International Journal of Foundations of Computer Science
Vol. 26, No. 6 (2015) 697–707

DOI: 10.1142/S0129054115500392

World Scientific
www.worldscientific.com

State Complexity of Boundary of Prefix-Free Regular Languages

Hae-Sung Eom* and Yo-Sub Han†

Department of Computer Science, Yonsei University
50, Yonsei-Ro, Seodaemun-Gu, Seoul 120-749, Republic of Korea
**haesung@cs.yonsei.ac.kr*
†*emmous@cs.yonsei.ac.kr*

Received 4 January 2015
Accepted 24 March 2015
Communicated by Arto K. Salomaa

Recently, researchers studied the state complexity of boundary — $L^* \cap L^{c*}$ — of regular languages L motivated from the famous Kuratowski's 14-theorem. Prefix codes — a set of languages — play an important role in several applications. We consider prefix-free regular languages and investigate the state complexity of two operations, L^{c*} and $L^* \cap L^{c*}$ for prefix-free regular languages. Based on the unique structural properties of a prefix-free minimal DFA, we compute the precise state complexity of L^{c*} and $L^* \cap L^{c*}$. We then present the tight bound over a quaternary alphabet for L^{c*} and $L^* \cap L^{c*}$. Our results are smaller than the composition of the state complexity function for individual operations of prefix-free regular languages.

Keywords: Finite-state automata; prefix-free regular languages; boundary operation; state complexity.

1. Introduction

State complexity is one of the most intensively studied topics in automata and formal language theory in recent years [1, 2, 4, 5, 8, 11, 13, 15, 16, 18, 22, 23]. Many applications including ClamAV,[a] PROSITE[b] and Snort[c] use regular languages and finite-state automata (FAs). The size of FAs used in those applications increase steadily. State complexity is a measurement of a FA size by the number of states in the FA. Since measuring the size of FAs become more and more important, state complexity also becomes important in automata theory. For instance the estimated upper bound of the state complexity useful in practice since it may help to manage resource efficiently. Moreover, it is a challenging quest to verify whether or not an estimated upper bound can be reached. The state complexity of an operation for regular languages is defined as the number of states that are necessary and sufficient

†Corresponding author.
[a]http://www.clamav.net.
[b]http://prosite.expasy.org/.
[c]https://www.snort.org/.

in the worst-case for the minimal DFA to accept the language resulting from the operation, considered as a function of the state complexities of operands. Maslov [14] obtained the state complexity of concatenation and other basic operations; however, his short paper did not include many proofs. Later, unaware of the earlier work, Yu *et al.* [23] reintroduced the study of operational state complexity in a more systematic way.

While researchers mainly looked at the state complexity of single operations (union, intersection, catenation and so on), Yu and his co-authors started to investigate the state complexity of combined operations (star-of-union, star-of-intersection and so on) [6, 7, 17, 19]. They showed that the state complexity of a combined operation is usually not equal to the function composition of the state complexities of the participating individual operations. They also observed that, in a few cases, the state complexity of a combined operation is very close to the composition of the individual state complexities.

Recently, Jirásek and Jirásková studied the state complexity of boundary of regular languages [12]. The boundary of a language is defined as $L^* \cap L^{c*}$. The problem is motivated from the famous Kuratowski's "14-theorem" states that, in a topological space, at most 14 sets can be produced by applying the operations of closure and complement to a given set. They found the tight bound for the boundary of a language over a five-letter alphabet.

Here we consider the state complexity of boundary of prefix-free regular languages. Note that prefix-free regular languages preserve unique structural properties in minimal DFAs, and these properties are crucial to obtain the state complexity bounds that are often significantly lower than for general regular languages [9, 10]. We first compute the state complexity of L^{c*} for prefix-free regular languages and then compute the state complexity of boundary of prefix-free regular languages. In Sec. 2, we define some basic notions. Then we present the state complexity of L^{c*} and $L^* \cap L^{c*}$ for prefix-free regular languages, respectively, in Secs. 3 and 4. We summarize the results and conclude the paper in Sec. 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. Let $|w|_b$ be the number of occurrences of symbol $b \in \Sigma$ in the string w. For strings x, y and z, we say that x is a *prefix* of z, if $z = xy$. We define a language L to be prefix-free if for any two distinct strings x and y in L, x is not a prefix of y. For a DFA A, we say that A is prefix-free if A accepts a prefix-free regular language.

A DFA A is specified by a tuple $(Q, \Sigma, \delta, s, F)$, where Q is a finite set of states, Σ is an input alphabet, $\delta : Q \times \Sigma \to Q$ is a transition function, $s \in Q$ is the start state and $F \subseteq Q$ is a set of final states. Given a DFA A, we assume that A is complete;

namely, each state has $|\Sigma|$ out-transitions and, therefore, A may have a dead state (a non-final state where all out-transitions are self-loops). We assume that A has a unique dead state since all dead states are equivalent and can be merged into a single state. Let $|Q|$ be the number of states in Q. We define the size $|A|$ of A to be $|Q|$. For a transition $\delta(p, a) = q$ in A, we say that p has an *out-transition* and q has an *in-transition*. Furthermore, p is a *source state* of q and q is a *target state* of p. We say that A is *non-returning* if the start state of A does not have any in-transitions and A is *non-exiting* if all out-transitions of any final state in A go to the dead state.

A nondeterministic finite-state automaton (NFA) is a tuple $A = (Q, \Sigma, \delta, Q_0, F)$ where Q, Σ, F are as in a DFA, Q_0 is the set of start states, and $\delta\colon Q \times \Sigma \to 2^Q$ is the transition function. Every NFA A can be converted to an equivalent DFA $A' = (2^Q, \Sigma, \delta', Q_0, F')$ by the subset construction. If $A = (Q, \Sigma, \delta, s, F)$ is an NFA, where $\delta : Q \times \Sigma \to 2^Q$ is nondeterministic transition function, which can be naturally extended to the domain $2^Q \times \Sigma^*$, then the subset automaton of A gives rise to a DFA $A' = (2^Q, \Sigma, \delta'\{s\}, F')$, where $\delta'(R, a) = \delta(R, a)$ for each R in 2^Q and each a in Σ, and $F' = \{R \in 2^Q \mid R \cup T \neq \emptyset\}$. The subset automaton may not be minimal since some of its states may be unreachable or equivalent.

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

For more background in automata theory, the reader may refer to the textbooks [20, 21].

3. State Complexity of L^{c*}

The state complexity is a good measurement for the size of a regular language. There are two well-known observations on state complexity:

(1) The state complexities for combined operations are usually smaller than the composition of the state complexity function for individual operations [6, 7, 17, 19].
(2) The state complexity of subfamilies of regular languages is usually smaller than the state complexity of regular languages [3, 9, 10].

With respect to the boundary of a regular language $L^* \cap L^{c*}$, these observations lead us to expect that

(1) The state complexity of L^{c*} would be smaller than the composition of the state complexity of L^c and L^*.
(2) The state complexity of the boundary operation $L^* \cap L^{c*}$ for prefix-free regular languages would be smaller than the general regular language case.

The state complexity of L^* for prefix-free regular languages is already studied but the state complexity of L^{c*} has been not proved yet. Thus, before we tackle the state complexity of boundary of prefix-free regular languages, we investigate the state complexity of L^{c*}. For simplicity, let an n-state prefix-free minimal DFA mean a DFA that is prefix-free, minimal, and has exactly n states.

Lemma 1. *Given an n-state prefix-free minimal DFA for L, $2^{n-3}+2$ states are sufficient for a DFA to accept L^{c*}, where $n \geq 3$.*

Proof. Let a prefix-free language L be accepted by a DFA $A = (\{s, q_1, q_2, \ldots, q_{n-3}, f, d\}, \Sigma, \delta, s, \{f\})$ with a dead state d. We can construct an NFA N for L^{c*} from the DFA A by exchanging the final and non-final states, and by adding a transition on a from a state q to the state s whenever $\delta(q, a) \in \{q_1, q_2, \ldots, q_{n-3}\}$ (notice that there is no need to add a new transition if $\delta(q, a) = d$).

Now, consider the subset automaton of the NFA N:

- the empty set is unreachable since A is deterministic and complete;
- each reachable non-empty subset of $\{s, q_1, q_2, \ldots, q_{n-3}\}$ must contain the state s;
- each set $S \cup \{d\}$ is equivalent to the state $\{d\}$;
- if S is a non-empty subset of $\{s, q_1, q_2, \ldots, q_{n-3}\}$, then $S \cup \{f\}$ is equivalent to the state $\{d\}$.

This guarantees that there are at most $2^{n-3}+2$ reachable and pairwise distinguishable subsets in the subset automaton of N as follows:

- 2^{n-3} subsets of $\{s, q_1, q_2, \ldots, q_{n-3}\}$ containing s;
- $\{d\}$;
- $\{f\}$. □

Lemma 2. *Given an n-state prefix-free minimal DFA for L, 1 state is necessary in the worst-case for a DFA that accepts L^{c*} for $n = 1, 2$.*

Proof. Since we consider prefix-free regular languages, $L_1 = \emptyset$ (when $n = 1$) or $L_2 = \{\lambda\}$ (when $n = 2$). Note that $L_1^{c*} = \Sigma^*$ and $L_2^{c*} = (\Sigma^+)^* = \Sigma^*$. Therefore, the state complexity is 1 for $n = 1, 2$. □

Lemma 3. *Given an n-state prefix-free minimal DFA for L, $2^{n-3}+2$ states are necessary in the worst-case for a DFA to accept L^{c*}, where $n \geq 3$.*

Proof. We define a prefix-free minimal DFA A such that the state complexity of L^{c*} reaches the upper bound in Lemma 1.

Let $A = (Q, \Sigma, \delta, 0, \{n-2\})$, where $Q = \{0, 1, 2, \ldots, n-1\}$, $\Sigma = \{a, b, c, d\}$, and δ is defined as follows (see Fig. 1 for illustration):

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

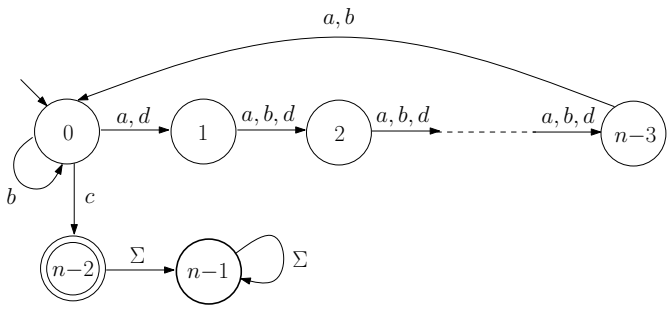

Fig. 1. The prefix-free DFA of a language L with $\mathcal{SC}(L^{c*}) = 2^{n-3} + 2$. The state $n-1$ is the dead state.

Note that the unique final state $n-2$ always goes to the dead state $n-1$. Thus, A is prefix-free.

Now we show that the state complexity $2^{n-3} + 2$ is reachable. Let a DFA $D = (Q_D, \Sigma, \delta_D, \{0\}, Q_D \setminus \{n-2\})$ be the resulting DFA constructed from a DFA A as in the proof of Lemma 1. We first show that all states of D are pairwise inequivalent and then reachable.

Inequivalence: We consider three types of states separately:

(1) Let P_1 and P_2 be two distinct states in $Q_D \setminus \{\{n-2\}, \{n-1\}\}$. Since $P_1 \neq P_2$, we can choose a state $j \neq 0$ that is in P_1 but not in P_2. While P_1 reaches the final state by the string $b^{n-3-j}db^{n-2}c$, P_2 cannot reach the final state by the same string. Thus all states in $Q_D \setminus \{\{n-2\}, \{n-1\}\}$ are inequivalent.
   - $P_1 \xrightarrow{b^{n-3-j}} P_1' \xrightarrow{db^{n-2}} \{0, n-1\} \xrightarrow{c} \{n-2, n-1\}$,
   - $P_2 \xrightarrow{b^{n-3-j}} P_2' \xrightarrow{db^{n-2}} \{0\} \xrightarrow{c} \{n-2\}$,

   where $n-3 \in P_1'$ and $n-3 \notin P_2'$. Note that $\{n-2, n-1\}$ is a final state of D and $\{n-2\}$ is not a final state of D. Thus, P_1 and P_2 are inequivalent.
(2) The state $\{0, n-1\}$ is inequivalent to the states above since it goes to the final state by the string $b^n c$ while all the states as described above go to non-final state.

(3) The state $\{n-2\}$ is the unique non-final state in D. Thus, it is inequivalent with any other states of D.

Therefore all states in D are pairwise inequivalent.

Reachability: We consider four types of states separately:

(1) The start state $\{0\}$ is reachable.
(2) Let $P = \{0, i_1, i_2, \ldots, i_k\}$ be a state in Q_D, where $0 < i_1 < i_2 < \cdots < i_k < n-2$. Let us prove by induction on the size of subsets that P is always reachable in D. Every set $\{0, i\}$ is reached from the start state $\{0\}$ by ab^{i-1}. Assume that $2 \leq k \leq n-1$ and that every set P of size k is reachable. Then, every set P of size $k+1$ is reached from the set $\{0, i_2 - i_1, i_3 - i_1, \ldots, i_k - i_1\}$ by the string ab^{i_1-1}.
(3) The state $\{n-2\}$ is reachable from the start state by c.
(4) The state $\{0, n-1\}$ is reachable from the start state by ca.

Therefore, all states in D are reachable. □

Theorem 1. *Let L be a prefix-free regular language over an alphabet Σ with $\mathcal{SC}(L) = n$ for $n \geq 3$. Then $\mathcal{SC}(L^{c*}) \leq 2^{n-3} + 2$, and the bound is tight if $|\Sigma| \geq 4$.*

4. State Complexity of the Boundary of Prefix-Free Regular Languages

We now consider the state complexity of $L^* \cap L^{c*}$ for prefix-free regular languages.

Lemma 4. *Given an n-state prefix-free minimal DFA for L, $(n-1)(2^{n-4}+1)+2$ states are sufficient for a DFA to accept $L^* \cap L^{c*}$, where $n \geq 4$.*

Proof. Let $A = (Q, \Sigma, \delta, s, \{f\})$ be a DFA for L with the state set $Q = \{s, q_1, q_2, \ldots, q_{n-3}, f, d\}$. Let d be the dead state of A and f be the unique final state of A.

From A, we construct an NFA N for the language L^{c*} as described in the proof of Lemma 1. We obtain an NFA D for L^* by adding several transitions and changing the start state. Before adding transitions, we remove the transition from f to d. For all transitions from s to $q \in Q$ on $a \in \Sigma$, we add a new transition from f to q by a. Next we change the start state from s to f. Then, we obtain a DFA D that accepts L^*. Since a state always goes to only one state in D, D is deterministic. Now we obtain a DFA D' that recognizes L^{c*} using the subset construction of N as described in the proof of Lemma 1. Then, $L^* \cap L^{c*}$ is accepted by the product automaton $D \times D'$. Let us count the number of reachable and distinguishable states in $D \times D'$.

The start state of $D \times D'$ is $(f, \{s\})$. Let a be a character in Σ. If $\delta(s, a) = d$, then $(f, \{s\})$ goes on a to $(d, \{d\})$ which is a dead state of $D \times D'$. If $\delta(s, a) = f$,

then $(f, \{s\})$ goes on a to $(f, \{f\})$. If $\delta(s, a) = p \notin \{f, d\}$, then $(f, \{s\})$ on a goes to $(p, \{p, s\})$. That is, we either go to a dead state, or to a state (p, P) such that $p \in P$.

Now let (p, P) with $p \in P$ be a reachable state. If $\delta(p, a) = d$, then (p, P) goes by a to a state (d, P') with $d \in P'$, and such a state is equivalent to $(d, \{d\})$. If $\delta(p, a) = f$, then (p, P) goes by a to (f, P') where $f \in P'$. The state $(f, \{f\})$ is equivalent to $(s, \{d\})$, and each state $(f, P' \cup \{f\})$ with P' nonempty is equivalent to $(f, \{d\})$. If $\delta(p, a) = p'$ which is not in $\{f, d\}$, then (p, P) goes by a to a state (p', P') with $p' \in P'$ and $s \in P'$. Now if $d \in P'$ or $f \in P'$, then (p', P') is equivalent to $(p', \{d\})$. Otherwise, P' is a subset of $\{s, q_1, q_2, \ldots, q_{n-3}\}$ containing p' and s. Thus, there are at most $1 + n + 2^{n-3} + (n-3) \cdot 2^{n-4} = (n-1) \cdot 2^{n-4} + n + 1$ reachable and pairwise distinguishable states:

- the start state $(f, \{s\})$;
- the n states $(p, \{d\})$ with $p \in Q$;
- the 2^{n-3} states (s, P), where P is a subset of $\{s, q_1, q_2, \ldots, q_{n-3}\}$ containing s;
- the $(n-3) \cdot 2^{n-4}$ states (p, P), where $p \in \{q_1, q_2, \ldots, q_{n-3}\}$ and P is a subset of $\{s, q_1, q_2, \ldots, q_{n-3}\}$ such that $s \in P$ and $p \in P$. □

In Lemma 4, we compute the upper bound for $L^* \cap L^{c*}$. Now, we present lower bound examples that reach the upper bound.

Lemma 5. *Given an n-state prefix-free minimal DFA for L, 2 states are necessary in the worst-case for a DFA that accepts $L^* \cap L^{c*}$ for $n = 1, 2$.*

Proof. Since we consider prefix-free regular languages, $L = \emptyset$ (when $n = 1$) or $L = \{\lambda\}$ (when $n = 2$). This follows that, for $n = 1$, $L^* \cap L^{c*} = \emptyset^* \cap \Sigma^* = \{\lambda\} \cap \Sigma^* = \{\lambda\}$, and, thus the state complexity is 2. For $n = 2$, $L^* \cap L^{c*} = \{\lambda\}^* \cap (\Sigma \setminus \{\lambda\})^* = \{\lambda\} \cap \Sigma^* = \{\lambda\}$, and, thus the state complexity is 2. □

Lemma 6. *Given an n-state prefix-free minimal DFA for L, $(n-1)(2^{n-4}+1)+2$ states are necessary in the worst-case for a DFA that accepts $L^* \cap L^{c*}$ for $n \geq 4$.*

Proof. We define a prefix-free minimal DFA A such that state complexity of $L^* \cap L^{c*}$ reaches the upper bound in Lemma 4. Let $A = (Q, \Sigma, \delta, 0, \{n-2\})$, where $Q = \{0, 1, 2, \ldots, n-1\}$, for $\Sigma = \{a, b, c, d\}$, and δ is defined as follows (see Fig. 2 for illustration):

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

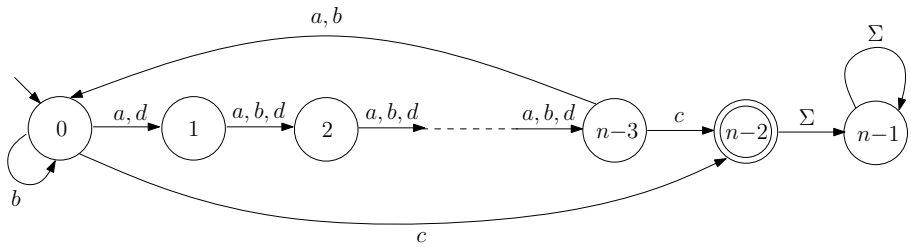

Fig. 2. The prefix-free DFA of a language L with $\mathcal{SC}(L^* \cap L^{c*}) = (n-1)(2^{n-4}+1)+2$. The state $n-1$ is the dead state.

Note that the unique final state $n-2$ has no out-transitions whose target state is not the dead state $n-1$. Thus, A is prefix-free.

Now we show that the state complexity $(n-1)(2^{n-4}+1)+2$ is reachable. From A, we construct a DFA $D \times D' = (Q_D, \Sigma, \delta_D, (n-2, \{0\}), F)$ for $L^* \cap L^{c*}$ as in the proof of Lemma 4. We first show that all states of D are pairwise inequivalent and then show that all states are reachable.

Inequivalence: Let (k_1, P_1) and (k_2, P_2) be two distinct states in Q_D. Since $(k_1, P_1) \neq (k_2, P_2)$, either $k_1 \neq k_2$ or $P_1 \neq P_2$ holds:

(1) $k_1 \neq k_2$: By the string $a^{n-k_1-3}c$, the state (k_1, P_1) does not go to a final state while the state (k_2, P_2) goes to a final state.
(2) $P_1 \neq P_2$: Since $P_1 \neq P_2$, without loss of generality, we can choose a state $j \in P_1 \setminus P_2$. There are two possibilities. If $k_1 > j$, then let $t_1 = 0$, otherwise, let $t_1 = k_1 + n - 3 - j$. Similarly, if $k_2 > j$, then let $t_2 = 0$, otherwise, let $t_2 = k_2 + n - 3 - j$. Then,

- $(k_1, P_1) \xrightarrow{b^{n-3-j}} (t_1, P_1') \xrightarrow{db^{n-2}} (0, \{0, n-1\}) \xrightarrow{c} (n-2, \{n-2, n-1\})$,
- $(k_2, P_2) \xrightarrow{b^{n-3-j}} (t_2, P_2') \xrightarrow{db^{n-2}} (0, \{0\}) \xrightarrow{c} (n-2, \{n-2\})$,

where $n-3 \in P_1'$ and $n-3 \notin P_2'$. Note that $(n-2, \{n-2, n-1\})$ is a final state of D and $(n-2, \{n-2\})$ is not a final state of D. Thus, (k_1, P_1) and (k_2, P_2) are inequivalent.

We now consider the state $(k, \{0, n-1\})$. For $0 \leq k \leq n-3$, a state $(k, \{0, n-1\})$ goes to a final state by the string $b^{n-2}c$ while other states considered in Case (1) and Case (2) cannot reach a final state. For two arbitrary states k_1 and k_2, where $k_1 \neq k_2$, $(k_1, \{0, n-1\})$ reaches a final state by the string $a^{n-2-k_1}c$, but $(k_2, \{0, n-1\})$ goes to a non-final state by the same string. Notice that the state $(n-2, \{0, n-1\})$ is inequivalent with states $(k, \{0, n-1\})$, where $1 \leq k \leq n-3$, because $(n-2, \{0, n-1\})$ is a final state. It is also inequivalent with other states considered above by the same reason. Therefore, all states in D are pairwise inequivalent.

Reachability: We consider five types of states separately:

(1) The start state $(n-2, \{0\})$ is reachable.
(2) Let a state $(0, P) \in Q_D$, where $P = \{0, i_1, i_2, \ldots, i_t\}$ and $0 < i_1 < i_2 < \cdots < i_t < n-2$. If $t > 1$, then from the start state, we can reach a state (k, P) by the string $ab^{n-3-i_t}ab^{i_t-i_{t-1}-1}ab^{i_{t-1}-i_{t-2}-1}\cdots ab^{i_2-i_1-1}ab^{i_1-1}$. If $t = 1$, then from the start state, we can reach a state (k, P) by the string $ab^{n-3-i_1}ab^{i_1-1}$.
(3) Let a state $(k, P) \in Q_D$, where $k \in Q \setminus \{0, n-2, n-1\}$, $P = \{0, i_1, i_2, \ldots, i_t\}$ and $0 < i_1 < i_2 < \cdots < i_t < n-2$, where $k \in P$. If $t > 1$, then from the start state, we can reach a state (k, P) by the string $ab^{n-i_t+k-3}ab^{i_t-i_{t-1}-1}ab^{i_{t-1}-i_{t-2}-1}\cdots ab^{i_2-i_1-1}ab^{i_1-1}$. If $t = 1$, then from the start state, we can reach a state (k, P) by the string $ab^{n-i_1+k-3}ab^{i_1-1}$.
(4) If $P = \{n-1\}$, then we can reach the state $(k, \{n-1\})$, where $0 \le k \le n-3$ by cca^k from the start state. We can reach the state $(n-2, \{n-1\})$ by cc from the start state.
(5) From the start state, the dead state is reachable by d^{n-2}. The character d is only used at this place.

Therefore, all states in D are reachable. □

Theorem 2. *Let L be a prefix-free regular language over an alphabet Σ with $\mathcal{SC}(L) = n$ for $n \ge 4$. Then $\mathcal{SC}(L^* \cap L^{c*}) \le (n-1)(2^{n-4}+1)+2$, and the bound is tight if $|\Sigma| \ge 4$.*

5. Conclusions

We have examined the state complexity of two operations, L^{c*} and $L^* \cap L^{c*}$, for prefix-free regular languages and established the tight state complexity bound for L^{c*} and $L^* \cap L^{c*}$ using a quaternary alphabet. The state complexity of subfamilies of regular languages (such as finite regular languages, unary regular languages, prefix-free or suffix-free regular languages) is often smaller than the state complexity of regular languages [1, 8–10, 16]. Our results are also smaller than the state complexities for general regular languages as summarized in Table 1.

Table 1. Comparison table between the state complexity of L^{c*} and $L^* \cap L^{c*}$ for prefix-free regular languages.

operation	prefix-free	general [12]
L^{c*}	$2^{n-3}+2$	$(3/4)\cdot 2^n$
$L^* \cap L^{c*}$	$(n-1)(2^{n-4}+1)+2$	$(3/8)\cdot 4^n + 2^{n-2} - 2\cdot 3^{n-2} - n + 2$

For both operations, we obtain smaller bounds than the general case. Table 2 represents the state complexities of intersection, complement, star for prefix-free regular languages and general case.

Table 2. The state complexities of intersection, complement, star for prefix-free regular languages and general case.

operation	prefix-free [10]	general [23]
$L_1 \cap L_2$	$mn - 2(m+n) - 3$	mn
L^c	n	n
L^*	n	$2^{n-1} + 1$

Note that the state complexity of L^{c*} is smaller than the composition of the state complexity of L^c for prefix-free regular languages and the state complexity of L^* for general case. Since a language L^c may not be prefix-free, we consider the state complexity of L^* for general case. It is easy to know that the state complexity of boundary of prefix-free regular languages is also smaller than the composition of results in Table 2.

We use a four-letter alphabet for the tight bounds for L^{c*} and $L^* \cap L^{c*}$. Tight bounds for L^{c*} and $L^* \cap L^{c*}$ in the binary or ternary case remain open.

Acknowledgments

We wish to thank the referees for the careful reading of the paper and many valuable suggestions. As usual, however, we alone are responsible for any remaining sins of omission and commission.

This research was supported by the Basic Science Research Program through NRF funded by MEST (2012R1A1A2044562) and the Yonsei University Future-leading Research Initiative of 2014.

References

[1] C. Câmpeanu, K. Culik II, K. Salomaa, and S. Yu. State complexity of basic operations on finite languages. In *Proceedings of WIA*'99, Lecture Notes in Computer Science 2214, pages 60–70, 2001.

[2] C. Câmpeanu, K. Salomaa, and S. Yu. Tight lower bound for the state complexity of shuffle of regular languages. *Journal of Automata, Languages and Combinatorics*, 7(3):303–310, 2002.

[3] R. Cmorik and G. Jirásková. Basic operations on binary suffix-free languages. In *Proceeding of MEMICS*'11, Lecture Notes in Computer Science 7119, pages 94–102, 2011.

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

[5] M. Domaratzki and K. Salomaa. State complexity of shuffle on trajectories. *Journal of Automata, Languages and Combinatorics*, 9(2–3):217–232, 2004.

[6] Z. Ésik, Y. Gao, G. Liu, and S. Yu. Estimation of state complexity of combined operations. *Theoretical Computer Science*, 410(35):3272–3280, 2009.

[7] 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.

[8] 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.

[9] Y.-S. Han and K. Salomaa. State complexity of basic operations on suffix-free regular languages. *Theoretical Computer Science*, 410(27–29):2537–2548, 2009.

[10] Y.-S. Han, K. Salomaa, and D. Wood. Operational state complexity of prefix-free regular languages. In *Automata, Formal Languages, and Related Topics — Dedicated to Ferenc Gécseg on the occasion of his* 70*th birthday*, pages 99–115. Institute of Informatics, University of Szeged, Hungary, 2009.

[11] M. Hricko, G. Jirásková, and A. Szabari. Union and intersection of regular languages and descriptional complexity. In *Proceedings of DCFS'*05, pages 170–181. Università degli Studi di Milano, Milan, Italy, 2005.

[12] J. Jirásek and G. Jirásková. On the boundary of regular languages. In *CIAA*, pages 208–219, 2013.

[13] 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.

[14] A. Maslov. Estimates of the number of states of finite automata. *Soviet Mathematics Doklady*, 11:1373–1375, 1970.

[15] C. Nicaud. Average state complexity of operations on unary automata. In *Proceedings of MFCS'*99, Lecture Notes in Computer Science 1672, pages 231–240, 1999.

[16] 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.

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

[18] 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.

[19] K. Salomaa and S. Yu. On the state complexity of combined operations and their estimation. *International Journal of Foundations of Computer Science*, 18:683–698, 2007.

[20] J. Shallit. *A Second Course in Formal Languages and Automata Theory*. Cambridge University Press, New York, NY, USA, 2008.

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

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

[23] 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.