

## State Complexity of $k$ -Union and $k$ -Intersection for Prefix-Free Regular Languages\*

Hae-Sung Eom<sup>†</sup> and Yo-Sub Han<sup>‡</sup>

*Department of Computer Science, Yonsei University  
50, Yonsei-Ro, Seodaemun-Gu, Seoul 120-749, Republic of Korea*

<sup>†</sup>*haesung@cs.yonsei.ac.kr*

<sup>‡</sup>*emmous@cs.yonsei.ac.kr*

Kai Salomaa

*School of Computing, Queen's University  
Kingston, Ontario K7L 3N6, Canada  
ksalomaa@cs.queensu.ca*

Received 26 November 2013

Accepted 11 September 2014

Communicated by Cristian S. Calude

We investigate the state complexity of multiple unions and of multiple intersections for prefix-free regular languages. Prefix-free deterministic finite automata have their own unique structural properties that are crucial for obtaining state complexity upper bounds that are improved from those for general regular languages. We present a tight lower bound construction for  $k$ -union using an alphabet of size  $k + 1$  and for  $k$ -intersection using a binary alphabet. We prove that the state complexity upper bound for  $k$ -union cannot be reached by languages over an alphabet with less than  $k$  symbols. We also give a lower bound construction for  $k$ -union using a binary alphabet that is within a constant factor of the upper bound.

*Keywords:* State complexity; prefix-free regular languages;  $k$ -union;  $k$ -intersection.

### 1. Introduction

State complexity is one of the most intensively studied topics in automata and formal language theory in recent years [1, 2, 4, 6, 11, 14, 15, 20, 21, 24, 29, 30]. The state complexity problem is both interesting theoretically and relevant for practical applications. For example, in a regular-expression pattern matching, it is very useful to be able to estimate the size of a finite-state automaton for describing patterns, which helps to manage memory resources efficiently. On the other hand,

\*A preliminary version of this paper appeared in Proceedings of the 15th International Workshop on Descriptive Complexity of Formal Systems, DCFS 2013, Lect. Notes Comput. Sci. 8031, Springer-Verlag, 2013, pp. 78–89.

<sup>‡</sup>Corresponding author.

state complexity is a basic foundational property of regular languages. We find out more about structural properties of regular languages by establishing tight state complexity bounds for them.

The state complexity of a  $k$ -ary regularity-preserving language operation  $f$  is, roughly speaking, a function that associates with positive integers  $n_1, \dots, n_k$  the worst-case size of a minimal DFA for a language  $f(L_1, \dots, L_k)$  where  $L_i$  has a DFA of size  $n_i$ ,  $i = 1, \dots, k$ . Maslov [19] obtained the state complexity of concatenation and other basic operations; however, his short paper did not include many proofs. Later, unaware of the earlier work, Yu *et al.* [30] reintroduced the study of operational state complexity in a more systematic way. The state complexity of an operation is calculated based on the structural properties of input regular languages and a given operation.

While researchers mainly looked at the state complexity of single operations (union, intersection, catenation and so on), Yu and his co-authors started to investigate the state complexity of combined operations (star-of-union, star-of-intersection and so on) [7, 10, 23, 25]. They showed that the state complexity of a combined operation is usually not equal to the function composition of the state complexities of the participating individual operations. They also observed that, in a few cases, the state complexity of a combined operation is very close to the composition of the individual state complexities. In addition, Yu and his co-authors considered the state complexity of combined Boolean operations including multiple unions and multiple intersections [7–9]. They conjectured that the upper bound cannot be reached, in general, over an alphabet of a fixed size. Researchers also considered the state complexity of multiple operations such as several concatenations or several intersections [5, 7, 22]. Jirásková and her co-authors studied the state complexity of some operations for binary languages [3, 17, 18]. Binary languages allow us to prove the tightness of the upper bound also in the case of reversal of deterministic union-free languages, that is, languages represented by one-cycle-free-path deterministic automata, in which from each state there exists exactly one cycle-free accepting path [16].

Here we consider the state complexity of multiple unions ( $L_1 \cup L_2 \cup \dots \cup L_k$ ) and of multiple intersections ( $L_1 \cap L_2 \cap \dots \cap L_k$ ) for prefix-free regular languages. Note that prefix-free regular languages preserve unique structural properties in minimal DFAs, and these properties are crucial to obtain the state complexity bounds that are often significantly lower than for general regular languages [12, 13]. We first compute the upper bound for  $k$ -union and prove that the bound cannot be reached using a fixed alphabet and that, more precisely, the bound cannot be reached by languages defined over any alphabet of size less than  $k$ . We also present a tight lower bound construction using an alphabet of size  $k + 1$ . For  $k$ -union we also give a lower bound construction over a binary alphabet that is within a fraction of  $\frac{1}{2}$  of the general upper bound. For  $k$ -intersection, we compute the upper bound and present a tight lower bound construction using a binary alphabet.

In Sec. 2, we define some basic notions. Then we present the state complexity of  $k$ -union and  $k$ -intersection for prefix-free regular languages, respectively, in Secs. 3 and 4. We summarize the results and conclude the paper in Sec. 5.

## 2. Preliminaries

For  $k \in \mathbb{N}$ , we denote  $[1, k] = \{1, 2, \dots, k\}$ . We say that a set of positive integers  $\{m_1, \dots, m_k\}$  is pairwise relatively prime if, for any  $1 \leq i < j \leq k$ , the greatest common divisor of  $m_i$  and  $m_j$  is 1.

Let  $\Sigma$  denote a finite alphabet of characters and  $\Sigma^*$  denote the set of all strings over  $\Sigma$ . The size  $|\Sigma|$  of  $\Sigma$  is the number of characters in  $\Sigma$ . A language over  $\Sigma$  is any subset of  $\Sigma^*$ . The symbol  $\emptyset$  denotes the empty language and the symbol  $\lambda$  denotes the null string. Let  $|w|_b$  be the number of occurrences of symbol  $b \in \Sigma$  in the string  $w$ . For strings  $x, y$  and  $z$ , we say that  $x$  is a *prefix* of  $z$  and  $y$  is a *suffix* of  $z$  if  $z = xy$ . We define a language  $L$  to be prefix-free if for any two distinct strings  $x$  and  $y$  in  $L$ ,  $x$  is not a prefix of  $y$ .

A DFA  $A$  is specified by a tuple  $(Q, \Sigma, \delta, s, F)$ , where  $Q$  is a finite set of states,  $\Sigma$  is an input alphabet,  $\delta : Q \times \Sigma \rightarrow Q$  is a transition function,  $s \in Q$  is the start state and  $F \subseteq Q$  is a set of final states. Given a DFA  $A$ , we assume that  $A$  is complete; namely, each state has  $|\Sigma|$  out-transitions and, therefore,  $A$  may have a sink state (a non-final state where all outgoing transitions are self-loops). We assume that  $A$  has a unique sink state since all sink states are equivalent and can be merged into a single state. Let  $|Q|$  be the number of states in  $Q$ . By the size of  $A$  we mean  $|Q|$ . For a transition  $\delta(p, a) = q$  in  $A$ , we say that  $p$  has an *out-transition* and  $q$  has an *in-transition*. Furthermore,  $p$  is a *source state* of  $q$  and  $q$  is a *target state* of  $p$ . We say that  $A$  is *non-returning* if the start state of  $A$  does not have any in-transitions and  $A$  is *non-exiting* if all out-transitions of any final state in  $A$  go to the sink state.

A string  $x$  over  $\Sigma$  is accepted by  $A$  if there is a labeled path from  $s$  to a final state such that this path spells out  $x$ . We call this path an *accepting path*. The language  $L(A)$  of  $A$  is the set of all strings spelled out by accepting paths in  $A$ . For a minimal DFA  $A$ ,  $L(A)$  is prefix-free if and only if  $A$  has exactly one accept state and all transitions from the accept state go to the sink state, that is,  $A$  is non-exiting. We define a state  $q$  of  $A$  to be *reachable* (respectively, *co-reachable*) if there is a path from the start state to  $q$  (respectively, a path from  $q$  to a final state). In the following, unless otherwise mentioned, we assume that all states are reachable and all states except the sink state are co-reachable and a DFA has at most one sink state. The state complexity  $\mathcal{SC}(L)$  of a regular language  $L$  is defined to be the size of the minimal DFA recognizing  $L$ .

For more knowledge in automata theory, the reader may refer to the textbooks [26–28].

## 3. Union of $k$ Prefix-Free Languages

We first consider the state complexity of  $L_1 \cup L_2 \cup \dots \cup L_k$  ( $k$ -union operation) for prefix-free regular languages  $L_i \subseteq \Sigma^*$ , where  $1 \leq i \leq k$  for  $k \geq 3$ . Note that

the case  $k = 2$  has been dealt with in [13]. Also, in the construction below we restrict consideration to prefix-free DFAs having at least three states because the only prefix-free language having a DFA of size two is  $\{\lambda\}$ .

### 3.1. Construction of a DFA for $L_1 \cup \dots \cup L_k$

For  $1 \leq i \leq k$ , consider minimal prefix-free DFAs  $A_i = (Q_i, \Sigma, \delta_i, q_{0,i}, \{f_i\})$  of size  $m_i \geq 3$ . Note that a minimal prefix-free DFA has a sink state and a unique final state and hence we can denote  $Q_i = P_i \cup \{f_i, d_i\}$ ,  $P_i = \{q_{0,i}, q_{1,i}, \dots, q_{m_i-3,i}\}$  where  $d_i$  is the sink state of  $A_i$  and  $f_i$  is the unique final state of  $A_i$ . The states of  $P_i$  are non-final and each state of  $P_i$  can reach a final state,  $1 \leq i \leq k$ .

The union  $L(A_1) \cup \dots \cup L(A_k)$  is recognized by a DFA

$$B = (R, \Sigma, \gamma, r_0, F), \quad (1)$$

where  $R = (P_1 \times \dots \times P_k) \cup R_1 \cup \{d_B, f_B\}$  and  $R_1 = \bigcup_{\emptyset \neq S \subsetneq [1,k]} (\prod_{i \in S} P_i) \times \{\text{acc}, \text{rej}\}$ . The notation  $\prod_{i \in S} P_i$  above denotes the Cartesian product of sets  $P_i$ ,  $i \in S$ , taken in order of increasing  $i$ . That is, if  $S = \{j_1, \dots, j_r\}$ ,  $1 \leq j_1 < j_2 < \dots < j_r \leq k$ , the product is  $P_{j_1} \times P_{j_2} \times \dots \times P_{j_r}$ . (The order of the components of  $\prod_{i \in S} P_i$  is not important. However, for the construction we need that the order is fixed.)

The initial state  $r_0$  of  $B$  is the tuple  $(q_{0,1}, \dots, q_{0,k})$  consisting of the initial states of each of the  $A_i$ 's. The set of final states  $F$  consists of  $f_B$  and all tuples in  $R_1$  where the last component is acc. Before defining the transitions we explain the intuitive idea of the construction of the DFA  $B$  which, hopefully, makes also the choice of the set of states more transparent. The DFA  $B$  simulates all the DFAs  $A_i$ ,  $1 \leq i \leq k$ , in parallel. The states of  $P_1 \times \dots \times P_k$  simulate computations where none of the  $A_i$ 's has reached a final state or a sink state.

When the simulated computation of  $A_i$  reaches the final state  $f_i$ , in the next step  $A_i$  necessarily goes to the sink state. The states of  $R_1$  simulate computations where at least one  $A_i$  but not all the  $A_i$ 's have reached the final state or the sink state. Suppose that  $\emptyset \neq S \subsetneq [1, k]$  is the set of indices of DFAs  $A_i$  that, in the simulated computation, have not yet reached the final nor the sink state. Now the state of  $B$  (belonging to  $R_1$ ) keeps track of only the corresponding states of  $A_i$ ,  $i \in S$  and does not need to store the information whether the state of  $A_j$ , for each  $j \notin S$ , is  $d_j$  or  $f_j$ . Instead, in the last component, the state of  $R_1$  stores only a binary choice. If the last component is acc, this means that at least one  $A_j$ , for  $j \notin S$ , is in the final state  $f_j$  and the last component being rej encodes the situation where all the DFAs  $A_j$ ,  $j \notin S$ , are in the sink state.

Finally, the state  $f_B \in F$  encodes the situation where, in the simulated computation, all the  $A_i$ 's are in the accept state and  $d_B$  the situation where all the  $A_i$ 's are in the sink state.

It remains to define the transition relation  $\gamma$  of  $B$ . First, for every  $b \in \Sigma$ , we define  $\gamma(f_B, b) = \gamma(d_B, b) = d_B$ . Note that  $d_B$  is the sink state of  $B$  and  $f_B$  is a special final state from which all transitions lead to the sink state.

Second, we define the general transitions. Let  $S = \{j_1, \dots, j_s\} \subset [1, k]$ ,  $1 \leq j_1 < \dots < j_s \leq k$ . For  $\bar{z} = (z_{j_1}, \dots, z_{j_s})$ ,  $z_{j_x} \in P_{j_x}$ ,  $1 \leq x \leq s$ , and  $b \in \Sigma$ , we define  $S_{\bar{z},b} \subseteq S$  to consist of those indices of  $j_i \in S$  such that  $\delta_{j_i}(z_{j_i}, b) \in P_{j_i}$ . That is, if  $S$  gives the indices of the DFAs  $A_i$  that in the simulated computation have not reached the final state nor the sink state (i.e.,  $A_i$  is in a state of  $P_i$ ), then  $S_{\bar{z},b}$  gives the indices of the DFAs that after processing a further input symbol  $b \in \Sigma$  have still not reached the final state nor the sink state.

Let  $y \in \{\text{acc}, \text{rej}\}$  be arbitrary. Let  $S$  and  $\bar{z}$  be as above and denote  $S_{\bar{z},b} = \{h_1, \dots, h_t\}$ ,  $h_1 < \dots < h_t$ ,  $0 \leq t \leq s$ . Now when  $\emptyset \neq S_{\bar{z},b} \neq [1, k]$  we define

$$\gamma((z_{j_1}, \dots, z_{j_s}, y), b) = (\delta_{h_1}(z_{h_1}, b), \dots, \delta_{h_t}(z_{h_t}, b), y') \quad (2)$$

where  $y'$  is acc if there exists  $i \in S - S_{\bar{z},b}$  such that  $\delta_{h_i}(z_{h_i}, b) = f_i$  and  $y'$  is rej otherwise. Note that the transition step simply simulates the computation step of each of the individual  $A_{z_{h_i}}$  and eliminates from the tuple the states of the DFAs that go to the accept state or the sink state. The last component  $y'$  of the new state simply encodes the information whether or not some of the eliminated computations entered a final state. Note that the computation step in Eq. (2) does not depend on the last component  $y$  of the original state.

There remain only two special cases to define separately that correspond to situations where either  $S_{\bar{z},b} = \emptyset$  or the original set of indices  $S$  consists of the entire set  $S$ .

If  $S_{\bar{z},b} = \emptyset$ , we define

$$\gamma((z_{j_1}, \dots, z_{j_s}, y), b) = \begin{cases} f_B, & \text{if } (\exists 1 \leq i \leq s) \delta_{j_i}(z_{j_i}, b) = f_{j_i}; \\ d_B, & \text{otherwise.} \end{cases} \quad (3)$$

Finally, if  $\bar{z} = (z_1, \dots, z_k)$ ,  $z_i \in P_i$ ,  $1 \leq i \leq k$ , we define

$$\gamma((z_1, \dots, z_k), b) = \begin{cases} (\delta_1(z_1, b), \dots, \delta_k(z_k, b)), & \text{if } \delta_j(z_j, b) \in P_j; \\ (\delta_{h_1}(z_{h_1}, b), \dots, \delta_{h_t}(z_{h_t}, b), y'), & \text{otherwise.} \end{cases} \quad (4)$$

In the latter case of Eq. (4),  $S_{\bar{z},b} = \{h_1, \dots, h_t\}$  and  $y' = \text{acc}$  if there exists  $i \in [1, k] - \{h_1, \dots, h_t\}$  such that  $\delta_i(z_i, b) = f_i$  and  $y' = \text{rej}$  otherwise.

The rules in Eq. (4) are used in the initial part of the computation where none of the components  $A_i$  has reached the accept nor the sink state. During this part of the computation, the state of  $B$  is a tuple of  $P_1 \times \dots \times P_k$  and we do not need a component of  $\{\text{acc}, \text{rej}\}$ . Finally, the transitions in Eq. (3) pertain to the situation where all components reach the accept or the sink state, and in this case the state of  $B$  is  $f_B$  if at least one component is in the accept state and the sink state  $d_B$  otherwise.

From the above description it is clear that  $B$  accepts an input string  $w$  if and only if at least one of the components  $A_i$ ,  $1 \leq i \leq k$ , accepts  $w$ , that is  $L(B) = L(A_1) \cup \dots \cup L(A_k)$ .

**Lemma 1.** *Let  $A_i$  be a prefix-free DFA of size  $m_i \geq 3$ ,  $i = 1, \dots, k$ . The union  $L(A_1) \cup \dots \cup L(A_k)$  can be recognized by a DFA of size*

$$2 \cdot \left( \sum_{\emptyset \neq S \subseteq [1,k]} \prod_{i \in S} (m_i - 2) \right) + \left( \prod_{i=1}^k (m_i - 2) \right) + 2.$$

**Proof.** Without loss of generality we can assume that each  $A_i$  is minimal. (If it is not, we replace it with an equivalent minimal DFA.)

In the notations of the construction for the DFA  $B$  we recall that  $|P_i| = m_i - 2$ . Thus the first term in the sum appearing in the statement of the lemma is the cardinality of  $R_1$  and the second term is the cardinality of  $P_1 \times \dots \times P_k$  and, additionally, the state set of  $B$  has only the two special states  $d_B$  and  $f_B$ .  $\square$

### 3.2. The upper bound cannot be reached with a fixed alphabet

We begin by observing that the upper bound of Lemma 1 cannot be reached for arbitrary  $k$  when the alphabet  $\Sigma$  is fixed.

**Lemma 2.** *If  $k > |\Sigma|$ , the upper bound of Lemma 1 cannot be reached.*

**Proof.** Let  $A_i = (Q_i, \Sigma, \delta_i, q_{0,i}, \{f_i\})$ ,  $i = 1, \dots, k$ , be minimal prefix-free DFAs and let  $B$  be constructed for the union  $L(A_1) \cup \dots \cup L(A_k)$  as in Eq. (1). In the following we use, without further mention, the notation for the DFAs  $A_i$  and  $B$  (as given in Sec. 3.1).

For  $1 \leq i \leq k$ , we define  $\Omega_i \subseteq \Sigma$  as follows:  $\Omega_i = \{c \in \Sigma \mid (\exists p \in P_i) \delta_i(p, c) = f_i\}$ . The set  $\Omega_i$  consists of alphabet symbols that take some state of  $P_i$  (where  $P_i$  consists of states of  $A_i$  that are neither final nor the sink state) to the unique final state. Since  $L(A_i) \neq \emptyset$ , we know that  $\Omega_i \neq \emptyset$ ,  $i = 1, \dots, k$ . The following observation will be the basis of our argument.

**Claim 1.** Let  $b \in \Sigma$ . If  $b \in \Omega_i$ ,  $1 \leq i \leq k$ , then the function  $\delta_i(\cdot, b)$  is not surjective on the set  $P_i$ . To verify the claim, we note that in the DFA  $A_i$  the states of  $P_i$  can be reached only from states of  $P_i$ . Since  $b \in \Omega_i$ , we know that the  $b$ -transitions take some state of  $P_i$  outside  $P_i$ . It follows that some state of  $P_i$  must be outside the range of  $\delta_i(\cdot, b)$ . Suppose now that for some  $1 \leq j \leq k$ ,

$$(\exists c_i \in \Omega_i, i = 1, \dots, j - 1, j + 1, \dots, k) \Omega_j \subseteq \{c_1, \dots, c_{j-1}, c_{j+1}, \dots, c_k\}. \quad (5)$$

The condition above means that there exists an index  $j$  such that by choosing one element from each of the sets  $\Omega_i$ ,  $i \neq j$ , we get all elements of  $\Omega_j$ .

We show that, assuming the tuple (5) holds, some states of the form

$$(p_1, \dots, p_{j-1}, p_{j+1}, \dots, p_k, \text{acc}), \quad p_i \in P_i, i = 1, \dots, j - 1, j + 1, \dots, k, \quad (6)$$

cannot be reachable in the DFA  $B$ . Since the tuple (6) is missing only the  $j$ th component corresponding to a state of  $A_j$ , according to rules in Eq. (4), a state of

the form tuple (6) can be reached only from a state of  $P_1 \times \dots \times P_k$  by a transition on a symbol  $b \in \Omega_j$  that takes a state of  $P_j$  to the final state  $f_j$  of  $A_j$ . Note that a state of the form tuple (6) cannot be reached from a state of the same form because in transitions of Eq. (2) the last component becomes “rej” unless one of the states  $p_1, \dots, p_{j-1}, p_{j+1}, \dots, p_k$  reaches the final state of the corresponding DFA, and in this case that component would also be omitted.

Now according to tuple (5) we can choose  $c_i \in \Omega_i$ ,  $i = 1, \dots, j - 1, j + 1, \dots, k$ , such that  $\Omega_j \subseteq \{c_1, \dots, c_{j-1}, c_{j+1}, \dots, c_k\}$ . By Claim 1 we know that  $\delta_i(\cdot, c_i)$  is not surjective on  $P_i$  and hence there exist  $p'_i \in P_i$  that cannot be reached in  $A_i$  by any transition on input  $c_i$ ,  $i = 1, \dots, j - 1, j + 1, \dots, k$ . In the previous paragraph it was observed that a state of the form tuple (5) can be reached only by an element of  $\Omega_j$ . Since each element of  $\Omega_j$  is one of the  $c_i$ 's, it follows that the state  $(p'_1, \dots, p'_{j-1}, p'_{j+1}, \dots, p'_k, \text{acc})$  will be unreachable in the DFA  $B$ .

Finally, it is easy to see using induction on  $k$  that when  $|\Sigma| < k$  we cannot choose nonempty subsets  $\Omega_i$  of  $\Sigma$ ,  $i = 1, \dots, k$ , such that tuple (5) fails to hold. That is, no matter how the individual DFAs  $A_i$  are defined, for some  $1 \leq j \leq k$ , the condition tuple (5) holds and, consequently,  $B$  has unreachable states.  $\square$

### 3.3. Lower bound with a binary alphabet

Next we give for the  $k$ -union of prefix-free languages a lower bound construction over a binary alphabet that reaches the upper bound of Lemma 1, within a constant factor. The motivation for first considering the binary alphabet case is that using a simple modification of the binary alphabet construction we obtain, in Sec. 3.4, an optimal lower bound using an alphabet of size  $k + 1$ . Furthermore, the same languages over a binary alphabet will be used in the next section to give a tight lower bound for the state complexity of  $k$ -intersection of prefix-free languages.

Our results leave open the question whether or not the upper bound can be reached using an alphabet of size exactly  $k$ . Note that from Lemma 2 we know that  $k - 1$  alphabet symbols are not sufficient.

We choose  $\Sigma = \{a, b\}$ . For  $w \in \Sigma^*$  of length  $m$ , the set of *positions* of  $w$  is  $\{1, \dots, m\}$ . For  $x \in \{a, b\}$  and  $1 \leq u \leq m$ , we say that  $u$  is an  $x$ -position of  $w$  if the  $u$ 'th symbol of  $w$  is  $x$ .

The set of all  $b$ -positions of string  $w$  is denoted  $\text{pos}_b(w) \subseteq \{1, \dots, |w|\}$ . Consider a  $b$ -position  $u_b$  of  $w \in \Sigma^*$ . By the  $a$ -count of position  $u_b$ ,  $\text{count}_a(u_b)$ , we mean the number of occurrences of symbol  $a$  that precede the occurrence of  $b$  at position  $u_b$ .

For each  $m \in \mathbb{N}$  we define the prefix-free language

$$L_m \subseteq \{a, b\}^* \tag{7}$$

to consist of all strings  $w$  such that

- (1)  $|w| \in \text{pos}_b(w)$ ,
- (2)  $\text{count}_a(|w|) \equiv 0 \pmod{m}$ , and,
- (3)  $(\forall u \in \text{pos}_b(w)) u < |w|$  implies  $\text{count}_a(u) \not\equiv 0 \pmod{m}$ .

That is,  $L_m$  consists of strings  $w$  ending with symbol  $b$  where the number of occurrences of symbols  $a$  is a multiple of  $m$ . Furthermore, any occurrence of  $b$  in  $w$  except the last one is preceded by a number of  $a$ 's that is not a multiple of  $m$ .

**Lemma 3.** For  $m \geq 1$ , the language  $L_m$  is prefix-free and is recognized by a DFA  $A_{m+2}$  with  $m + 2$  states.

**Proof.** The prefix-freeness of  $L_m$  follows directly from the definition of the language.

We choose  $A_{m+2} = (Q, \Sigma, \delta, 0, \{m\})$  where  $Q = \{0, 1, \dots, m + 1\}$  and the transition relation  $\delta$  is defined by setting

- (1) for  $0 \leq i \leq m - 2$ ,  $\delta(i, a) = i + 1$ ,
- (2)  $\delta(m - 1, a) = 0$ ,  $\delta(m, a) = \delta(m + 1, a) = m + 1$ ,
- (3) for  $1 \leq i \leq m - 1$  and  $i = m + 1$ ,  $\delta(i, b) = i$ ,
- (4)  $\delta(0, b) = m$ ,  $\delta(m, b) = m + 1$ .

The DFA  $A_7$  is illustrated in Fig. 1. □

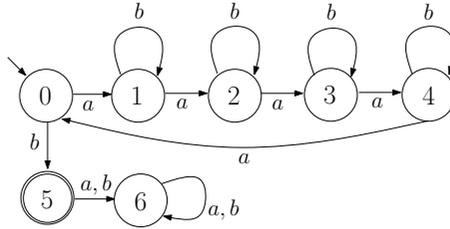


Fig. 1. The DFA  $A_7$  recognizing  $L_5$ .

**Lemma 4.** Let  $\{m_1, \dots, m_k\}$  be a set of relatively prime integers where  $m_i \geq 2$ ,  $i = 1, \dots, k$ . Let  $L_{m_i}$ ,  $1 \leq i \leq k$ , be the language defined in language (7).

Then the minimal DFA for  $L_{m_1} \cup \dots \cup L_{m_k}$  has

$$\left( \sum_{\emptyset \neq S \subseteq [1, k]} \prod_{i \in S} m_i \right) + \left( \sum_{\emptyset \neq S \subseteq [1, k]} \prod_{i \in S} (m_i - 1) \right) + \left( \prod_{i=1}^k m_i \right) + 2 \quad (8)$$

states.

**Proof.** According to Lemma 3, the language  $L_{m_i}$ ,  $1 \leq i \leq k$ , is recognized by a DFA  $A_{m_i+2} = (Q_i, \Sigma, \delta_i, q_0, \{q_{m_i}\})$  where  $Q_i = \{q_{0,i}, q_{1,i}, \dots, q_{m_i,i}, q_{m_i+1,i}\}$  and  $\delta_i$  is defined as in Lemma 3. To make our notations consistent with the upper bound construction from Sec. 3.1, we rename the final state  $q_{m_i,i}$  as  $f_i$  and the sink state  $q_{m_i+1,i}$  as  $d_i$ .

Now as in the upper bound construction we denote  $Q_i = P_i \cup \{f_i, d_i\}$ ,  $P_i = \{q_{0,i}, q_{1,i}, \dots, q_{m_i-1,i}\}$ ,

$$R = (P_1 \times \dots \times P_k) \cup R_1 \cup \{d_B, f_B\}$$

and

$$R_1 = \bigcup_{\emptyset \neq S \subset [1,k]} \left( \prod_{i \in S} P_i \right) \times \{\text{acc}, \text{rej}\}.$$

Then  $L_{m_1} \cup \dots \cup L_{m_k}$  is recognized by a DFA  $B = (R, \Sigma, \gamma, r_0, F)$  defined as in Eq. (1). In order to prove the lemma it is sufficient to show that

(i) all states of  $B$  except states of the form

$$(p_{j_1}, \dots, p_{j_r}, \text{acc}), \{j_1, \dots, j_r\} \subset [1, k], r \geq 1, \text{ and } (\exists 1 \leq x \leq r) p_{j_x} = q_{0,j_x}$$

are reachable, and,

(ii) all states of  $B$  are pairwise inequivalent.

Note that the number of states as given in (i) is exactly Eq. (8) because, from the states of the original DFA  $B$  the tuples where the last component is “acc” and some of the previous components is the initial state of some  $A_i$  are omitted. Thus, the number of states with last component “acc” that are included in (i) is  $\sum_{\emptyset \neq S \subset [1,k]} \prod_{i \in S} (m_i - 1)$ .

Note also that for (ii) we establish that all states of  $B$ , including the unreachable states, are pairwise inequivalent. Naturally for our current purpose it would not be necessary to consider the unreachable states, however, this stronger property will be used in the next subsection to argue that modified construction with alphabet of size  $k + 1$  will reach the upper bound of Lemma 1.

We show that all states as in (i) are reachable. Since each  $A_{m_i+2}$  contains a cycle of  $a$ -transitions of length  $m_i$  beginning from the start state and all the  $m_i$ 's are relatively prime, the states of  $P_1 \times \dots \times P_k$  can be reached just using  $a$ -transitions. The state  $f_B$  is reached by reading a  $b$  from the start state and  $d_B$  is reached from  $f_B$ .

Inductively on the size of  $S$ ,  $1 \leq |S| \leq k - 1$ , we show that all states of  $(\prod_{i \in S} P_i) \times \{\text{acc}, \text{rej}\}$  are reachable, with the exception of states that contain an initial state of some  $A_i$  and have as the last component “acc”.

• Suppose  $|S| = k - 1$  and consider a state

$$(q_{1,i_1}, \dots, q_{j-1,i_{j-1}}, q_{j+1,i_{j+1}}, \dots, q_{k,i_k}, \text{zz}), \tag{9}$$

$$0 \leq i_x \leq m_x - 1, x = 1, \dots, k, \text{zz} \in \{\text{acc}, \text{rej}\}.$$

If none of the  $q_{x,i_x}$ 's is the initial state of  $A_x$  (that is,  $i_x \neq 0$  for all  $x$ ), then when zz is acc (respectively, when zz is rej) the state (9) is reached from state

$$(q_{1,i_1}, \dots, q_{j-1,i_{j-1}}, q_{j,0}, q_{j+1,i_{j+1}}, \dots, q_{k,i_k})$$

by reading a symbol  $b$  (respectively, by reading two symbols  $b$ ).

Consider next the situation where the  $q_{x,i_x}$ 's are arbitrary and  $zz = \text{rej}$ . Now from a state  $(q_{1,1}, \dots, q_{j-1,1}, q_{0,j}, q_{j+1,1}, \dots, q_{k,1})$  by reading two symbols  $b$  we reach the state  $(q_{1,1}, \dots, q_{1,j-1}, q_{1,j+1}, \dots, q_{1,k}, \text{rej})$ . From this state we reach state (9) by reaching  $y \in \mathbb{N}$  symbols  $a$ , where

$$y \equiv i_x - 1 \pmod{m_x}, \quad x = 1, \dots, j - 1, j + 1, \dots, k.$$

Since all the integers  $m_x$  are relatively prime, a positive integer  $y$  satisfying the above exists.

- In the inductive step, exactly as above, a state of  $(\prod_{i \in S} P_i) \times \{\text{acc}\}$ , (respectively, of  $(\prod_{i \in S} P_i) \times \{\text{rej}\}$ ) with  $|S| < k - 1$  that is of the form (i) is obtained by reading a  $b$  (respectively, two symbols  $b$ ) from a state of  $(\prod_{i \in S'} P_i) \times \{\text{acc}, \text{rej}\}$ ,  $S' = S \cup \{j\}$  ( $j \notin S$ ), that contains as the  $j$ th component the start state  $q_{j,0}$ .

We show that all states of the DFA  $B$  are pairwise inequivalent. The state  $d_B$  is the unique sink state and  $f_B$  is the only state where all transitions lead to the sink state and hence these states cannot be equivalent with any other state.

Consider two distinct states

$$\bar{r} = (r_{i_1}, \dots, r_{i_u}, z_r), \quad 1 \leq i_1 < \dots < i_u \leq k, \quad r_{i_x} \in P_{i_x}, \quad x = 1, \dots, u,$$

$$\bar{s} = (s_{j_1}, \dots, s_{j_v}, z_s), \quad 1 \leq j_1 < \dots < j_v \leq k, \quad s_{j_x} \in P_{j_x}, \quad x = 1, \dots, v,$$

where  $z_r$  and  $z_s$  are  $\text{acc}$ ,  $\text{rej}$ , or are omitted (in which case we have an element of  $P_1 \times \dots \times P_k$ ). If  $u = v$ ,  $i_x = j_x$  and  $r_{i_x} = s_{j_x}$ ,  $x = 1, \dots, u$ , then  $\{z_r, z_s\} = \{\text{acc}, \text{rej}\}$  and states  $\bar{r}$  and  $\bar{s}$  are inequivalent because one of them is final and the other is not. Note that in this case it is not possible, for example, that  $z_r = \text{rej}$  and  $z_s$  is omitted, because  $z_r = \text{rej}$  implies that  $u < k$  and  $z_s$  is omitted only when  $v = k$ .

Second consider the situation where the sequences  $r_{i_1}, \dots, r_{i_u}$  and  $s_{j_1}, \dots, s_{j_v}$  are not identical.

- (a) Suppose there exists  $1 \leq x \leq u$  such that  $i_x \notin \{j_1, \dots, j_v\}$ . Since the integers  $m_x, m_{j_1}, \dots, m_{j_v}$  are all relatively prime, we can select an integer  $y$  such that by reading  $y$  symbols  $a$  the DFA  $A_{m_x}$  takes the state  $r_{i_x}$  to  $q_{i_x,0}$  and each DFA  $A_{j_h}$  takes the state  $s_{j_h}$  to  $q_{j_h,1}$ , for  $h = 1, \dots, v$ . This means that, in the computation of  $B$ , the string  $a^y b$  takes the state  $\bar{r}$  to a final state and takes the state  $\bar{s}$  to a nonfinal state.

The case where the sequence  $j_1, \dots, j_v$  contains an index not appearing in the sequence  $i_1, \dots, i_u$  is completely symmetric.

- (b) The remaining case is where  $u = v$ ,  $i_x = j_x$ ,  $x = 1, \dots, u$  and there exists  $1 \leq h \leq u$  such that  $r_{i_h} \neq s_{i_h}$ . Again we can select an integer  $y$  such that by reading  $y$  symbols  $a$  the DFA  $A_{i_h}$  takes the state  $r_{i_h}$  to  $q_{i_h,0}$  and each DFA  $A_{i_t}$ ,  $t \neq h$ ,  $1 \leq t \leq u$  takes the state  $s_{i_t}$  to  $q_{i_t,1}$ . Furthermore, since  $r_{i_h} \neq s_{i_h}$ , we know that  $A_{i_h}$  by reading  $y$  symbols  $a$  does not take  $s_{i_h}$  to the start state  $q_{i_h,0}$ . This means that by reading the string  $a^y b$  the DFA  $B$  takes the state  $\bar{r}$  to a final state and the state  $\bar{s}$  to a nonfinal state. □

When comparing the value in Eq. (8) to the upper bound of Lemma 1 we recall that the minimal DFA for  $L_{m_i}$  had  $m_i + 2$  states. Thus for the union of prefix-free DFAs of size  $m_i$ ,  $1 \leq i \leq k$ , over a binary alphabet the construction of Lemma 4 gives a lower bound

$$\left( \sum_{\emptyset \neq S \subseteq [1,k]} \prod_{i \in S} (m_i - 2) \right) + \left( \sum_{\emptyset \neq S \subseteq [1,k]} \prod_{i \in S} (m_i - 3) \right) + \left( \prod_{i=1}^k (m_i - 2) \right) + 2.$$

This differs from the upper bound of Lemma 1 by

$$\left( \sum_{\emptyset \neq S \subseteq [1,k]} \prod_{i \in S} (m_i - 2) \right) - \left( \sum_{\emptyset \neq S \subseteq [1,k]} \prod_{i \in S} (m_i - 3) \right).$$

**Corollary 3.1.** *For arbitrary  $k \geq 1$  the lower bound for the union of  $k$  prefix-free DFAs over a binary alphabet is more than half of the general upper bound of Lemma 1.*

### 3.4. Tight lower bound with alphabet of size $k + 1$

As a simple modification of the binary DFA construction from the previous section, we obtain a lower bound matching the upper bound of Lemma 1 using an alphabet of size  $k + 1$ .

In the following let  $k \in \mathbb{N}$  be arbitrary but fixed and  $\Sigma = \{a, b_1, \dots, b_k\}$ . For  $m \in \mathbb{N}$  and  $1 \leq i \leq k$  we define the language  $L_{i,m}$  to consist of all strings  $w \in \Sigma^*$  such that  $w$  ends with  $b_i$ ,  $|w|_a \equiv 0 \pmod m$  and the number of  $a$ 's preceding any other occurrence of  $b_i$  in  $w$  except the last one is not a multiple of  $m$ . In strings of  $L_{i,m}$  the symbols  $b_j$ ,  $j \neq i$ , can occur in arbitrary positions, except that the string must end with  $b_i$ .

Clearly  $L_{i,m}$  is prefix-free,  $1 \leq i \leq m$ , and  $L_{i,m}$  is recognized by a DFA  $A'_{i,m+2} = (Q, \Sigma, \delta, 0, \{m\})$  where  $Q = \{0, 1, \dots, m + 1\}$  and the transition relation  $\delta$  is defined by setting

- (1) for  $0 \leq j \leq m - 2$ ,  $\delta(j, a) = j + 1$ ,
- (2)  $\delta(m - 1, a) = 0$ ,  $\delta(m, a) = \delta(m + 1, a) = m + 1$ ,
- (3) for  $1 \leq j \leq m - 1$  and  $j = m + 1$ ,  $\delta(j, b_i) = j$ ,
- (4)  $\delta(0, b_i) = m$ ,
- (5) for  $0 \leq j \leq m - 1$  and  $h \neq i$ ,  $\delta(j, b_h) = j$ ,
- (6) for all  $1 \leq h \leq k$ ,  $\delta(m, b_h) = \delta(m + 1, b_h) = m + 1$ .

The transitions of the DFA  $A'_{i,m+2}$  restricted to subalphabet  $\{a, b_i\}$  are an isomorphic copy of the DFA  $A_{m+2}$  defined in the proof of Lemma 3. In  $A'_{i,m+2}$  the transitions on  $b_j$ ,  $j \neq i$ , are the identity on states  $\{0, 1, \dots, m - 1\}$  and take  $m$  and  $m + 1$  to the sink state  $m + 1$ .

Let  $\{m_1, \dots, m_k\}$  be a set of relatively prime integers. As in the proof of Lemma 1 we construct a DFA  $B'$  for  $L_{1,m_1} \cup \dots \cup L_{k,m_k}$ . Now similarly as in the proof

of Lemma 4 we verify that all states of  $B'$  are reachable and pairwise inequivalent. The latter property was shown to hold already in the case of a binary alphabet. The only unreachable states in the construction used in the proof of Lemma 4 were states of the form  $(p_{j_1}, \dots, p_{j_r}, \text{acc})$ , where some  $p_{j_x}$ ,  $1 \leq x \leq r$ , was the final state of  $A_{m_{j_x}+2}$ . In  $B'$  the above state is reached from a state  $(p_{j_1}, \dots, q_{h,0}, \dots, p_{j_r}, \text{rej})$  by reading a symbol  $b_h$ . Here  $1 \leq h \leq k$  is an index not appearing in the sequence  $j_1, \dots, j_r$ , and hence the  $B'$ -transition on  $b_h$  does not change the components  $p_{j_1}, \dots, p_{j_r}$ . (Strictly speaking, it may be the case that we must choose  $h < j_1$  or  $h > j_r$  in which case the notations above are slightly different.)

Thus, as a consequence of Lemma 1 and Proposition 2, we have the following result.

**Theorem 1.** *Let  $A_i$  be a prefix-free DFA of size  $m_i \geq 3$ ,  $i = 1, \dots, k$ . The union  $L(A_1) \cup \dots \cup L(A_k)$  can be recognized by a DFA of size*

$$2 \cdot \left( \sum_{\emptyset \neq S \subseteq [1,k]} \prod_{i \in S} (m_i - 2) \right) + \left( \prod_{i=1}^k (m_i - 2) \right) + 2.$$

*For any integers  $n_1, \dots, n_k$ , there exist prefix-free DFAs  $A_i$  over an alphabet of size  $k + 1$  having size  $m_i \geq n_i$ ,  $i = 1, \dots, k$ , such that the minimal DFA for  $L(A_1) \cup \dots \cup L(A_k)$  needs this number of states. Moreover, the bound cannot be reached by prefix-free DFAs over an alphabet with less than  $k$  symbols.*

Theorem 1 leaves open the question whether or not the worst-case state complexity of  $k$ -union of prefix-free languages can be reached by DFAs over an alphabet of size exactly  $k$ . We conjecture a positive answer to this question but the required lower bound construction would likely be much more complicated than the construction used above for proving Theorem 1.

#### 4. Intersection of $k$ Prefix-Free Languages

We consider the state complexity of  $L_1 \cap L_2 \cap \dots \cap L_k$  (the  $k$ -intersection operation) for prefix-free regular languages. First we give an upper bound construction. Recall, as in the previous section, that we can restrict consideration to DFAs of size at least three because any non-trivial prefix-free DFA has at least three states.

**Lemma 5.** *Let  $A_i$  be a DFA with  $m_i$  states,  $m_i \geq 3$ , that recognizes a prefix-free language  $L_i$ ,  $1 \leq i \leq k$ . Then  $2 + \prod_{i=1}^k (m_i - 2)$  states are sufficient for a DFA to*

*recognize  $\bigcap_{i=1}^k L_i$ .*

**Proof.** Let  $A_i = (Q_i, \Sigma, \delta_i, s_i, \{f_i\})$  where  $|Q_i| = m_i$ ,  $1 \leq i \leq k$ . We construct a DFA  $B = (R, \Sigma, \gamma, r_0, F)$  that accepts  $L(A_1) \cap L(A_2) \cap \dots \cap L(A_k)$ . Denote

$P_i = Q_i \setminus \{d_i, f_i\}$ , where  $d_i$  is the unique sink state of  $A_i$ ,  $1 \leq i \leq k$ . As the state set of  $B$  we choose

$$R = (P_1 \times P_2 \times \cdots \times P_k) \cup \{d_B, f_B\}.$$

Here  $d_B$  is the sink state of  $B$  and  $F = \{f_B\}$ . As the initial state of  $B$  we choose  $r_0 = (s_1, s_2, \dots, s_k)$ .

The transition function  $\gamma$  is defined as follows. For  $z \in \Sigma$  and  $q_i \in P_i$ , for  $1 \leq i \leq k$ ,

$$\gamma((q_1, q_2, \dots, q_k), z) = \begin{cases} d_B, & \text{if } \delta_i(q_i, z) = d_i \text{ for some } 1 \leq i \leq k, \\ f_B & \text{if } \delta_i(q_i, z) = f_i \text{ for all } 1 \leq i \leq k, \\ (\delta_1(q_1, z), \delta_2(q_2, z), \dots, \delta_k(q_k, z)), & \text{otherwise.} \end{cases} \quad (10)$$

Additionally, we define  $\gamma(d_B, z) = \gamma(f_B, z) = d_B$ , for all  $z \in \Sigma$ .

The computation of the DFA  $B$  simulates all the  $A_i$ 's in parallel and goes to the sink state  $d_B$  if any of the  $A_i$ 's goes to the sink state, and goes to the accepting state  $f_B$  if all the  $A_i$ 's simultaneously go to the accepting state. Note that since the languages  $L(A_i)$  are prefix-free, if a string  $w$  is in the intersection of the languages, no  $A_i$  can accept a proper prefix of  $w$ . This means that  $B$  accepts  $w$  if and only if all the  $A_i$ 's accept  $w$ . The claim follows since  $|R| = 2 + \prod_{i=1}^k (m_i - 2)$ .  $\square$

We note that the state complexity upper bound of Lemma 5 coincides with the  $(k-1)$ -fold function composition of the state complexity for the intersection of two prefix-free regular languages [13].

The result of Lemma 5 is, perhaps, not surprising because the family of prefix-free regular languages is closed under intersection. A more interesting question is whether or not the upper bound can be reached using a fixed alphabet. Note that we have shown that this is not possible for  $k$ -union in Lemma 2. On the other hand, for  $k$ -intersection, we present a positive answer using a binary alphabet. We use the prefix-free regular languages  $L_m$ , defined by language (7), that were used also in Lemma 3, and prove that with an appropriate choice of the values  $m$  the languages  $L_m$  yield a tight lower bound for  $k$ -intersection.

**Lemma 6.** *Let  $\Sigma = \{a, b\}$  and, for  $m \geq 1$ , let  $L_m \subset \{a, b\}^*$  be the prefix-free regular language defined in language (7). Let  $\{m_1, \dots, m_k\}$  be a set of relatively prime integers, where  $m_i \geq 2$  for  $1 \leq i \leq k$ . Then the minimal DFA for  $L_{m_1} \cap \cdots \cap L_{m_k}$  has*

$$2 + \prod_{i=1}^k m_i$$

states.

**Proof.** From Lemma 3, we know that the language  $L_{m_i}$ , for  $1 \leq i \leq k$ , is recognized by a DFA  $A_{m_i+2} = (Q_i, \Sigma, \delta_i, 0, \{m_i\})$  with  $m_i+2$  states,  $Q_i = \{0, 1, \dots, m_i, m_i+1\}$ .

Note that here  $m_i$  is the unique final state (denoted as  $f_i$  in the construction of Lemma 5) and  $m_i + 1$  is the sink state (denoted as  $d_i$  in the construction of Lemma 5). Following the notations of the proof of Lemma 5, let  $P_i = \{0, 1, \dots, m_i - 1\}$ .

Now let  $B = (R, \Sigma, \gamma, r_0, F)$  be the DFA for the language  $L_{m_1} \cap \dots \cap L_{m_k}$  constructed as in the proof of Lemma 5 based on the DFAs  $A_{m_i+2}$ , for  $1 \leq i \leq k$ . Thus,  $R = P_1 \times \dots \times P_k \cup \{f_B, d_B\}$ . We show that

- (i) all states of  $B$  are reachable and
- (ii) all states of  $B$  are pairwise inequivalent.

First, we show that all states are reachable. Since each  $A_{m_i+2}$  contains a cycle of  $a$ -transitions of length  $m_i$  beginning from the start state and all the  $m_i$ 's are relatively prime, the states of  $P_1 \times \dots \times P_k$  can be reached just using  $a$ -transitions. The state  $f_B$  is reached by reading a  $b$  from the start state and  $d_B$  is reached from  $f_B$  by reading either  $a$  or  $b$ .

Next, we prove that all states of the DFA  $B$  are pairwise inequivalent. The state  $d_B$  is the unique sink state and  $f_B$  is the only accepting state where all out-transitions go to  $d_B$  and, hence, these two states cannot be equivalent with any other state. Now let  $X = (x_1, x_2, \dots, x_k)$  and  $Y = (y_1, y_2, \dots, y_k)$  be arbitrary distinct states of  $P_1 \times \dots \times P_k$ , where  $0 \leq x_i, y_i \leq m_i - 1$ , for  $1 \leq i \leq k$ . Since  $X \neq Y$ , there exists an index  $h$  such that  $x_h \neq y_h$ . Since the set  $\{m_1, \dots, m_k\}$  is pairwise relatively prime, there exists  $t \in \mathbb{N}$  such that  $x_i + t \equiv 0 \pmod{m_i}$  for all  $i = 1, \dots, k$ . This means that the computation of  $A_{m_i+2}$  when started in the state  $x_i$  will be in state 0 after reading the unary string  $a^t$ . Thus, the DFA  $B$  when started in state  $X$  and reading the string  $a^t b$ , first ends in the state  $(0, \dots, 0)$  after reading the prefix  $a^t$  and then, according to rule in Eq. (10), goes to the final state  $f_B$  by reading  $b$ .

On the other hand, since  $y_h \neq x_h$ , we know that  $y_h + t \not\equiv 0 \pmod{m_h}$  and, consequently, the computation of  $B$  started in state  $Y$  does not accept after reading  $a^t b$ .

We have shown that the DFA  $B$  with  $2 + \prod_{i=1}^k m_i$  states is minimal. □

Note that the minimal DFA for  $L_{m_i}$  had  $m_i + 2$  state as illustrated in language (7). Thus for the union of prefix-free DFAs of size  $m_i$ ,  $1 \leq i \leq k$ , the construction of Lemma 6 gives a lower bound

$$2 + \prod_{i=1}^k (m_i - 2).$$

The below theorem summarizes the results from Lemmas 5 and 6.

**Theorem 2.** Let  $A_i$  be a prefix-free DFA of size  $m_i \geq 3$ ,  $i = 1, \dots, k$ . The intersection  $L(A_1) \cap \dots \cap L(A_k)$  can be recognized by a DFA of size

$$2 + \prod_{i=1}^k (m_i - 2).$$

Furthermore, there exist prefix-free DFAs as above defined over a binary input alphabet such that the minimal DFA for  $L(A_1) \cap \dots \cap L(A_k)$  needs this number of states.

### 5. Conclusion

We have examined the state complexity of two multiple operations, the  $k$ -union and  $k$ -intersection operations, for prefix-free regular languages. We have established a tight state complexity bound for  $k$ -union using an alphabet of size  $k + 1$  and a tight state complexity bound for  $k$ -intersection using a binary alphabet. Tables 1 and 2 summarize the state complexity bounds.

Table 1. State complexity of union of  $k$  prefix-free languages.

$ \Sigma $	state complexity	$k = 2$ [13]
$k + 1$	$2 \cdot \sum_{\emptyset \neq S \subseteq [1, k]} \prod_{i \in S} (m_i - 2) + \prod_{i=1}^k (m_i - 2) + 2$	$m_1 m_2 - 2$
2 (lower bound)	$\sum_{\emptyset \neq S \subseteq [1, k]} \prod_{i \in S} (m_i - 2) + \sum_{\emptyset \neq S \subseteq [1, k]} \prod_{i \in S} (m_i - 3) + \prod_{i=1}^k (m_i - 2) + 2$	

Table 2. State complexity of intersection of  $k$  prefix-free languages.

$ \Sigma $	state complexity	$k = 2$ [13]
2	$\prod_{i=1}^k (m_i - 2) + 2$	$m_1 m_2 - 2(m_1 + m_2) + 6$

Note that the state complexity for  $k$ -union is smaller than the function composition of the state complexity of several unions whereas the state complexity for  $k$ -intersection is the same as the  $(k-1)$ -fold composition of the state complexity function for the intersection of two languages. This phenomenon can be viewed to be caused by the fact that the family of (regular) prefix-free languages is closed under intersection but not closed under union. Since prefix-free languages are closed under concatenation, analogously, by extending the construction used in [13], the state complexity of  $k$ -fold concatenation of prefix-free languages can be shown to coincide with the  $k$ -fold function composition of the state complexity of the concatenation of two languages.

For  $k$ -union, additionally, we have considered the binary alphabet case and given a lower bound construction that is within the constant  $\frac{1}{2}$  from the general upper bound. We have proved that the state complexity of  $k$ -union cannot be reached when the alphabet size is less than  $k$ . This leaves open only whether or not the state complexity of  $k$ -union can be reached by prefix-free languages over an alphabet of size exactly  $k$ .

## Acknowledgments

We wish to thank the referee for the careful reading of the paper and valuable suggestions.

Eom and Han were supported by the Basic Science Research Program through NRF funded by MEST (2012R1A1A2044562) and Salomaa was supported by the Natural Sciences and Engineering Research Council of Canada Grant OGP0147224.

## References

- [1] C. Câmpeanu, K. Culik II, K. Salomaa, and S. Yu. State complexity of basic operations on finite languages. In *Proceedings of WIA '99*, Lecture Notes in Computer Science 2214, 60–70, 2001.
- [2] C. Câmpeanu, K. Salomaa, and S. Yu. Tight lower bound for the state complexity of shuffle of regular languages. *Journal of Automata, Languages and Combinatorics*, 7(3):303–310, 2002.
- [3] R. Cmorik and G. Jirásková. Basic operations on binary suffix-free languages. In *Proceeding of MEMICS'11*, Lecture Notes in Computer Science 7119, 94–102, 2011.
- [4] M. Domaratzki. State complexity of proportional removals. *Journal of Automata, Languages and Combinatorics*, 7(4):455–468, 2002.
- [5] M. Domaratzki and A. Okhotin. State complexity of power. *Theoretical Computer Science*, 410(24-25):2377–2392, 2009.
- [6] M. Domaratzki and K. Salomaa. State complexity of shuffle on trajectories. *Journal of Automata, Languages and Combinatorics*, 9(2-3):217–232, 2004.
- [7] Z. Ésik, Y. Gao, G. Liu, and S. Yu. Estimation of state complexity of combined operations. *Theoretical Computer Science*, 410(35):3272–3280, 2009.
- [8] Y. Gao and L. Kari. State complexity of star and square of union of  $k$  regular languages. *Theoretical Computer Science*, 499:38–50, 2013.
- [9] Y. Gao, L. Kari, and S. Yu. State complexity of union and intersection of square and reversal on  $k$  regular languages. *Theoretical Computer Science*, 454:164–171, 2012.
- [10] Y. Gao, K. Salomaa, and S. Yu. The state complexity of two combined operations: Star of catenation and star of reversal. *Fundamenta Informaticae*, 83(1-2):75–89, 2008.
- [11] Y.-S. Han and K. Salomaa. State complexity of union and intersection of finite languages. *International Journal of Foundations of Computer Science*, 19(3):581–595, 2008.
- [12] Y.-S. Han and K. Salomaa. State complexity of basic operations on suffix-free regular languages. *Theoretical Computer Science*, 410(27-29):2537–2548, 2009.
- [13] Y.-S. Han, K. Salomaa, and D. Wood. Operational state complexity of prefix-free regular languages. In *Automata, Formal Languages, and Related Topics - Dedicated to Ferenc Gécseg on the occasion of his 70th birthday*, 99–115. Institute of Informatics, University of Szeged, Hungary, 2009.

- [14] M. Hricko, G. Jirásková, and A. Szabari. Union and intersection of regular languages and descriptive complexity. In *Proceedings of DCFs'05*, 170–181. Università degli Studi di Milano, Milan, Italy, 2005.
- [15] J. Jirásek, G. Jirásková, and A. Szabari. State complexity of concatenation and complementation. *International Journal of Foundations of Computer Science*, 16(3):511–529, 2005.
- [16] G. Jirásková and T. Masopust. Complexity in union-free regular languages. *International Journal of Foundations of Computer Science*, 22(7):1639–1653, 2011.
- [17] G. Jirásková and P. Olejár. State complexity of intersection and union of suffix-free languages and descriptive complexity. In *Proceedings of NCMA'09*, 151–166. Austrian Computer Society, 2009.
- [18] G. Jirásková and J. Sebej. Reversal of binary regular languages. *Theoretical Computer Science*, 449:85–92, 2012.
- [19] A. Maslov. Estimates of the number of states of finite automata. *Soviet Mathematics Doklady*, 11:1373–1375, 1970.
- [20] C. Nicaud. Average state complexity of operations on unary automata. In *Proceedings of MFCS'99*, Lecture Notes in Computer Science 1672, 231–240, 1999.
- [21] G. Pighizzini and J. Shallit. Unary language operations, state complexity and Jacobsthal's function. *International Journal of Foundations of Computer Science*, 13(1):145–159, 2002.
- [22] N. Rampersad. The state complexity of  $L^2$  and  $L^k$ . *Information Processing Letters*, 98(6):231–234, 2006.
- [23] A. Salomaa, K. Salomaa, and S. Yu. State complexity of combined operations. *Theoretical Computer Science*, 383(2-3):140–152, 2007.
- [24] A. Salomaa, D. Wood, and S. Yu. On the state complexity of reversals of regular languages. *Theoretical Computer Science*, 320(2-3):315–329, 2004.
- [25] K. Salomaa and S. Yu. On the state complexity of combined operations and their estimation. *International Journal of Foundations of Computer Science*, 18:683–698, 2007.
- [26] J. Shallit. *A Second Course in Formal Languages and Automata Theory*. Cambridge University Press, New York, NY, USA, 2008.
- [27] D. Wood. *Theory of Computation*. John Wiley & Sons, Inc., New York, NY, 1987.
- [28] S. Yu. Regular languages. In G. Rozenberg and A. Salomaa, editors, *Word, Language, Grammar*, volume 1 of *Handbook of Formal Languages*, 41–110. Springer-Verlag, 1997.
- [29] S. Yu. State complexity of regular languages. *Journal of Automata, Languages and Combinatorics*, 6(2):221–234, 2001.
- [30] S. Yu, Q. Zhuang, and K. Salomaa. The state complexities of some basic operations on regular languages. *Theoretical Computer Science*, 125(2):315–328, 1994.