State Complexity of Basic Operations on Suffix-Free Regular Languages

Yo-Sub Han[1,⋆] and Kai Salomaa[2,⋆⋆]

[1] Intelligence and Interaction Research Center,
Korea Institute of Science and Technology
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

Abstract. We investigate the state complexity of basic operations for suffix-free regular languages. The state complexity of an operation for regular languages is the number of states that are necessary and sufficient in the worst-case for the minimal deterministic finite-state automaton that accepts the language obtained from the operation. We establish the precise state complexity of catenation, Kleene star, reversal and the Boolean operations for suffix-free regular languages.

1 Introduction

Codes are useful in many areas such as information processing, data compression, cryptography and information transmission [16]. Some of well-known codes are prefix codes, suffix codes, bifix codes and infix codes. People use different codes for different applications based on the characteristic of each code [1,16]. Since codes are sets of strings over an alphabet, they are closely related to formal languages: a code is a language. Thus, the condition defining a class of codes defines a corresponding subfamily of each language family. For regular languages, for example, suffix-freeness defines suffix-free regular languages, which constitute a subfamily of regular languages.

There are different ways to define the complexity of a regular language L. One classical definition is the total number of states in the minimal deterministic finite-state automaton (DFA) for L since the minimal DFA for L is unique (up to isomorphism) [13,21]. Based on this definition, Yu and his co-authors [24] defined 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. Yu [23] gave a comprehensive survey of the state complexity of regular languages. Salomaa et al. [20] studied classes of languages for which the reversal operation reaches the

⋆ Han was supported by the KIST Research Grants 2E20050 and 2Z03050.
⋆⋆ Salomaa was supported by the Natural Sciences and Engineering Research Council of Canada Grant OGP0147224.

L. Kučera and A. Kučera (Eds.): MFCS 2007, LNCS 4708, pp. 501–512, 2007.

exponential upper bound. As special cases of the state complexity, researchers examined the state complexity of finite languages [3,8], the state complexity of unary language operations [19] and the nondeterministic descriptional complexity of regular languages [11,12]. There are several other results with respect to the state complexity of different operations [4,5,6,14,15,18].

Recently, Han et al. [9] examined 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 [10]. It turns out that the state complexity for the prefix-free case is strictly less than the corresponding state complexity for regular languages over some basic operations. We know that if a language L is prefix-free, then its reversal L^R is suffix-free by definition. Moreover, if L is regular and non-empty, then the start state of a DFA for L^R should not have any in-transitions. However, this condition is necessary but not sufficient. Due to this fact, the state complexity of suffix-free regular languages is not symmetric to the prefix-free case. This leads us to investigate the state complexity of basic operations on suffix-free regular languages. Interestingly, the results for catenation and Kleene star turn out to be of a totally different order than in the case of prefix-free regular languages.

In Section 2, we define some basic notions. In Section 3, we examine the state complexity of Kleene star and reversal of suffix-free regular languages. We then look at the catenation of two suffix-free minimal DFAs in Section 4. Next, we investigate the state complexity of intersection and union of suffix-free regular languages based on the Cartesian product of states in Section 5. We present the comparison table of the state complexity on different types of regular languages and conclude the paper in Section 6.

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^*. Given a set X, 2^X denotes the power set of X. For a string $x \in \Sigma^*$ and a character a, $|x|_a$ denotes the number of symbol a occurrences in x. We say that a string x is a *suffix* of a string y if $y = ux$ for some string u. We define a set X of strings to be a *suffix-free set* if a string from X is not a suffix of any other string in X. Given a string x from a set X, let x^R be the reversal of x, in which case $X^R = \{x^R \mid x \in X\}$.

The symbol $\emptyset$ denotes the empty language and the character λ denotes the null string. A finite-state automaton (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, we use f instead of $\{f\}$ for simplicity. $|Q|$ denotes the number of states in Q. We define a state d to be a *sink state* if d is reachable from s of A and, for any $a \in \Sigma$, $\delta(d, a) = d$ and $d \notin F$. Since all sink states are always equivalent, we can assume that A has a unique sink state. 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. The transition function δ can be extended to a function $Q \times \Sigma^* \to 2^Q$ that reflects sequences of inputs. A string x over Σ is accepted by A if there is a labeled path from s to a state in F such that this path spells out the string x. Namely, $\delta(s, x) \cap F \neq \emptyset$. The language $L(A)$ of an FA A is the set of all strings that are spelled out by paths from s to a final state in F. 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 every final state in A go to the sink state.

Given an FA $A = (Q, \Sigma, \delta, s, F)$, we define the *right language* L_q of a state q to be the set of strings that are spelled out by some path from q to a final state in A; namely, L_q is the language accepted by the FA obtained from A by changing the start state to q. We say that two states p and q are *equivalent* if $L_p = L_q$.

We define an FA A to be a DFA if the number of target states for each pair of a state q and a character $a \in \Sigma$ is one: namely, $|\delta(q, a)| = 1$. Given a DFA A, we assume that A is complete; namely, each state has $|\Sigma|$ out-transitions. If A has m states, then we say that A is an m-state DFA.

We define a (regular) language L to be suffix-free if L is a suffix-free set. A regular expression E is suffix-free if $L(E)$ is suffix-free. Similarly, an FA A is suffix-free if $L(A)$ is suffix-free. Moreover, if $L(A)$ is suffix-free and non-empty, then A must be non-returning. Similarly, we can define prefix-free regular expressions and languages. Note that if a language L is suffix-free, then L^R is prefix-free.

For complete background knowledge in automata theory, the reader may refer to textbooks [13,21].

Due to the limit on the number of pages, we omit all the proofs in the following sections. The proofs can be found in a full version [7].

3 Kleene Star and Reversal

Before examining the state complexity of various operations, we establish that any suffix-free (complete) DFA must always have a sink state. Recall that the state complexity of a regular language L is the number of states in its minimal DFA. If L is a regular language, its minimal DFA does not necessarily have have a sink state. However, if L is prefix-free, then its minimal DFA A must have a sink state since A is non-exiting. Therefore, we have to verify the existence of the sink state in a suffix-free minimal DFA before investigating the state complexity for each operation. This is crucial for computing the correct state complexity.

Lemma 1. *Let $A = (Q, \Sigma, \delta, s, F)$ be a minimal DFA for a suffix-free language and $k = |Q|$. Then, A has a sink state $d \in Q$ and for every string $w \in \Sigma^+$, $\delta(s, w^k) = d$.*

Lemma 1 shows that we must always consider the sink state for computing the state complexity of suffix-free regular languages. From now, we assume that a suffix-free minimal DFA has the unique sink state.

3.1 Kleene Star of Suffix-Free Regular languages

We first start with the Kleene star operation.

Lemma 2. *Given an m-state suffix-free minimal DFA $A = (Q, \Sigma, \delta, s, F)$, $2^{m-2}+1$ states are sufficient for $L(A)^*$.*

We now define a DFA A such that $L(A)$ is suffix-free and the state complexity of $L(A)^*$ reaches the upper bound in Lemma 2. Let $A = (Q, \Sigma, \delta, s, F)$, where $Q = \{0, 1, \ldots, m-1\}$, for $m \geq 4$, $\Sigma = \{a, b, c, d\}$, $s = m-2$, $F = \{0\}$ and δ is defined as follows:

(i) $\delta(m-2, c) = 0$,
(ii) $\delta(i, a) = i+1$, for $0 \leq i \leq m-4$, and $\delta(m-3, a) = 0$,
(iii) $\delta(i, d) = i$, for $1 \leq i \leq m-3$,
(iv) $\delta(m-2, b) = 1$, $\delta(0, b) = 0$, $\delta(i, b) = i$ for $2 \leq i \leq m-3$,
(v) all transitions not defined above go to the sink state $m-1$.

Fig. 1 depicts the DFA A. The figure omits the sink state $m-1$.

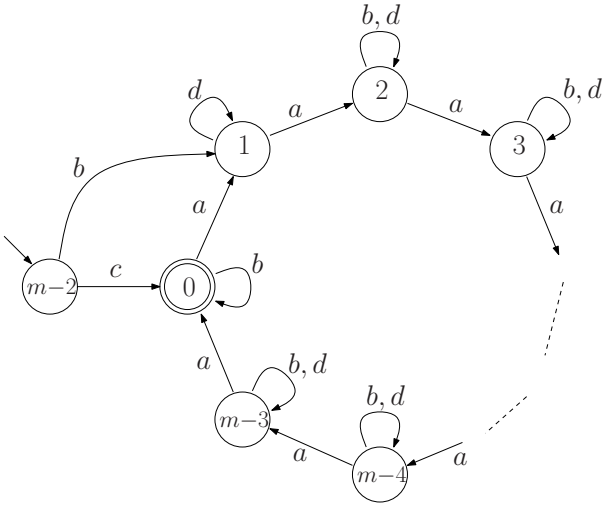

Fig. 1. The DFA A for the worst-case lower bound for the Kleene star of $L(A)$, for $m \geq 4$. Note that we omit the sink state $m-1$.

Lemma 3. *Let A be the DFA in Fig. 1 for $m \geq 4$.*

1. *The language $L(A)$ is suffix-free.*
2. *The state complexity of $L(A)^*$ is $2^{m-2}+1$.*

Combining Lemma 2 and Lemma 3, we have the following result.

Theorem 1. *Given an m-state suffix-free minimal DFA A, $2^{m-2}+1$ states are necessary and sufficient in the worst-case for the minimal DFA of $L(A)^*$.*

The proof of Lemma 3 uses a four character alphabet. It remains an open question whether the bound of Theorem 1 can be reached using an alphabet of size 2 or 3.

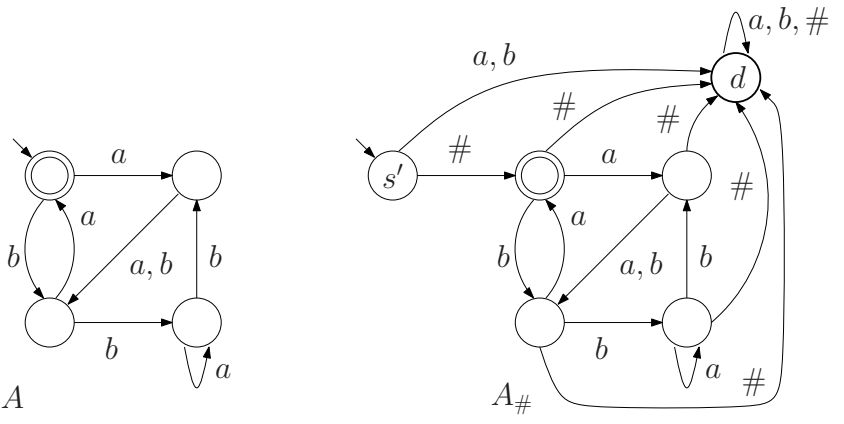

Fig. 2. An example of a minimal DFA A in Proposition 1. Note that $A_{\#}$ is also a minimal DFA and $L(A_{\#})$ is suffix-free.

3.2 Reversal of Suffix-Free Regular Languages

We examine the reversal operation of suffix-free regular languages. First, we recall the state complexity of reversal on regular languages. If a regular language L is accepted by an m-state minimal DFA, then its reversal L^R is accepted by an m-state NFA. By the well-known subset argument, we can conclude that the state complexity of L^R is at most 2^m.

Proposition 1 (Leiss [17] and Salomaa et al. [20]). *There are classes of regular languages for which 2^m states are necessary and sufficient for the reversal of an m-state minimal DFA. Note that such an m-state minimal DFA does not have the sink state.*

Given a suffix-free minimal DFA $A = (Q, \Sigma, \delta, s, F)$, we flip all transition directions in A and obtain a new FA A^R for $L(A)^R$. If we apply the subset construction on A^R, then the resulting DFA is the minimal DFA for $L(A^R)$ [2,22].

Lemma 4. *Given an m-state suffix-free minimal DFA A, $2^{m-2}+1$ states are sufficient in the worst-case for the minimal DFA of $L(A)^R$.*

Next, we show that $2^{m-2}+1$ states are necessary for the reversal of a suffix-free minimal DFA. Given a (regular) language L over Σ, $\#L$ is suffix-free if the character $\#$ is not in Σ.

We construct a suffix-free minimal DFA that has m states as follows: Let $A = (Q, \Sigma, \delta, s, F)$ be a minimal DFA as in Proposition 1 over Σ, which is not suffix-free in general. We introduce a new start state s' and a new transition $\delta(s', \#) = s$. We also introduce a sink state d. Note that a minimal DFA for a regular language in Proposition 1 does not have a sink state. Consequently, d is not equivalent with any of the states of A. Then, the new FA $A_{\#}$ is deterministic and minimal by construction. Furthermore, $L(A_{\#})$ is suffix-free. Thus, if A has $m-2$ states, then $A_{\#}$ has m states. See Fig. 2 for an example.

Lemma 5. *Given an m-state suffix-free minimal DFA $A_\#$ as shown in Fig. 2, $2^{m-2}+1$ states are necessary for the minimal DFA of $L(A_\#)^R$, where $\# \notin \Sigma$.*

We establish the following theorem from Lemmas 4 and 5. Note that Salomaa et al. [20] established that the result of Proposition 1 holds also for binary alphabets.

Theorem 2. *Given an m-state suffix-free minimal DFA A over Σ, $2^{m-2}+1$ states are necessary and sufficient in the worst-case for the minimal DFA of $L(A)^R$, where $|\Sigma| \geq 3$.*

4 Catenation

We investigate the state complexity of the catenation of two suffix-free regular languages. We first compute the upper bound and after that present a matching lower bound example.

Lemma 6. *Given two suffix-free minimal DFAs $A = (Q_1, \Sigma, \delta_1, s_1, F_1)$ and $B = (Q_2, \Sigma, \delta_2, s_2, F_2)$, $(m-1)2^{n-2}+1$ states are sufficient for the minimal DFA of $L(A) \cdot L(B)$, where $m = |Q_1|$ and $n = |Q_2|$.*

We present two suffix-free minimal DFAs A and B such that the state complexity of $L(A)L(B)$ reaches the upper bound in Lemma 6. In the following, let $\Sigma = \{a, b, c, d\}$. We define

$$A = (Q_1, \Sigma, \delta_1, s_1, F_1), \tag{1}$$

where $Q_1 = \{0, 1, \ldots, m-1\}$, $m \geq 3$, $s_1 = 0$, $F_1 = \{1\}$ and δ_1 is defined as follows:

(i) $\delta_1(0, c) = 1$,
(ii) $\delta_1(i, a) = i+1$, $1 \leq i \leq m-3$, $\delta_1(m-2, a) = 1$,
(iii) $\delta_1(i, b) = i$, $1 \leq i \leq m-2$,
(iv) $\delta_1(1, d) = 1$,
(v) all transitions not defined above go to the sink state $m-1$.

The DFA A is depicted in Fig. 3. The figure does not show the sink state $m-1$ or the transitions into the sink state.

Next we define

$$B = (Q_2, \Sigma, \delta_2, s_2, F_2), \tag{2}$$

where $Q_2 = \{0, 1, \ldots, n-1\}$, $n \geq 3$, $s_2 = 0$, $F_2 = \{1\}$, and δ_2 is defined by the following:

1. $\delta_2(0, d) = 1$,
2. $\delta_2(i, b) = i+1$, $1 \leq i \leq n-3$, $\delta_2(n-2, b) = 1$,
3. $\delta_2(i, a) = \delta_2(i, c) = i$, $1 \leq i \leq n-2$,
4. $\delta_2(i, d) = i$, $2 \leq i \leq n-2$,
5. all transitions not defined above go to the sink state $n-1$.

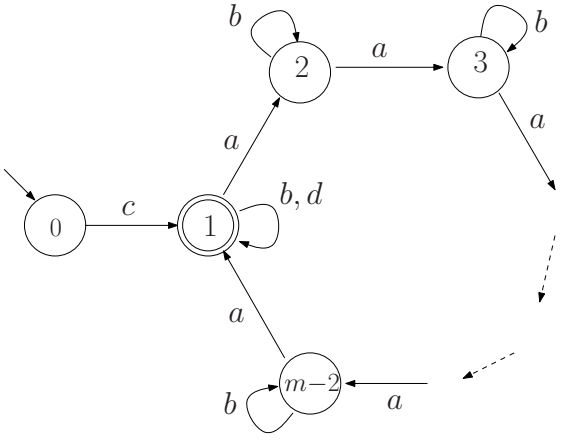

Fig. 3. The DFA A for the worst-case lower bound for catenation

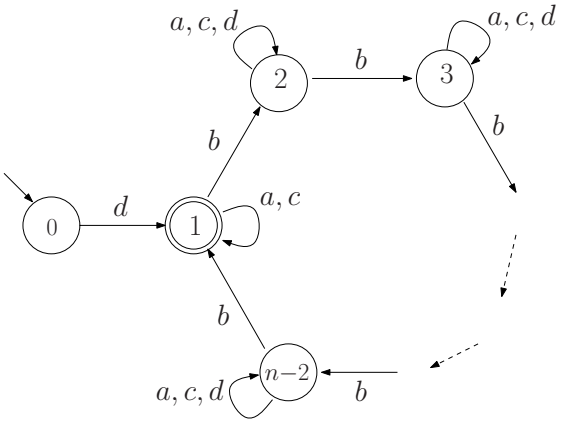

Fig. 4. The DFA B for the worst-case lower bound for catenation

The DFA B is depicted in Fig. 4. Again the figure does not show the sink state $n-1$.

Lemma 7. *Let A be as in (1) and B as in (2), for $m, n \geq 3$.*

1. *The languages $L(A)$ and $L(B)$ are suffix-free.*
2. *The state-complexity of $L(A) \cdot L(B)$ is $(m-1)2^{n-2}+1$.*

Lemma 7 shows that the upper bound in Lemma 6 is tight when $|\Sigma| \geq 4$.

Theorem 3. *For arbitrary $m, n \geq 3$, $(m-1)2^{n-2}+1$ states are necessary and sufficient in the worst-case for the catenation of, respectively, an m-state and an n-state suffix-free minimal DFAs.*

The worst-case example in Lemma 7 uses an alphabet with 4 characters. We do not know whether the upper bound can be reached using an alphabet of size 2 or 3.

5 Intersection and Union

Note that for the complement operation of an m-state suffix-free DFA, it is easy to verify that m states are necessary and sufficient. In the following, we consider the operations of intersection and union.

5.1 Intersection of Suffix-Free Regular Languages

Given two DFAs A and B, we can construct a DFA for the intersection of $L(A)$ and $L(B)$ based on the Cartesian product of states. For details on the Cartesian product construction, refer to Hopcroft and Ullman [13].

Proposition 2. *Given two DFAs $A = (Q_1, \Sigma, \delta_1, s_1, F_1)$ and $B = (Q_2, \Sigma, \delta_2, s_2, F_2)$, let $M = (Q_1 \times Q_2, \Sigma, \delta, (s_1, s_2), F_1 \times F_2)$, where for all $p \in Q_1$ and $q \in Q_2$ and $a \in \Sigma$,*

$$\delta((p, q), a) = (\delta_1(p, a), \delta_2(q, a)).$$

Then, $L(M) = L(A) \cap L(B)$.

Since the automaton M constructed in Proposition 2 is deterministic, it follows that mn states are sufficient for the intersection of $L(A)$ and $L(B)$, where $|A| = m$ and $|B| = n$. Note that mn is a tight bound for the intersection of two regular languages [24].

We assign a unique number for each state from 1 to m in A and from 1 to n in B, where $|A| = m$ and $|B| = n$. Assume that the mth state and the nth state are the sink states in A and B, respectively. Let $A \cap_c B$ denote the resulting intersection automaton that we compute using the Cartesian product of states. By the construction, $A \cap_c B$ is deterministic since A and B are deterministic. Therefore, we obtain a DFA for $L(A) \cap L(B)$. Next, we minimize $A \cap_c B$ by merging all equivalent states and removing unreachable states from the start state.

Proposition 3 (Han et al. [9]). *For a state (i, j) in $A \cap_c B$, the right language $L_{(i,j)}$ of (i, j) is the intersection of L_i in A and L_j in B.*

Since a suffix-free DFA A has the sink state as proved in Lemma 1, $L_{(m,i)} = \emptyset$, for $1 \leq i \leq n$, by Proposition 3, where m is the sink state of A. Therefore, we can merge all these states. Similarly, all states (j, n), for $1 \leq j \leq m$, of $A \cap_c B$ are equivalent and, therefore, can be merged.

Observation 1. *Given suffix-free minimal DFAs A and B, all states (m, i) for $1 \leq i \leq n$ and all states (j, n) for $1 \leq j \leq m$ of $A \cap_c B$ are equivalent.*

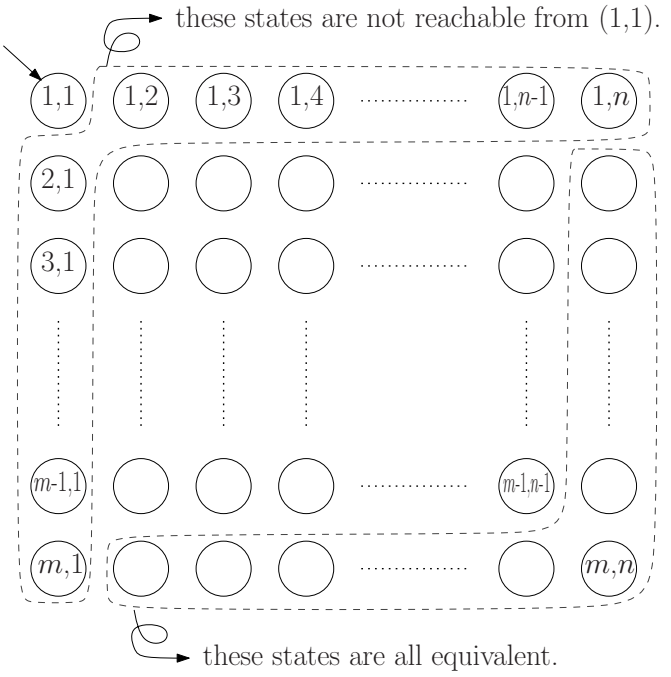

Fig. 5. The figure depicts the intersection automaton $A \cap_c B$ constructed for two suffix-free minimal DFAs A and B. Note that, by Observation 1, all states in the last row and in the last column are equivalent. Similarly, by Observation 2, all states, except for the start state (1,1), in the first row and in the first column are unreachable from (1,1).

Consider all states $(1, i)$, for $1 < i \leq n$, of $A \cap_c B$. Since $L(A)$ is suffix-free, the start state of A has no in-transitions. It implies that $(1, i)$ is not reachable from $(1, 1)$ in $A \cap_c B$ and, therefore, these states are useless as shown in Fig. 5. We can establish a similar result for the the states $(j, 1)$, for $1 < j \leq m$.

Observation 2. *Given suffix-free minimal DFAs A and B, all states $(1, i)$, for $1 < i \leq m$, and all states $(j, 1)$, for $1 < j \leq n$, are useless in $A \cap_c B$.*

Once we minimize $A \cap_c B$ based on Observations 1 and 2, the resulting minimal DFA has $mn - 2(m + n) + 6$ states.

Theorem 4. *Given two suffix-free minimal DFAs A and B, $mn - 2(m+n)+6$ states are necessary and sufficient in the worst-case for the minimal DFA of $L(A) \cap L(B)$, where $|\Sigma| \geq 3$.*

5.2 Union of Suffix-Free Regular Languages

We now investigate the union of two suffix-free regular languages. We compute the union DFA for $L(A)$ and $L(B)$ using the Cartesian product of states. Given

two suffix-free minimal DFAs $A = (Q_1, \Sigma, \delta_1, s_1, F_1)$ and $B = (Q_2, \Sigma, \delta_2, s_2, F_2)$, let $M = (Q_1 \times Q_2, \Sigma, \delta, (s_1, s_2), F)$, where for all $p \in Q_1$ and $q \in Q_2$ and $a \in \Sigma$,

$$\delta((p, q), a) = (\delta(p, a), \delta(q, a))$$

and $F = (F_1 \times Q_2) \cup (Q_1 \times F_2)$. Then, $L(M) = L(A) \cup L(B)$ and M is deterministic. Let $A \cup_c B$ denote M. Consider the right language of a state (i, j) in $A \cup_c B$.

Proposition 4 (Han et al. [9]). *For a state (i, j) in $A \cup_c B$, the right language $L_{(i,j)}$ of (i, j) is the union of L_i in A and L_j in B.*

Note that the two constructions for $A \cap_c B$ and $A \cup_c B$ are different. This implies that we may not be able to apply the same approach that we used for $A \cap_c B$ for computing the upper bound for $L(A) \cup L(B)$. For example, since $L_{(n,j)} = L_n \cup L_j \neq \emptyset$ by Proposition 4, all states (m, i) and (j, n) for $1 \leq i \leq n$ and $1 \leq j \leq m$, in $A \cup_c B$ are not necessarily equivalent. Thus, these states cannot be merged. On the other hand, we observe that all states $(1, i)$ and $(j, 1)$, for $1 < i \leq n$ and $1 < j \leq m$, are useless since $L(A)$ and $L(B)$ are suffix-free. Therefore, we minimize $A \cup_c B$ by removing these $m + n - 2$ states.

Theorem 5. *Given two suffix-free minimal DFAs A and B, $mn - (m + n) + 2$ states are necessary and sufficient in the worst-case for the minimal DFA of $L(A) \cup L(B)$, where $|\Sigma| \geq 5$.*

6 Conclusion

The state complexity of an operation for regular languages is the number of states that are necessary and sufficient for the minimal DFA that accepts the language obtained from the operation. Yu et al. [24] studied the operational state complexity of general regular languages and Han et al. [9] examined the state complexity of basic operations on prefix-free regular languages. Since suffix-freeness is reversal of prefix-freeness, it was a natural problem to examine the state complexity of basic operations on suffix-free regular languages.

operation	regular languages	prefix-free case	suffix-free case
L_1^*	$2^{m-1} + 2^{m-2}$	m	$2^{m-2} + 1$
L_1^R	2^m	$2^{m-2} + 1$	$2^{m-2} + 1$
$L_1 \cdot L_2$	$(2m - 1)2^{n-1}$	$m + n - 2$	$(m - 1)2^{n-2} + 1$
$L_1 \cap L_2$	mn	$mn - 2(m + n) + 6$	$mn - 2(m + n) + 6$
$L_1 \cup L_2$	mn	$mn - 2$	$mn - (m + n) + 2$

Fig. 6. Operational state complexity of general, prefix-free and suffix-free regular languages

Based on the structural property that a suffix-free minimal DFA must be non-returning, we have tackled Kleene star, reversal, catenation, intersection and union cases and obtained the tight bound for each operation.

Fig. 6 shows the comparison table of the state complexity on regular languages, prefix-free regular languages and suffix-free regular languages. We have established the tight state complexity bounds for each of the operations using languages over a fixed alphabet. However, the constructions usually require an alphabet of size 3 or 4 and, then, for most operations, it is open whether or not the upper bound for the state complexity of each operation can be reached using a small size alphabet such as $|\Sigma| = 2$ or 3.

References

1. Berstel, J., Perrin, D.: Theory of Codes. Academic Press, Inc., London (1985)
2. Brzozowski, J.: A survey of regular expressions and their applications. IEEE Transactions on Electronic Computers 11, 324–335 (1962)
3. 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)
4. 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)
5. Domaratzki, M.: State complexity of proportional removals. Journal of Automata, Languages and Combinatorics 7(4), 455–468 (2002)
6. Domaratzki, M., Salomaa, K.: State complexity of shuffle on trajectories. Journal of Automata, Languages and Combinatorics 9(2-3), 217–232 (2004)
7. Han, Y.-S., Salomaa, K.: State complexity of basic operations on suffix-free regular languages. Technical Report Technical Report 2007-534, Queen's University (2007) `http://www.cs.queensu.ca/TechReports/Reports/2007-534.pdf`
8. Han, Y.-S., Salomaa, K.: State complexity of union and intersection of finite languages. In: Proceedings of DLT'07. LNCS, vol. 4588, pp. 217–228 (2007)
9. Han, Y.-S., Salomaa, K., Wood, D.: State complexity of prefix-free regular languages. In: Proceedings of DCFS'06, pp. 165–176 (2006) (Full version is submitted for publication)
10. Han, Y.-S., Wood, D.: The generalization of generalized automata: Expression automata. International Journal of Foundations of Computer Science 16(3), 499–510 (2005)
11. Holzer, M., Kutrib, M.: Unary language operations and their nondeterministic state complexity. In: Ito, M., Toyama, M. (eds.) DLT 2002. LNCS, vol. 2450, pp. 162–172. Springer, Heidelberg (2003)
12. Holzer, M., Kutrib, M.: Nondeterministic descriptional complexity of regular languages. International Journal of Foundations of Computer Science 14(6), 1087–1102 (2003)
13. Hopcroft, J., Ullman, J.: Introduction to Automata Theory, Languages, and Computation, 2nd edn. Addison-Wesley, Reading, MA (1979)
14. Hricko, M., Jirásková, G., Szabari, A.: Union and intersection of regular languages and descriptional complexity. In: Proceedings of DCFS'05, pp. 170–181 (2005)

15. 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)
16. Jürgensen, H., Konstantinidis, S.: Codes. In: Rozenberg, G., Salomaa, A. (eds.) Word, Language, Grammar. Handbook of Formal Languages, vol. 1, pp. 511–607. Springer, Heidelberg (1997)
17. Leiss, E.L.: Succint representation of regular languages by boolean automata. Theoretical Computer Science 13, 323–330 (1981)
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. 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)
20. Salomaa, A., Wood, D., Yu, S.: On the state complexity of reversals of regular languages. Theoretical Computer Science 320(2-3), 315–329 (2004)
21. Wood, D.: Theory of Computation. John Wiley & Sons, Inc., New York, NY (1987)
22. Yu, S.: Regular languages. In: Rozenberg, G., Salomaa, A. (eds.) Word, Language, Grammar. Handbook of Formal Languages, vol. 1, pp. 41–110. Springer, Heidelberg (1997)
23. Yu, S.: State complexity of regular languages. Journal of Automata, Languages and Combinatorics 6(2), 221–234 (2001)
24. Yu, S., Zhuang, Q., Salomaa, K.: The state complexities of some basic operations on regular languages. Theoretical Computer Science 125(2), 315–328 (1994)