State Complexity of Insertion

Yo-Sub Han, Sang-Ki Ko*

Department of Computer Science, Yonsei University
50 Yonsei-Ro, Seodaemun-Gu, Seoul 120-749, Republic of Korea
{*emmous, narame7*}*@yonsei.ac.kr*

Timothy Ng, Kai Salomaa

School of Computing, Queen's University
Kingston, Ontario K7L 3N6, Canada
{*ng, ksalomaa*}*@cs.queensu.ca*

Received (Day Month Year)
Accepted (Day Month Year)
Communicated by (xxxxxxxxxx)

It is well known that the resulting language obtained by inserting a regular language to a regular language is regular. We study the nondeterministic and deterministic state complexity of the insertion operation. Given two incomplete DFAs of sizes m and n, we give an upper bound $(m+2) \cdot 2^{mn-m-1} \cdot 3^m$ and find a lower bound for an asymptotically tight bound. We also present the tight nondeterministic state complexity by a fooling set technique. The deterministic state complexity of insertion is $2^{\Theta(mn)}$ and the nondeterministic state complexity of insertion is precisely $mn+2m$, where m and n are the size of input finite automata. We also consider the state complexity of insertion in the case where the inserted language is bifix-free or non-returning.

Keywords: insertion operation, state complexity, regular languages

1. Introduction

The state complexity problem is one of the fundamental topics in automata and formal language theory [4, 9, 18, 24]. Most results are about the descriptional complexity of finite automata and regular languages. The operational state complexity is the size of a DFA required to recognize the language obtained by applying the operation to given DFAs. For example, Maslov [17] obtained the operational state complexity of catenation and Yu et al. [24] investigated the state complexity for basic operations.

Insertion is one of the crucial operations on formal languages [12–14]. The insertion $u \leftarrow v$ of a string v into a string u is to insert v in an arbitrary place in u; namely, $u \leftarrow v = \{u_1 v u_2 \mid u = u_1 u_2, \ \ u_1, u_2 \in \Sigma^*\}$. We can extend the insertion operation to languages: $L_1 \leftarrow L_2$ denotes the resulting language of inserting L_2 to

*Corresponding author

L_1. Since we insert a string in any place whereas we insert a string at the right extremity in the catenation, the insertion operation is the most natural generalization of catenation [13]. For two regular languages L_1 and L_2 recognized by DFAs with m and n states, respectively, it is known that the state complexity of the catenation L_1L_2 is $m \cdot 2^n - 2^{n-1}$ [24].

Due to the fact that insertion and deletion are inverse to each other, there have been several approaches concerning these operations. For example, researchers relied on the insertion-deletion systems in many bio-applications [2,5,15,16,21] and cryptography [1]. Recently, some of the authors investigated the state complexity of deletion and bipolar deletion [8].

Here we give state complexity bounds for the language obtained by inserting a regular language into a regular language. First, we establish the nondeterministic state complexity of insertion by presenting an NFA construction as an upper bound and finding a fooling set for the matching lower bound. The nondeterministic state complexity of insertion is $mn+2m$. Note that the nondeterministic state complexity of concatenation is $m + n$ [11]. Then, we give state complexity bounds for the language obtained by inserting a regular language into a regular language. We show that if L_1 is recognized by an incomplete DFA with m states and L_2 is recognized by an incomplete DFA with n states, we can construct an incomplete DFA for $L_1 \leftarrow L_2$ with $(m+2) \cdot 2^{mn-m-1} \cdot 3^m$ states. We also show that it is impossible to reach the upper bound with a fixed-sized alphabet and present a lower bound with a fixed-sized alphabet for an asymptotically tight bound $2^{\Theta(mn)}$.

Lastly, we consider the case when the inserted language L_2 is bifix-free or non-returning. We present a slightly improved tight bound mn if L_2 is bifix-free for the nondeterministic state complexity. We also establish a tight bound $m \cdot 2^{mn-m}$ if L_2 is a non-returning regular language.

We give the basic notations and definitions in Section 2. We present the results for nondeterministic state complexity in Section 3 and the deterministic state complexity results in Section 4. We also consider the special case where the inserted language is bifix-free or non-returning and establish tight bounds in Section 5. In Section 6, we conclude the paper.

2. Preliminaries

We assume that the reader is familiar with the basics of finite automata and formal languages and recall here some definitions and notation. For a general introduction to the topic the reader may consult the textbooks [20, 22] or the survey [23]. More information on state complexity of operations can be found in the survey [7].

In the following Σ always stands for a finite alphabet and the set of strings over Σ is Σ^*. A language is a subset of Σ^*. The cardinality of a finite set S is denoted $|S|$. For strings x, y and z, we say that x is a *prefix* of y and z is a *suffix* of y if $y = xz$. We define a (regular) language L to be *prefix-free* (*suffix-free*) if a string $x \in L$ is not a prefix (suffix) of any other strings in L. We say that L is *bifix-free* if

L is prefix-free and suffix-free.

The set of strings obtained by inserting a string $v \in \Sigma^*$ to the string $u \in \Sigma^*$ is

$$u \leftarrow v = \{u_1 v u_2 \mid u = u_1 u_2,\ \ u_1, u_2 \in \Sigma^*\}.$$

The result of the insertion is a set of strings instead of a single string. For example, $abc \leftarrow d = \{dabc, adbc, abdc, abcd\}$, where $a, b, c, d \in \Sigma$.

The *insertion operation* is extended in the natural way for languages $L_1, L_2 \subseteq \Sigma^*$ by setting

$$L_1 \leftarrow L_2 = \bigcup_{u \in L_1, v \in L_2} u \leftarrow v.$$

A *nondeterministic finite automaton with λ-transitions* (λ-NFA) is a five-tuple $A = (Q, \Sigma, Q_0, F, \delta)$ where Q is a finite set of states, Σ is a finite alphabet, $Q_0 \subseteq Q$ is the set of initial states, $F \subseteq Q$ is the set of final states and δ is a multi-valued transition function from $Q \times (\Sigma \cup \{\lambda\})$ into 2^Q. For a transition $p \in \delta(q, a)$ in A, we say that q has an *out-transition* and p has an *in-transition*.

By an NFA we mean a nondeterministic automaton without λ-transitions, that is, A is an NFA if δ is a function from $Q \times \Sigma$ into 2^Q. The automaton A is *deterministic* (a DFA) if Q_0 is a singleton set and δ is a (total single-valued) function $Q \times \Sigma \to Q$. A DFA is *incomplete* if δ is a partial function $Q \times \Sigma \to Q$. A *sink state* (also called as a dead state) is a non-final state q such that, for all characters $a \in \Sigma$, $\delta(q, a) = q$. This implies that there is no way to reach a final state whenever a DFA enters a sink state.

For $q \in Q$, $P \subseteq Q$, $b \in \Sigma$ and $L \subseteq \Sigma^*$ we also denote

$$\delta(P, b) = \{\delta(p, b) \mid p \in P\} \quad \text{and} \quad \delta(q, L) = \{\delta(q, w) \mid w \in L\}.$$

The transition function δ is defined as a partial function $2^Q \times \Sigma \to 2^Q$ in the former notation while $Q \times \Sigma^* \to 2^Q$ in the latter. We say that an NFA (or a DFA) A is *non-returning* if the start state of A does not have any in-transitions and A is *non-exiting* if every final state in A has no out-transition. We assume that A has only useful states; namely, all states of A are reachable from the start state.

Note that λ-NFAs, NFAs and DFAs all recognize regular languages [19, 20, 22].

Proposition 1 ([22]) *A λ-NFA has an equivalent NFA without λ-transitions and the same number of states.*

The (right) *Kleene congruence* of a language $L \subseteq \Sigma^*$ is the relation $\equiv_L \subseteq \Sigma^* \times \Sigma^*$ defined by setting, for $x, y \in \Sigma^*$,

$$x \equiv_L y \text{ iff } [(\forall z \in \Sigma^*)\ xz \in L \Leftrightarrow yz \in L].$$

It is well known that L is regular if and only if the index of $\equiv_L$ is finite and, in this case, the number of classes of $\equiv_L$ is equal to the size of the minimal DFA for L. Note that we define the size of an automaton to be the number of its states.

The deterministic (respectively, nondeterministic) state complexity of a regular language L, $\mathrm{sc}(L)$ (respectively, $\mathrm{nsc}(L)$) is the size of the minimal DFA (respectively,

the size of a minimal NFA) recognizing L. The *incomplete state complexity* of L, $\mathrm{isc}(L)$, is the size of the minimal incomplete DFA recognizing L. Here we only consider the incomplete case for the deterministic state complexity. For each regular language L either $\mathrm{sc}(L) = \mathrm{isc}(L) + 1$ or $\mathrm{sc}(L) = \mathrm{isc}(L)$. Note that $\mathrm{sc}(L)$ is equal to the number of classes of $\equiv_L$.

The nondeterministic state complexity of a language can be estimated using the *fooling set technique* that gives a lower bound for the size of NFAs [3].

Proposition 2 ([3, 20]) *Let $L \subseteq \Sigma^*$ be a regular language. Suppose that there exists a set $P = \{(x_i, w_i) \mid 1 \leq i \leq n,\ \ x_i, w_i \in \Sigma^*\}$ of pairs such that*

(i) $x_i w_i \in L$ for $1 \leq i \leq n$;
(ii) if $i \neq j$, then $x_i w_j \notin L$ or $x_j w_i \notin L$, $1 \leq i, j \leq n$.

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

The set P satisfying the conditions of Proposition 2 is called a *fooling set* for the language L.

3. Nondeterministic State Complexity of Insertion

We establish the precise nondeterministic state complexity of the insertion operation. For the lower bound construction the languages can be defined over a binary alphabet. The result is best possible because over a unary alphabet insertion coincides with concatenation and the nondeterministic state complexity of concatenation is $m + n$ [11].

Lemma 3. *Consider $L_1, L_2 \subseteq \Sigma^*$ where L_1 is recognized by an NFA with m states and L_2 is recognized by an NFA with n states. Then,*

$$\mathrm{nsc}(L_1 \leftarrow L_2) \leq mn + 2m.$$

Proof. Let $A = (Q, \Sigma, \delta, q_0, Q_F)$ and $B = (P, \Sigma, \delta', p_0, P_F)$ be NFAs for L_1 and L_2, respectively, where $|Q| = m$ and $|P| = n$. We define a λ-NFA $C = (S, \Sigma, \gamma, q_0, \overline{Q_F})$ for the language $L_1 \leftarrow L_2$ where $S = Q \cup \underbrace{P_0 \cup \cdots \cup P_{m-1}}_{m \text{ copies of } P} \cup \overline{Q}$.

Here $P_i = \{p_{j,i} \mid p_j \in P,\ \ 0 \leq j \leq n-1\}, 0 \leq i \leq m-1$ is the ith copy of the state set of B. We denote the ith copy of the NFA B as follows:

$$B_i = (P_i, \Sigma, \delta_i, p_{0,i}, P_{F,i}).$$

We also note that the state set $\overline{Q}$ is a copy of the state set $Q = \{q_i \mid 0 \leq i \leq m-1\}$ of A and used to continue the computation of A after simulating a string in L_2 by one of m copies of B. Here $\overline{Q} = \{\overline{q_i} \mid q_i \in Q,\ \ 0 \leq i \leq m-1\}$. The set of final states of C is $\overline{Q_F}$, where $\overline{Q_F} = \{\overline{q_f} \mid q_f \in Q_F\}$.

Then, the transition function γ is defined as follows:

(i) For all $q_i, q_j \in Q, 0 \leq i, j \leq m-1$ and $a \in \Sigma$, if $q_j \in \delta(q_i, a)$, then $q_j \in \gamma(q_i, a)$.

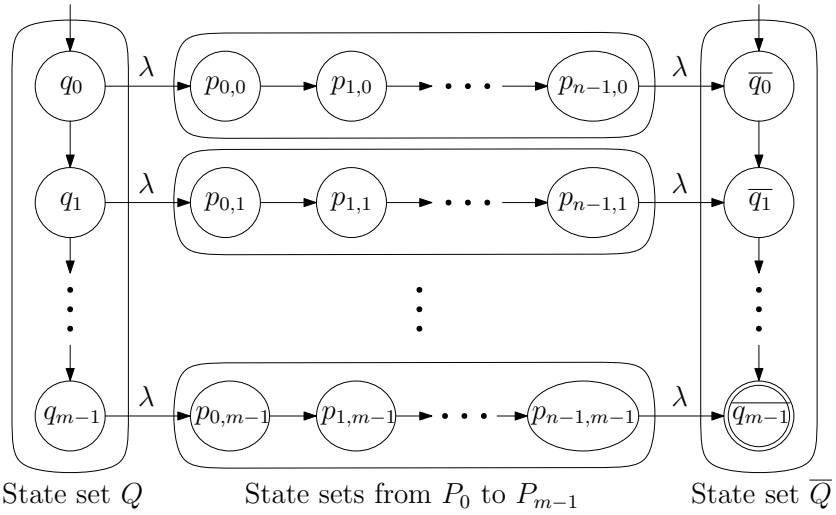

Fig. 1: The NFA C constructed from two NFAs $A = (Q, \Sigma, \delta, q_0, \{q_{m-1}\})$ and $B = (P, \Sigma, \delta', p_0, \{p_{n-1}\})$, where $Q = \{q_i \mid 0 \leq i \leq m-1\}$ and $P = \{p_i \mid 0 \leq i \leq n-1\}$.

(ii) For all $\overline{q_i}, \overline{q_j} \in \overline{Q}, 0 \leq i, j \leq m-1$ and $a \in \Sigma$, if $q_j \in \delta(q_i, a)$, then $\overline{q_j} \in \gamma(\overline{q_i}, a)$.
(iii) For all $q_i \in Q, 0 \leq i \leq m-1$, $p_{0,i} \in \gamma(q_i, \lambda)$.
(iv) For all $p_{j,i}, p_{k,i} \in P_i, 0 \leq i \leq m-1, 0 \leq j, k \leq n-1$ and $a \in \Sigma$, if $p_{k,i} \in \delta_i(p_{j,i}, a)$, then $p_{k,i} \in \gamma(p_{j,i}, a)$.
(v) For all $p_{j,i} \in P_{F,i}, 0 \leq i \leq m-1, 0 \leq j \leq n-1$, $\overline{q_i} \in \gamma(p_{j,i}, \lambda)$.

We give a graphical description of the NFA C in Fig. 1. The NFA C operates as follows. The computation of A begins at the initial state of the original state set Q. For any state $q \in Q$, we nondeterministically move to one of the copies of the NFA B using a λ-transition. After the computation of a string in L_2 in the copied NFA, again we nondeterministically move to the copied set $\overline{Q}$ of the NFA A. This implies that if we are at a state of $\overline{Q}$, then the insertion of a string of L_2 has been already done. Therefore, the simulation is completed if we arrive at a final state of the copied set $\overline{Q}$ of final states. The constructed NFA C has λ-transitions. The claim follows by Proposition 1. □

Lemma 4. *Let $\Sigma = \{a, b\}$. For $m, n \in \mathbb{N}$, there exist an incomplete DFA A with m states and an incomplete DFA B with n states over Σ such that any NFA for $L(A) \leftarrow L(B)$ needs $mn + 2m$ states.*

Proof. Choose $L_1 = (a^m)^*$ and $L_2 = (b^n)^*$. Since L_1 (respectively, L_2) has an incomplete DFA with m (respectively, n) states it is sufficient to give a fooling set of size $mn + 2m$ for $L_1 \leftarrow L_2$.

Define $S \subseteq \Sigma^* \times \Sigma^*$ to consist of pairs of strings listed in (i), (ii) and (iii) below.

(i) $(a^i b^{n+j},\ b^{n-j}a^{m-i})$, $0 \leq i < m$, $0 \leq j < n$.
(ii) $(a^i, a^{2m-i}b^n)$, $0 \leq i < m$.
(iii) $(b^n a^{2m-i}, a^i)$, $0 \leq i < m$.

We verify that S is a fooling set. Consider two distinct pairs of S, $P_1 = (X_1, Y_1)$ and $P_2 = (X_2, Y_2)$. If P_1 and P_2 are both of the type (i) (respectively, both of type (ii); both of type (iii)), then the number of a's in $X_1 \cdot Y_2$ is not divisible by m or the number of b's in $X_1 \cdot Y_2$ is not divisible by n and $X_1 \cdot Y_2$ is not in $L_1 \leftarrow L_2$.

If P_1 is of type (i) and P_2 is of type (ii), we note that $X_1 \cdot Y_2$ is not in $a^*b^*a^*$ and hence not in $L_1 \leftarrow L_2$. Similarly, if P_1 is of type (i) and P_2 is of type (iii), $X_2 \cdot Y_1$ is not in $a^*b^*a^*$. Finally, if P_1 is of type (ii) and P_2 is of type (iii), $X_2 \cdot Y_1$ is again not in $a^*b^*a^*$.

The above are all possible cases up to symmetry between P_1 and P_2 and we have verified that S is a fooling set for $L_1 \leftarrow L_2$. □

As a consequence of Lemma 3 and Lemma 4 we have:

Theorem 5. *For languages $L_1, L_2 \subseteq \Sigma^*$ where L_1 and L_2 are regular,*

$$\mathrm{nsc}(L_1 \leftarrow L_2) \leq mn + 2m$$

and this bound can be reached in the worst-case.

4. Deterministic State Complexity of Insertion

From the nondeterministic state complexity of insertion, we already have an upper bound 2^{mn+2m} for the deterministic state complexity. Here we present an upper bound construction with incomplete DFAs which is slightly improved from 2^{mn+2m}.

Lemma 6. *Consider $L_1, L_2 \subseteq \Sigma^*$ where L_1 is recognized by an incomplete DFA with m states and L_2 is recognized by an incomplete DFA with n states. Then,*

$$\mathrm{isc}(L_1 \leftarrow L_2) \leq (m+2) \cdot 2^{mn-m-1} \cdot 3^m.$$

Proof. Let $A = (Q, \Sigma, \delta, q_0, Q_F)$ and $B = (P, \Sigma, \delta', p_0, P_F)$ be incomplete DFAs for L_1 and L_2, respectively, where $|Q| = m$ and $|P| = n$. We define the completion of δ as a function $\overline{\delta} : (Q \cup \{\mathrm{sink}\}) \times \Sigma \to Q \cup \{\mathrm{sink}\}$ by setting for $r \in Q \cup \{\mathrm{sink}\}$ and $b \in \Sigma$,

$$\overline{\delta}(r, b) = \begin{cases} \delta(r, b), & \text{if } r \in Q \text{ and } \delta(r, b) \text{ is defined;} \\ \mathrm{sink}, & \text{otherwise.} \end{cases}$$

We define a DFA $C = (T, \Sigma, \gamma, t_0, T_F)$, where

- $T = \{(q, R, S) \mid q \in Q \cup \{\mathrm{sink}\}, R \in 2^{P \times Q}, S \in 2^Q\}$,
- $t_0 = \{(q_0, \{(p_0, q_0)\}, S) \mid S = \emptyset$ if $p_0 \notin P_F$ otherwise $\{q_0\}\}$, and
- $T_F = \{(q, R, S) \mid S \cap Q_F \neq \emptyset\}$.

It remains to define the transitions of γ. For $(q, R, S) \in T$ and $a \in \Sigma$, we set

$$\gamma((q, R, S), a) = (q', X, Y),$$

where

- $q' = \overline{\delta}(q, a)$,
- $X = \{(\delta'(p, a), r) \mid (p, r) \in R\} \cup \{(p_0, \delta(q, a))\}$, and
- $Y = \delta(S, a) \cup \{r \in Q \mid (\exists (s, r) \in R)\delta'(s, a) \in P_F\}$.

The transitions of C operate as follows. Consider a state $(q, R, S) \in T$. The first component q simulates the computation of A on a string where no insertion has taken place. The elements of R keep track of the computation of B on an inserted string while at the same time remembering the state of A before the corresponding string was inserted. On an element $(p, r) \in R$, the first component simulates the transitions of B and the second component keeps track of the state of A at the point where the simulated computation of A began. The pair $(p_0, \delta(q, a))$ is added to take care of the situation where the string of L_2 is inserted directly after the current symbol a. The computation of A after inserting a string of L_2 is continued on states of S. Additionally, of each pair $(p, r) \in R$ where the first component simulates a computation of B, if input symbol a takes this state to a final state, we add r to the component Y.

The states of Y keep track of all possible computations of A where a string of L_2 has been inserted previously, and the choice of the set of final states then guarantees that C recognizes the language $L_1 \leftarrow L_2$.

Note that we only consider the states reachable from the initial state t_0. For each $q \in Q$ and $(q, R, S) \in T$, R should contain (p_0, q). Therefore, there exist $m \cdot 2^{mn-1}$ combinations for the first and second components. If $q = \text{sink}$, we have 2^{mn} more combinations since we can have any set of state pairs for R. Now we have

$$m \cdot 2^{mn-1} + 2^{mn} = (m + 2) \cdot 2^{mn-1}$$

combinations for the first and second components of the states of T. We calculate the number of reachable states of C by considering the number of combinations that we can have for the third component. Note that if R has a pair (p, q) where $p \in P, q \in Q$ such that $p \in P_F$, then S should have q in the set to continue the computation of A after the insertion.

Assume that B has l final states. Let us define a natural number k as follows:

$$k = |\{q \mid (p, q) \in R, p \in P_F\}|.$$

The set S should contain the k states by the definition of the transition function γ. Since the number k can be from 0 to m, we have the following number of reachable

states:

$$(m+2)\cdot 2^{mn-ml-1}\cdot\sum_{k=0}^{m}\binom{m}{k}\cdot(2^l-1)^k\cdot 2^{m-k} = (m+2)\cdot 2^{mn-ml-1}\cdot(2^l+1)^m$$
$$= (m+2)\cdot 2^{mn-1}\cdot\left(\frac{2^l+1}{2^l}\right)^m$$
$$= (m+2)\cdot 2^{mn-1}\cdot\left(1+\frac{1}{2^l}\right)^m,$$

which is maximal if $l=1$. Therefore, the upper bound for the state complexity of the insertion is

$$(m+2)\cdot 2^{mn-m-1}\cdot 3^m. \qquad \square$$

Before we discuss the lower bound of the state complexity, we first prove that it is impossible to reach the upper bound with a fixed-sized alphabet.

Theorem 7. *Let A be a DFA with m states and B be a DFA with n states. Then, there exists a DFA C recognizing $L(A) \leftarrow L(B)$ with*

$$(m+2)\cdot 2^{mn-m-1}\cdot 3^m$$

states and this bound cannot be reached with a fixed-sized alphabet.

Proof. Let $A=(Q,\Sigma,\delta,q_0,Q_F)$ and $B=(P,\Sigma,\delta',p_0,P_F)$ be incomplete DFAs, where $|Q|=m$ and $|P|=n$. Suppose $|\Sigma|<m-1$. Since there are fewer than m letters, there must be a state $k\neq q_0$ and $k\in Q$ that is not reachable from the initial state q_0 by reading any character in Σ. In other words, $k\neq\delta(q_0,a)$ for all $a\in\Sigma$.

Now we construct a DFA C as in the proof of Lemma 6. Consider a state (k,R,S) of C, where

$$R=\{(p,q)\mid p\in P,\ \ q\in\{q_0,k\}\}.$$

Assume that we can reach the state (k,R,S) from the initial state of C by reading a string w. Since k cannot be reached from q_0 by reading a string of length 1 by assumption, there should be a longest prefix w' of w that makes C proceed to the state (k',R',S'), where $k'\notin\{q_0,k\}$.

Now $(p_0,k')\in R'$ but recall that $(p,k')\notin R$ for all $p\in P$. This implies that we need to remove the state (p_0,k') by reading the remaining suffix of w. However, if we remove the state (p_0,k') from R', then we also remove all the state pairs with the same first components such as (p_0,q_0). Moreover, since the string w' is the longest prefix of w, there is no chance to restore the pair. Therefore, it is impossible to reach the state (k,R,S) if we have fewer than $m-1$ letters. Since the state (k,R,S) is included in the upper bound $(m+2)\cdot 2^{mn-m-1}\cdot 3^m$ claimed in Lemma 6, the theorem holds. $\square$

Next we give a lower bound example that asymptotically reaches the upper bound $2^{O(mn)}$ using a fixed-sized alphabet.

Lemma 8. *For every $m, n \geq 3$ there exist incomplete DFAs A and B over the alphabet $\Sigma = \{a, b, c, d, e\}$ with m and n states, respectively, such that*

$$\mathrm{isc}(L(A) \leftarrow L(B)) \geq m \cdot 2^{mn-m}.$$

Proof. Now we show that there exists a lower bound example with a fixed-sized alphabet that asymptotically reaches the upper bound.

Choose $A = (Q, \Sigma, \delta, 0, \{0\})$ where $Q = \{0, 1, \ldots, m-1\}$ and the transitions of δ are defined by setting

- $\delta(i, a) = i + 1$ for $0 \leq i \leq m-2$ and $\delta(m-1, a) = 0$;
- $\delta(i, b) = i$ for $0 \leq i \leq m-1$;
- $\delta(i, c) = i$ for $0 \leq i \leq m-1$;
- $\delta(i, d) = i$ for $0 \leq i \leq m-1$.

We also choose $B = (P, \Sigma, \delta', 0, \{n-1\})$ where $P = \{0, 1, \ldots, n-1\}$ and the transitions of δ' are defined by setting

- $\delta'(i, a) = i$ for $1 \leq i \leq n-2$;
- $\delta'(i, b) = i + 1$ for $1 \leq i \leq n-2$;
- $\delta'(0, c) = 1$ and $\delta'(i, c) = i$ for $1 \leq i \leq n-2$;
- $\delta'(i, e) = i$ for $1 \leq i \leq n-2$.

The DFAs A and B are depicted in Fig. 2 and Fig. 3.

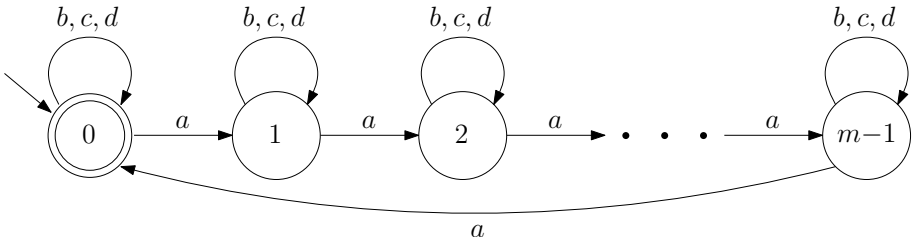

Fig. 2: The incomplete DFA A used in the proof of Lemma 8.

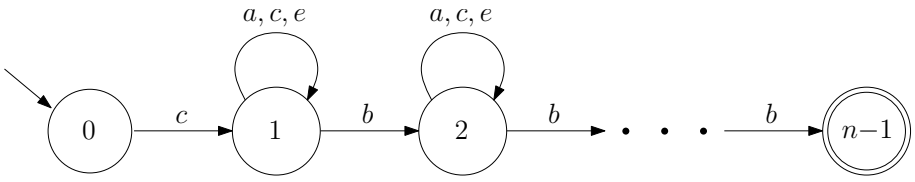

Fig. 3: The incomplete DFA B used in the proof of Lemma 8.

Let $C = (T, \Sigma, \gamma, t_0, T_F)$ be the incomplete DFA recognizing the language $L(A) \leftarrow L(B)$ constructed as in the proof of Lemma 6. Recall that a state $t = (r, R, S)$ of C is a triple, where $r \in Q$ is a state (including the sink state) of A, $R \subseteq 2^{P \times Q}$ is a set of state pairs, and S is a set of states of A. Here we only consider the reachability of states (r, R, S) of C, where $r \in Q$, R is a set of state pairs $(p, q) \in (P \setminus \{0\}) \times Q$. By considering only these combinations, we can show that at least $m \cdot 2^{mn-m}$ states are reachable in the new DFA C.

Now we show that $m \cdot 2^{mn-m}$ states are reachable from the initial state $t_0 = (0, \{(0, 0)\}, \emptyset)$. For $r \in Q$ and $R \in 2^{(P \setminus \{0\}) \times Q}$, we claim that there exists a state $t = (r, R \cup R', S) \in T$, where $R' \in 2^{\{0\} \times Q}$ and $S \in 2^Q$, which is reachable from the initial state t_0.

We first denote a subset $R_i \subseteq R$ of state pairs in which the first component of the pair is i by R_i. Formally, we define

$$R_i = \{(i, q) \mid (i, q) \in R\}, \;\; 1 \leq i \leq n - 1.$$

We also define

$$Y_i = \{q \mid (i, q) \in R\}, \;\; 1 \leq i \leq n - 1.$$

Lastly, we define

$$R^{(k)} = \bigcup_{j=1}^{k} (\{j\} \times Y_{n-k+j-1})$$

to represent the second component of the kth intermediate state in the path from the initial state $t_0 = (0, \{(0, 0)\}, \emptyset)$ to the state $t = (r, R \cup R', S)$. Note that $R^{(0)} = \emptyset$ and $R^{(n-1)} = R$ since

$$R^{(n-1)} = \bigcup_{j=1}^{n-1} (\{j\} \times Y_j).$$

We show how we reach a state where the second component contains $R^{(n-1)}$ as a subset from a state where the second component contains $R^{(0)}$ as a subset.

Let $Y_{n-1} = \{s_1, s_2, \ldots, s_l\}$, where $0 \leq s_1 < s_2 < \cdots < s_l \leq m - 1$. By reading a string

$$a^{s_1} c a^{s_2 - s_1} c \cdots c a^{s_{l-1} - s_{l-2}} c a^{s_l - s_{l-1}} c,$$

we make the second component of the state contain $R^{(1)}$ as a subset. Now we are at a state where the first component is s_l and the second component contains $R^{(1)}$. Let $Y_{n-2} = \{z_1, z_2, \ldots, z_h\}$, where $0 \leq z_1 < z_2 < \cdots < z_h \leq m - 1$. First we move the first component back to 0 by reading $a^{m - s_l}$. After, we read b to shift the set of pairs R_1 of states in the second component to R_2. Then, we can let the second component of the state to contain $R^{(2)}$ as a subset by reading a string

$$a^{z_1} c a^{z_2 - z_1} c \cdots c a^{z_{h-1} - z_{h-2}} c a^{z_h - z_{h-1}} c.$$

By repeating the above steps, we can reach the state where the second component contains $R^{(n-1)}$ as a subset. After then, we can also move the first component of the state to r by reading a sequence of a's.

Since the number of combinations for the first and the second components is $m \cdot 2^{mn-m}$, we can say that at least $m \cdot 2^{mn-m}$ states of C are reachable and it remains to show that they are all pairwise inequivalent.

First consider two states $(r, R, S), (r', R', S') \in T$ where $S \neq S'$. Without loss of generality, we find $s \in S \setminus S'$ since the other possibility is completely symmetric. By reading d from the two states, we can remove all state pairs except $(0, r)$ and $(0, r')$ from R and R', respectively. This step ensures that we do not change the elements of S and S' by the effect of the first or second second component if we read a sequence of a. Then, the state (r, R, S) moves to a final state by reading a^{m-s} while the state (r', R', S') moves to a non-final state.

Now we consider two states $(r, R, S), (r', R', S') \in T$, where $R \neq R'$. As we have defined for the set R above, we use similar notations for R and R' as follows:

$$R_i = \{(i, q) \mid (i, q) \in R\} \text{ and } R'_i = \{(i, q) \mid (i, q) \in R'\}, \;\; 1 \leq i \leq n-1.$$

We also define

$$Y_i = \{q \mid (i, q) \in R\} \text{ and } Y'_i = \{q \mid (i, q) \in R'\}, \;\; 1 \leq i \leq n-1.$$

Since $R \neq R'$, there should exist $0 \leq i \leq n-1$ such that $R_i \neq R'_i$. Note that $Y_i \neq Y'_i$ because $R_i \neq R'_i$. We read b^{n-i-2} from the two states (r, R, S) and (r', R', S'). Now the first components of the state pairs in R_i and R'_i become $n-2$ and, therefore, $R_{n-2} \neq R'_{n-2}$ and $Y_{n-2} \neq Y'_{n-2}$. We now move the first component of states to the sink state and empty the third component of states by reading a character e, which is undefined in A. Then, we move the first components of the state pairs in R_{n-2} and R'_{n-2} to $n-1$ by reading the symbol b while the third components become Y_{n-2} and Y'_{n-2}, respectively.

Since we have already shown that two states are pairwise inequivalent if the third components of the states are inequivalent, the two states $(r, R, S), (r', R', S'), R \neq R'$ are also pairwise inequivalent.

Lastly, we consider two states $(r, R, S), (r', R', S')$, where $r \neq r'$. Without loss of generality, we assume $r < r'$.

As there is no transition defined for the character d in B, the two states (r, R, S) and (r', R', S') become $(r, \{(0, r)\}, S)$ and $(r', \{(0, r')\}, S')$ by reading d. Since we have shown that two states are pairwise inequivalent if the second components are inequivalent, two states $(r, R, S), (r', R', S')$ with $r \neq r'$ are also pairwise inequivalent.

We have shown that at least $m \cdot 2^{mn-m}$ states of C are reachable and they are all pairwise inequivalent. Therefore, the deterministic state complexity of insertion has a lower bound of $m \cdot 2^{mn-m}$. □

As a consequence of Lemma 6 and Lemma 8 we have:

Theorem 9. *For languages $L_1, L_2 \subseteq \Sigma^*$ where L_1 and L_2 are regular,*

$$\mathrm{isc}(L_1 \leftarrow L_2) \in 2^{\Theta(mn)}.$$

Note that the deterministic state complexity of insertion is strictly worse than the deterministic state complexity of concatenation $m \cdot 2^n - 2^{n-1}$.

5. Special Case: if L_2 is bifix-free or non-returning

Now we briefly consider the case when the inserted language L_2 is contained in a subclass of regular languages called *bifix-free regular languages*.

A regular language L is prefix-free if and only if any DFA accepting L is non-exiting. For suffix-free regular languages, we have a restriction that DFAs accepting suffix-free regular languages should be non-returning, but the converse does not always hold [10]. Since the non-returning property is only a necessary condition for a minimal DFA to be suffix-free, we call a family of regular languages where the minimal DFAs accepting the languages are non-returning, *non-returning regular languages* [6].

For NFAs, any NFA accepting a prefix-free regular language should be non-exiting and any NFA accepting a suffix-free regular language should be non-returning. We also mention that the converse does not always hold.

We first present the nondeterministic state complexity of insertion.

Theorem 10. *For languages $L_1, L_2 \subseteq \Sigma^*$ where L_1 and L_2 are regular and L_2 is bifix-free,*

$$\mathrm{nsc}(L_1 \leftarrow L_2) \leq mn$$

and this bound can be reached in the worst-case.

Proof. The upper bound mn is immediate from the proof of Lemma 3. See Fig. 1 for example. Since a bifix-free NFA is always non-exiting and non-returning, the initial state of the NFA has no in-transition and the only final state of the NFA has no out-transition. Therefore, we can merge m state pairs q_i and $p_{0,i}$ for $0 \leq i \leq m-1$ as the states $p_{0,i}, 0 \leq i \leq m-1$ have no in-transitions. We can also merge m state pairs $p_{n-1,i}$ and $\overline{q_i}$ for $0 \leq i \leq m-1$ as the states $p_{n-1,i}, 0 \leq i \leq m-1$ have no in-transitions.

After merging $2m$ pairs of states, we have an upper bound $mn + 2m - 2m = mn$ for the nondeterministic state complexity of insertion.

Now we give a matching lower bound with a fooling set technique.

Choose $L_1 = (a^m)^*$ and $L_2 = c(b^{n-2})^*c$. Since L_1 (respectively, L_2) has an incomplete DFA with m (respectively, n) states it is sufficient to give a fooling set of size mn for $L_1 \leftarrow L_2$.

Define $S \subseteq \Sigma^* \times \Sigma^*$ to consist of pairs of strings listed in (i), (ii) and (iii) below.

(i) $(a^i c b^{n+j-2},\ b^{n-j-2} c a^{m-i})$, $0 \leq i < m$, $0 \leq j < n-2$.

(ii) $(a^i, a^{2m-i}cb^{n-2}c)$, $0 \leq i < m$.
(iii) $(cb^{n-2}ca^{2m-i}, a^i)$, $0 \leq i < m$.

We verify that S is a fooling set. Consider two distinct pairs of S, $P_1 = (X_1, Y_1)$ and $P_2 = (X_2, Y_2)$. If P_1 and P_2 are both of the type (i) (respectively, both of type (ii); both of type (iii)), then the number of a's in $X_1 \cdot Y_2$ is not divisible by m or the number of b's in $X_1 \cdot Y_2$ is not divisible by $n-2$ and $X_1 \cdot Y_2$ is not in $L_1 \leftarrow L_2$.

If P_1 is of type (i) and P_2 is of type (ii), we note that $X_1 \cdot Y_2$ is not in $a^*cb^*ca^*$ and hence not in $L_1 \leftarrow L_2$. Similarly, if P_1 is of type (i) and P_2 is of type (iii), $X_2 \cdot Y_1$ is not in $a^*cb^*ca^*$. Finally, if P_1 is of type (ii) and P_2 is of type (iii), $X_2 \cdot Y_1$ is again not in $a^*cb^*ca^*$.

The above are all possible cases up to symmetry between P_1 and P_2 and we have verified that S is a fooling set for $L_1 \leftarrow L_2$. Moreover, since S has mn pairs of strings, we have a matching lower bound for the claimed upper bound. □

We also consider the deterministic state complexity of insertion when L_2 is bifix-free. Since a bifix-free DFA is always non-returning and non-exiting, we present an upper bound for non-returning and non-exiting incomplete DFAs.

Theorem 11. *For languages $L_1, L_2 \subseteq \Sigma^*$ where L_1 and L_2 are regular and L_2 is non-returning and non-exiting,*

$$\mathrm{isc}(L_1 \leftarrow L_2) \leq m \cdot 2^{mn-m}$$

and this bound can be reached in the worst-case.

Proof. Recall the upper bound construction used in Lemma 6. Here we assume that the DFA B is non-returning and non-exiting. Since a non-exiting DFA has a single final state, we say $P_F = \{p_F\}$. Now we show that some states of C are unreachable or pairwise equivalent by the construction and the properties of bifix-free DFAs.
Claim 1. All states $(q, R, S) \in T$, where $R \cap \{(p_0, q') \mid q' \in Q, \quad q' \neq q\} \neq \emptyset$, are unreachable from the initial state t_0.
Claim 2. Any two states $(q, R, S) \in T$ and $(q, R \setminus R', S) \in T$, where $R' = \{(p_F, q') \mid q' \in Q\}$, are pairwise equivalent.

Proof of **Claim 1**: Since the DFA B is non-returning, the initial state p_0 of B has no in-transition. This implies that any state pair of the form (p_0, q), where $q \in Q$, in the second component should be removed by reading any character in Σ. The only way to have a state pair of the form (p_0, q) is to have the state q in the first component. Therefore, the claim holds.

Proof of **Claim 2**: Since the DFA B is non-exiting, all state pairs in R' are always removed from the second component by reading any character in Σ. Moreover, a state pair in the second component of the state of C can generate a state in the third component only when the first component of the state pair becomes the only final state p_F of B. Therefore, the claim holds.

By the above claims, we have at most $m \cdot 2^{mn-m}$ reachable and pairwise inequivalent states by the upper bound construction if the DFA accepting L_2 is non-returning and non-exiting.

Note that the DFA B used in the proof of Lemma 8 is non-returning and non-exiting. Therefore, we establish the tight bound $m \cdot 2^{mn-m}$ for the deterministic state complexity of insertion when L_2 is accepted by a non-returning and non-exiting DFA. □

Since a bifix-free DFA is always non-returning and non-exiting, we have a following corollary as a result.

Corollary 12. *For languages $L_1, L_2 \subseteq \Sigma^*$ where L_1 and L_2 are regular and L_2 is bifix-free,*

$$\mathrm{isc}(L_1 \leftarrow L_2) \leq m \cdot 2^{mn-m}.$$

As a final note, we mention that the tight bound for the deterministic state complexity of insertion when L_2 is bifix-free still remains open.

6. Conclusions

Insertion is a fundamental operation on strings and languages and has been used in practice such as bio-applications or cryptography. Regular languages are closed under insertion. This has led us to examine the state complexity of regular languages with respect to insertion.

We have studied tight state complexity bounds for the insertion operation. For the nondeterministic state complexity, we have given an upper bound construction with $mn+2m$ states and a matching fooling set to show the tightness of the bound.

For the deterministic state complexity, we have considered the case where L_1 and L_2 can be recognized by incomplete DFAs with m and n states, respectively. We have established an upper bound construction of $(m+2) \cdot 2^{mn-m-1} \cdot 3^m$ states and proved that it is impossible to reach the upper bound using a fixed-sized alphabet. In addition, we have presented a lower bound with a fixed-sized alphabet for an asymptotic tight bound $2^{\Theta(mn)}$. We leave the problem of finding a tight lower bound with a fixed-sized alphabet or a new upper bound construction open for future research.

In addition, we have investigated the case when the inserted language L_2 is bifix-free or non-returning. When L_2 is bifix-free, we have a slightly improved tight bound mn for the nondeterministic state complexity. We also have a tight bound $m \cdot 2^{mn-m}$ even with a fixed-sized alphabet if L_2 is a non-returning regular language. Note that the tight bound on the deterministic state complexity when L_2 is bifix-free is an open problem.

Acknowledgments

We thank the anonymous referees for a careful reading of an earlier version of the paper and many useful suggestions that have improved the presentation.

Han and Ko were supported by the Basic Science Research Program through NRF funded by MEST (2015R1D1A1A01060097), the Yonsei University Future-leading Research Initiative of 2015 and the International Cooperation Program managed by NRF of Korea (2014K2A1A2048512), and Ng and Salomaa were supported by the Natural Sciences and Engineering Research Council of Canada Grant OGP0147224.

References

[1] M. Andraşiu, G. Păun, J. Dassow, and A. Salomaa. Language-theoretic problems arising from richelieu cryptosystems. *Theoretical Computer Science*, 116(2):339–357, 1993.

[2] F. Biegler, M. J. Burrell, and M. Daley. Regulated RNA rewriting: Modelling RNA editing with guided. *Theoretical Computer Science*, 387(2):103–112, 2007.

[3] J.-C. Birget. Intersection and union of regular languages and state complexity. *Information Processing Letters*, 43(4):185–190, 1992.

[4] C. Câmpeanu, K. Culik II, K. Salomaa, and S. Yu. State complexity of basic operations on finite languages. In *Proceedings of the 4th International Workshop on Implementing Automata*, 60–70, 2001.

[5] M. Daley, L. Kari, G. Gloor, and R. Siromoney. Circular contextual insertions/deletions with applications to biomolecular computation. In *Proceedings of the String Processing and Information Retrieval Symposium, 1999 and International Workshop on Groupware*, 47–54, 1999.

[6] H.-S. Eom, Y.-S. Han, and G. Jirásková. State complexity of basic operations on non-returning regular languages. *Fundamenta Informaticae*. To appear.

[7] Y. Gao, N. Moreira, R. Reis, and S. Yu. A review on state complexity of individual operations. Technical report, Technical report, Universidade do Porto, Technical Report Series DCC-2011-08, Version 1.1 (September 2012).

[8] Y.-S. Han, S.-K. Ko, and K. Salomaa. State complexity of deletion and bipolar deletion. *Acta Informatica*, 53:67–85, 2016.

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

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

[12] L. Kari. *On insertion and deletion in formal languages*. PhD thesis, University of Turku, 1991.

[13] L. Kari. Insertion operations: Closure properties. *Bulletin of the EATCS*, 51:181–191, 1993.

[14] L. Kari. On language equations with invertible operations. *Theoretical Computer Science*, 132(1-2):129–150, 1994.

[15] A. Krassovitskiy, Y. Rogozhin, and S. Verlan. Computational power of insertion-deletion (p) systems with rules of size two. *Natural Computing*, 10(2):835–852, 2011.

[16] M. Margenstern, G. Păun, Y. Rogozhin, and S. Verlan. Context-free insertion-deletion systems. *Theoretical Computer Science*, 330(2):339–348, 2005.
[17] A. Maslov. Estimates of the number of states of finite automata. *Soviet Mathematics Doklady*, 11:1373–1375, 1970.
[18] 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.
[19] G. Rozenberg and A. Salomaa, editors. *Handbook of Formal Languages, Vol. 3: Beyond Words*. Springer-Verlag New York, Inc., 1997.
[20] J. Shallit. *A Second Course in Formal Languages and Automata Theory*. Cambridge University Press, New York, NY, USA, 1 edition, 2008.
[21] A. Takahara and T. Yokomori. On the computational power of insertion-deletion systems. *Natural Computing*, 2(4):321–336, 2003.
[22] D. Wood. *Theory of Computation*. John Wiley & Sons, Inc., New York, NY, 1987.
[23] S. Yu. State complexity of regular languages. *Journal of Automata, Languages and Combinatorics*, 6(2):221–234, 2001.
[24] 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.