Fundamenta Informaticae 90 (2009) 93–106
DOI 10.3233/FI-2009-0008
IOS Press

Nondeterministic State Complexity of Basic Operations for Prefix-Free Regular Languages

Yo-Sub Han[*†]
Intelligence and Interaction Research Center, Korea Institute of Science and Technology
P.O.BOX 131, Cheongryang, Seoul, Korea
emmous@kist.re.kr

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

Derick Wood
Department of Computer Science, The Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, Hong Kong SAR
dwood@cs.ust.hk

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

Keywords: prefix-freeness, regular languages, nondeterministic state complexity, descriptional complexity

[*]Han was partly supported by the IT R&D program of MKE/IITA 2008-S-024-01 and the KRCF research grant.
[†]Address for correspondence: Intelligence and Interaction Research Center, Korea Institute of Science and Technology, P.O.BOX 131, Cheongryang, Seoul, Korea
[‡]Salomaa was supported by the Natural Sciences and Engineering Research Council of Canada Grant OGP0147224.

1. Introduction

Codes play a crucial role in many areas such as information processing, data compression, cryptography, information transmission and so on [18]. They are categorized with respect to different conditions (for example, *prefix-free*, *suffix-free*, *infix-free* or *outfix-free*) according to applications [1, 18]. Since codes are 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 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 [6] and for expression automata [12]. Recently, Han et al. [11] 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.

Regular languages are given by finite-state automata (FAs) or regular expressions. There are two main types of FAs: deterministic finite-state automata (DFAs) and nondeterministic finite-state automata (NFAs). NFAs provide exponential savings in space compared with DFAs but the problem to convert a given DFA to an equivalent minimal NFA is PSPACE-complete [15]. For finite languages, Salomaa and Yu [23] showed that $O(k^{\frac{n}{\log_2 k+1}})$ is a tight bound for converting an n-state NFA to a DFA, where k is the size of an input alphabet.

There are at least two different models for the state complexity of operations: The deterministic state complexity model considers minimal DFAs and the nondeterministic state complexity considers minimal NFAs.

Yu et al. [25, 27] investigated the deterministic state complexity for various operations on regular languages. Recently, Yu and his co-authors [5, 21, 26] examined the deterministic state complexity of combined operations on regular languages. As special cases of state complexity, Câmpeanu et al. [2] and Han and Salomaa [9] examined the deterministic state complexity of finite languages. Pighizzini and Shallit [20] investigated the deterministic state complexity of unary language operations. Moreover, Han et al. [10] studied the deterministic state complexity of prefix-free regular languages and Han and Salomaa [8] looked into the deterministic state complexity of suffix-free regular languages.

Holzer and Kutrib [13] studied the nondeterministic state complexity of regular languages. Jirásek et al. [16] examined the nondeterministic state complexity of complementation of regular languages. Here we consider the operational nondeterministic state complexity of prefix-free regular languages. The results of Holzer and Kutrib [13] provide upper bounds for the prefix-free case but are usually not tight. Since prefix-freeness is a fundamental code property, it is important to calculate the precise bounds. There are several other results with respect to the state complexity of various operations [3, 4, 22].

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. We observe that we can convert an arbitrary (prefix-free) NFA to a (prefix-free) NFA with a single final state that does not have any out-transitions without adding any states. Based on this observation, we compute the nondeterministic state complexity of basic operations for prefix-free regular languages.

In Section 2, we define some basic notions. In Section 3, we examine the worst-case nondeterministic state complexity of binary operations (catenation, union and intersection) of prefix-free regular languages. The results are tight in the sense that we give lower bound examples that match the upper bounds. Given an m-state prefix-free minimal NFA A and an n-state prefix-free minimal NFA B, we show that

- The nondeterministic state complexity of $L(A) \cdot L(B)$ is $m + n - 1$.
- The nondeterministic state complexity of $L(A) \cup L(B)$ is $m + n$.
- The nondeterministic state complexity of $L(A) \cap L(B)$ is $mn - (m + n) + 2$.

In Section 4, we study the nondeterministic state complexity of Kleene star and reversal operations for prefix-free regular languages. Given an m-state prefix-free minimal NFA A, we show that

- The nondeterministic state complexity of $L(A)^*$ is m.
- The nondeterministic state complexity of $L(A)^R$ is m.
- The nondeterministic state complexity of $\overline{L(A)}$ is 2^{m-1} or $2^{m-1} + 1$.

We give a comparison table between the deterministic state complexity and the nondeterministic state complexity in Section 5.

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 a string $x \in L$ is not a prefix of any other strings in L. 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. Let $|Q|$ be the number of states in Q. We define the size $|A|$ of A to be the number of states in A; namely $|A| = |Q|$. 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 final states of A do not have any out-transitions. If $\delta(q, a)$ has a single element q', then we denote $\delta(q, a) = q'$ instead of $\delta(q, a) = \{q'\}$ for simplicity.

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 spells out 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 of an FA 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. Note that if a prefix-free FA A has several final states, then all final states are equivalent and, thus, can be merged into a single final state since A is non-exiting. Therefore, a minimal NFA for a prefix-free regular language must have a single final state. We assume that a given NFA has no λ-transitions since we can always transform an n-state NFA with λ-transitions to an equivalent n-state NFA without λ-transitions [14].

For complete background knowledge in automata theory, the reader may refer to textbooks [14, 24].

Before tackling the problem, we present a nice technique that gives a lower bound for the size of NFAs suggested by Glaister and Shallit [7] and establish a lemma that is crucial to prove the tight bound for the nondeterministic state complexity in the following sections. Notice that an FA for a non-trivial prefix-free regular language L (namely, $L \neq \{\lambda\}$) must have at least 2 states since such FA needs at least one start state and one final state.

Proposition 2.1. (Glaister and Shallit [7])
Let $L \subseteq \Sigma^*$ be a regular language. Suppose that there exists a set of pairs

$$P = \{(x_i, w_i) \mid 1 \leq i \leq n\}$$

such that

1. $x_i w_i \in L$ for $1 \leq i \leq n$;
2. $x_i w_j \notin L$ for $1 \leq i, j \leq n$, and $i \neq j$.

Then, a minimal NFA for L has at least n states.

The set P satisfying the conditions of Proposition 2.1 is called a *fooling set* for L.

Lemma 2.1. Let $n \geq 2$ be an arbitrary integer. Then, n states are necessary and sufficient in the worst-case for an NFA of a prefix-free regular language $L((a^{n-1})^*b)$.

Proof:
The language $L = L((a^{n-1})^*b)$ is accepted by an NFA with n states that consist of a cycle of length $n-1$ of a-transitions and one b-transition from the start state to a final state.

Consider the following set of pairs of strings

$$P = \{(a, a^{n-2}b), (a^2, a^{n-3}b), \ldots, (a^{n-2}, ab), (a^{n-1}, b), (a^{n-1}b, \lambda)\}.$$

Now P is a fooling set for L and its cardinality is n. By Proposition 2.1, it follows that the nondeterministic state complexity of L is exactly n. □

We use $\mathcal{NSC}(L)$ to denote the number of states of a minimal NFA for L; namely, $\mathcal{NSC}(L)$ is the nondeterministic state complexity of L.

3. Binary Operations

We examine the nondeterministic state complexity of catenation, union and intersection of prefix-free regular languages.

3.1. Catenation of prefix-free regular languages

For the catenation of regular languages, $(2m-1)2^{n-1}$ is the state complexity for the DFA case [27] and $m+n$ is the state complexity for the NFA case [13]. From Han et al. [10], we know that in the prefix-free case, the deterministic state complexity is linear in the sizes of the component automata and the nondeterministic state complexity cannot be very much different.

Theorem 3.1. Given two prefix-free regular languages L_1 and L_2, the nondeterministic state complexity $\mathcal{NSC}(L_1L_2)$ for L_1L_2 is $m+n-1$, where $m = \mathcal{NSC}(L_1)$ and $n = \mathcal{NSC}(L_2)$.

Proof:
Let $A = (Q_1, \Sigma, \delta_1, s_1, f_1)$ and $B = (Q_2, \Sigma, \delta_2, s_2, f_2)$ be two NFAs recognizing prefix-free languages. As observed above, without loss of generality, we can assume that A and B have only one final state that is non-exiting. We construct an NFA C for $L(A)L(B)$ by merging the final state f_1 of A and the start state s_2 of B to give a single state that also eliminates all out-transitions of f_1 of A. Namely, $C = (Q, \Sigma, \delta, s_1, f_2)$, where $Q = Q_1 \cup Q_2 \setminus \{f_1\}$ and for each state $p \in Q$ and a character $a \in \Sigma$,

$$\delta(p,a) = \begin{cases} s_2 & \text{if } p \in Q_1 \text{ and } \delta_1(p,a) = f_1, \\ \delta_1(p,a) & \text{if } p \in Q_1 \text{ and } \delta_1(p,a) \neq f_1, \\ \delta_2(p,a) & \text{otherwise.} \end{cases}$$

It is easy to verify that $L(C) = L(A)L(B)$ from the construction. Since $Q = Q_1 \cup Q_2 \setminus \{f_1\}$, $|Q| = m+n-1$ and, therefore, the construction shows that $m+n-1$ states are sufficient.

We next prove that $m+n-1$ states are necessary for $L(A)L(B)$ in the worst-case. Let $L_1 = L((a^{m-1})^*b)$ and $L_2 = L((a^{n-1})^*b)$. Then, by Lemma 2.1, we know that $\mathcal{NSC}(L_1) = m$ and $\mathcal{NSC}(L_2) = n$.

Consider the following set consisting of $m+n-1$ pairs of strings:

$$\begin{aligned} P = \quad & \{(a, a^{m-2}ba^{n-1}b), (a^2, a^{m-3}ba^{n-1}b), \ldots, (a^{m-2}, aba^{n-1}b), (a^{m-1}, ba^{n-1}b)\} \cup \\ & \{(a^{m-1}ba, a^{n-2}b), (a^{m-1}ba^2, a^{n-3}b), \ldots (a^{m-1}ba^{n-1}, b), (a^{m-1}ba^{n-1}b, \lambda)\}. \end{aligned}$$

It is easy to verify that P is a fooling set of L_1L_2. By Proposition 2.1, $m+n-1$ states are necessary and, therefore, $\mathcal{NSC}(L_1L_2)$ is $m+n-1$.

Note that we cannot add the pair $(a^{m-1}b, a^{n-1}b)$ to P since $(a^{m-1}ba^{n-1})(a^{n-1}b) \in L_1L_2$, and $(a^{m-1}b)b \in L_1L_2$. □

Han et al. [10] showed that $m+n-2$ is the state complexity for the catenation of two prefix-free minimal DFAs whose sizes are m and n. Theorem 3.1 shows that there is no exponential gap between the DFA case and the NFA case. This is caused by the structural property that a minimal FA for a prefix-free regular language must be non-exiting and has only a single final state, and this structural property gives rise to a simple construction for the automaton recognizing the catenation of prefix-free regular languages.

3.2. Union of prefix-free regular languages

Han et al. [10] proved that $mn-2$ is the state complexity of the union of an m-state prefix-free DFA and an n-state prefix-free DFA using the Cartesian product of states. For the NFA case, we directly construct an NFA for the union of two prefix-free regular languages without the Cartesian product of states.

Theorem 3.2. Given two prefix-free regular languages L_1 and L_2, the nondeterministic state complexity $\mathcal{NSC}(L_1 \cup L_2)$ for $L_1 \cup L_2$ is $m+n$, where $m = \mathcal{NSC}(L_1)$ and $n = \mathcal{NSC}(L_2)$.

Proof:
Let $A = (Q_1, \Sigma, \delta_1, s_1, f_1)$ be a minimal NFA for L_1 and $B = (Q_2, \Sigma, \delta_2, s_2, f_2)$ be a minimal NFA for L_2. We construct an NFA C for $L_1 \cup L_2$ by introducing a new start state for the alternation of $L(A)$ and $L(B)$. Namely, $C = (Q, \Sigma, \delta, s, \{f_1, f_2\})$, where $Q = Q_1 \cup Q_2 \cup \{s\}$ and for a state $q \in Q$ and a character $a \in \Sigma$,

$$\delta(p,a) = \begin{cases} \delta_1(s_1,a) \cup \delta_2(s_2,a) & \text{if } p = s, \\ \delta_1(p,a) & \text{if } p \in Q_1, \\ \delta_2(p,a) & \text{if } p \in Q_2. \end{cases}$$

From C, we observe that the two final states f_1 and f_2 are equivalent and, thus, merge f_1 and f_2. Now we have

$$|C| = m + n + 1 - 1 = m + n$$

states. Therefore, $m+n$ states are sufficient for $L_1 \cup L_2$.

For the necessary condition, consider $L_1 = L((a^{m-1})^*b)$ and $L_2 = L((c^{n-1})^*d)$, where $\mathcal{NSC}(L_1)$ = m and $\mathcal{NSC}(L_2)$ = n by Lemma 2.1. Define the following set of pairs of strings:

$$\begin{aligned} P = \quad & \{(\lambda, a^{m-1}b), (a, a^{m-2}b), \ldots, (a^{m-2}, ab), (a^{m-1}, b)\} \cup \\ & \{(c, c^{n-2}d), (c^2, c^{n-3}d), \ldots, (c^{n-1}, d), (c^{n-1}d, \lambda)\}. \end{aligned}$$

The set P has $m+n$ elements and is a fooling set for $L_1 \cup L_2$. This shows that $m+n$ states are necessary by Proposition 2.1. Therefore, $\mathcal{NSC}(L_1 \cup L_2)$ is $m+n$. □

Theorem 3.2 shows that the state complexity of union of two prefix-free NFAs is linear in the sizes of input FAs. The size of the resulting NFA differs by one from the case dealing with union of general NFAs [13] due to the fact that the two final states of two prefix-free FAs are equivalent. We note that the state complexity of union of two prefix-free DFAs is quadratic in the sizes of inputs [10].

3.3. Intersection of prefix-free regular languages

Given two FAs $A = (Q_1, \Sigma, \delta_1, s_1, F_1)$ and $B = (Q_2, \Sigma, \delta_2, s_2, F_2)$, we can construct an FA $M = (Q_1 \times Q_2, \Sigma, \delta, (s_1, s_2), F_1 \times F_2)$ for the intersection of $L(A)$ and $L(B)$ based on the Cartesian product of states (see Hopcroft and Ullman [14] for details), where

$$\delta((p,q),a) = (\delta_1(p,a), \delta_2(q,a)) \text{ for } p \in Q_1, q \in Q_2 \text{ and } a \in \Sigma.$$

Since the construction does not require input FAs to be deterministic, we know that mn states are sufficient for $L(A) \cap L(B)$ if $|A| = m$ and $|B| = n$. Note that mn is the tight bound for the intersection

of regular languages given by either DFAs or NFAs [13, 27]. On the other hand, we observe that, in the case when $L(A)$ and $L(B)$ are prefix-free, M has some useless states and, thus, is not minimal because $L(A)$ and $L(B)$ are prefix-free. Our approach is to reduce the number of states by identifying and removing these useless states from M based on structural properties of prefix-free NFAs.

First, we assign a unique number to 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 final states in A and B, respectively. The 1st state is the start state in both A and B. 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, n) is the unique final state.

Proposition 3.1. Let $A \cap_c B$ be the Cartesian product of states for $L(A) \cap L(B)$. Then, all states (m, i) and (j, n), for $1 \leq i \leq n-1$ and $1 \leq j \leq m-1$, do not appear in an accepting path in $A \cap_c B$ since $L(A)$ and $L(B)$ are prefix-free.

If a state (m, i) for $i \neq n$ appears in an accepting path, then the final state m of A has an out-transition and this contradicts that $L(A)$ is prefix-free. Therefore, Proposition 3.1 is valid. Now from the observation, we know that we can remove all states (m, i) and (j, n), for $1 \leq i \leq n-1$ and $1 \leq j \leq m-1$ since they are useless.

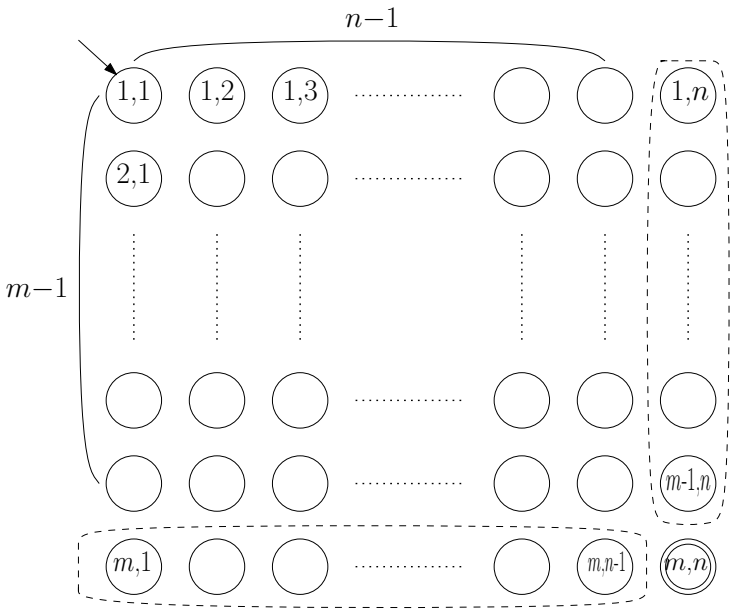

Figure 1. An example of computing the intersection of two prefix-free minimal NFAs based on the Cartesian product of states. We omit all transitions. All states inside the two dotted boxes are useless.

Once we remove all useless states, the resulting automaton has

$$mn - (m-1) - (n-1) = mn - (m+n) + 2$$

states. This implies that $mn - (m+n) + 2$ states are sufficient for $L(A) \cap L(B)$, where $|A| = m$ and $|B| = n$.

Theorem 3.3. Given two prefix-free regular languages L_1 and L_2, the nondeterministic state complexity $\mathcal{NSC}(L_1 \cap L_2)$ for $L_1 \cap L_2$ is $mn-(m+n)+2$, where $m = \mathcal{NSC}(L_1)$, $n = \mathcal{NSC}(L_2)$ and $|\Sigma| \geq 3$.

Proof:
The previous consideration together with Fig. 1 shows that $mn-(m+n)+2$ states are sufficient.

We prove the necessary condition by giving two prefix-free regular languages that reach the bound. Assume that $\Sigma = \{a, b, c\}$. Given a string w over Σ, let $|w|_a$ denote the number of a's in w. Let

$$L_1 = \{wc \mid |w|_a \equiv 0 \pmod{m-1}, \text{ for } w \in \{a,b\}^*\}$$

and

$$L_2 = \{wc \mid |w|_b \equiv 0 \pmod{n-1}, \text{ for } w \in \{a,b\}^*\}.$$

As in Lemma 2.1, one can show that $\mathcal{NSC}(L_1) = m$ and $\mathcal{NSC}(L_2) = n$.

Define the following set of pairs of strings:

$$P = \{(c, \lambda)\} \cup \{(a^i b^j, a^{m-1-i} b^{n-1-j} c) \mid 1 \leq i \leq m-1, 1 \leq j \leq n-1\}.$$

Then, P has $(m-1)(n-1)+1$ elements and P is a fooling set for $L_1 \cap L_2$. This implies that a minimal NFA for $L_1 \cap L_2$ needs at least $(m-1)(n-1)+1$ states. Therefore, $\mathcal{NSC}(L_1 \cap L_2) = mn-(m+n)+2$. □

4. Unary Operations

We consider the nondeterministic state complexity of Kleene star, reversal and complementation of prefix-free regular languages.

4.1. Kleene star of prefix-free regular languages

We examine the Kleene star operation of prefix-free NFAs. For the prefix-free minimal DFA case, Han et al. [10] showed that m is the tight bound, where m is the number of states.

We first consider the Kleene plus operation and study Kleene star later. Given a prefix-free minimal NFA $A = (Q, \Sigma, \delta, s, f)$, we construct a new NFA $A' = (Q, \Sigma, \delta', s, f)$, where

$$\delta'(p,a) = \begin{cases} \delta(p,a) & \text{if } p \neq f, \\ \delta(s,a) & \text{if } p = f. \end{cases} \tag{1}$$

Note that, by the construction, $L(A') = L(A)^+$. We show that A' must be minimal.

Lemma 4.1. A is a minimal NFA if and only if A' is a minimal NFA.

Proof:

$\Longrightarrow$ Assume that A' is not minimal. This implies that there is a minimal NFA B such that $L(B) = L(A)^+$ and $|B| < |A'|$. Since $L(B) = L(A)^+$, B must accept all stings in $L(A)$. We mark all states and transitions of accepting paths for the strings of $L(A)$ in B. The newly marked states and transitions give rise to a new NFA C.

Claim 1. C is non-exiting.
Assume that C is not non-exiting. Then, there is an accepting path for a string $w \in L(A)$ that passes through a final state f_B in B. Let u be a proper prefix string of w that is spelled out by reaching f_B. Since $u \in L(B) = L(A)^+$, $u = u_1u_2 \cdots u_k$, for $k \geq 1$, and $u_i \in L(A)$ for $1 \leq i \leq k$. Note that if u is a prefix string of w, then u_1 is also a prefix of w since u_1 is a prefix of u. This implies that both w and u_1 are in $L(A)$ — a contradiction. Therefore, C is non-exiting.

Claim 2. $L(A) = L(C)$.
Since $L(A) \subseteq L(C)$ by construction, we only need to show that $L(A) \supseteq L(C)$. Assume that $L(A) \not\supseteq L(C)$. This implies that there is an accepting path for a string $w \in L(C) \setminus L(A)$. Since C is a subgraph of B, $w \in L(B) = L(A)^+$. Note that $w = w_1w_2 \cdots w_k$, for $k > 1$, and $w_i \in L(A)$ for $1 \leq i \leq k$. Therefore, the accepting path for w in C must spell out $w_1w_2 \cdots w_k$. Since $w_1 \in L(A)$, there is a final state on the accepting path for w in C and it contradicts Claim 1. Therefore, $L(A) \supseteq L(C)$ and, thus, the claim is true.

Note that $|C| < |Q|$ since C is a subgraph of B. Then, Claim 2 shows that there exists a smaller NFA for $L(A)$ — a contradiction.

$\Longleftarrow$ Assume that A is not minimal. This implies that there exists a minimal NFA C such that $L(C) = L(A)$ and $|C| < |A|$. Since $L(C)$ is prefix-free, it is non-exiting and has a single final state. Then, we use the same construction above and obtain a new NFA C' such that $L(C') = L(C)^+ = L(A)^+$. Since $|C'| = |C| < |A| = |A'|$, there is a smaller NFA for $L(A')$ — a contradiction. □

Lemma 4.1 guarantees that m states are sufficient for $L(A)^+$. Since for every $m \geq 1$ there exists a prefix-free language L with $\mathcal{NSC}(L) = m$, we obtain the following statement from Lemma 4.1.

Theorem 4.1. Given a prefix-free regular language L, the nondeterministic state complexity $\mathcal{NSC}(L^+)$ for L^+ is m, where $m = \mathcal{NSC}(L)$.

Given a prefix-free minimal NFA $A = (Q, \Sigma, \delta, s, f)$, we have constructed a minimal NFA $A' = (Q, \Sigma, \delta', s, f)$ such that $L(A') = L(A)^+$. Now we change the start state of A' to f and denote the resulting NFA as B; namely $B = (Q, \Sigma, \delta', f, f)$.

Lemma 4.2. Given such $A' = (Q, \Sigma, \delta', s, f)$ and $B = (Q, \Sigma, \delta', f, f)$,

1. $L(B) = L(A') \cup \{\lambda\}$.
2. A' is minimal if and only if B is minimal.

Proof:
We only prove the first result. The second result can be proved using an argument similar to that for the proof of Lemma 4.1.

Since f is now a start state, $\lambda \in L(B)$. Let $w = w_1w_2 \cdots w_k$ be a string in $L(A')$. This implies that there is an accepting path for w in A'. Since $\delta'(s, w_1) = \delta'(f, w_1)$ by (1), we reach the same set of states after reading w_1 in both A' and B. Consequently, we can follow the same accepting path in B since δ' is the same. Similarly, we can show that if a string $w \neq \lambda \in L(B)$, then $w \in L(A')$. □

Lemma 4.2 gives an upper bound of the nondeterministic state complexity for L^*. We can easily show that the upper bound is reachable based on Theorem 4.1.

Theorem 4.2. Given a prefix-free regular language L, the nondeterministic state complexity $\mathcal{NSC}(L^*)$ for L^* is m, where $m = \mathcal{NSC}(L)$.

For the Kleene star of regular languages, the deterministic state complexity is $2^{n-1} + 2^{n-2}$ [27] and the nondeterministic state complexity is $m + 1$ [13], where m is the number of states in both cases. On the other hand, we have the same bound m if the input regular language is prefix-free. Intuitively, the reason can be viewed to be the fact that minimal FAs for a prefix-free regular language have a single final state that is non-exiting.

4.2. Reversal of prefix-free regular languages

We investigate the nondeterministic state complexity of reversal on prefix-free regular languages. Note that if L is prefix-free, then L^R is suffix-free. Therefore, the complexity is also related to the reversal of suffix-free regular languages.

Theorem 4.3. Given a prefix-free regular language L, the nondeterministic state complexity $\mathcal{NSC}(L^R)$ is m, where $m = \mathcal{NSC}(L)$.

Proof:
Since $m = \mathcal{NSC}(L)$, there is a minimal NFA $A = (Q, \Sigma, \delta, s, f)$ that accepts L, where $|Q| = m$. Remark that A has a single final state and it is non-exiting. Based on this structural property, we can obtain an NFA A^R for L^R by flipping the directions of all transitions and interchange the start state and the final state. Namely, the new NFA $A^R = (Q, \Sigma, \delta^R, f, s)$. Since $L^R = L(A^R)$, and both A and A^R have the same set Q of states, we know that m states are sufficient for L^R.

Next, we demonstrate that the bound is reachable. Let $L = L((a^{m-1})^*b)$. It is already proved in Lemma 2.1 that $\mathcal{NSC}(L) = m$. Consider the following set of pairs of strings:

$$P' = \{(\lambda, ba^{m-1}), (b, a^{m-1}), (ba, a^{m-2}), \ldots, (ba^{m-3}, a^2), (ba^{n-2}, a)\}.$$

Note that P' is the reversal of the fooling set P used in Lemma 2.1. It is clear that P' is a fooling set for L^R and has m elements. Then, a minimal NFA for L^R needs at least m states by Proposition 2.1. Therefore, $\mathcal{NSC}(L^R) = m$. □

If we compute the minimal DFA for $L(A)^R$, then we need an exponential number of states since we have to determinize A^R. For example, the state complexity of reversal of a prefix-free minimal DFA is $2^{m-2} + 1$, where m is the number of states [10].

4.3. Complementation of prefix-free regular languages

The complementation of NFA is a hard operation with respect to state complexity. It is well known that 2^n is the tight upper bound for transforming an n-state NFA to a DFA [19]. The complementation of an n-state DFA does not change the state complexity since it simply interchanges final states and non-final states. Thus, based on the subset construction, we know that 2^n states are sufficient for the complementation of an n-state NFA. For the tight bound, Jirásková [17] recently showed that 2^n states are necessary when $|\Sigma| = 2$.

We use a modification of the construction of Jirásková [17] for computing the nondeterministic state complexity of the complementation of prefix-free regular languages.

Lemma 4.3. Given an n-state prefix-free NFA $A = (Q, \Sigma, \delta, s, f)$, $2^{n-1} + 1$ states are sufficient for its complement language $\overline{L(A)}$.

Proof:
We first determinize A using the subset construction. Let A' be the resulting DFA. Note that A' has 2^n states (some of them may be useless states), and the final states are the subsets of Q that contain f. Since $L(A)$ is prefix-free, $L(A')$ is also prefix-free and, thus, we can merge all 2^{n-1} final states into a single final state f' in A'. Therefore, the number of states for a DFA for $L(A)$ is $2^{n-1} + 1$. Now we interchange final states and non-final states and the resulting FA is an NFA for $\overline{L(A)}$. □

Next we show that the upper bound of Lemma 4.3 for the complement of prefix-free languages can almost be reached. We need the following two lemmas.

Lemma 4.4. Let $\Sigma = \Omega \cup \{\#\}$ be an alphabet, where $\# \notin \Omega$. Let $L_1 \subseteq \Omega^*$ and $L = L_1 \cdot \#$. Then $\mathcal{NSC}(L) = \mathcal{NSC}(L_1) + 1$.

Proof:
We note that L is prefix-free and hence the minimal NFA for L has exactly one final state and it is non-exiting. Any in-transition of the unique final state f has to be labeled by $\#$. Hence from a minimal NFA for L we can construct an NFA for L_1 simply by making all source states of f to be final states and removing f. This shows that if the minimal NFA for L has n states, then the minimal NFA for L_1 needs at most $n - 1$ states.

On the other hand, let $A = (Q, \Omega, \delta, s, F)$ be a minimal NFA for L_1. (Since strings of L_1 do not contain occurrences of $\#$, the transitions of A do not use the symbol $\#$.) Then we can construct for L an NFA $A' = (Q \cup \{f\}, \Sigma, \delta_1, s, f)$, $f \notin Q$, where for $q \in Q \cup \{f\}$, $x \in \Sigma$,

$$\delta_1(q, x) = \begin{cases} \delta(q, x) \text{ if } q \in Q, x \in \Omega, \\ f \text{ if } q \in F, x = \#, \\ \emptyset \text{ otherwise.} \end{cases}$$

This means that if the minimal NFA for L has n states, then the minimal NFA for L_1 needs at least $n - 1$ states. □

Lemma 4.5. Let $\Sigma = \Omega \cup \{\#\}$ be an alphabet, where $\# \notin \Omega$. Let $L \subseteq \Sigma^*$ and $L_1 \subseteq \Omega^*$ be regular languages such that

$$L \cap (\Omega^* \cdot \#) = L_1 \cdot \#. \tag{2}$$

Then $\mathcal{NSC}(L) \geq \mathcal{NSC}(L_1)$.

Proof:
Let $A = (Q, \Sigma, \delta, s, F)$ be a minimal NFA for L. It is sufficient to show that L_1 can be recognized by an NFA that has the same set of states Q.

We define $B = (Q, \Omega, \gamma, s, F_1)$ where γ is the restriction of δ to $Q \times \Omega$ (that is, γ is undefined for the symbol $\#$) and

$$F_1 = \{q \in Q \mid \delta(q, \#) \in F\}.$$

By the choice of F_1, it follows that B accepts $w \in \Omega^*$ if and only if A accepts $w\#$. By (2), it follows that $L(B) = L_1$. □

Now the following lemma gives a lower bound for the complement of prefix-free languages.

Lemma 4.6. Let Σ be an alphabet of cardinality at least three and $n \geq 2$. There exists a prefix-free regular language L_3 over Σ with $\mathcal{NSC}(L_3) = n$ such that $\mathcal{NSC}(\overline{L_3}) \geq 2^{n-1}$.

Proof:
We denote $\Sigma = \Omega \cup \{\#\}$, $\# \notin \Omega$. Since Ω has at least two letters, from Jirásková [17] we know that there exists a regular language $L_2 \subseteq \Omega^*$ such that

$$\mathcal{NSC}(L_2) = n - 1 \text{ and } \mathcal{NSC}(\Omega^* - L_2) = 2^{n-1}. \tag{3}$$

We define the prefix-free language L_3 as $L_3 = L_2 \cdot \#$. By (3) and Lemma 4.4, we know that $\mathcal{NSC}(L_3) = n$. We note that $\Sigma^* - L_3$ consists of all strings over Σ that do not end with $\#$ and the strings of $(\Sigma^* - L_2) \cdot \#$. Hence $(\Sigma^* - L_3) \cap (\Omega^*\#) = (\Omega^* - L_2) \cdot \#$. Consequently, by Lemma 4.5 and (3) it follows that

$$\mathcal{NSC}(\Sigma^* - L_3) \geq 2^{n-1}. \qquad \square$$

We conclude with the following result that follows from Lemma 12 and Lemma 4.6.

Theorem 4.4. The nondeterministic state complexity of the complement of an n-state NFA language is at most $2^{n-1} + 1$. For each $n \geq 1$, there exists a language L over a three letter alphabet recognized by an NFA with n states such that the nondeterministic state complexity of $\overline{L}$ is at least 2^{n-1}.

The result of Theorem 4.4 means that the the worst-case nondeterministic state complexity of complementation is either $2^{n-1} + 1$ or 2^{n-1}. The lower bound construction obtained as an extension of the construction from Jirásková [17] requires an alphabet of size 3 and the question remains open for alphabet size 2. Holzer and Kutrib [13] have established a lower bound 2^{n-2} for the nondeterministic state complexity of complementation of general regular languages over a binary alphabet.

5. Conclusions

We have investigated the nondeterministic state complexity of basic operations for prefix-free regular languages. If a minimal NFA A is prefix-free, then A has only one final state and A is non-exiting. Based on these structural properties, we have examined the nondeterministic state complexity with respect to

operation	prefix-free DFAs	prefix-free NFAs
$L_1 \cdot L_2$	$m + n - 2$	$m + n - 1$
$L_1 \cup L_2$	$mn - 2$	$m + n$
$L_1 \cap L_2$	$mn - 2(m + n) + 6$	$mn - (m + n) + 2$
L_1^*	m	m
L_1^R	$2^{m-2} + 1$	m
$\overline{L_1}$	m	2^{m-1} or $2^{m-1} + 1$

Table 1. State complexity of basic operations between prefix-free DFAs [10] and prefix-free NFAs.

catenation, union, intersection, Kleene star, reversal and complementation. Table 1 shows the comparison between the deterministic state complexity and the nondeterministic the state complexity.

We know that if a language L is prefix-free, then its reversal L^R is suffix-free by definition. Therefore, it is natural to investigate whether or not similar results hold for the nondeterministic state complexity of suffix-free regular languages.

Acknowledgements

We wish to thank the referees for the care they put into reading the previous version of this manuscript. Their comments were invaluable in depth and detail, and the current version owes much to their efforts. As usual, however, we alone are responsible for any remaining sins of omission and commission.

References

[1] J. Berstel and D. Perrin. *Theory of Codes*. Academic Press, Inc., 1985.

[2] 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.

[3] 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.

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

[5] 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.

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

[7] I. Glaister and J. Shallit. A lower bound technique for the size of nondeterministic finite automata. *Information Processing Letters*, 59(2):75–77, 1996.

[8] Y.-S. Han and K. Salomaa. State complexity of basic operations on suffix-free regular languages. In *Proceedings of MFCS'07*, Lecture Notes in Computer Science 4708, 501–512, 2007.

[9] 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.

[10] Y.-S. Han, K. Salomaa, and D. Wood. State complexity of prefix-free regular languages. In *Proceedings of DCFS'06*, 165–176, 2006. *Full version is submitted for publication*.

[11] Y.-S. Han, Y. Wang, and D. Wood. Prefix-free regular languages and pattern matching. *Theoretical Computer Science*, 389(1-2):307–317, 2007.

[12] 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.

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

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

[15] T. Jiang and B. Ravikumar. Minimal NFA problems are hard. *SIAM Journal on Computing*, 22(6):1117–1141, 1993.

[16] 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.

[17] G. Jirásková. State complexity of some operations on binary regular languages. *Theoretical Computer Science*, 330(2):287–298, 2005.

[18] 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.

[19] A. R. Meyer and M. J. Fischer. Economy of description by automata, grammars, and formal systems. In *Proceedings of the Twelfth Annual IEEE Symposium on Switching and Automata Theory*, 188–191, 1971.

[20] 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.

[21] A. Salomaa, K. Salomaa, and S. Yu. State complexity of combined operations. *Theoretical Computer Science*, 383(2-3):140–152, 2007.

[22] 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.

[23] K. Salomaa and S. Yu. NFA to DFA transformation for finite languages over arbitrary alphabets. *Journal of Automata, Languages and Combinatorics*, 2(3):177–186, 1998.

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

[25] S. Yu. State complexity of regular languages. *Journal of Automata, Languages and Combinatorics*, 6(2):221–234, 2001.

[26] S. Yu. On the state complexity of combined operations. In *Proceeding of CIAA'06*, Lecture Notes in Computer Science 4094, 11–22, 2006.

[27] 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.