **10**(4), 407–437 (2005)

4. Gao, Y., Moreira, N., Reis, R., Yu, S.: A review of state complexity of individual operations. Technical report, Universidade do Porto, Technical Report Series DCC-2011-08, Version 1.1, September (2012). www.dcc.fc.up.pt/Pubs (To appear in Computer Science Review, 2015)
5. Goč, D., Palioudakis, A., Salomaa, K.: Nondeterministic state complexity of proportional removals. In: Jurgensen, H., Reis, R. (eds.) DCFS 2013. LNCS, vol. 8031, pp. 102–111. Springer, Heidelberg (2013)
6. Goldstine, J., Kappes, M., Kintala, C.M.R., Leung, H., Malcher, A., Wotschke, D.: Descriptional complexity of machines with limited resources. J. UCS **8**(2), 193–234 (2002)
7. Han, Y.S., Salomaa, K.: State complexity of union and intersection of finite languages. Int. J. Found. Comput. Sci. **19**(03), 581–595 (2008)
8. Holzer, M., Kutrib, M.: State complexity of basic operations on nondeterministic finite automata. In: Champarnaud, J.-M., Maurel, D. (eds.) CIAA 2002. LNCS, vol. 2608, pp. 148–157. Springer, Heidelberg (2003)
9. Holzer, M., Kutrib, M.: Descriptional and computational complexity of finite automata – a survey. Inf. Comput. **209**(3), 456–470 (2011)
10. Kleene, S.C.: Representation of events in nerve nets and finite automata. Technical report, DTIC Document (1951)
11. Lavado, G.J., Pighizzini, G., Seki, S.: Operational state complexity under Parikh equivalence. In: Jürgensen, H., Karhumäki, J., Okhotin, A. (eds.) DCFS 2014. LNCS, vol. 8614, pp. 294–305. Springer, Heidelberg (2014)
12. Lupanov, O.: A comparison of two types of finite sources. Problemy Kibernetiki **9**, 328–335 (1963)
13. Maslov, A.: Estimates of the number of states of finite automata. In: Soviet Mathematics Doklady, Translation from Doklady Akademii Nauk SSSR 194, vol. 11, pp. 1266–1268, 1373–1375 (1970)
14. McCulloch, W.S., Pitts, W.: A logical calculus of the ideas immanent in nervous activity. Bull. Math. Biophys. **5**(4), 115–133 (1943)
15. Meyer, A.R., Fischer, M.J.: Economy of description by automata, grammars, and formal systems. In: SWAT (FOCS), pp. 188–191. IEEE Computer Society (1971)
16. Moore, F.: 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**(10), 1211–1214 (1971)
17. Parikh, R.J.: On context-free languages. J. ACM (JACM) **13**(4), 570–581 (1966)
18. Rabin, M.O., Scott, D.S.: Finite automata and their decision problems. IBM J. Res. Dev. **3**(2), 114–125 (1959)
19. Shallit, J.: A Second Course in Formal Languages and Automata Theory. Cambridge University Press, Cambridge (2008)
20. Yu, S., Zhuang, Q., Salomaa, K.: The state complexities of some basic operations on regular languages. Theor. Comput. Sci. **125**(2), 315–328 (1994)
21. Yu, S.: Handbook of Formal Languages, Volume 1, Chap. Regular Languages, pp. 41–110. Springer, Heidelberg (1998)