State Complexity of Inversion Operations

Da-Jung Cho[1], Yo-Sub Han[1], Sang-Ki Ko[1], and Kai Salomaa[2]

[1] Department of Computer Science, Yonsei University,
50, Yonsei-Ro, Seodaemun-Gu, Seoul 120-749, Republic of Korea
{dajung,emmous,narame7}@cs.yonsei.ac.kr
[2] School of Computing, Queen's University,
Kingston, Ontario K7L 3N6, Canada
ksalomaa@cs.queensu.ca

Abstract. The reversal operation is well-studied in literature and the deterministic (respectively, nondeterministic) state complexity of reversal is known to be 2^n (respectively, n). We consider the inversion operation where some substring of the given string is reversed. Formally, the inversion of a language L consists of all strings ux^Rv such that $uxv \in L$. We show that the nondeterministic state complexity of inversion is in $\Theta(n^3)$. We establish that the deterministic state complexity of the inversion is $2^{\Omega(n \cdot \log n)}$, which is strictly worse than the worst case state complexity of the reversal operation. We also study the state complexity of different variants of the inversion operation, including prefix-, suffix-, and pseudo-inversion.

Keywords: State complexity, Inversion operations, Regular languages.

1 Introduction

Questions of descriptional complexity belong to the very foundations of automata and formal language theory [10, 12, 23, 27]. The state complexity of finite automata has been studied since the 60's [13, 16, 17]. Maslov [15] originated the study of operational state complexity and Yu et al. [27] investigated the state complexity for basic operations. Later, Yu and his co-authors [7, 8, 20, 21] initiated the study on the state complexity of combined operations such as star-of-union, star-of-intersection and so on.

In biology, a *chromosomal inversion* occurs when a segment of a single chromosome breaks and rearranges within itself in reverse order [18]. It is known that the chromosomal inversion often causes genetic diseases [14]. Informally, the inversion operation reverses an infix of a given string. This can be viewed as a generalization of the reversal operation which reverses the whole string. The inversion of a language L is defined as the union of all inversions of strings in L. Therefore, the inversion of L always contains the reversal of L since a string is always an infix of itself.

Many researchers [2, 4–6, 11, 24] have considered the inversion of DNA sequences in terms of formal language theory. Searls [22] considered closure properties of languages under various bio-inspired operations including inversion. Later,

H. Jürgensen et al. (Eds.): DCFS 2014, LNCS 8614, pp. 102–113, 2014.

Yokomori and Kobayashi [26] showed that inversion can be simulated by the set of primitive operations and languages. Dassow et al. [6] investigated a generative mechanism based on some operations inspired by mutations in genomes such as deletion, transposition, duplication and inversion. Daley et al. [4] considered a hairpin inverse operation, which replaces the hairpin part of a string with the inversion of the hairpin part. Note that the hairpin inversion operation is a variation of the inversion operation that reverses substrings of a string. Recently, Cho et al. [3] defined the pseudo-inversion operation and examined closure properties and decidability problems regarding the operation. Moreover, several string matching problems allowing inversions have been studied [2, 24].

Reversal is an "easy" operation for NFAs. The reversal of a regular language L can be, roughly speaking, recognized by an NFA that is obtained by reversing the transitions of an NFA for L and, consequently, the nondeterministic state complexity of the reversal operation is n for NFAs that allow multiple initial states [9][1]. However, a corresponding simple NFA construction does not work for inversion and here we show that the nondeterministic state complexity of inversion is $\Theta(n^3)$. Also we show that the nondeterministic state complexity of prefix- and suffix-inversion is $\Theta(n^2)$. Moreover, we establish the nondeterministic complexity of the pseudo-inversion, which is defined as the reversal of inversion, and the pseudo-prefix- and pseudo-suffix-inversion operations.

It is known that the deterministic state complexity of the reversal operation is 2^n [19]. The inversion operation is, in some sense, an extension of the reversal operation and using this correspondence it is easy to verify that the state complexity of inversion is at least exponential. Based on their nondeterministic state complexity we establish an upper bound 2^{n^3+2n} for the deterministic state complexity of inversion and an upper bound 2^{n^2+n} for the deterministic state complexity of prefix- and suffix-inversion. Also using a non-constant alphabet (of exponential size) we give a lower bound $2^{\Omega(n \cdot \log n)}$ for inversion and prefix-inversion. This establishes that the deterministic state complexity of these operations is strictly worse than the deterministic state complexity of ordinary reversal. For the nondeterministic and deterministic state complexity of pseudo-inversion we establish exactly the same bounds as for inversion.

There remains a possibility that there could be a more efficient DFA construction for the inversion of a language recognized by a given DFA A than first constructing an NFA for the inversion of $L(A)$ and then determinizing the NFA. The precise deterministic state complexity of inversion remains open.

We give the basic notations and definitions in Section 2. We introduce the inversion and related operations in Section 3 and present the state complexity results in Section 4 and in Section 5. In Section 6, we conclude the paper.

[1] The result stated in [9] is $n + 1$ because the NFA model used there allows only one initial state.

2 Preliminaries

We briefly present definitions and notations used throughout the paper. The reader may refer to the books [23, 25] for more details on language theory.

Let Σ be a finite alphabet and Σ^* be a set of all strings over Σ. A language over Σ is any subset of Σ^*. The symbol λ denotes the null string and Σ^+ denotes $\Sigma^* \setminus \{\lambda\}$. Given a string $w = z_1 z_2 \cdots z_m$, $z_i \in \Sigma$, $1 \leq i \leq m$, we denote the reversal of w by $w^R = z_m z_{m-1} \cdots z_1$.

A *nondeterministic finite automaton with λ-transitions* (λ-NFA) is a five-tuple $A = (\Sigma, Q, Q_0, F, \delta)$ where Σ is a finite alphabet, Q is a finite set of states, $Q_0 \subseteq Q$ 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. 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 $Q \times \Sigma \to Q$. It is well known that the λ-NFAs, NFAs and DFAs all recognize the regular languages [19, 23, 25]. Moreover, the language recognized by a λ-NFA A can be recognized also by an NFA (without λ-transitions) of the same size as A [25].

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 [19, 23, 25].

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. Thus, $\mathrm{sc}(L)$ is equal to the number of classes of $\equiv_L$.

For the nondeterministic state complexity problem, we show a technique called the *fooling set technique* that gives a lower bound for the size of NFAs.

Proposition 1 (Fooling set technique [1]). *Let $L \subseteq \Sigma^*$ be a regular language. Suppose that there exists a set $P = \{(x_i, w_i) \mid 1 \leq i \leq n\}$ 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 1 is called a *fooling set* for the language L.

3 Inversion Operations

We give the formal definition of the inversion as follows:

Definition 1 (Yokomori and Kobayashi [26]). *The inversion of a string w is defined as the set*

$$\mathbb{INV}(w) = \{ux^Rv \mid w = uxv, \ \ u, x, v \in \Sigma^*\}.$$

For instance, given a string $w = abcd$, we have

$$\mathbb{INV}(w) = \{abcd, bacd, cbad, dcba, acbd, abdc, adcb\}.$$

Note that $\mathbb{INV}(\lambda) = \{\lambda\}$. The inversion operation is extended to the languages in the natural way:

$$\mathbb{INV}(L) = \bigcup_{w \in L} \mathbb{INV}(w).$$

We define the *prefix-inversion* that reverses a prefix of a given string and the *suffix-inversion* that reverses a suffix of a given string.

Definition 2. *For a string w, we define the* prefix-inversion *of w as*

$$\mathrm{Pref}\mathbb{INV}(w) = \{u^Rx \mid w = ux, \ u, x \in \Sigma^*\}.$$

Definition 3. *For a string w, we define the* suffix-inversion *of w as*

$$\mathrm{Suf}\mathbb{INV}(w) = \{ux^R \mid w = ux, \ \ u, x \in \Sigma^*\}.$$

See Fig. 1 for examples. Note that the sets $\mathrm{Pref}\mathbb{INV}(L)$ and $\mathrm{Suf}\mathbb{INV}(L)$ are always included in the set $\mathbb{INV}(L)$.

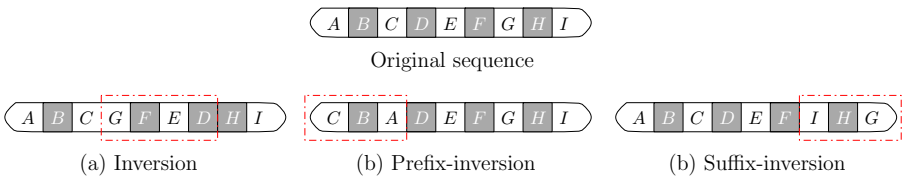

Fig. 1. Examples of the inversion operations

As variants of the inversion operations, we consider the *pseudo-inversion* operations [3] which are defined as the reversal of the inversion operations. Informally, the pseudo-inversion of a given string is defined as a set of strings that are obtained by reversing the given string while maintaining a central substring.

Definition 4. *For a string w, we define the* pseudo-inversion *of w as*

$$\mathbb{PI}(w) = \{v^Rxu^R \mid w = uxv, \ \ u, x, v \in \Sigma^*\}.$$

Furthermore, given a set L of strings, $\mathbb{PI}(L) = \bigcup_{w \in L} \mathbb{PI}(w)$.

We call the operation the *pseudo-inversion* in the sense that the inversion is not properly performed. Note that $\mathbb{PI}(L) = \mathbb{INV}(L)^R$. We also define similar operations called the *prefix-pseudo-inversion* and *suffix-pseudo-inversion* as follows:

Definition 5. *We define the* prefix-pseudo-inversion *of a string* w *as*

$$\mathrm{Pref}\mathbb{PI}(w) = \{xu^R \mid w = ux,\ u, x \in \Sigma^*\}.$$

Definition 6. *We define the* suffix-pseudo-inversion *of a string* w *as*

$$\mathrm{Suf}\mathbb{PI}(w) = \{x^R u \mid w = ux,\ u, x \in \Sigma^*\}.$$

See Fig. 2 for examples of the pseudo-inversion operations.

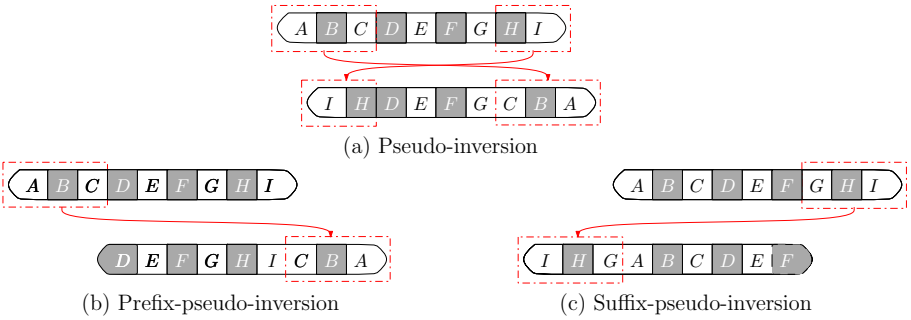

Fig. 2. Examples of the pseudo-inversion operations

Lastly, we consider one more non-trivial inversion operation called the *non-overlapping-inversion*. The non-overlapping-inversion operation allows any character in the string to be involved in at most one inversion operation.

Definition 7. *For a string* w*, we define the* non-overlapping-inversion *of* w *as*

$$\mathrm{NonO}\mathbb{INV}(w) =$$
$$\{w'_1 w'_2 \cdots w'_n \mid w = w_1 w_2 \cdots w_n,\ w_i \in \Sigma^*,\ w'_i = w_i \text{ or } w'_i = w_i^R \text{ for } 1 \le i \le n\}.$$

4 Nondeterministic State Complexity

We establish upper and lower bounds for the nondeterministic state complexity of inversion, prefix-inversion and suffix-inversion. We begin with the upper bound construction of an NFA for $\mathbb{INV}(L)$ when we are given an NFA for a regular language L.

Lemma 1. *Let* L *be a regular language recognized by an NFA with* n *states. Then,* $\mathbb{INV}(L)$ *is recognized by an NFA with* $n^3 + 2n$ *states.*

Proof. Let $A = (\Sigma, Q, Q_0, F_A, \delta)$ be an NFA for L. We define a λ-NFA $B = (\Sigma, P, P_0, F_B, \gamma)$ for the language $\mathbb{INV}(L)$ where

$$P = Q^3 \cup Q \cup \overline{Q},$$

$\overline{Q} = \{\overline{q} \mid q \in Q\}, P_0 = Q_0, F_B = F_A \cup \overline{F_A}$ and the transition function $\gamma : P \times (\Sigma \cup \{\lambda\}) \to 2^P$ is defined as follows:

(i) For all $q, p \in Q$ and $a \in \Sigma$, if $p \in \delta(q, a)$, then $p \in \gamma(q, a)$ and $\overline{p} \in \gamma(\overline{q}, a)$.
(ii) For all $q, p \in Q$, $(p, q, q) \in \gamma(p, \lambda)$.
(iii) For all $q, p, r_1, r_2 \in Q$ and $a \in \Sigma$, if $r_2 \in \delta(r_1, a)$, then $(p, r_1, q) \in \gamma((p, r_2, q), a)$.
(iv) For all $q, p \in Q, \overline{q} \in \gamma((p, p, q), \lambda)$.

The automaton B operates as follows. The transitions in (i) simulate the original computation of A. For any state $p \in Q$, we choose a state q nondeterministically using a λ-transition, and we reach a state (p, q, q) according to the transitions in (ii). The transitions in (iii) allow B to simulate the computation of A in reverse. Note that the first and third elements in Q^3 remember the start and ending positions of the reversed part, while the second element simulatTes the computation of A in reverse. After B reaches the state (p, p, q), it can make a λ-transition to the state $\overline{q}$, and B continues the original computation of A following the transition (i). Fig. 3 shows the computation of B as an illustrative example. As a consequence of the transitions, B recognizes a string ux^Rv if A has an accepting computation for uxv. □

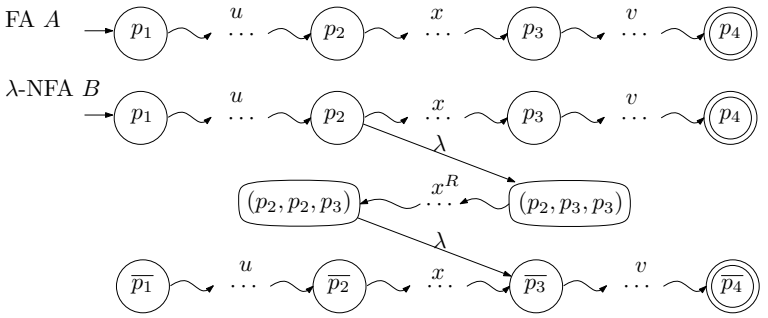

Fig. 3. An illustrative example of constructing an NFA B recogninzing $\mathbb{INV}(L(A))$. Note that if A accepts a string uxv, then B accepts $ux^Rv \in \mathbb{INV}(L(A))$.

We present the following lower bound using the fooling set technique.

Lemma 2. *For every $n_0 \in \mathbb{N}$, there exists an NFA $A = (Q, \Sigma, Q_0, F_A, \delta)$ with $n \geq n_0$ states over an alphabet of size 4 such that* $\mathrm{nsc}(\mathbb{INV}(L(A))) \geq \frac{1}{8}n^3 - f(n)$, *where $f(n) \in O(n^2)$.*

Proof. Let $m \geq 1$ be an integer and consider the language $L = \{\#a^m\$, \#b^m\$\}^*$ over the alphabet $\Sigma = \{a, b, \#, \$\}$. We construct a fooling set P for the language $\mathbb{INV}(L(A))$.

Take the set of pairs P to be the set

$$P = \{(\#a^i b^j \#\$b^k, b^{m-k}\#\$a^{m-i}b^{m-j}\$) \mid 1 \leq i, j, k \leq m\}.$$

Consider

$$(x, w) = (\#a^i b^j \#\$b^k, b^{m-k}\#\$a^{m-i}b^{m-j}\$) \in P.$$

Now the string xw is in $\mathbb{INV}(L)$ because we can write

$$xw = \#a^i(a^{m-i}\$\#b^m\$\#b^j)^R b^{m-j}\$.$$

On the other hand, consider another pair

$$(x', w') = (\#a^{i'} b^{j'} \#\$b^{k'}, b^{m-k'}\#\$a^{m-i'}b^{m-j'}\$) \in P,$$

where $(i, j, k) \neq (i', j', k')$. Now

$$x \cdot w' = \#a^i b^j \#\$b^k \cdot b^{m-k'}\#\$a^{m-i'}b^{m-j'}\$.$$

Now if $xw' \in \mathbb{INV}(L)$ it must be obtained from a string of L by inverting one substring. Since in strings of L the markers # and \$ alternate (when we disregard symbols a and b), the only way we could obtain a string of L from $x \cdot w'$ is to invert a substring z that begins between the first two markers # and ends between the last two markers \$. If $k \neq k'$, the resulting string is not in L. If $k = k'$, necessarily we have $i \neq i'$ or $j \neq j'$ which means again that inverting z cannot produce a string in L.

Hence there are at least $|P| = m^3$ states for any NFA accepting $\mathbb{INV}(L)$ by Proposition 1. It is easy to verify that $n = 2m + 2$ states are sufficient for an NFA that accepts L. Therefore, we have the lower bound $\frac{1}{8}n^3 - f(n)$ for the nondeterministic state complexity of $\mathbb{INV}(L)$, where $f(n) \in O(n^2)$. □

As a consequence of Lemma 1 and Lemma 2, we have:

Theorem 1. *The nondeterministic state complexity of inversion is in* $\Theta(n^3)$.

Now we consider the upper bound construction for the prefix-inversion which is a restricted variant of the general inversion in the sense that only the prefixes of the given string can be reversed.

Lemma 3. *Let L be a regular language recognized by an NFA with n states. Then,* $\mathrm{Pref}\mathbb{INV}(L)$ *is recognized by an NFA with $n^2 + n$ states.*

Proof. Let $A = (\Sigma, Q, Q_0, F_A, \delta)$ be an NFA for L. We define a λ-NFA $B = (\Sigma, P, P_0, F_B, \gamma)$ for the language $\mathrm{Pref}\mathbb{INV}(L)$. We choose

$$P = Q^2 \cup Q,$$

where $P_0 = \{(q, q) \mid q \in Q\}$, $F_B = F_A$ and the transition function $\gamma : P \times (\Sigma \cup \{\lambda\}) \to 2^P$ is defined as follows:

(i) For all $p, q \in Q$ and $a \in \Sigma$, if $p \in \delta(q, a)$, then $p \in \gamma(q, a)$.
(ii) For all $p, r_1, r_2 \in Q$ and $a \in \Sigma$, if $r_2 \in \delta(r_1, a)$, then $(p, r_1) \in \gamma((p, r_2), a)$.
(iii) For all $q_0 \in Q_0$ and $r \in Q$, $r \in \gamma((r, q_0), \lambda)$.

The simulation begins in an arbitrary state (q, q), $q \in Q$. The transitions (ii) simulate a computation of A in reverse in the second component of the state, while the first component of the state pair remembers the state where B starts reverse computation of A. After B reaches a state (q, q_0), where $q_0 \in Q_0$, it can make a λ-transition to state q using the rule (iii). The transitions (i) allow B to simulate the original computation of A from q to a final state. Therefore, B accepts exactly all strings $u^R x$ where A has an accepting computation for ux. □

We also establish that the nondeterministic state complexity of the suffix-inversion coincides with that of the prefix-inversion:

Lemma 4. *Let L be a regular language recognized by an NFA with n states. Then,* $\mathrm{Suf}\mathbb{INV}(L)$ *is recognized by an NFA with $n^2 + n$ states.*

Next we give the following lower bound for the nondeterministic state complexity of the prefix- and suffix-inversion. Using an analogous fooling set construction as in the proof of Lemma 2 we can easily get an $\Omega(n^2)$ lower bound for the nondeterministic state complexity of suffix-inversion.

Lemma 5. *For every $n_0 \in \mathbb{N}$, there exists an NFA A with $n \geq n_0$ states over an alphabet Σ of size 4 such that* $\mathrm{nsc}(\mathrm{Pref}\mathbb{INV}(L(A))) \geq \frac{1}{4}n^2 - f(n)$, *where $f(n) \in O(n)$.*

We have the following statement based on Lemma 3, Lemma 4 and Lemma 5.

Theorem 2. *The nondeterministic state complexity of prefix- and suffix-inversion is in $\Theta(n^2)$.*

The following Observation 3 is now immediate since the state complexity of the reversal operation is n.

Observation 3. *The following statements hold:*

(i) $nsc(\mathrm{Suf}\mathbb{INV}(L)) = nsc(\mathrm{Pref}\mathbb{PI}(L))$,
(ii) $nsc(\mathrm{Pref}\mathbb{INV}(L)) = nsc(\mathrm{Suf}\mathbb{PI}(L))$, *and*
(iii) $nsc(\mathbb{INV}(L)) = nsc(\mathbb{PI}(L))$.

Based on Observation 3, we establish the following results.

Corollary 1. *The nondeterministic state complexity of prefix- and suffix-pseudo-inversion is in $\Theta(n^2)$.*

Corollary 2. *The nondeterministic state complexity of pseudo-inversion is in $\Theta(n^3)$.*

Now we discuss the nondeterministic state complexity of non-overlapping-inversion. Interestingly, we have slightly smaller upper bound for the non-overlapping-inversion than the upper bound for the general inversion.

Lemma 6. *Let L be a regular language recognized by an NFA with n states. Then,* $\mathrm{NonO}\mathbb{INV}(L)$ *is recognized by an NFA with n^3+n states.*

Proof. Let a λ-NFA B be an automaton for $\mathrm{NonO}\mathbb{INV}(L)$. The computation of the automaton B is similar to the computation in the proof of Lemma 1. But, the set of states $\overline{Q}$ and the transitions in (iv) of the proof of Lemma 1 are useless for the language $\mathrm{NonO}\mathbb{INV}(L)$ since the non-overlapping-inversion $\mathrm{NonO}\mathbb{INV}$ allows more than one inversion without overlap. Note that the automaton B can make a λ-transition to the state q if B reaches the state (p, p, q). □

5 Deterministic State Complexity

We first consider the deterministic state complexity of $\mathrm{Pref}\mathbb{INV}(L)$. Recall that if A is an NFA with n states, Lemma 3 gives a construction of an NFA with $n^2 + n$ states for the language $\mathrm{Pref}\mathbb{INV}(L(A))$. This implies an upper bound for the deterministic state complexity 2^{n^2+n} of prefix-inversion. Next we present a lower bound $2^{\Omega(n \cdot \log n)}$ using an alphabet of size 5.

Lemma 7. *For $n \in \mathbb{N}$ there exists an alphabet of size 5 and a DFA A with $2n+3$ states such that the minimal DFA for* $\mathrm{Pref}\mathbb{INV}(L(A))$ *has size at least $2^{n \cdot \log n}$.*

Proof. We define $L_n \subseteq \Sigma_n^3$ by setting

$$L_n = \{1^j \cdot [1^{i_0}, \ldots, 1^{i_m}] \cdot 1^{i_j} \mid j < n, m \geq j\}.$$

Note that all strings of L_n have length exactly three.

Consider a DFA $A = (\Sigma_n, Q, q_0, \{q_{\mathrm{acc}}\}, \delta)$, where

$$Q = \{q_0, q_1, \ldots, q_n, p_1, \ldots, p_n\} \cup \{q_{\mathrm{acc}}, q_{\mathrm{dead}}\},$$

and the transitions of δ are defined by setting

(i) $\delta(q_0, i) = q_i$, $i \in [n]$,
(ii) $\delta(q_i, f) = p_{f(i)}$, $i \in [n]$, $f \in \mathrm{func}_n$,
(iii) $\delta(p_i, i) = q_{\mathrm{acc}}$, $i \in [n]$,
(iv) all transitions not defined above go to the dead state q_{dead}.

It is clear that $L(A) = L_n$. We show that any distinct alphabet symbols $f_1, f_2 \in \mathrm{func}_n$ belong to distinct classes of $\equiv_{\mathrm{Pref}\mathbb{INV}(L_n)}$ which gives a lower bound for the size of a minimal DFA for $\mathrm{Pref}\mathbb{INV}(L_n)$.

If $f_1 \neq f_2$, there exists $i \in [n]$ such that $f_1(i) \neq f_2(i)$. Now $i \cdot f_1 \cdot f_1(i) \in L_n$ and hence $f_1 \cdot i \cdot f_1(i) \in \mathrm{Pref}\mathbb{INV}(L_n)$.

On the other hand, since all words of L_n have length three and have an element of func_n in the middle position, the only way $f_2 \cdot i \cdot f_1(i)$ could be in $\mathrm{Pref}\mathbb{INV}(L_n)$

is that $i \cdot f_2$ would be the prefix of a word of L_n that has been reversed. However, since $f_2(i) \neq f_1(i)$, $i \cdot f_2 \cdot f_1(i) \notin L_n$ and hence $f_2 \cdot i \cdot f_1(i) \notin \mathrm{Pref}\mathbb{INV}(L_n)$ and, consequently, $f_1 \not\equiv_{\mathrm{Pref}\mathbb{INV}(L_n)} f_2$.

Thus, $\equiv_{\mathrm{Pref}\mathbb{INV}(L_n)}$ has at least

$$|\mathrm{func}_n| = n^n = 2^{n \cdot \log n}$$

equivalence classes. □

Note that the deterministic state complexity of the prefix-inversion is strictly worse than the deterministic state complexity of reversal since the latter is known to be 2^n. Moreover, it is easily verified that the same lower bound as in the proof of Lemma 7 applies to the inversion and non-overlapping-inversion operations.

Lemma 8. *A lower bound for the deterministic state complexity of inversion and of non-overlapping-inversion is* $2^{\Omega(n \cdot \log n)}$.

As a consequence of Lemma 1,3 and 7 we have:

Theorem 4. *Let L be a regular language having a DFA with n states. Then* $sc(\mathrm{Pref}\mathbb{INV}(L)) \leq 2^{n^2+n}$. *There exist languages $L(n)$ defined over an alphabet depending on n such that* $sc(L(n)) \in O(n), sc(\mathrm{Pref}\mathbb{INV}(L(n))) \geq 2^{n \cdot \log n}$.

Theorem 5. *Let L be a regular language having a DFA with n states. Then* $sc(\mathbb{INV}(L)) \leq 2^{n^3+2n}$. *There exist languages $L(n)$ defined over an alphabet depending on n such that* $sc(L(n)) \in O(n)$ *and* $sc(\mathbb{INV}(L(n))) \geq 2^{n \cdot \log n}$.

Theorem 5 leaves a larger gap between the state complexity upper and lower bounds for inversion than was the case for prefix-inversion.

The method of Lemma 7 is based on the idea that, in order to force the DFA to remember more information, we want to move the function symbols to the beginning of the string, and this construction does not seem to work for suffix-inversion. For the state complexity of suffix-inversion, the best immediate lower bound is 2^n. The same situation applies for suffix-pseudo-inversion.

Lastly, we observe that similar state complexity lower bounds apply for pseudo-inversion and prefix-pseudo-inversion. The below corollary again follows from the proof of Lemma 7 in the same way as Lemma 8.

Lemma 9. *A lower bound for the deterministic state complexity of pseudo-inversion and prefix-pseudo-inversion is* $2^{\Omega(n \cdot \log n)}$.

6 Conclusion

We have considered the (non)deterministic state complexity of inversion operations that are motivated by evolutionary operations on DNA sequences. While the reversal operation completely reverses the whole string, the inversion operation reverses any infix of a string. Initially, one might think that the state

complexity of inversion operations should be similar to that of the reversal operation. However, both the nondeterministic and deterministic state complexity of the inversion have turned out to be strictly worse than the known bounds for the reversal operation. The prefix- and suffix-inversions which are simplified variants of inversion were also considered. We have shown that the nondeterministic state complexity of prefix- and suffix-inversion is $\Theta(n^2)$ while that of the inversion operation is $\Theta(n^3)$.

We have also obtained a deterministic state complexity lower bound $2^{\Omega(n \cdot \log n)}$ for inversion and prefix-inversion operations using an exponential-size alphabet. This is strictly worse than the deterministic state complexity of reversal, however, it does not match the corresponding upper bounds. It is possible that given a DFA A for L, there is a more efficient construction of a DFA for $\mathbb{INV}(L)$ than first constructing an NFA and then determinizing it. However, when working on this question it seems that a DFA for $\mathbb{INV}(L)$ needs to remember sets of triples of states of A (and similarly a DFA for the prefix-inversion or suffix-inversion needs to remember sets of pairs of states of A). The main open question is to determine the precise deterministic state complexity for inversion and its variants.

References

1. Birget, J.-C.: Intersection and union of regular languages and state complexity. Information Processing Letters 43(4), 185–190 (1992)
2. Cantone, D., Cristofaro, S., Faro, S.: Efficient string-matching allowing for non-overlapping inversions. Theoretical Computer Science 483, 85–95 (2013)
3. Cho, D.-J., Han, Y.-S., Kang, S.-D., Kim, H., Ko, S.-K., Salomaa, K.: Pseudo-inversion on formal languages. In: Proceeding of the 13th International Conference on Unconventional and Natural Computation (to appear)
4. Daley, M., Ibarra, O.H., Kari, L.: Closure and decidability properties of some language classes with respect to ciliate bio-operations. Theoretical Computer Science 306(1-3), 19–38 (2003)
5. Dassow, J., Mitrana, V.: Operations and language generating devices suggested by the genome evolution. Theoretical Computer Science 270(12), 701–738 (2002)
6. Dassow, J., Mitrana, V., Salomaa, A.: Context-free evolutionary grammars and the structural language of nucleic acids. Biosystems 43(3), 169–177 (1997)
7. Ésik, Z., Gao, Y., Liu, G., Yu, S.: Estimation of state complexity of combined operations. Theoretical Computer Science 410(35), 3272–3280 (2009)
8. Gao, Y., Salomaa, K., Yu, S.: The state complexity of two combined operations: Star of catenation and star of reversal. Fundamenta Informaticae 83(1-2), 75–89 (2008)
9. Holzer, M., Kutrib, M.: Nondeterministic descriptional complexity of regular languages. International Journal of Foundations of Computer Science 14(6), 1087–1102 (2003)
10. Holzer, M., Kutrib, M.: Descriptional and computational complexity of finite automata – a survey. Information and Computation 209, 456–470 (2011)
11. Kececioglu, J.D., Sankoff, D.: Exact and approximation algorithms for the inversion distance between two chromosomes. In: Apostolico, A., Crochemore, M., Galil, Z., Manber, U. (eds.) CPM 1993. LNCS, vol. 684, pp. 87–105. Springer, Heidelberg (1993)

12. Kutrib, M., Pighizzini, G.: Recent trends in descriptional complexity of formal languages. Bulletin of the EATCS 111, 70–86 (2013)
13. Lupanov, O.: A comparison of two types of finite sources. Problemy Kibernetiki 9, 328–335 (1963)
14. Lupski, J.R.: Genomic disorders: structural features of the genome can lead to DNA rearrangements and human disease traits. Trends in Genetics 14(10), 417–422 (1998)
15. Maslov, A.: Estimates of the number of states of finite automata. Soviet Mathematics Doklady 11, 1373–1375 (1970)
16. Meyer, A., Fisher, M.: Economy of description by automata, grammars and formal systems. In: Proceedings of the 12th Annual Symposium on Switching and Automata Theory, pp. 188–191 (1971)
17. Moore, F.: On the bounds for state-set size in the proofs of equivalence between deterministic, nondeterministic and two-way finite automata. IEEE Transactions on Computers C-20, 1211–1214 (1971)
18. Painter, T.S.: A New Method for the Study of Chromosome Rearrangements and the Plotting of Chromosome Maps. Science 78, 585–586 (1933)
19. Rozenberg, G., Salomaa, A. (eds.): Handbook of Formal Languages. Beyond Words, vol. 3. Springer-Verlag New York, Inc. (1997)
20. Salomaa, A., Salomaa, K., Yu, S.: State complexity of combined operations. Theoretical Computer Science 383(2-3), 140–152 (2007)
21. Salomaa, K., Yu, S.: On the state complexity of combined operations and their estimation. International Journal of Foundations of Computer Science 18, 683–698 (2007)
22. Searls, D.B.: The Computational Linguistics of Biological Sequences. In: Artificial Intelligence and Molecular Biology, pp. 47–120 (1993)
23. Shallit, J.: A Second Course in Formal Languages and Automata Theory, 1st edn. Cambridge University Press, New York (2008)
24. Vellozo, A.F., Alves, C.E.R., do Lago, A.P.: Alignment with non-overlapping inversions in $O(n^3)$-time. In: Bücher, P., Moret, B.M.E. (eds.) WABI 2006. LNCS (LNBI), vol. 4175, pp. 186–196. Springer, Heidelberg (2006)
25. Wood, D.: Theory of Computation. Harper & Row (1986)
26. Yokomori, T., Kobayashi, S.: DNA evolutionary linguistics and RNA structure modeling: A computational approach. In: Proceedings of INBS 1995, pp. 38–45. IEEE Computer Society (1995)
27. Yu, S., Zhuang, Q., Salomaa, K.: The state complexities of some basic operations on regular languages. Theoretical Computer Science 125(2), 315–328 (1994)