HKUST Theoretical Computer Science Center Research Report HKUST-TCSC-2006-02

State Complexity of Prefix-Free Regular Languages

Yo-Sub Han*
System Technology Division, Korea Institute of Science and Technology
emmous@kist.re.kr

Kai Salomaa†
School of Computing, Queen's University
ksalomaa@cs.queensu.ca

Derick Wood
Department of Computer Science, The Hong Kong University of Science and Technology
dwood@cs.ust.hk

Abstract

We investigate the state complexities of basic operations for prefix-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 (DFA) that accepts the language obtained from the operation. We know that a regular language is prefix-free if and only if its minimal DFA has only one final state and the final state has no out-transitions whose target state is not a sink state. Based on this observation, we reduce the state complexities for prefix-free regular languages compared with the state complexities for (general) regular languages. For both catenation and Kleene star operations of (general) regular languages, the state complexities are exponential in the size of given minimal DFAs. On the other hand, if both regular languages are prefix-free, then the state complexities are at most linear. We also demonstrate that we can reduce the state complexities of intersection and union operations based on the structural properties of prefix-free minimal DFAs.

1 Introduction

Codes play a crucial role in many areas such as information processing, date compression, cryptography, information transmission and so on [12]. They are categorized with respect to different conditions (for example, *prefix-free, suffix-free, infix-free* or *outfix-free*) according to applications [7, 10, 11]. Since codes deal with sets of strings, they are closely related to formal languages: a code is a *language*. The conditions that classify code types define proper subfamilies of given language families. For regular languages, for example, prefix-freeness defines the family of prefix-free regular languages, which is a proper subfamily of regular languages. Prefix-freeness is fundamental in coding theory; for example, Huffman

*Han was supported by the KIST Tangible Space Initiative Grant 2E19020.
†Salomaa was supported by the Natural Sciences and Engineering Research Council of Canada Grant OGP0147224.

codes are prefix-free sets. The advantage of prefix-free codes is that we can decode a given encoded string deterministically. Prefix-free regular languages have already been used to define *determinism* for generalized automata [4] and for expression automata [6]. Recently, Han et al. [5] considered prefix-free regular expressions as patterns in text searching and designed an efficient algorithm for the prefix-free regular-expression matching problem based on prefix-freeness.

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) [9, 15]. Based on this definition, Yu and his co-authors [16] 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. They showed that $m2^n - 2^{n-1}$ states are necessary and sufficient in the worst-case for the catenation of two DFAs whose sizes are m and n, respectively. They also demonstrated that $2^{n-1} + 2^{n-2}$ states are necessary and sufficient for the Kleene star operation of a given DFA whose size is n. As special cases of state complexity, Câmpeanu et al. [1] examined the state complexity of finite languages, Pighizzini and Shallit [13] investigated the state complexity of unary language operations and Holzer and Kutrib [8] studied the nondeterministic descriptional complexity of regular languages. There are several other results with respect to the state complexities of difference operations [2, 3, 14].

Prefix-free languages are used in many coding theory applications, and for this reason results on state complexity of prefix-free regular languages may be useful. Furthermore, determining the state complexity of operations on fundamental subfamilies of the regular languages can provide valuable insights on connections between restrictions placed on language definitions and descriptional complexity. Thus, we investigate the state complexities of basic operations, such as catenation, Kleene star, intersection and union, for prefix-free regular languages. Since prefix-free regular languages are a proper subfamily of regular languages and prefix-free minimal DFAs have unique properties, we aim to obtain better state complexity based on the structural properties compared with the state complexity of general regular languages.

In Section 2, we define some basic notions. In Section 3, we examine the state complexities of catenation and Kleene star of prefix-free regular languages. Given two prefix-free minimal DFAs of m states and n states, respectively, we show that $m + n - 2$ states are necessary and sufficient for catenation of two languages and m states are necessary and sufficient for Kleene star of the first language. The proof is based on the fact that a prefix-free minimal DFA has only one final state and the final state has no out-transitions whose target state is not a sink state. In Section 4, we study the state complexities of intersection and union of prefix-free regular languages based on the Cartesian product of states. We demonstrate that $mn - 2(m + n) + 6$ states are necessary and sufficient for intersection and $mn - 2$ states are necessary and sufficient for union.

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 character $\emptyset$ 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, $\delta \subseteq Q \times \Sigma \times Q$ is a (finite) set of transitions, $s \in Q$ 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. Let $|Q|$ be the number of states in Q and $|\delta|$ be the number of transitions in δ. Then, the size $|A|$ of A is $|Q| + |\delta|$. Given a transition (p, a, q) in δ, where $p, q \in Q$ and $a \in \Sigma$, 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. 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. Thus, 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 a final state of A does not have any out-transitions whose target state is not a sink state. We assume that A has only *useful* states.

We define a (regular) language L to be prefix-free if L is a prefix-free set. A regular expression E is prefix-free if $L(E)$ is prefix-free. 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.

3 Catenation and Kleene star of prefix-free regular languages

Let A and B be minimal DFAs for two regular languages L_1 and L_2 and $|A| = m$ and $|B| = n$. Yu et al. [16] obtained the following results for catenation $(L_1 L_2)$ and for Kleene star (L_1^*):

1. The size of the minimal DFA for $L_1 L_2$ is $m2^n - 2^{n-1}$ in the worst-case.
2. The size of the minimal DFA for L_1^* is $2^{m-1} + 2^{m-2}$ in the worst-case.

Note that L_1 and L_2 are not necessarily non-returning or non-exiting. On the other hand, a regular language is prefix-free if and only if its minimal DFA is non-exiting [6]. Let us consider what is possible when L_1 and L_2 are prefix-free and, thus, A and B are non-exiting. Observe that both A and B have only one final state.

Theorem 1. *Given two prefix-free minimal DFAs A and B, $m+n-2$ states are necessary and sufficient in the worst-case for the minimal DFA of $L(A)L(B)$, where $m = |A|$ and $n = |B|$.*

Proof. We construct a DFA for $L(A)L(B)$ by merging the final state of A and the start state of B to give a single state which also eliminates all out-transitions from the final state of A as shown in Fig. 1. We also merge two sink states to give a single sink state. The resulting automaton is deterministic and, therefore, $m + n - 2$ states are sufficient.

Let A be the minimal DFA for $L((a^{m-2})^*b)$ and B be the minimal DFA for $L((a^{n-2})^*b)$. They imply that $|A| = m$ and $|B| = n$. The size of the minimal DFA for $L(A)L(B)$ is $m + n - 2$ and, therefore, $m + n - 2$ states are necessary. $\square$

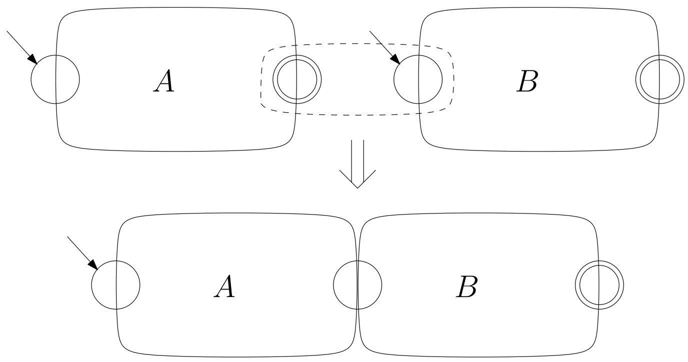

Figure 1: An example of a catenation of two prefix-free minimal DFAs.

We next examine the Kleene star operation. Given an FA $A = (Q, \Sigma, \delta, s, F)$, we construct an FA $A' = (Q, \Sigma, \delta', s, F)$, where

$$\delta' = \delta \cup \{(f, a, q) \mid f \in F \text{ and } (s, a, q) \in \delta \text{ for } a \in \Sigma\}.$$

That is, for each out-transition from s in A, we add new out-transitions with the same label from all final states as illustrated in Fig. 2. Then, $L(A') = L(A)^+$.

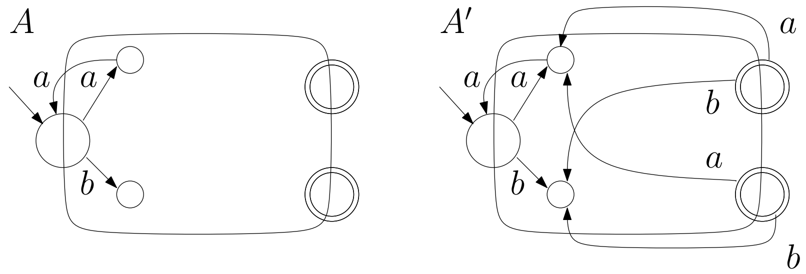

Figure 2: Given an FA A, we add new out-transitions with the same label from all final states for each out-transition from the start state. Note that $L(A') = L(A)^+$.

If A is a prefix-free minimal DFA, then A has only one final state and, therefore, A' also has one final state. Since we are computing the minimal DFA for $L(A)^*$, we should consider λ as well. We introduce a new start state s' and make a null-transition from s' to s in A'. Next, we ensure that s' is a final state as shown in Fig. 3 (a).

Given an FA $A = (Q, \Sigma, \delta, s, F)$, we define the *right language* L_q of a state q to be a set of strings that are spelled out by some path from q to a final state in A; namely, we ensure that q is the start state. We say that two states p and q are *equivalent* if $L_p = L_q$ [15]. If we identify two equivalent states, then we merge them and reduce the size of the corresponding FA. Since $L_{s'} = L_f$ in A'', we merge s' and f. Let B denote the resulting FA; see Fig. 3 (b) for example. Since merging two equivalent states does not affect the language, $L(A'') = L(B)$.

Lemma 2. *Given a prefix-free minimal DFA A, let B be the FA that we construct as shown in Fig. 3. Then, B is a minimal DFA and $L(B) = L(A)^*$.*

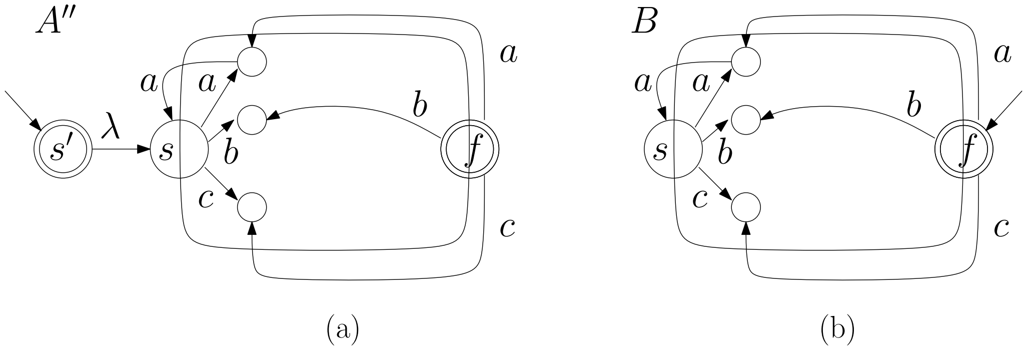

Figure 3: (a) Given a prefix-free minimal DFA A, we construct A'' such that $L(A'') = L(A') \cup \{\lambda\} = L(A)^+ \cup \{\lambda\} = L(A)^*$. (b) We merge two equivalent states s' and f in A'' and obtain B, where $L(B) = L(A)^*$.

Proof. Since $L(A)$ is prefix-free, f has no out-transitions whose target state is not a sink state. Furthermore, all new out-transitions that we add to f in A' have different labels and it guarantees that A' is also deterministic. Note that A' and B have the same transition function and, therefore, B is deterministic.

We prove the minimality of B as follows: Assume that B is not minimal. Then, there exist two equivalent states p and q. Since there is only one final state, both p and q cannot be final states. However, if p and q are equivalent in B, then p and q are also equivalent in A — a contradiction. Therefore, B is minimal. $\square$

Based on Lemma 2, we establish the following result.

Theorem 3. *Given a prefix-free minimal DFA A, m states are necessary and sufficient in the worst-case for the minimal DFA of $L(A)^*$, where $m = |A|$.*

Proof. Lemma 2 shows that m states are sufficient.

Let A be the minimal DFA for $L((a^{m-2})^*b)$. It implies that $|A| = m$. Since the size of the minimal DFA for $L(A)^*$ is m, m states are necessary. $\square$

4 Intersection and union of prefix-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.

Proposition 4 (Hopcroft and Ullman [9]). *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 resulting automaton M in Proposition 4 is deterministic, the construction shows that mn states are sufficient for the intersection of $L(A)$ and $L(B)$, where $|A| = m$

and $|B| = n$. It turns out that mn, which looks trivial, is a tight bound [16]. Yu et al. [16] also showed that mn states are both necessary and sufficient in the worst-case for the union of $L(A)$ and $L(B)$ as well.

We investigate the state complexities of intersection and union of prefix-free regular languages based on the structural properties of these minimal DFAs. We, first, consider the intersection of two prefix-free minimal DFAs $A = (Q_1, \Sigma, \delta_1, s_1, f_1)$ and $B = (Q_2, \Sigma, \delta_2, s_2, f_2)$ using the Cartesian product suggested by Hopcroft and Ullman [9].

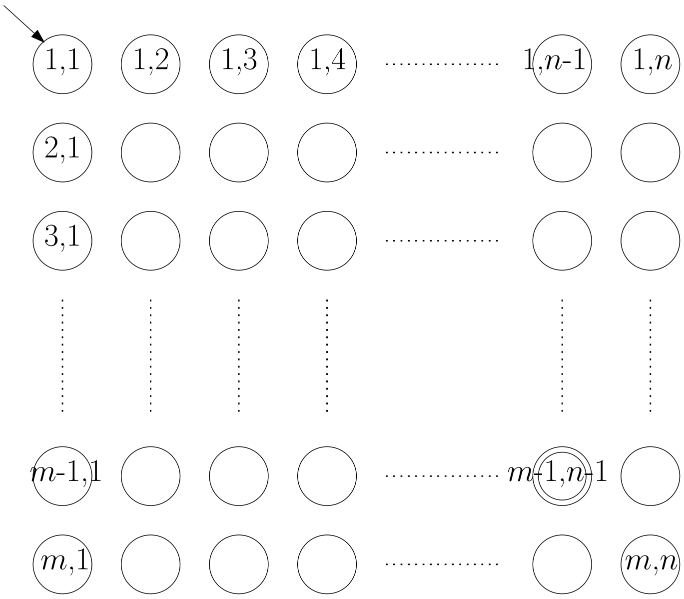

Figure 4: An example of computing the intersection of two prefix-free minimal DFAs based on the Cartesian product of states. We omit all transitions.

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 $(m-1)$th state and $(n-1)$th state are final states, and mth state and nth state are sink states in A and B, respectively. Let $A \cap_c B$ denote the resulting intersection automaton that we compute based on the Cartesian product of states. Then, $(1, 1)$ is the start state and $(m-1, n-1)$ is a final state. By the construction, $A \cap_c B$ is deterministic since A and B are deterministic. Therefore, we obtain a DFA for $L(A) \cap L(B)$. Now we identify all equivalent states and merge them in $A \cap_c B$ to compute the minimal DFA for $L(A) \cap L(B)$.

Lemma 5. *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.*

Proof. Assume that a string w is in $L_{(i,j)}$. It implies that there is an accepting path from (i, j) to $(m-1, n-1)$ for w in $A \cap_c B$. The accepting path for w is a sequence of states and each state has two indices, one is from A and the other is from B. Therefore, there exists the corresponding path from i to m in A that spells out w and, thus, $w \in L_i$ in A. Similarly, $w \in L_j$ in B. Therefore, $w \in L_{(i,j)}$ if and only if $w \in L_i \cap L_j$. $\square$

For all states in the last row of $A \cap_c B$, indices of these states are (m, i), for $1 \le i \le n$, and $L_{(m,i)} = \emptyset$ by Lemma 5 since $L_m = \emptyset$ in A. Therefore, we can merge all these states. Similarly, all states in the last column of $A \cap_c B$ are equivalent and, therefore, can be merged.

Observation 6. *Given prefix-free minimal DFAs A and B, all states in the last row and all states in the last column of $A \cap_c B$ are equivalent.*

Consider all states in the second-last row whose indices are $(m-1, j)$, for $1 \le j \le n-2$, of $A \cap_c B$. Since $L(A)$ is prefix-free, the final state $(m-1)$ of A has no out-transitions whose target state is not a sink state; namely, $L_{m-1} = \{\lambda\}$. Note that $\lambda \notin L_j$, for $1 \le j \le n-2$, in B since $L(B)$ is prefix-free and state j is not a final state of B. Therefore, all states $(m-1, j)$ of $A \cap_c B$ are equivalent by Lemma 5. Observe that we exclude $(m-1, n-1)$ since it is a final state of $A \cap_c B$. We can establish a similar result for the second-last column.

Observation 7. *Given prefix-free minimal DFAs A and B, all states $(m-1, j)$, for $1 \le j \le n-2$, and all states $(i, n-1)$, for $1 \le i \le m-2$, in $A \cap_c B$ are equivalent.*

Based on Observations 6 and 7, we merge all equivalent states in $A \cap_c B$ as shown in Fig. 5.

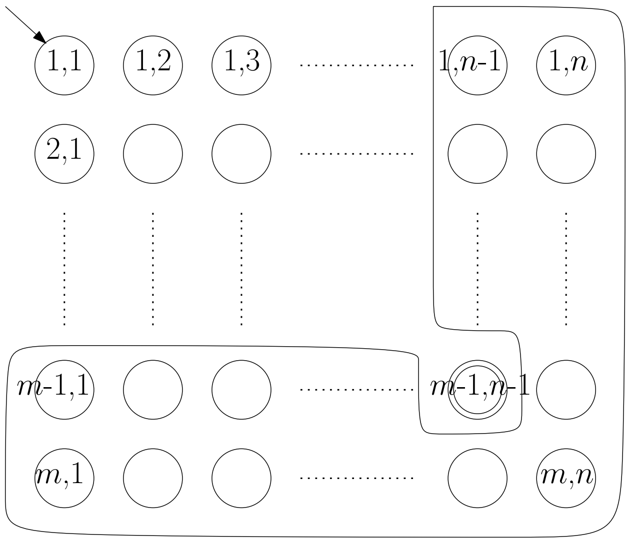

Figure 5: Based on Observations 6 and 7, we merge all equivalent states in $A \cap_c B$.

The resulting automaton in Fig. 5 has $mn - 2(m + n) + 6$ states and we have obtained the following theorem.

Theorem 8. *Given two prefix-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 $L(A) \cap L(B)$.*

Proof. Fig. 5 shows that $mn - 2(m + n) + 6$ states are sufficient.

Assume that $\Sigma = \{a, b, c\}$. Given a string w over Σ, let $|w|_a$ denote the number of a's in w. Let A be the minimal DFA for

$$L = \{wc \mid (m-2) \text{ divides } |w|_a, \text{ for } w \in \{a,b\}^*\}$$

and B be the minimal DFA for

$$L = \{wc \mid (n-2) \text{ divides } |w|_b, \text{ for } w \in \{a,b\}^*\}.$$

Then, $|A|$ is m and $|B|$ is n. Let M be the minimal DFA for the intersection of $L(A)$ and $L(B)$. M has to count both a's and b's before M reads c and, thus, M requires $(m-2)(n-2)$ states. M also needs one final state and one sink state. It shows that M needs $(m-2)(n-2)+2$ states and, therefore, $mn - 2(m+n) + 6$ states are necessary. □

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

$$F = \{(p, f_2) \mid p \in Q_1\} \cup \{(f_1, q) \mid q \in Q_2\}.$$

Then, $L(M) = L(A) \cup L(B)$ and M is deterministic. Fig. 6 illustrates M. Let $A \cup_c B$ denote M.

Consider the right language of a state (i, j) in $A \cup_c B$.

Lemma 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.*

Proof. The proof is similar to the proof for Lemma 5. □

Although the structure of $A \cap_c B$ illustrated in Fig. 4 and the structure of $A \cup_c B$ illustrated in Fig. 6 are similar, $A \cup_c B$ has two distinct differences compared with $A \cap_c B$:

1. All states in the second-last row and all states in the second-last column of $A \cup_c B$ are final states.

2. All states in the last row and all states in the last column of $A \cup_c B$ are not necessarily sink states except for state (n, m).

Because of these two distinct properties, these states are not equivalent anymore in $A \cup_c B$. On the other hand, we observe that

$$L_{(m,n-1)} = L_m \cup L_{n-1} = \{\lambda\} \text{ and } L_{(m-1,n)} = L_{m-1} \cup L_n = \{\lambda\}$$

since A and B are non-exiting. Therefore,

$$L_{(m,n-1)} = L_{(m-1,n)} = L_{(m-1,n-1)}.$$

We merge the three states $(m, n-1)$, $(m-1, n)$ and $(m-1, n-1)$ to reduce the number of states. It shows that $mn - 2$ states are sufficient for the union of the two prefix-free regular languages.

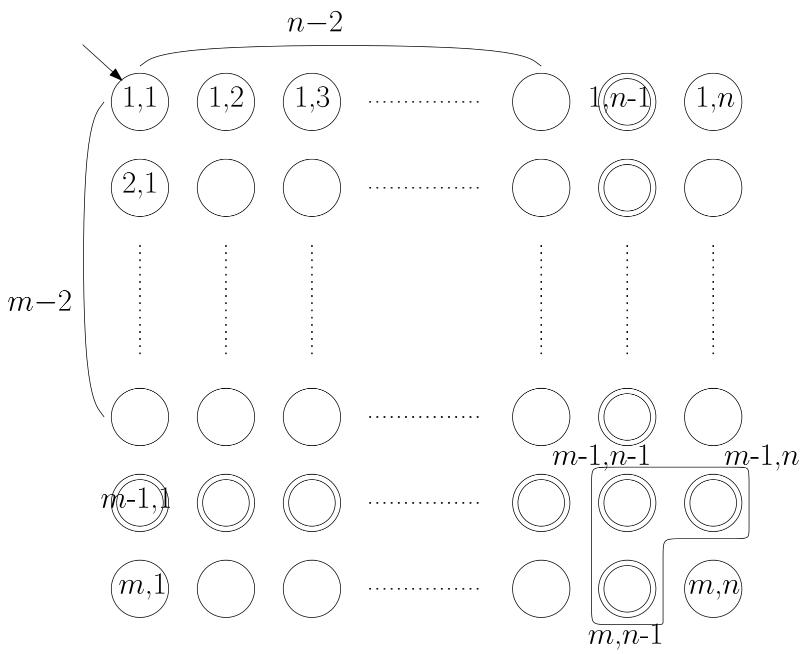

Figure 6: An example of computing the union of two prefix-free minimal DFAs based on the Cartesian product of states.

Theorem 10. *Given two prefix-free minimal DFAs A and B, $mn-2$ states are necessary and sufficient in the worst-case for the minimal DFA of $L(A) \cup L(B)$.*

Proof. Fig. 6 shows that $mn-2$ states are sufficient.

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 be the minimal DFA for

$$L = \{wc \mid (m{-}2) \text{ divides } |w|_a, \text{ for } w \in \{a, b, d\}^*\}$$

and B be the minimal DFA for

$$L = \{wd \mid (n{-}2) \text{ divides } |w|_b, \text{ for } w \in \{a, b, c\}^*\}.$$

Then, $|A|$ is m and $|B|$ is n. Let M be the minimal DFA for the union of $L(A)$ and $L(B)$. Let us examine the structure of M: M has to count both a's and b's before M reads any c or d: M requires $(m{-}2)(n{-}2)$ states. (This corresponds to all states in the $(m{-}2) \times (n{-}2)$ array in Fig. 6.

S1. Once M reads c, M only counts the number of b's. If the preceding number of a's is divisible by $(m{-}2)$, then, the current state is a final state. Then, after reading any character a, b or c, M goes to a non-accepting state that counts b's. This gives $2(n{-}2)$ inequivalent states. (These states correspond to all states in the second-last row and in the last row except for the last two columns in Fig. 6.) If M reads d, then it goes to a final state q_f whose target state is always a sink state.

S2. Symmetrically, once M reads d, M only counts the number of a's and M needs both final and non-final states for reading any character a, b or d. This gives $2(m-2)$ inequivalent states. (These states correspond to all states in the second-last column and in the last column except for the last two rows in Fig. 6.) If M reads c, then it goes to a final state q_f whose target state is always a sink state.

Therefore, the total number of states in M is

$$(m-2)(n-2) + 2(m-2) + 2(n-2) + 2 = mn - 2.$$

The constant 2 is from the common final state q_f for both cases and a sink state. Note that each state of the first mentioned $(m-2)(n-2)$ states can reach a final state with some strings having c and with some strings having d. On the other hand, each state of $2(n-2)$ states in S1 (respectively, $2(m-2)$ states in S2) can reach a final state with some strings having d (respectively, c) but with no strings having c (respectively, d). Thus, no two states from two different groups are equivalent. Clearly, the final state q_f and the sink state are not equivalent with any other states. It follows that $mn - 2$ states are necessary. □

5 Conclusions

We have investigated the state complexities of basic operations for prefix-free regular languages. Given a minimal DFA A, $L(A)$ is prefix-free if and only if A has only one final state and A is non-exiting [7]. Based on these structural properties, we have shown that given two prefix-free minimal DFAs A and B, $m+n-2$ states are necessary and sufficient for the catenation of $L(A)$ and $L(B)$, where $|A| = m$ and $|B| = n$. For the Kleene star operation of $L(A)$, m states are necessary and sufficient.

Hopcroft and Ullman [9] demonstrated that we can use the Cartesian product of states for computing the intersection of two FAs. If both FAs for the Cartesian product of states are deterministic, then the resulting FA M is also deterministic and $|M|$ is at most mn. Yu et al. [16] already showed that mn is a tight bound for intersection and union of (general) regular languages. We have examined the case when both DFAs A and B are prefix-free. Let A_c be the resulting DFA from the Cartesian product of states for the intersection of $L(A)$ and $L(B)$. We have observed that some states in A_c are equivalent since A and B are non-exiting. Based on this observation, we have proved that $mn - 2(m+n) + 6$ states are necessary and sufficient for intersection. We can use the Cartesian product of states to compute the union of two DFAs as well. The resulting DFA for union has three equivalent states and, thus, can be minimized. We have shown that $mn - 2$ states are necessary and sufficient for the union of two prefix-free regular languages.

We know that if a language L is prefix-free, then its reversal L^R is suffix-free by definition. If L is regular, then an FA for L^R must be non-returning. However, this condition is necessary but not sufficient. It implies that the suffix-free case is not symmetric to the prefix-free case for obtaining the state complexity. Therefore, natural future work is to investigate the state complexity of the suffix-free regular languages.

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*, 60–70. Springer-Verlag, 2001. Lecture Notes in Computer Science 2214.

[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] M. Domaratzki. State complexity of proportional removals. *Journal of Automata, Languages and Combinatorics*, 7(4):455–468, 2002.

[4] D. Giammarresi and R. Montalbano. Deterministic generalized automata. *Theoretical Computer Science*, 215:191–208, 1999.

[5] Y.-S. Han, Y. Wang, and D. Wood. Prefix-free regular-expression matching. In *Proceedings of CPM'05*, 298–309. Springer-Verlag, 2005. Lecture Notes in Computer Science 3537.

[6] Y.-S. Han and D. Wood. The generalization of generalized automata: Expression automata. *International Journal of Foundations of Computer Science*, 16(3):499–510, 2005.

[7] Y.-S. Han and D. Wood. Outfix-free regular languages and prime outfix-free decomposition. In *Proceedings of ICTAC'05*, 96–109. Springer-Verlag, 2005. Lecture Notes in Computer Science 3722.

[8] M. Holzer and M. Kutrib. Nondeterministic descriptional complexity of regular languages. *International Journal of Foundations of Computer Science*, 14(6):1087–1102, 2003.

[9] J. Hopcroft and J. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, Reading, MA, 2 edition, 1979.

[10] M. Ito, H. Jürgensen, H.-J. Shyr, and G. Thierrin. N-prefix-suffix languages. *International Journal of Computer Mathematics*, 30:37–56, 1989.

[11] M. Ito, H. Jürgensen, H.-J. Shyr, and G. Thierrin. Outfix and infix codes and related classes of languages. *Journal of Computer and System Sciences*, 43:484–508, 1991.

[12] H. Jürgensen and S. Konstantinidis. Codes. In G. Rozenberg and A. Salomaa, editors, *Word, Language, Grammar*, volume 1 of *Handbook of Formal Languages*, 511–607. Springer-Verlag, 1997.

[13] 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.

[14] 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.

[15] D. Wood. *Theory of Computation.* John Wiley & Sons, Inc., New York, NY, 1987.

[16] 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.