State Complexity of Combined Operations for Prefix-Free Regular Languages

Yo-Sub Han[1], Kai Salomaa[2], and Sheng Yu[3]

[1] Intelligence and Interaction Research Center, KIST
P.O.BOX 131, Cheongryang, Seoul, Korea
emmous@kist.re.kr

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

[3] Department of Computer Science, University of Western Ontario
London, Ontario N6A 5B7, Canada
syu@csd.uwo.ca

Abstract. We investigate the state complexity of combined operations for prefix-free regular languages. Prefix-free minimal deterministic finite-state automata have a unique structural property that plays an important role to obtain the precise state complexity of basic operations. Based on the same property, we establish the precise state complexity of four combined operations: star-of-union, star-of-intersection, star-of-reversal and star-of-catenation.

1 Introduction

Regular languages are widely used in many applications such as text searching, speech processing or software engineering [1,2,3]. Given a regular language L, researchers often use the number of states in the minimal DFA for L to represent the complexity of L. Based on this notation, we, then, define the state complexity of an operation for regular languages to be the number of states that are necessary and sufficient in the worst-case for the minimal DFA that accepts the language obtained from the operation [4]. The state complexity of an operation is calculated based on the the structural properties of given regular languages and the function of a given operation. Recently, due to large amount of memory and fast CPUs, many applications using regular languages require huge size of finite-state automata (FAs). This makes the estimated upper bound of the state complexity useful in practice since it is directly related to the efficient resource management in applications. Moreover, it is a challenging quest to verify whether or not an estimated upper bound can be reached.

Yu [5] gave a comprehensive survey of the state complexity of regular languages. Salomaa et al. [6] studied classes of languages for which the reversal operation reaches the exponential upper bound. As special cases of the state complexity, researchers examined the state complexity of finite languages [7,8], the state complexity of unary language operations [9] and the nondeterministic descriptional complexity of regular languages [10]. For regular language codes, Han et al. [11] studied the state complexity of prefix-free regular languages. They tackled the problem based on the structural property of prefix-free DFAs: A prefix-free DFA must be non-exiting assuming all states are useful [11]. Similarly, based on suffix-freeness, Han and Salomaa [12] looked at the state complexity of suffix-free regular languages. There are several other results with respect to the state complexity of different operations [13,14,15,16,17,18].

While people mainly looked at the state complexity of single operations (union, intersection, catenation and so on), Yu and his co-authors [19,20,21] recently started investigating the state complexity of combined operations (star-of-union, star-of-intersection and so on). They showed that the state complexity of a combined operation is usually not equal to the composition of the state complexities of the participating individual operations. On the other hand, they also observed that in a few cases, the state complexity of a combined operation is very close to the composition of the state complexities. This leads us to study the state complexity of combined operations and examine the cases that give a similar state complexity to the composition of the state complexities of single operations.

We choose prefix-free regular languages for the state complexity of combined operations. Note that state complexity of prefix-free regular languages is very different from the state complexity of regular languages because prefix-freeness gives a unique structural property in a prefix-free minimal DFA [11,22]. Moreover, prefix-free languages are used in many coding theory applications (Huffman coding is an example), and for this reason results on state complexity of combined operations for prefix-free regular languages may be useful. Furthermore, determining the state complexity of combined operations on fundamental subfamilies of the regular languages can provide valuable insights on connections between restrictions placed on language definitions and descriptional complexity.

In Section 2, we define some basic notions. Then, we present the state complexities of four combined operations in Section 3. We compare the state complexity of basic operations and the state complexity of combined operations for prefix-free regular languages, and conclude the paper in Section 4.

2 Preliminaries

Let Σ denote a finite alphabet of characters and Σ^* denote the set of all strings over Σ. The size $|\Sigma|$ of Σ is the number of characters in Σ. A language over Σ is any subset of Σ^*. The symbol $\emptyset$ denotes the empty language and the symbol λ denotes the null string. For strings x, y and z, we say that x is a *prefix* of y if $y = xz$. We define a (regular) language L to be prefix-free if for any two distinct strings x and y in L, x is not a prefix of y. Given a string x in a set X of strings, let x^R be the reversal of x, in which case $X^R = \{x^R \mid x \in X\}$.

An FA A is specified by a tuple $(Q, \Sigma, \delta, s, F)$, where Q is a finite set of states, Σ is an input alphabet, $\delta : Q \times \Sigma \to 2^Q$ is a transition function, $s \in Q$ is the start state and $F \subseteq Q$ is a set of final states. If F consists of a single state f, then we use f instead of $\{f\}$ for simplicity. 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. 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. The size $|A|$ of A is $|Q|$. 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 Σ is accepted by A if there is a labeled path from s to a final state such that this path reads x. We call this path an *accepting path*. Then, the language $L(A)$ of A is the set of all strings spelled out by accepting paths in A. We say that a state of A is *useful* if it appears in an accepting path in A; otherwise, it is *useless*. Unless otherwise mentioned, in the following we assume that all states are useful.

A regular expression E is prefix-free if $L(E)$ is prefix-free and an FA A is prefix-free if $L(A)$ is prefix-free. Moreover, if $L(A)$ is prefix-free, then A must be non-exiting. We recall that an arbitrary minimal DFA recognizing a prefix-free language has exactly one final state and all of its out-transitions go to the sink state [11].

For complete background knowledge in automata theory, the reader may refer to the textbook [23].

3 State Complexity of Combined Operations

We consider four combined operations of prefix-free regular languages: star-of-union, star-of-reversal, star-of-catenation and star-of-intersection. For each operation, we compare the state complexity of a combined operation and the composition of the state complexities of two individual operations. In the following section, let $\mathcal{SC}(L)$ denote the state complexity of L.

3.1 Star of Union

First we give an upper bound construction for the state complexity of star-of-union of two prefix-free languages. Let $A_i = (Q_i, \Sigma, \delta_i, q_{0,i}, f_i)$, $|Q_i| = m_i$, $i = 1, 2$ be arbitrary minimal DFAs recognizing prefix-free languages. Here $f_i \in Q_i$ is the unique final state and we can assume that $f_i \neq q_{i,0}$. (If the final state is the start state, A_i must recognize $\{\lambda\}$.) We denote by $d_i \in Q_i$ the sink state of A_i, $Q'_i = Q_i \setminus \{f_i, d_i\}$ is the set of states of A_i excluding the final state and the sink state. Without loss of generality we assume that $Q_1 \cap Q_2 = \emptyset$.

We construct a DFA

$$A = (Q, \Sigma, \delta, \{q_0, q_{0,1}, q_{0,2}\}, F) \tag{1}$$

for the language $(L(A_1) \cup L(A_2))^*$. We choose Q to be the collection of subsets of $P \subseteq \{q_0\} \cup Q'_1 \cup Q'_2$ such that

$$\text{if } q_0 \in P, \text{ then } q_{0,1}, q_{0,2} \in P. \tag{2}$$

The set of final states F consists of all elements of Q that contain q_0 and the transition function δ is defined as follows. Let $P = X \cup P_1 \cup P_2$, where $P_i \subseteq Q'_i$, $i = 1, 2$, X is $\{q_0\}$ or $\emptyset$, and $c \in \Sigma$. Then we define:

$$\delta(P, c) = \begin{cases} \delta_1(P_1, c) \cup \delta_2(P_2, c), \text{ if } f_1 \notin \delta_1(P_1, c) \text{ and } f_2 \notin \delta_2(P_2, c), \\ \delta_1(P_1, c) \cup \delta_2(P_2, c) \cup \{q_{0,1}, q_{0,2}, q_0\}, \text{ otherwise.} \end{cases}$$

The transitions defined above clearly preserve the property (2), that is, for any $P \in Q$, $\delta(P, c) \in Q$.

The construction of A can be viewed as constructing an NFA for the star-of-union of the original languages, and performing a subset construction on the NFA. In the subset construction, we can merge the final states of the original DFAs. The construction is illustrated in Fig. 1.

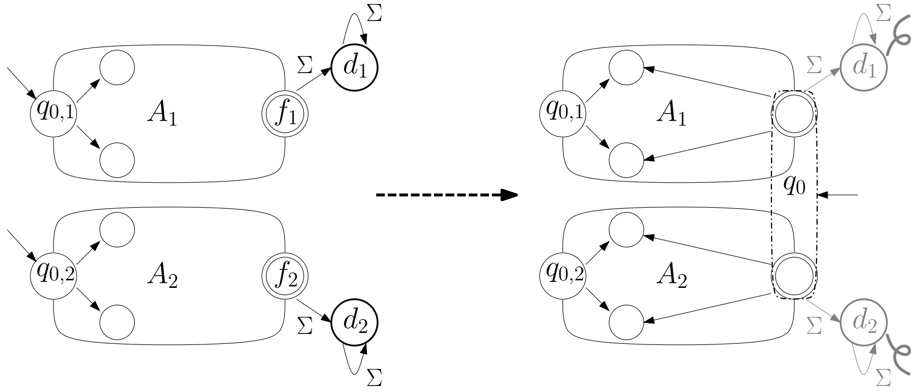

Fig. 1. Construction of an NFA for the star-of-union. Note that the merged state q_0, which is a final state as well, is the start state.

The DFA A uses the symbol q_0 to represent the merging of the two final states f_1 and f_2 of the original DFAs. The start state of A is $\{q_0, q_{0,1}, q_{0,2}\}$ which means that the computation begins by simulating both the computation of A_1 and the computation of A_2. Whenever one of the simulated computations enters the final state, the computation of A adds both $q_{0,1}$ and $q_{0,2}$ to the current set of states and begins new computations simulating A_1 and A_2. Namely, q_0 in the current set indicates that the previously simulated computation step is accepted either in A_1 or in A_2. Note that the presence of $q_{0,1}$ or $q_{0,2}$ in the current state of A is not sufficient to guarantee this property. The choice of the final states guarantees that A recognizes exactly the language $(L(A_1) \cup L(A_2))^*$.

Assuming, $q_{0,i}$, f_i, and d_i are all distinct, for $i = 1, 2$, the set Q of states defined by (2) contains $2^{m_1+m_2-4}$ subsets of $Q'_1 \cup Q'_2$ that do not contain the merged final state q_0. Additionally, Q contains $2^{m_1+m_2-6}$ subsets of the form $\{q_0, q_{0,1}, q_{0,2}\} \cup P$ where $P \subseteq (Q_1 \setminus \{q_{0,1}, f_1, s_1\}) \cup (Q_2 \setminus \{q_{0,2}, f_2, s_2\})$. Note that if $q_{0,i}$, f_i and d_i, $i \in \{1, 2\}$, are not distinct, then A_i recognizes one of the trivial languages $\{\lambda\}$ or $\emptyset$.

Now we obtain the following upper bound for the state complexity of star-of-union of prefix-free regular languages from the construction.

Lemma 1. *Let L_i be a prefix-free regular language with $\mathcal{SC}(L_i) = m_i$, $m_i \geq 3$, $i = 1, 2$. Then*

$$\mathcal{SC}((L_1 \cup L_2)^*) \leq 5 \cdot 2^{m_1+m_2-6}.$$

In the following, we give a worst-case construction that reaches the upper bound of Lemma 1. Let $\Sigma = \{a, b, c, d, e\}$ and $m, n \geq 3$. We define

$$A_1 = (R, \Sigma, \delta_1, r_0, \{r_{m-2}\}), \tag{3}$$

where $R = \{r_0, \ldots, r_{m-1}\}$, and the transitions of δ_1 are defined as:

- $\delta_1(r_i, a) = r_{i+1}$, $i = 0, \ldots, m-4$, $\delta_1(r_{m-3}, a) = r_0$.
- $\delta_1(r_i, b) = \delta_1(r_i, d) = r_i$, $i = 0, \ldots, m-3$.
- $\delta_1(r_0, c) = r_{m-2}$, $\delta_1(r_i, c) = r_i$, $i = 1, \ldots, m-3$.

The state r_{m-1} is the sink state of A_1 and above all undefined transitions go to r_{m-1}. In particular, note that all transitions of A_1 on input symbol e go to the sink state. The language $L(A_1)$ is prefix-free since all out-transitions from the final state r_{m-2} go to the sink state.

We define the second DFA as

$$A_2 = (S, \Sigma, \delta_2, s_0, \{s_{n-2}\}), \tag{4}$$

where $S = \{s_0, \ldots, s_{n-1}\}$, and δ_2 is defined by setting:

- $\delta_2(s_i, b) = s_{i+1}$, $i = 0, \ldots, n-4$, $\delta_2(s_{n-3}, b) = s_0$.
- $\delta_2(s_i, a) = \delta_2(s_i, e) = s_i$, $i = 0, \ldots, n-3$.
- $\delta_2(s_0, c) = s_{n-2}$, $\delta_2(s_i, c) = s_i$, $i = 1, \ldots, n-3$.

Again, s_{n-1} is the sink state of A_2 and all above undefined transitions go to the sink state. In particular, any state of A_2 transitions with input symbol d to the sink state.

The DFAs A_1 and A_2 are depicted in Fig. 2.

We show that the DFAs in Fig. 2 reach the upper bound of Lemma 1 when $m, n \geq 3$. Note that any complete DFA recognizing a prefix-free language that is not $\{\lambda\}$ or $\emptyset$ has to have at least three states.

Lemma 2. *Let $m, n \geq 3$ and let A_1 and A_2 be DFAs as defined in (3) and (4), respectively.*

We claim that $\mathcal{SC}((L(A_1) \cup L(A_2))^*)$ is $5 \cdot 2^{m+n-6}$.

By Lemmas 1 and 2 we get a precise bound for the state complexity of star-of-union of prefix-free regular languages.

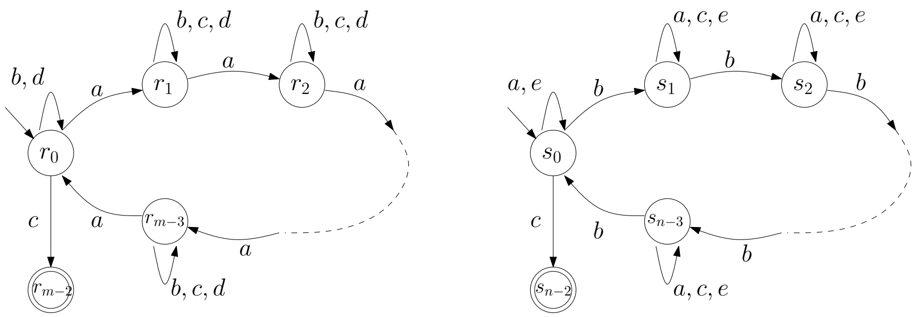

Fig. 2. The DFAs A_1 and A_2. The figure does not show the sink states r_{m-1} and s_{n-1} and their in-transitions.

Theorem 1. *The worst-case state complexity of the star-of-union of an m_1-state and an m_2-state, $m_1, m_2 \geq 3$, prefix-free regular languages is precisely $5 \cdot 2^{m_1+m_2-6}$, where $|\Sigma| \geq 5$.*

Theorem 1 implies that the upper bound can be reached for all values $m_1, m_2 \geq 3$. If, for example, $m_2 = 2$, the state complexity is m_1. The result follows from the state complexity of star for prefix-free languages [11] because with $m_2 = 2$, the worst-case example corresponds to the case where $L(A_2) = \{\lambda\}$ and $(L(A_1) \cup L(A_2))^* = L(A_1)^*$.

Note that the state complexity of the union of two prefix-free regular languages is $m_1 m_2 - 2$ [11] and the state complexity of the star of an m-state regular language is $3 \cdot 2^{m-2}$ [4]. Thus, the composition of two complexities is $3 \cdot 2^{m_1 m_2 - 4}$. Therefore, the state complexity of the star-of-union is much lower (one has a linear exponent and the other has a quadratic exponent) than the composition of two complexities.

The lower bound construction of Lemma 2 uses an alphabet of size five. It remains an open question whether the worst-case bound can be reached by prefix-free regular languages over alphabets of size 2, 3 or 4.

3.2 Star of Reversal

Let A be a minimal DFA for a regular language and $|A| = m$. Since the state complexity of $L(A)$ is m, the reversal $L(A)^R$ of $L(A)$ can be accepted by an NFA A^R with m states, where A^R can have multiple start states. We, then, use the subset construction on A^R and the resulting DFA can have at most 2^m states. Thus, the upper bound for the reversal of regular languages is 2^m. Leiss [24] demonstrated that some classes of languages can reach the upper bound. Later, Salomaa et al. [6] showed the conditions for such regular languages and obtained the following result.

Proposition 1 (Salomaa et al. [6]). *Let Σ be an alphabet with at least 2 characters. There exists a minimal DFA A that has a maximal blow-up, 2^m, in the transition to its reversal $L(A)^R$, where the transition function of A is functionally complete, $L(A) \neq \emptyset, \Sigma^*$ and $m = |A|$.*

Given a minimal prefix-free DFA $A = (Q, \Sigma, \delta, s, f)$, we can obtain an FA for $L((A)^R)^*$ as follows: We first flip all transition directions in A such that f is the start state and s is the final state. Then, the resulting FA $A^R = (Q, \Sigma, \delta^R, f, s)$ is the reversal of A; $L(A)^R = L(A^R)$. The sink state d of A is now unreachable from f in A^R and, thus, we remove it. Next, we add all out-transitions of f to s and make f to be final in A^R. Note that we keep original out-transition of s in A^R as well. Therefore, we have an FA $A' = (Q \setminus \{d\}, \Sigma, \delta', f, \{s, f\})$, where

$$\delta'(q,a) = \begin{cases} \delta^R(q,a) \text{ if } q \neq s, \\ \delta^R(q,a) \cup \delta^R(f,a) \text{ otherwise.} \end{cases}$$

It is clear from the construction that $L(A') = L((A)^R)^*$. Note that A' is not necessarily deterministic since we have flipped transition directions. Fig. 3 illustrates this construction.

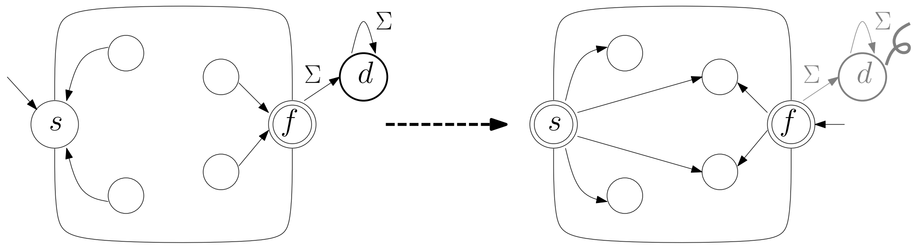

Fig. 3. Construction of an NFA for the star-of-reversal

Lemma 3. *Let L be a prefix-free regular language with $\mathcal{SC}(L) = m$. Then,*

$$\mathcal{SC}((L^R)^*) \leq 2^{m-2} + 1.$$

Next, we demonstrate that $2^{m-2} + 1$ states are necessary for the star-of-reversal of an m-state prefix-free regular language L. Given a (regular) language L over Σ, $L\#$ is prefix-free if the character $\#$ is not in Σ. Our approach is an extension of the method used by Han et al. [11] for computing the reversal of prefix-free minimal DFAs.

Let $A = (Q, \Sigma, \delta, s, F)$ be a minimal DFA in Proposition 1 over Σ, which is not prefix-free in general. We construct a prefix-free minimal DFA $A_\# = (Q', \Sigma \cup \{\#\}, \delta', s, f')$ that requires m states as follows:

$Q' = Q \cup \{d, f'\}$,
for $q \in Q$ and $a \in \Sigma \cup \{\#\}$,

$$\delta'(q,a) = \begin{cases} \delta(q,a) & \text{if } q \in Q \text{ and } a \neq \#, \\ f' & \text{if } q \in F \text{ and } a = \#, \\ d & \text{otherwise.} \end{cases}$$

Namely, we introduce a new final state f' and connect all states in F of A to f' with label $\#$. We also introduce a sink state d. Since A is functionally complete, that is, the transition function of A consists of all mappings from Q to Q, A cannot have a sink state, and consequently, d is not equivalent with any of the states of A. Note that by the construction, $A_\#$ is deterministic and minimal. Furthermore, $L(A_\#)$ is prefix-free. Thus, if A has $m-2$ states, then $A_\#$ has m states.

Proposition 2 (Han et al. [11]). *Given a prefix-free minimal DFA $A_\#$ as constructed above, $2^{m-2}+1$ states are necessary for the minimal DFA of $L(A_\#)^R$, where $m = |A_\#|$ and $\# \notin \Sigma$.*

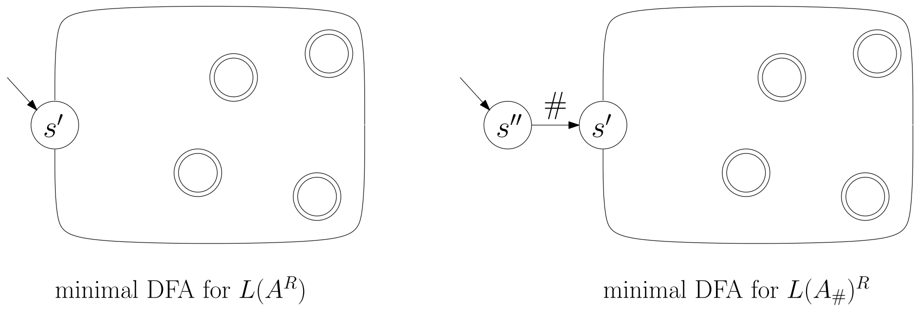

Fig. 4. Construction of the minimal DFA for $L(A_\#)^R$ from the minimal DFA for $L(A^R)$. Note that the left DFA is defined over Σ and the right DFA is defined over $\Sigma \cup \{\#\}$.

The brief sketch of Proposition 2 is that we flip the transition directions of A and apply the subset construction on A^R. Let s' be the start state of the resulting DFA. Then, we introduce a new start state s'' and connect s'' to s' with label $\#$. Then, the new FA is the minimal DFA for $L(A_\#)^R$ and the number of states is $2^{m-2}+1$ if $|A| = m-2$. See Fig. 4.

We are ready to compute the star-of-reversal for $A_\#$. From the minimal DFA $M = (Q, \Sigma, \delta, s'', F)$ for $L(A_\#)^R$ as depicted in Fig. 4, we add $\#$-transitions from all final states in F to s', which is the only target state of s'' except for the sink state d. We also make s'' to be a final state. Let $M_\#$ be the resulting FA. It is easy to verify that $L(M_\#) = (L(A_\#)^R)^*$. We only need to show that $M_\#$ is a minimal DFA. Since $\delta(f, \#) = d$ for $f \in F$ in M, $M_\#$ is deterministic.

We show that all states of $M_\#$ are pairwise inequivalent. First consider two nonfinal states of $M_\#$, r_1 and r_2. Now since r_1 and r_2 are inequivalent as states of M, there exists $w \in \Sigma^*$ such that $\delta(r_1, w) \in F$ and $\delta(r_2, w) \notin F$. Note that since r_1 and r_2 are distinct from s'', we can assume that w does not contain occurrences of $\#$, and the same string w distinguishes r_1 and r_2 as states of $M_\#$.

Next, consider two final states of $M_\#$. If they are both also final states in M, then we can use the same argument as above. It remains to show that s'' is not equivalent with any other final state r of $M_\#$. We note that since the original DFA A is functionally complete, r must have an out-transition on a symbol of Σ to some state of Q. On the other hand, the only out-transition from s'' (either in M or $M_\#$) whose target state is not the sink state d is on the input symbol $\#$. This means that s'' and r are inequivalent. Therefore, $M_\#$ is minimal.

Note that, from M, we have constructed the minimal DFA for the star-of-reversal of $A_{\#}$ without adding new states. This implies that we get a precise bound for the state complexity of star-or-reversal of prefix-free regular languages.

Theorem 2. *The worst-case state complexity of the star-of-reversal of an m-state prefix-free regular language is precisely $2^{m-2}+1$, where $|\Sigma| \geq 3$.*

Note that the state complexity of the reversal of an m-state prefix-free regular language is $2^{m-2}+1$ [11] and the state complexity of the star of an m-state suffix-free regular language is $2^{m-2}+1$ [12]. Thus, the composition of two complexities is $2^{2^{m-2}+1-2}+1$. Therefore, the state complexity of the star-of-reversal is much lower than the composition of two complexities.

3.3 Star of Catenation and Intersection

We examine two operations, star-of-catenation and star-of intersection, based on the following observations:

1. Prefix-freeness is preserved by catenation and intersection.
2. Given an m-state prefix-free DFA A, $\mathcal{SC}(L(A)^*) = m$ [11].

Theorem 3. *The worst-case state complexity of the star-of-catenation of an m_1-state and an m_2-state, $m_1, m_2 \geq 3$, prefix-free regular languages is precisely $m_1 + m_2 - 2$.*

Theorem 3 shows that the state complexity of the star-of-catenation is equal to the state complexity calculated by the composition of state complexities of star and catenation. This is due to the fact that prefix-freeness is closed under catenation. Since prefix-freeness is also closed under intersection, we expect to see a similar case for the state complexity of the star-of-intersection.

Theorem 4. *The worst-case state complexity of the star-of-intersection of an m_1-state and an m_2-state, $m_1, m_2 \geq 3$, prefix-free regular languages is precisely $m_1 m_2 - 2(m_1 + m_2) + 6$, where $|\Sigma| \geq 4$.*

Proof. Han et al. [11] gave a construction for the intersection of two prefix-free DFAs based on the Cartesian product of states. The main idea of the construction is to remove unreachable states and merge equivalent states that are identified based on the structural properties of prefix-free DFAs. Based on the construction, they showed that $m_1 m_2 - 2(m_1+m_2)+6$ states are sufficient for the intersection of an m_1-state prefix-free minimal DFA and an m_2-state prefix-free minimal DFA. Since prefix-freeness is closed under intersection, the resulting minimal DFA is also prefix-free. Therefore, $m_1 m_2 - 2(m_1 + m_2) + 6$ states are sufficient for the star-of-intersection since the state complexity of the Kleene star is m for an m-state prefix-free minimal DFA [11].

We only need to prove the necessary part. Assume that $\Sigma = \{a, b, c, d\}$. Given a string w over Σ, let $|w|_a$ denote the number of a's in w. Let A_1 be the minimal DFA for

$$L(A_1) = \{wc \mid |w|_a \equiv 0 \pmod{m_1 - 2}, \text{ for } w \in \{a, b\}^*\}$$

and A_2 be the minimal DFA for

$$L(A_2) = \{wc \mid |w|_b \equiv 0 \pmod{m_2 - 2},\ \text{for } w \in \{a,b\}^*\}.$$

It is easy to verify that $L(A_1)$ and $L(A_2)$ are prefix-free and $|A_1| = m$ and $|A_2| = n$. Let $L = (L(A_1) \cap L(A_2))^*$. We claim that the minimal DFA for L needs $mn - 2(m+n) + 6$ states. To prove the claim, it is sufficient to show that there exists a set R of $mn - 2(m+n) + 6$ strings over Σ that are pairwise inequivalent modulo the right invariant congruence of L, $\equiv_L$.

Let $R = R_1 \cup R_2$, where

$R_1 = \{\lambda, d\}$,
$R_2 = \{a^i b^j \mid 1 \leq i \leq m_1 - 2 \text{ and } 1 \leq j \leq m_2 - 2\}$.

Any string $a^i b^j$ from R_2 cannot be equivalent with λ since $a^i b^j \cdot \lambda \notin L$ but $\lambda \cdot \lambda \in L$. Similarly, $a^i b^j$ cannot be equivalent with d since $a^i b^j \cdot a^{m_1-2-i} b^{m_2-2-j} c \in L$ but $d \cdot a^{m_1-2-i} b^{m_2-2-j} c \notin L$. Note that λ and d are not equivalent as well.

Next, consider two distinct strings $a^i b^j$ and $a^k b^l$ from R_2. Since $a^i b^j \neq a^k b^l$, $a^i b^j \cdot a^{m_1-2-i} b^{m_2-2-j} c \in L$ but $a^k b^l \cdot a^{m_1-2-i} b^{m_2-2-j} c \notin L$. Therefore, any two distinct strings from R_2 are pairwise inequivalent.

Therefore, all $mn - 2(m+n) + 6$ strings in R are pairwise inequivalent. This concludes the proof. □

4 Conclusions

Recently, researchers started looking at state complexity of combined operations [19,25,26]. Usually, we can obtain a much lower state complexity for combined operations compared with the composition of state complexities of individual operations. However, in a few cases, the state complexity of combined operations and the composition of state complexities are similar. We have examined prefix-free regular languages and computed the state complexity of combined operations.

operation	complexity	operation	complexity
$L_1 \cup L_2$	$m_1 m_2 - 2$	$(L_1 \cup L_2)^*$	$5 \cdot 2^{m_1+m_2-6}$
$L_1 \cap L_2$	$m_1 m_2 - 2(m_1 + m_2) + 6$	$(L_1 \cap L_2)^*$	$m_1 m_2 - 2(m_1 + m_2) + 6$
$L_1 \cdot L_2$	$m_1 + m_2 - 2$	$(L_1 \cdot L_2)^*$	$m_1 + m_2 - 2$
L_1^R	$2^{m_1-2} + 1$	$(L_1^R)^*$	$2^{m_1-2} + 1$
L_1^*	m_1	$(L_1^*)^* = L_1^*$	m_1

The table summaries the state complexity of basic operations and the state complexity for combined operations of prefix-free regular languages. Note that prefix-freeness is closed under both intersection and catenation and both operations have the same state complexity. On the other hand, for union, the state complexity of the star-of-union is much lower. An interesting case is the reversal case. We know that prefix-freeness is not closed under reversal since the reversal of a prefix-free language is suffix-free. However, the complexities of single and combined operations are the same. We think this is because suffix-free minimal

DFAs preserve a certain structural property. Therefore, natural future work is to examine the state complexity of combined operations for suffix-free regular languages.

References

1. Friedl, J.E.F.: Mastering Regular Expressions. O'Reilly & Associates, Inc., Sebastopol (2002)
2. Harel, D., Politi, M.: Modeling Reactive Systems with Statecharts: The Statemate Approach. McGraw-Hill, New York (1998)
3. Karttunen, L.: Applications of finite-state transducers in natural language processing. In: Yu, S., Păun, A. (eds.) CIAA 2000. LNCS, vol. 2088, pp. 34–46. Springer, Heidelberg (2001)
4. Yu, S., Zhuang, Q., Salomaa, K.: The state complexities of some basic operations on regular languages. Theoretical Computer Science 125(2), 315–328 (1994)
5. Yu, S.: State complexity of regular languages. Journal of Automata, Languages and Combinatorics 6(2), 221–234 (2001)
6. Salomaa, A., Wood, D., Yu, S.: On the state complexity of reversals of regular languages. Theoretical Computer Science 320(2-3), 315–329 (2004)
7. Câmpeanu, C., Culik II, 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)
8. Han, Y.S., Salomaa, K.: State complexity of union and intersection of finite languages. International Journal of Foundations of Computer Science 19(3), 581–595 (2008)
9. Pighizzini, G., Shallit, J.: Unary language operations, state complexity and Jacobsthal's function. International Journal of Foundations of Computer Science 13(1), 145–159 (2002)
10. Holzer, M., Kutrib, M.: Nondeterministic descriptional complexity of regular languages. International Journal of Foundations of Computer Science 14(6), 1087–1102 (2003)
11. Han, Y.S., Salomaa, K., Wood, D.: State complexity of prefix-free regular languages. In: Proceedings of DCFS 2006, pp. 165–176 (2006); Full version is submitted for publication
12. Han, Y.S., Salomaa, K.: State complexity of basic operations on suffix-free regular languages. In: Kučera, L., Kučera, A. (eds.) MFCS 2007. LNCS, vol. 4708, pp. 501–512. Springer, Heidelberg (2007)
13. Câmpeanu, C., Salomaa, K., Yu, S.: Tight lower bound for the state complexity of shuffle of regular languages. Journal of Automata, Languages and Combinatorics 7(3), 303–310 (2002)
14. Domaratzki, M.: State complexity of proportional removals. Journal of Automata, Languages and Combinatorics 7(4), 455–468 (2002)
15. Domaratzki, M., Salomaa, K.: State complexity of shuffle on trajectories. Journal of Automata, Languages and Combinatorics 9(2-3), 217–232 (2004)
16. Hricko, M., Jirásková, G., Szabari, A.: Union and intersection of regular languages and descriptional complexity. In: Proceedings of DCFS 2005, pp. 170–181 (2005)
17. Jirásek, J., Jirásková, G., Szabari, A.: State complexity of concatenation and complementation of regular languages. In: Domaratzki, M., Okhotin, A., Salomaa, K., Yu, S. (eds.) CIAA 2004. LNCS, vol. 3317, pp. 178–189. Springer, Heidelberg (2005)

18. Nicaud, C.: Average state complexity of operations on unary automata. In: Kutyłowski, M., Wierzbicki, T., Pacholski, L. (eds.) MFCS 1999. LNCS, vol. 1672, pp. 231–240. Springer, Heidelberg (1999)
19. Gao, Y., Salomaa, K., Yu, S.: The state complexity of two combined operations: Star of catenation and star of reversal. Fundamenta Informaticae 83(1-2), 75–89 (2008)
20. Salomaa, A., Salomaa, K., Yu, S.: State complexity of combined operations. Theoretical Computer Science 383(2-3), 140–152 (2007)
21. Salomaa, K., Yu, S.: On the state complexity of combined operations and their estimation. International Journal of Foundations of Computer Science 18, 683–698 (2007)
22. Berstel, J., Perrin, D.: Theory of Codes. Academic Press, Inc., London (1985)
23. Wood, D.: Theory of Computation. John Wiley & Sons, Inc., New York (1987)
24. Leiss, E.L.: Succinct representation of regular languages by boolean automata. Theoretical Computer Science 13, 323–330 (1981)
25. Ellul, K., Krawetz, B., Shallit, J., Wang, M.W.: Regular expressions: New results and open problems. Journal of Automata, Languages and Combinatorics 9, 233–256 (2004)
26. Rampersad, N.: The state complexity of L^2 and L^k. Information Processing Letters 98(6), 231–234 (2006)