Theoretical Computer Science 610 (2016) 2–12

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

State complexity of inversion operations ☆

Da-Jung Cho [a], Yo-Sub Han [a,*], Sang-Ki Ko [a], Kai Salomaa [b]

[a] Department of Computer Science, Yonsei University, Seoul 120-749, Republic of Korea
[b] School of Computing, Queen's University, Kingston, Ontario K7L 3N6, Canada

ARTICLE INFO

Article history:
Received 31 October 2014
Received in revised form 25 March 2015
Accepted 14 April 2015
Available online 17 April 2015

Keywords:
State complexity
Inversion operations
Finite automata
Regular languages

ABSTRACT

The reversal operation is well-studied in the 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 (respectively, prefix-inversion) of a language L consists of all strings ux^Rv such that $uxv \in L$ (respectively, all strings u^Rx where $ux \in L$). We show that the nondeterministic state complexity of prefix-inversion is $\Theta(n^2)$ and that of inversion is $\Theta(n^3)$. We show that the deterministic state complexity of prefix-inversion is at most $2^{n \cdot \log n + n}$ and has lower bound $2^{\Omega(n \log n)}$. The same lower bound holds for the state complexity of inversion, but for inversion we do not have a matching upper bound. We also study the state complexity of other variants of the inversion operation.

1. Introduction

Questions of descriptional complexity belong to the very foundations of automata and formal language theory [10,14,26,30]. The state complexity of finite automata has been studied since the 60s [16,19,20]. Maslov [18] originated the study of operational state complexity and Yu et al. [30] presented complete proofs for the state complexity of basic operations. Later, Yu and his co-authors [7,8,23,24] 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 [21]. It is known that the chromosomal inversion often causes genetic diseases [17]. 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 [2,4–6,13,27] have considered the inversion of DNA sequences in terms of formal language theory. Searls [25] considered closure properties of languages under various bio-inspired operations including inversion. Later, Yokomori and Kobayashi [29] showed that inversion can be simulated by the set of primitive operations and languages. Dassow et al. [5] 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 inversion operation, which replaces the

☆ A preliminary version appeared in *Proceedings of the 16th International Workshop on Descriptional Complexity of Formal Systems*, DCFS 2014, LNCS **8614**, 102–113, Springer-Verlag, 2014.

* Corresponding author.
E-mail addresses: dajung@cs.yonsei.ac.kr (D.-J. Cho), emmous@cs.yonsei.ac.kr (Y.-S. Han), narame7@cs.yonsei.ac.kr (S.-K. Ko), ksalomaa@cs.queensu.ca (K. Salomaa).

http://dx.doi.org/10.1016/j.tcs.2015.04.017
0304-3975/

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. Strong decidability and undecidability results of related operations were given by Ibarra [11]. Moreover, several string matching problems allowing inversions have been studied [2,27].

From a descriptional complexity point of view, 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 $\Theta(n^3)$. The prefix- (respectively, suffix-) inversion operations reverses a prefix (respectively, a suffix) of a given string. We show that the nondeterministic state complexity of prefix- and suffix-inversion is $\Theta(n^2)$. Moreover, we establish the nondeterministic state complexity of the pseudo-inversion, which is defined as the reversal of inversion, and the prefix-pseudo- and suffix-pseudo-inversion operations.

It is known that the deterministic state complexity of the reversal operation is 2^n [22]. The inversion operation is, in some sense, an extension of the reversal operation and using this correspondence it is easy to see that the state complexity of inversion is at least exponential. We give an upper bound $2^{n\cdot\log n+n}$ for the deterministic state complexity of prefix-inversion and an almost matching lower bound $2^{(n-3)\cdot\log(n-3)}$. The lower bound uses an alphabet depending on n.

The nondeterministic state complexity of inversion implies an upper bound 2^{n^3+2n} for the corresponding deterministic state complexity. Using a direct construction we improve this bound to $n\cdot 2^{n^3+n}$. Also the construction used for prefix-inversion yields the same lower bound $2^{(n-3)\cdot\log(n-3)}$ for the deterministic state complexity of inversion, however, a large gap remains with the upper bound.

With the exception of prefix-inversion, the precise deterministic state complexity of inversion and different variants of the operation 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 results for nondeterministic state complexity in Section 4 and the deterministic state complexity results in Section 5. In Section 6, we conclude the paper.

2. Preliminaries

We briefly present definitions and notations used throughout the paper. The reader may refer to the textbooks [26,28] or the handbook [22] for more details on formal language theory.

The cardinality of a finite set S is denoted $|S|$. When there is no danger of confusion, a singleton set $\{x\}$ is denoted simply by x. The set of functions from S to itself is S^S and the composition of functions $f, g \in S^S$ is $f \circ g$, where $(f \circ g)(x) = f(g(x))$, for $x \in S$.

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 $w = w_1 w_2 \cdots w_m$, $w_i \in \Sigma$, $1 \le i \le m$, we denote the *reversal* of w by $w^R = w_m w_{m-1} \cdots w_1$. Note that $\lambda^R = \lambda$.

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 [22,26,28].

Proposition 2.1. *(See [28].) 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 [22,26,28].

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 $\equiv_L$.

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 [1].

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

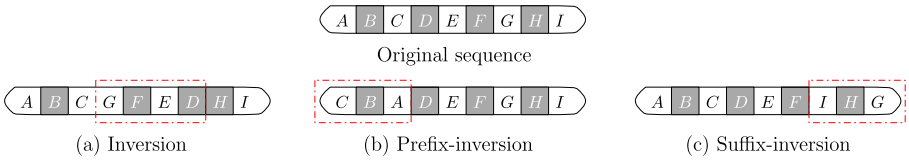


Fig. 1. Examples of the inversion operations.

Proposition 2.2. *(See [1,26].) 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* $x_j w_i \notin L$, *for* $1 \leq i, j \leq n$.

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

The set P satisfying the conditions of Proposition 2.2 is called a *fooling set* for the language L.

3. Inversion operations

We give the formal definition of the inversion as follows:

Definition 3.1. (See Yokomori and Kobayashi [29].) The inversion of a string w is defined as the set

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

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

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

Definition 3.3. 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 $\mathbb{INV}(L)$.

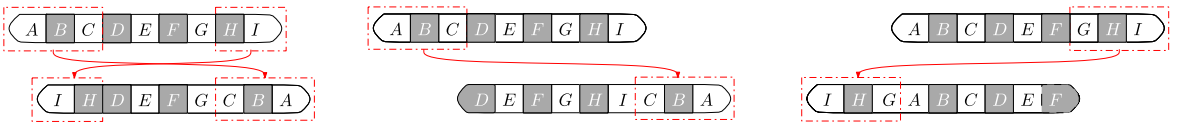
As variants of the inversion operations, we consider the *pseudo-inversion* operations [3] which are defined as the reversal of the corresponding inversion operations. Informally, the pseudo-inversion of a given string is defined as a set of strings that are obtained by reversing the given string while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

Definition 3.4. For a string w, we define the *pseudo-inversion* of w as

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

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

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:



(a) Pseudo-inversion (b) Prefix-pseudo-inversion (c) Suffix-pseudo-inversion

Fig. 2. Examples of the pseudo-inversion operations.

Definition 3.5. 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 3.6. 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. Notice that $\mathrm{Pref}\mathbb{PI}(w)$ and $\mathrm{Suf}\mathbb{PI}(w)$ are always subsets of $\mathbb{PI}(w)$.

Lastly, we consider one more non-trivial inversion operation called the *non-overlapping-inversion*. The non-overlapping-inversion operation allows the reversal of multiple substrings as long as any position in the string is involved in at most one reversal.

Definition 3.7. For a string w, we define the *non-overlapping-inversion* of w as

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

Note that $\mathbb{INV}(w)$, $\mathrm{Pref}\mathbb{INV}(w)$ and $\mathrm{Suf}\mathbb{INV}(w)$ are always subsets of $\mathrm{NonO}\mathbb{INV}(w)$.

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 4.1. *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

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

$\overline{Q} = \{\bar{q} \mid q \in Q\}$, $P_0 = Q_0$, $F_B = F_A \cup \overline{F_A}$. Note that $\sqcup$ is a disjoint union. 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 $\bar{p} \in \gamma(\bar{q}, a)$.
(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 $q, p \in Q$, $\bar{q} \in \gamma((p, p, q), \lambda)$.

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 simulates the computation of A in reverse. After B reaches the state (p, p, q), it can make a λ-transition to the state $\bar{q}$, 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 $ux^R v$ if A has an accepting computation for uxv.

By Proposition 2.1, L has also an NFA without λ-transitions having $n^3 + 2n$ states. □

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

Lemma 4.2. *For every $n_0 \in \mathbb{N}$, there exists an NFA $A = (Q, \Sigma, Q_0, F_A, \delta)$ with $n \ge n_0$ states over an alphabet of size 4 such that* $\mathrm{nsc}(\mathbb{INV}(L(A))) \ge \frac{1}{8}(n^3 - 3n^2 + 3n - 1)$.

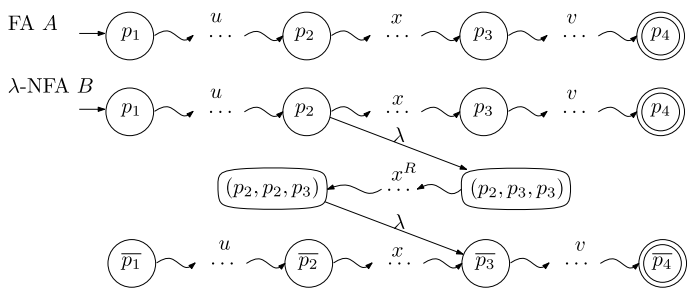


Fig. 3. An illustrative example of constructing a λ-NFA B recognizing $\mathbb{INV}(L(A))$. Note that if A accepts a string uxv, then B accepts $ux^Rv \in \mathbb{INV}(L(A))$.

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 $\mathbb{INV}(L)$.

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 $(i, j, k) \neq (i', j', k')$. Thus,

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

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, by Proposition 2.2, any NFA recognizing $\mathbb{INV}(L)$ has at least $|P| = m^3$ states. It is easy to verify that $n = 2m+1$ states are sufficient for an NFA that recognizes L. Therefore, we have the lower bound $\frac{1}{8}(n^3 - 3n^2 + 3n - 1)$ for the nondeterministic state complexity of $\mathbb{INV}(L)$. □

As a consequence of Lemma 4.1 and Lemma 4.2, we have:

Theorem 4.3. *The nondeterministic state complexity of inversion is in* $\Theta(n^3)$.

Next we consider an upper bound construction for 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 4.4. *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 begins to simulate a 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.

The claim follows by relying on Proposition 2.1. □

We also establish that the nondeterministic state complexity of the suffix-inversion coincides with that of the prefix-inversion. The construction used for the next lemma is analogous to that used in the proof of Lemma 4.4.

Lemma 4.5. *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 a lower bound for the nondeterministic state complexity of the prefix- and suffix-inversion operations.

Lemma 4.6. *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* $\mathrm{nsc}(\mathrm{Pref}\mathbb{INV}(L(A))) \geq \frac{1}{4}(n^2 - 2n+1)$.

Proof. Let $m \geq 1$ be an integer and consider the language $L = \{(\#a^m\$ + \#b^m\$)^*\}$ over the alphabet $\Sigma = \{a, b, \#, \$\}$.

As a fooling set for the language $\mathrm{Pref}\mathbb{INV}(L)$, we choose the set of pairs

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

Consider

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

Now $x \cdot w$ is in $\mathrm{Pref}\mathbb{INV}(L)$ because we can write

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

On the other hand, if

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

is a different element of P, where $(i, j) \neq (i', j')$, we note that xw' is not in $\mathrm{Pref}\mathbb{INV}(L)$. Note that $xw' = b^i\#\$a^j \cdot a^{m-j'}\#b^{m-i'}\$$. Since strings of L begin with a #, in order to get a word of L in xw' we would need to reverse one of the prefixes that ends with #. The shorter prefix does not work because $i' \geq 1$, and also the longer prefix ending with # does not work when $i \neq i'$ or $j \neq j'$.

Therefore, there are at least $|P| = m^2$ states for any NFA accepting $\mathrm{Pref}\mathbb{INV}(L)$ by Proposition 2.2. As in the proof of Lemma 4.2 it is verified that L has an NFA with $2m+1$ states and this yields the claimed lower bound for the nondeterministic state complexity of prefix-inversion. □

A fooling set construction establishing an $\Omega(n^2)$ lower bound for the nondeterministic state complexity of suffix-inversion is completely analogous to the above and the proof of the following lemma is omitted.

Lemma 4.7. *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* $\mathrm{nsc}(\mathrm{Suf}\mathbb{INV}(L(A))) \geq \frac{1}{4}(n^2 - 2n+1)$.

We have the following result based on Lemmas 4.4, 4.5, 4.6 and 4.7.

Theorem 4.8. *The nondeterministic state complexity of prefix- and suffix-inversion is in $\Theta(n^2)$.*

The following Observation 4.9 is now immediate since the state complexity of the reversal operation is n.

Observation 4.9. *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 4.9, we establish the following results.

Corollary 4.10. *The nondeterministic state complexity of prefix- and suffix-pseudo-inversion is in* $\Theta(n^2)$.

Corollary 4.11. *The nondeterministic state complexity of pseudo-inversion is in* $\Theta(n^3)$.

Finally 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 operation.

Lemma 4.12. *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. Based on an NFA A for L, similarly as in the proof of Lemma 4.1 we construct a λ-NFA B for the language $\mathrm{NonO}\mathbb{INV}(L)$. The construction can use the following simplification. The set of states $\overline{Q}$ and the transitions in (iv) of the proof of Lemma 4.1 are not needed for the λ-NFA B recognizing the language $\mathrm{NonO}\mathbb{INV}(L)$ since the non-overlapping-inversion operation allows more than one reversal without overlap. Instead the automaton B can make a λ-transition to the state q whenever B reaches a state (p, p, q). □

Since the lower bound discussed in Lemma 4.2 also applies to the non-overlapping-inversion, we establish the following result.

Corollary 4.13. *The nondeterministic state complexity of non-overlapping-inversion is in* $\Theta(n^3)$.

5. Deterministic state complexity

We give an almost tight bound for the deterministic state complexity of prefix-inversion on regular languages. The same construction yields a non-trivial lower bound for the state complexity of general inversion and other variants of the operation, however, the upper bounds that we have for the latter operations do not match the lower bound.

5.1. Prefix-inversion

Recall that, by Lemma 4.4, for a DFA A with n states the language $\mathrm{Pref}\mathbb{INV}(L(A))$ has an NFA with n^2+n states. This implies an upper bound 2^{n^2+n} for the deterministic state complexity of prefix-inversion. However, we get a significantly better upper bound if, when given a DFA A, instead of constructing an NFA for $\mathrm{Pref}\mathbb{INV}(L(A))$ and then determinizing it, the deterministic computation keeps track of the transition function of the original DFA A on the reversal of the prefix of the input processed up to that point together with the set of all possible states that the NFA recognizing $\mathrm{Pref}\mathbb{INV}(L(A))$ can be in, assuming the computation has passed by the point where some prefix of the input was reversed.

Lemma 5.1. *If A is a DFA with n states, then* $\mathrm{Pref}\mathbb{INV}(L(A))$ *can be recognized by a DFA with* $2^{n\cdot\log n+n}$ *states.*

Proof. Let $A=(\Sigma, Q, q_0, F_A, \delta)$ and denote by id_Q the identity function on Q. For $b\in\Sigma$, the transition function associated with b is denoted $\delta_b \in Q^Q$ (that is, $\delta_b(q)=\delta(q,b)$, for $q\in Q$).

To recognize the language $\mathrm{Pref}\mathbb{INV}(L(A))$, we define a DFA

$$B=(\Sigma, Q^Q\times 2^Q, (\mathrm{id}_Q, \{q_0\}), F_B, \gamma),$$

where

$$F_B=\{(f,P)\in Q^Q\times 2^Q \mid P\cap F_A\neq\emptyset\},$$

and the transitions of γ are defined by the following. For $f\in Q^Q$, $P\subseteq Q$ and $b\in\Sigma$ we define

$$\gamma((f,P),b)=(f\circ\delta_b,\ \ \delta(P,b)\cup\{f(\delta(q_0,b))\}),$$

where $\delta(P,b)=\bigcup_{p\in P}\delta(p,b)$. Using induction on the length of a string $w\in\Sigma^*$ we verify that if

$$\gamma((\mathrm{id}_Q,\{q_0\}),w)=(f,P)\ \ (\in Q^Q\times 2^Q), \tag{1}$$

then $f: Q\to Q$ is the transition function of A defined by w^R $(q\mapsto\delta(q,w^R))$, and $P=\{\delta(q_0,u)\mid u\in\mathrm{Pref}\mathbb{INV}(w)\}$, that is, P consists of all states that a computation of A (originating from q_0) could reach on some prefix-inversion of w.

In the base case $w=\lambda$ we have $\gamma((\mathrm{id}_Q,\{q_0\}),w)=(\mathrm{id}_Q,\{q_0\})$ and the claim holds. Now inductively assume that, using the notations of (1), the claim holds for the string w and consider the string $w'=wb$, $b\in\Sigma$. According to the definition of the transitions of γ we have

$$\gamma((\mathrm{id}_Q, \{q_0\}), w') = \gamma((f, P), b) = (f \circ \delta_b,\ \delta(P, b) \cup \{f(\delta(q_0, b))\}).$$

Since $f \in Q^Q$ is the transition function of A defined by w^R, $f \circ \delta_b$ is the transition function of A defined by $(wb)^R = bw^R$. To verify that the second component of the resulting state satisfies the claim we note that

$$\mathrm{Pref}\mathbb{INV}(w') = \mathrm{Pref}\mathbb{INV}(w) \cdot b \cup (wb)^R.$$

The states that A can reach after reading a string in $\mathrm{Pref}\mathbb{INV}(w) \cdot b$ consist of $\delta(P, b)$ and, by the inductive assumption, $\delta(q_0, (wb)^R) = f(\delta(q_0, b)))$. This concludes the proof of the claim.

Now $x \in \mathrm{Pref}\mathbb{INV}(L(A))$ if and only if there exists $y \in \mathrm{Pref}\mathbb{INV}(x)$ such that $y \in L(A)$. This together with the above claim and the choice of the set of final states of B implies that $L(B) = \mathrm{Pref}\mathbb{INV}(L(A))$. Since Q has n states, $|Q^Q \times 2^Q| = 2^{n \cdot \log n} \cdot 2^n$ and the number of states of B is as claimed. □

Note that the construction of Lemma 5.1 does not work if A is only an NFA. The construction keeps track of the transition function of A on the reversal of the current input, and this is not a one-valued function if A were to be nondeterministic.

Next we present a lower bound for the state complexity of prefix-inversion. The construction uses an alphabet of exponential size.

Lemma 5.2. *For $n \in \mathbb{N}$ there exists an alphabet Σ_n and a DFA A with $n + 3$ states such that the minimal DFA for $\mathrm{Pref}\mathbb{INV}(L(A))$ has size at least $2^{n \cdot \log n}$.*

Proof. Let $[n]$ be the set $\{1, 2, \ldots, n\}$ for $n \in \mathbb{N}$. Denote the set of functions $[n] \to [n]$ as func_n and choose $\Sigma_n = \mathrm{func}_n \cup [n]$. We define $L_n \subseteq \Sigma_n^*$ by setting

$$L_n = \{i \cdot f_1 \cdot f_2 \cdots f_k \cdot j \mid i, j \in [n],\ f_z \in \mathrm{func}_n, z = 1, \ldots, k, k \geq 0,\ (f_k \circ f_{k-1} \circ \cdots \circ f_1)(i) = j\}.$$

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

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

and the transitions of δ are defined by setting

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

A string accepted by A must begin with a symbol $i \in [n]$. After that the computation of A can read a sequence of function symbols $f_1, \ldots, f_k \in \mathrm{func}_n$ reaching a state q_r where $r = (f_k \circ f_{k-1} \circ \cdots \circ f_1)(i)$, and the computation accepts if the input ends with $r \in [n]$. Thus $L(A) = L_n$.

We show that any distinct alphabet symbols $f_1, f_2 \in \mathrm{func}_n$ belong to distinct classes of the Kleene congruence $\equiv_{\mathrm{Pref}\mathbb{INV}(L_n)}$ 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 length three in L_n have an element of func_n in the middle position, the only way that $f_2 \cdot i \cdot f_1(i)$ 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. □

Lemma 5.2 yields a lower bound $2^{(n-3) \cdot \log(n-3)}$ for the worst-case deterministic state complexity of the prefix-inversion of an n state regular language. Combining this with Lemma 5.1 we have:

Theorem 5.3. *The deterministic state complexity of the prefix-inversion of an n state regular language has lower bound $2^{(n-3) \cdot \log(n-3)}$ and upper bound $2^{(1+o(1))(n \cdot \log n)}$.*

Note that the deterministic state complexity of the prefix-inversion operation is strictly worse than the deterministic state complexity of reversal (in the binary case) which is known to be 2^n [30]. Note that this bound for the deterministic state complexity of reversal is tight over a binary alphabet [12,15].

5.2. State complexity of other inversion operations

For an NFA A with n states, Lemma 4.1 gives a construction of an NFA with n^3+2n states for the inversion of $L(A)$. Therefore, by the subset construction we have an upper bound 2^{n^3+2n} for the deterministic state complexity of inversion.

Here we present, for a given DFA A, an improved construction of a DFA B for $\mathbb{INV}(L(A))$ that relies on the observation that the initial part of the computation of B on the prefix u that is not reversed is deterministic (although B must guess nondeterministically where the prefix u ends). Unfortunately, the idea of keeping track of the transition function of A on a reversed string following u (as used in the proof of Lemma 5.1) does not work here, at least not directly. Because u is chosen nondeterministically, such a construction would need to remember for each state of A multiple possible transition functions. Instead, the construction we use below in the proof of Lemma 5.4, for each two states q and p of A, keeps track of all possible states that a simulated reverse computation from p to q could currently be in. This yields a bound strictly better than the bound implied by the nondeterministic state complexity of inversion.

Lemma 5.4. *Let L be a regular language recognized by a DFA with n states. Then, $\mathbb{INV}(L)$ is recognized by a DFA with $n \cdot 2^{n^3+n}$ states.*

Proof. Let $A = (\Sigma, Q, q_0, F_A, \delta)$ be a DFA for L. Assume that $Q = \{q_0, q_1, \ldots, q_{n-1}\}$. We define a DFA $B = (\Sigma, P, p_0, F_B, \gamma)$ for the language $\mathbb{INV}(L)$. We choose

$$P = Q \times (\underbrace{2^Q \times 2^Q \times \cdots \times 2^Q}_{n^2 \text{ times}}) \times 2^Q,$$

where

$$p_0 = (q_0, \underbrace{\{q_0\}, \{q_1\}, \ldots, \{q_{n-1}\}}_{n \text{ times}}, \underbrace{\emptyset, \emptyset, \ldots, \emptyset}_{n^2-n \text{ times}}, \{q_0\})$$

and

$$F_B = \{(q, \underbrace{Q_{0,0}, Q_{0,1}, \ldots, Q_{n-1,n-1}}_{n^2 \text{ pairs}}, \hat{Q}) \mid \hat{Q} \cap F_A \neq \emptyset\}.$$

We define the transition function γ as follows:

$$\gamma((q, \underbrace{Q_{0,0}, \ldots, Q_{n-1,n-1}}_{n^2 \text{ pairs}}, \hat{Q}), a) = (q', Q'_{0,0}, \ldots, Q'_{n-1,n-1}, \hat{Q}'),$$

where $q' = \delta(q, a)$,

$$Q'_{i,j} = \begin{cases} \{r \in Q \mid \delta(r, a) \in Q_{i,j}\} \cup \{q_j\} & \text{if } q' = q_i \\ \{r \in Q \mid \delta(r, a) \in Q_{i,j}\} & \text{otherwise} \end{cases}$$

and

$$\hat{Q}' = \bigcup_{r \in \hat{Q}} \delta(r, a) \cup \{q_j \in Q \mid q_i \in \bigcup_{j=0}^{n-1} Q'_{i,j} \text{ for } 0 \leq i \leq n-1\}.$$

Now we explain how the DFA B accepts $\mathbb{INV}(L)$. Consider a state

$$p = (q, Q_{0,0}, Q_{0,1}, \ldots, Q_{n-1,n-1}, \hat{Q}) \in P.$$

The first component q simulates the original computation of the DFA A. We have n^2 state sets to simulate n^2 reverse computations of the DFA A for each pair of states. For example, the state set $Q_{i,j}$ simulates the reverse computation of A from the state q_j to the state q_i. Whenever the first component enters the state q' which is equal to $q_i \in Q$, we add the state q_j to the state sets $Q_{i,j}$ for $0 \leq i \leq n-1$. We end the reverse computation of A in the state set $Q_{i,j}$ if there is a state q_i in $Q_{i,j}$. As soon as we have a state q_i in the state set $Q_{i,j}$, we add a state q_j to the last state set $\hat{Q}$ to resume the forward computation of the DFA A. Note that the DFA B accepts a string whenever the last component contains the final state of A.

It follows that the upper bound construction yields a DFA with $n \cdot 2^{n^3+n}$ states to accept $\mathbb{INV}(L)$ if we have a DFA with n states to accept L. □

The proof of Lemma 5.2 yields the same $2^{\Omega(n \cdot \log n)}$ lower bound also for the state complexity of inversion and several other variants of the operation.

		$\mathbb{INV}(L)$	Pref$\mathbb{INV}(L)$	Suf$\mathbb{INV}(L)$	NonO$\mathbb{INV}(L)$
nsc	upper	n^3+2n	n^2+n	n^2+n	n^3+n
	lower	$\frac{1}{8}(n^3-3n^2+3n-1)$	$\frac{1}{4}(n^2-2n+1)$	$\frac{1}{4}(n^2-2n+1)$	$\frac{1}{8}(n^3-3n^2+3n-1)$
sc	upper	$n\cdot 2^{n^3+n}$	$2^{n\log n+n}$	2^{n^2+n}	2^{n^3+n}
	lower	$2^{(n-3)\cdot\log(n-3)}$	$2^{(n-3)\cdot\log(n-3)}$	2^n	$2^{(n-3)\cdot\log(n-3)}$

Fig. 4. Operational state complexity of inversion, prefix-inversion, suffix-inversion and non-overlapping-inversion of regular languages.

Lemma 5.5. *For $n\in\mathbb{N}$, there exists an alphabet Σ_n and a regular language L_n recognized by a DFA with $n+3$ states such that the minimal DFA for each of the languages $\mathbb{INV}(L)$, NonO$\mathbb{INV}(L)$, $\mathbb{PI}(L)$ and Pref$\mathbb{PI}(L)$ needs $2^{n\cdot\log n}$ states.*

Proof. The alphabet Σ_n and the language L_n are defined as in the proof of Lemma 5.2 and the proof is exactly the same. In the proof we, basically, just replace prefix-inversion by, respectively, inversion, non-overlapping inversion, pseudo-inversion, or prefix-pseudo-inversion. In the latter two cases there are other minor differences. For example, for pseudo-inversion when verifying that $f_1\not\equiv_{\mathbb{PI}(L_n)} f_2$, it is observed that $f_1\cdot f_1(i)\cdot i$ is in $\mathbb{PI}(L_n)$ (instead of $f_1\cdot i\cdot f_1(i)\in$ Pref$\mathbb{INV}(L_n)$). □

The reason why the proof of Lemma 5.2 carries over to the operations inversion, non-overlapping inversion, pseudo-inversion, and prefix-pseudo-inversion is the fact that, using the notations of the proof, for $i,j\in[n]$, $f\in\text{func}_n$, the string $i\cdot f\cdot j$ is in L_n if and only if $f\cdot i\cdot j\in\mathbb{INV}(L_n)$ (respectively, $f\cdot i\cdot j\in$ NonO$\mathbb{INV}(L_n)$, $f\cdot j\cdot i\in\mathbb{PI}(L_n)$, $f\cdot j\cdot i\in$ Pref$\mathbb{PI}(L_n)$). That is, membership of $i\cdot f\cdot j$ in L_n is determined by membership in the inverted language of a string that begins with the function symbol, which then implies that a DFA for $\mathbb{INV}(L_n)$ (respectively, NonO$\mathbb{INV}(L_n)$, $\mathbb{PI}(L_n)$, Pref$\mathbb{PI}(L_n)$) has to "remember" the function.

The above argument does not work for suffix-inversion and suffix-pseudo-inversion and for these two operations we have only the state complexity lower bound 2^n implied by the state complexity of reversal.

As a consequence of Lemma 5.4 and Lemma 5.5 we have:

Theorem 5.6. *The state complexity of the inversion of an n state DFA language is at most $n\cdot 2^{n^3+n}$ and has lower bound $2^{\Omega(n\cdot\log n)}$.*

The upper bound of Theorem 5.6 is better, but not much better, than the bound implied by the nondeterministic state complexity of inversion.

For all inversion variants the nondeterministic state complexity determined in Section 4 naturally yields an upper bound also for deterministic state complexity. However, the bounds remain far apart from the lower bounds given by Lemma 5.5.

6. Conclusions

We have considered the deterministic and nondeterministic 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 the inversion operations could be similar to that of the reversal operation. However, both the nondeterministic and deterministic state complexity of 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.

Fig. 4 summarizes the state complexity of inversion and several other variants of the operation. 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 shown that the deterministic state complexity of prefix-inversion has upper bound $2^{(1+o(1))\cdot n\cdot\log n}$ and lower bound $2^{\Omega(n\cdot\log n)}$. The lower bound uses an alphabet depending on n and a topic for future work can be to find a lower bound construction for prefix-inversion based on a fixed alphabet.

The same $2^{\Omega(n\cdot\log n)}$ lower bound is shown to hold for inversion, however, for inversion we do not have a matching upper bound. We have shown 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, the construction improves the size only relatively little. The main open question is to determine the precise deterministic state complexity for inversion and other variants like suffix-inversion. It seems possible that a DFA for $\mathbb{INV}(L)$ needs to remember sets of triples of states of A, and if this is the case, then we should try to improve the lower bound for the deterministic state complexity of inversion.

Acknowledgements

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.

Cho, Han and Ko were supported by the Basic Science Research Program through NRF funded by MEST (2012R1A1A2044562), the International Cooperation Program managed by NRF of Korea (2014K2A1A2048512) and the Yonsei University Future-leading Research Initiative of 2014. Salomaa was supported by the Natural Sciences and Engineering Research Council of Canada Grant OGP0147224.

References

[1] J.-C. Birget, Intersection and union of regular languages and state complexity, Inform. Process. Lett. 43 (4) (1992) 185–190.
[2] D. Cantone, S. Cristofaro, S. Faro, Efficient string-matching allowing for non-overlapping inversions, Theoret. Comput. Sci. 483 (2013) 85–95.
[3] D.-J. Cho, Y.-S. Han, S.-D. Kang, H. Kim, S.-K. Ko, K. Salomaa, Pseudo-inversion on formal languages, in: Proceeding of the 13th International Conference on Unconventional and Natural Computation, 2014, pp. 93–104.
[4] M. Daley, O.H. Ibarra, L. Kari, Closure and decidability properties of some language classes with respect to ciliate bio-operations, Theoret. Comput. Sci. 306 (1–3) (2003) 19–38.
[5] J. Dassow, V. Mitrana, A. Salomaa, Context-free evolutionary grammars and the structural language of nucleic acids, Biosystems 43 (3) (1997) 169–177.
[6] J. Dassow, V. Mitrana, A. Salomaa, Operations and language generating devices suggested by the genome evolution, Theoret. Comput. Sci. 270 (12) (2002) 701–738.
[7] Z. Ésik, Y. Gao, G. Liu, S. Yu, Estimation of state complexity of combined operations, Theoret. Comput. Sci. 410 (35) (2009) 3272–3280.
[8] Y. Gao, K. Salomaa, S. Yu, The state complexity of two combined operations: star of catenation and star of reversal, Fund. Inform. 83 (1–2) (2008) 75–89.
[9] M. Holzer, M. Kutrib, Nondeterministic descriptional complexity of regular languages, Internat. J. Found. Comput. Sci. 14 (6) (2003) 1087–1102.
[10] M. Holzer, M. Kutrib, Descriptional and computational complexity of finite automata – a survey, Inform. and Comput. 209 (2011) 456–470.
[11] O.H. Ibarra, On decidability and closure properties of language classes with respect to bio-operations, in: Proceedings of the DNA Computing and Molecular Programming, 20th International Conference, DNA 20, Kyoto, Japan, September 22–26, 2014, 2014, pp. 148–160.
[12] G. Jirásková, J. Šebej, Reversal of binary regular languages, Theoret. Comput. Sci. 449 (2012) 85–92.
[13] J.D. Kececioglu, D. Sankoff, Exact and approximation algorithms for the inversion distance between two chromosomes, in: Proceedings of the 4th Annual Symposium on Combinatorial Pattern Matching, 1993, pp. 87–105.
[14] M. Kutrib, G. Pighizzini, Recent trends in descriptional complexity of formal languages, Bull. Eur. Assoc. Theor. Comput. Sci. EATCS 111 (2013) 70–86.
[15] E. Leiss, Succinct representation of regular languages by boolean automata, Theoret. Comput. Sci. 13 (3) (1981) 323–330.
[16] O. Lupanov, A comparison of two types of finite sources, Probl. Kibern. 9 (1963) 328–335.
[17] J.R. Lupski, Genomic disorders: structural features of the genome can lead to DNA rearrangements and human disease traits, Trends Genet. 14 (10) (1998) 417–422.
[18] A. Maslov, Estimates of the number of states of finite automata, Sov. Math., Dokl. 11 (1970) 1373–1375.
[19] A. Meyer, M. Fisher, Economy of description by automata, grammars and formal systems, in: Proceedings of the 12th Annual Symposium on Switching and Automata Theory, 1971, pp. 188–191.
[20] F. Moore, On the bounds for state-set size in the proofs of equivalence between deterministic, nondeterministic and two-way finite automata, IEEE Trans. Comput. C-20 (1971) 1211–1214.
[21] T.S. Painter, A new method for the study of chromosome rearrangements and the plotting of chromosome maps, Science 78 (1933) 585–586.
[22] G. Rozenberg, A. Salomaa (Eds.), Handbook of Formal Languages, vol. 3: Beyond Words, Springer-Verlag New York, Inc., 1997.
[23] A. Salomaa, K. Salomaa, S. Yu, State complexity of combined operations, Theoret. Comput. Sci. 383 (2–3) (2007) 140–152.
[24] K. Salomaa, S. Yu, On the state complexity of combined operations and their estimation, Internat. J. Found. Comput. Sci. 18 (2007) 683–698.
[25] D.B. Searls, The computational linguistics of biological sequences, in: Artificial Intelligence and Molecular Biology, 1993, pp. 47–120.
[26] J. Shallit, A Second Course in Formal Languages and Automata Theory, 1st edition, Cambridge University Press, New York, NY, USA, 2008.
[27] A.F. Vellozo, C.E.R. Alves, A.P. do Lago, Alignment with non-overlapping inversions in $O(n^3)$-time, in: Proceedings of the 6th International Workshop in Algorithms in Bioinformatics, 2006, pp. 186–196.
[28] D. Wood, Theory of Computation, Harper & Row, 1986.
[29] T. Yokomori, S. Kobayashi, DNA evolutionary linguistics and RNA structure modeling: a computational approach, in: Proceedings of INBS'95, IEEE Computer Society, 1995, pp. 38–45.
[30] S. Yu, Q. Zhuang, K. Salomaa, The state complexities of some basic operations on regular languages, Theoret. Comput. Sci. 125 (2) (1994) 315–328.