Theoretical Computer Science 592 (2015) 9–22

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Non-overlapping inversion on strings and languages

Hwee Kim, Yo-Sub Han *

Department of Computer Science, Yonsei University, Seoul 120-749, Republic of Korea

ARTICLE INFO

Article history:
Received 22 September 2014
Received in revised form 6 March 2015
Accepted 6 May 2015
Available online 13 May 2015
Communicated by R. Giancarlo

Keywords:
Non-overlapping inversion
Finite-state automata
Formal languages

ABSTRACT

Given a string, a non-overlapping inversion is to reverse some non-overlapping fragments of the string simultaneously. We define a non-overlapping inversion operation to be the computation of all possible non-overlapping inversions. We apply the operation on a string, which gives rise to a set of strings, and construct an NFA recognizing the set. Then, we design an efficient DFA reduction algorithm from the resulting NFA. We also consider the non-overlapping inversion operation on a language and show the closure properties for regular, context-free and context-sensitive languages. We furthermore examine iterative non-overlapping inversions and establish the closure properties. Finally, we introduce non-overlapping inversion-free languages and present the decidability results for regular and context-free languages.

1. Introduction

In modern biology, it is important to determine exact orders of DNA sequences, retrieve relevant information of DNA sequences and align these sequences [1,16,20,21]. Researchers considered several chromosomal operations on DNA sequences including inversions, pseudo-inversions, translocations, swaps and pseudoknots [2–5,10,15,17,21,23]. We focus on inversions. A *chromosomal inversion* occurs when a single chromosome undergoes breakage and rearrangement within itself [18]. For instance, for a string $w = w_1 w_2 \cdots w_{i-1} w_i w_{i+1} \cdots w_k$, we have a string

$$w' = w_1 \cdots w_{i-2} \overbrace{w_{i+2} w_{i+1} w_i w_{i-1}}^{\text{inversion}} w_{i+3} \cdots w_k$$

by an inversion on w.

A non-overlapping inversion is a set of inversions that do not overlap with each other. This operation is crucial in bio sequences since it ensures that all inversions occur in one mutation step [21]. There are several well-defined problems considering non-overlapping inversions such as the string alignment problem [3,5,21,23], the edit distance problem [15] and the approximate matching problem [2,11]. These problems consider non-overlapping inversions on strings. We notice that, as the size of the bio data grows, we often need to maintain a huge number of relevant patterns as a set—a language. When such a language is described by an automaton, we can process a streaming DNA sequence in the automaton without preprocessing the sequence. This motivates us to obtain a language by applying chromosomal operations to a set of strings and characterize the resulting language. Many researchers studied bio operations from a formal language viewpoint including inversion [8,12], synchronized insertion and deletion [6], hairpin inversion [6,12], hairpin inverted repeat [7], duplication

* Corresponding author.
E-mail addresses: kimhwee@cs.yonsei.ac.kr (H. Kim), emmous@cs.yonsei.ac.kr (Y.-S. Han).

http://dx.doi.org/10.1016/j.tcs.2015.05.008
0304-3975/

and repeat-deletion [8,13,14], and pseudo-inversion [4,12]. However, they do not consider the concept of non-overlapping operations, which is essential in ensuring that all operations can occur in one mutation step [21].

We, in particular, consider the non-overlapping inversion operation on strings and languages. We briefly recall basic definitions and fix notation in Section 2. We first consider a set $\mathbb{NOI}(w)$ of strings obtained by applying all possible non-overlapping inversions to a string w in Section 3. The resulting language represents a set of gene sequences mutated from an input gene sequence by one mutation step. We construct an NFA A_w recognizing $\mathbb{NOI}(w)$ and investigate the structural properties of the resulting NFA. Then, based on the properties, we present an efficient algorithm that constructs a DFA for $\mathbb{NOI}(w)$ by reducing states in A_w without the subset construction. We calculate the average number of reduced states in the construction. We also examine a language $\mathbb{NOI}(L)$ computed by applying all possible non-overlapping inversions to all strings in L in Section 4. The resulting language represents a set of gene sequences mutated from the input set of gene sequences by one mutation step. We investigate the closure properties for regular, context-free and context-sensitive languages. We also consider iterative non-overlapping inversions on strings and languages. We establish the closure property results of iterative non-overlapping inversions. Finally, we introduce non-overlapping inversion-free languages. A non-overlapping inversion-free language is a set of sequences where for each pair of sequences x and y, x is not in a non-overlapping inversions of y; $x \notin \mathbb{NOI}(y)$. We present the decidability results of non-overlapping inversion-free languages for regular and context-free languages. Our results on the language properties would help to design new matching or alignment algorithms considering non-overlapping inversions on multiple strings.

2. Preliminaries

Given a finite set Σ of characters and a string w over Σ, let $|w|$ be the length of w and $w[i]$ be the character of w at position i, for $1 \le i \le |w|$. For a character $\sigma \in \Sigma$, $|w|_\sigma$ denotes the number of σ's occurrences in w. Let $w[i:j]$ be a substring $w[i]w[i+1]\cdots w[j]$, where $1 \le i \le j \le |w|$. We use w^R to denote the reversal of w and $\pi(w)$ to denote the set of all permutations of w.

For a finite set Σ of characters, Σ^* denotes the set of all strings over Σ. A language over Σ is any subset of Σ^*. The symbol $\emptyset$ denotes the empty language and the symbol λ denotes the null string. A finite-state automaton (FA) A is specified by $A = (Q, \Sigma, \delta, s, F)$, where Q is a set of states, Σ is an alphabet, $\delta \subseteq Q \times \Sigma \times Q$ is a set of transitions, $s \in Q$ is the start state and $F \subseteq Q$ is a set of final states. Let $|Q|$ be the number of states in Q and $|\delta|$ be the number of transitions in δ. We define the size $|A|$ of A to be $|Q| + |\delta|$. For a transition $\delta(p, \sigma) = q$, we say that p has an *out-transition* and q has an *in-transition*. Moreover, we call q a *target state* of p. A string w is accepted by A if there is a labeled path from s to a final state in F such that the path spells out w. The language $L(A)$ of an FA A is the set of all strings accepted by A. If $|\{\delta(p, \sigma)\}| = 1$ for all $p \in Q$ and $\sigma \in \Sigma$, we say that A is a deterministic finite-state automaton (DFA); otherwise, A is a nondeterministic finite-state automaton (NFA). We define $\hat{\delta}(q, w)$ recursively: If $w = \sigma w'$ and $|w| \ge 1$, then $\hat{\delta}(q, w) = \hat{\delta}(\delta(q, \sigma), w')$. If $|w| = 1$, then $\hat{\delta}(q, w) = \delta(q, w)$. For an FA A, we can *merge* two states $p, q \in Q$ (say p to q) and obtain an FA $A' = (Q', \Sigma, \delta', s', F')$ by the following construction:

- $Q' = Q \setminus \{p\}$.
- $\delta'(r, \sigma) = \delta(r, \sigma)$ for all $r \in Q'$ and $\sigma \in \Sigma$.
- $\delta'(r, \sigma) = q$ for all $r \in Q'$ such that $\delta(r, \sigma) = p$ and $\sigma \in \Sigma$.
- $\delta'(q, \sigma) = \delta(p, \sigma)$ for all $\sigma \in \Sigma$.
- $s' = q$ if $s = p$ or $s = q$, $s' = s$ otherwise.
- $F' = F \setminus \{p\} \cup \{q\}$ if $p \in F$ and $q \notin F$, $F' = F \setminus \{p\}$ otherwise.

For more knowledge in automata theory, the reader may refer to textbooks [22,24].

3. Non-overlapping inversion operation on a string

We consider the non-overlapping inversion operation on a string. This operation is motivated from a bio operation that makes all partial inversions on a string occur in one mutation step [21].

Definition 3.1 *(Non-overlapping inversion operation on a string).* Given a string $w \in \Sigma^*$, we define a non-overlapping inversion on w to be $w'_1 w'_2 \cdots w'_k$, where $w = w_1 w_2 \cdots w_k$, $w_i \in \Sigma^*$ and $w'_i = (w_i \text{ or } w_i^R)$ for $1 \le i \le k$. We then define a non-overlapping inversion operation $\mathbb{NOI}(w)$ to be

$$\mathbb{NOI}(w) = \{w'_1 w'_2 \cdots w'_k \mid w = w_1 w_2 \cdots w_k \text{ and } w'_i = (w_i \text{ or } w_i^R) \text{ for } 1 \le i \le k\}.$$

Observation 3.2. *Given a string $w \in \Sigma^*$ of length n and its all possible non-overlapping inversions $\mathbb{NOI}(w)$:*

1. *For any string $u \in \mathbb{NOI}(w)$, $|w| = |u|$ and $|w|_\sigma = |u|_\sigma$ for all $\sigma \in \Sigma$.*
2. *When $w = w_1 w_2$, $\mathbb{NOI}(w_1) \cdot \mathbb{NOI}(w_2) \subseteq \mathbb{NOI}(w)$. Moreover, if $\mathbb{NOI}(w_1) \cdot \mathbb{NOI}(w_2) \subsetneq \mathbb{NOI}(w)$ and $w = w_1 w_2$, then both $w_1, w_2 \neq \lambda$.*

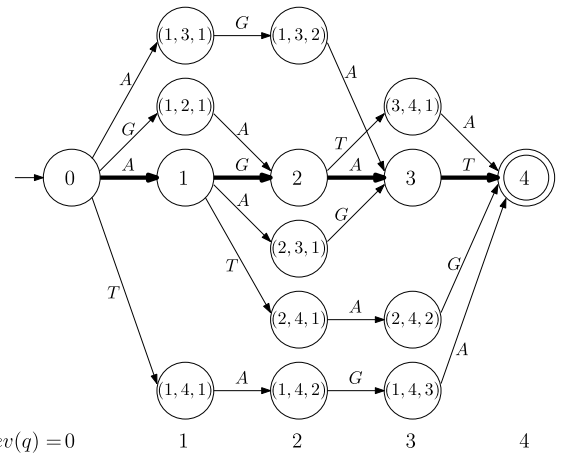

Fig. 1. An example of A_w for $w = AGAT$. We use thick transitions to highlight transitions between backbone states.

3. *Given a string* w, *let* $\mathbb{NOI}'(w) = \{w_1^R w_2^R \cdots w_k^R \mid w = w_1 w_2 \cdots w_k\}$. *Then,* $\mathbb{NOI}'(w) = \mathbb{NOI}(w)$.
4. $|\mathbb{NOI}(w)| \leq 2^{n-1}$.

The last observation comes from the following fact: A string in $\mathbb{NOI}(w)$ can be represented by inserting a number of delimiters between characters, where we perform inversion to the corresponding substrings between delimiters. Since the number of possible delimiters is $n-1$, $|\mathbb{NOI}(w)|$ is bounded by 2^{n-1}. This observation—we have at most 2^{n-1} non-overlapping inversions—leads us to examine the size of an NFA recognizing $\mathbb{NOI}(w)$.

Definition 3.3. For a string $w \in \Sigma^*$ of length n, we define an $\mathbb{NOI}$-NFA $A_w = (Q_w, \Sigma, \delta_w, s_w, F_w)$ as follows:

Let $Q_w^b = \{0, 1, 2, \ldots, n\}$ and $Q_w^v = Q_w^b \times Q_w^b \times Q_w^b$. We make $Q_w = Q_w^b \cup Q_w^v$, $s_w = 0$ and $F_w = \{n\}$. The transition function δ_w is

1. $\delta_w(i{-}1, w[i]) = i$ for $1 \leq i \leq n$,
2. $\delta_w(i{-}1, w[j]) = (i, j, 1)$ for $1 \leq i < j \leq n$,
3. $\delta_w((i, j, j{-}i), w[i]) = j$ for $1 \leq i < j \leq n$,
4. $\delta_w((i, j, k), w[j{-}k]) = (i, j, k{+}1)$ for $1 \leq i < j \leq n$ and $1 \leq k < j - i$.

In the $\mathbb{NOI}$-NFA construction, we rely on states in Q_w^b to simulate substrings of w, and states in Q_w^v to simulate reversed substrings of w. Namely, a sequence of transitions via states $k \in Q_w^b$ from $i \in Q_w^b$ to $j \in Q_w^b$ spells out the substring $w[i{+}1 : j]$ and a sequence of transitions via states $(i, j, k) \in Q_w^v$ from $i \in Q_w^b$ to $j \in Q_w^b$ spells out the reversed substring $w[i{+}1 : j]^R$. We call states in Q_w^b *backbone states* and states in Q_w^v *inversion states*. Remark that the backbone states are indexed from 0 (start state) to n (final state) in order, where $n = |w|$. Fig. 1 shows an example of A_w when $w = AGAT$. We define the level $lev(q)$ of a state $q \in Q_w$ as follows: If $q \in Q_w^b$, then $lev(q) = q$. If $q = (i, j, k) \in Q_w^v$, then $lev(q) = i + k - 1$. In other words, if $q \in \hat{\delta}_w(s_w, w')$, then $lev(q) = |w'|$—the number of transitions in a path from s_w to q.

We next show that A_w indeed recognizes $\mathbb{NOI}(w)$. Suppose that $w = w_1 w_2 \cdots w_k$ and $w[p_i{+}1 : p_{i+1}] = w_i$ for $1 \leq i \leq k-1$. It follows from the construction that $p_{i+1} \in \hat{\delta}_w(p_i, w_i^R)$. Since $w \in \mathbb{NOI}(w)$, we have $p_1 = 0$ and $p_{k+1} = n$. Let $w' = w'_1 w'_2 \cdots w'_k \in \mathbb{NOI}(w)$, where $w'_i = w_i$ or w_i^R.

1. If $w'_i = w_i$, then $p_{i+1} \in \hat{\delta}_w(p_i, w_i)$.
2. If $w'_i = w_i^R$, then $p_{i+1} \in \hat{\delta}_w(p_i, w_i^R)$.

Namely, for both cases, A_w moves from p_i to p_{i+1} by reading either w_i or w_i^R. Thus, $w' \in L(A_w)$ and $L(A_w) = \mathbb{NOI}(w)$. We establish the following result from the $\mathbb{NOI}$-NFA construction:

Lemma 3.4. *Given a string* $w \in \Sigma^*$ *of length* n, *there is an NFA* A_w *recognizing* $\mathbb{NOI}(w)$ *with* $\frac{n^3}{6} + \frac{5n}{6} + 1$ *states and* $\frac{n^3}{6} + \frac{n^2}{2} + \frac{n}{3}$ *transitions.*

Proof. Note that in the $\mathbb{NOI}$-NFA construction, A_w uses $n+1$ states for reading w and $(n-l+1)(l-1)$ states for reading reversed substrings of length l. Therefore, the total number $|Q_w|$ of states of A_w is

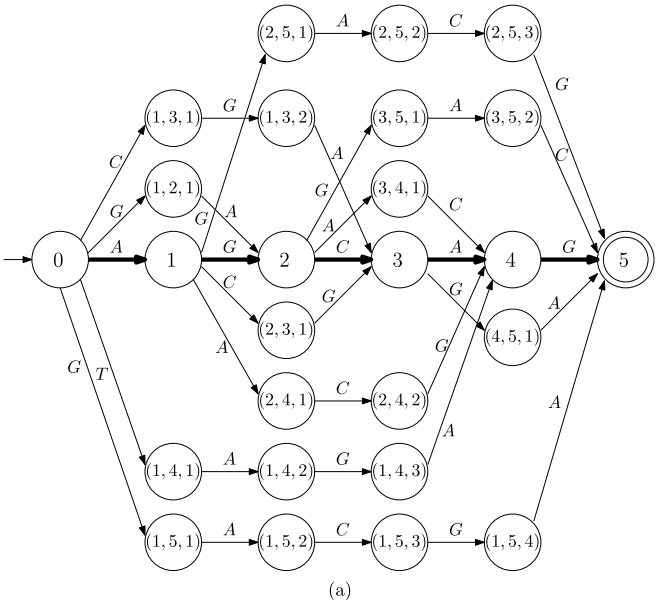

Fig. 2. (a) An example of an NOI-NFA A_w for $w = AGCAG$.

$$|Q_w| = n + 1 + \sum_{i=2}^{n}(n-i+1)(i-1) = \frac{n^3}{6} + \frac{5n}{6} + 1.$$

For the number of transitions, A_w uses n transitions for reading w and $(n-l+1)\times l$ transitions for reading reversed substrings of length l. Hence, the total number $|\delta_w|$ of transitions of A_w is

$$|\delta_w| = n + \sum_{i=2}^{n}(n-i+1)i = \frac{n^3}{6} + \frac{n^2}{2} + \frac{n}{3}. \qquad \square$$

From the $\mathbb{NOI}$-NFA construction, we observe the following properties:

Observation 3.5. *Given a string w and its $\mathbb{NOI}$-NFA $A_w = (Q_w, \Sigma, \delta_w, s_w, F_w)$,*

1. *The NFA A_w is acyclic.*
2. *If $q \in Q_w^b$, then q has $n-q$ out-transitions and q in-transitions. If $q \in Q_w^v$, then q has one out-transition and one in-transition.*

We notice that we can obtain a DFA that recognizes $\mathbb{NOI}(w)$ from A_w by merging some states of A_w in polynomial time; we simply merge all target states p of a state q by reading a character σ (line 4 of Algorithm 1). Algorithm 1 is a pseudo-description of the DFA construction procedure. Fig. 2 and Fig. 3 show an example of the DFA construction by Algorithm 1.

Algorithm 1: A DFA construction from $\mathbb{NOI}$-NFA A_w.

Input: $\mathbb{NOI}$-NFA $A_w = (Q_w, \Sigma, \delta_w, s_w, F_w)$
Output: DFA $A_{w(det)}$
1 **for** $i \leftarrow 1$ **to** $n-1$ **do**
2 **for each** $q \in Q_w$ *where* $lev(q) = i-1$ **do**
3 **for each** $\sigma \in \Sigma$ **do**
4 merge all p such that $\delta_w(q,\sigma) = p$;
5 $A_{w(det)} \leftarrow A_w$;
6 **return** $A_{w(det)}$

Before we prove the correctness of the algorithm, we establish the following statement on an acyclic NFA and on the non-overlapping inversion operation.

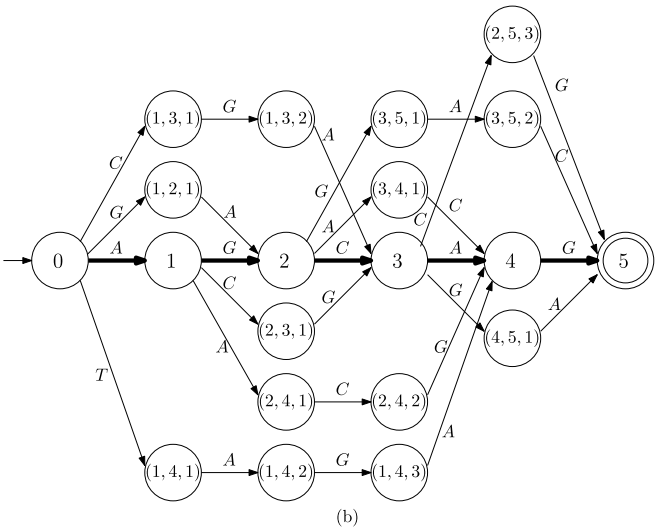

Fig. 3. (b) The reduced DFA $A_{w(det)}$ for $w = AGCAG$.

Lemma 3.6. *For an acyclic NFA $A = (Q, \Sigma, \delta, s, F)$, let $L_h(p) = \{u \mid p \in \hat{\delta}(s, u)\}$ and $L_t(p) = \{u \mid f \in \hat{\delta}(p, u)$ and $f \in F\}$ for a state p. If $L_h(p) \subseteq L_h(q)$ and $L_t(p) \subseteq L_t(q)$ for two states p, q, then we can merge p to q and obtain a smaller FA A' such that $L(A) = L(A')$.*

Proof. It is straightforward to verify that $L(A) \subseteq L(A')$. Thus, we only give a formal proof for $L(A) \supseteq L(A')$. Suppose $w \in L(A')$. We consider the following cases:

1. The accepting path for w does not contain both p and q in A': In this case, it is easy to verify that $w \in L(A)$ since merging p and q does not affect the accepting path for w.
2. There exist strings w_1, w_2 such that $w = w_1 w_2$ and
   (a) $w_1 \in L_h(p)$ and $w_2 \in L_t(p)$: $w = w_1 w_2 \in L_h(p) \cdot L_t(p) \subset L(A)$.
   (b) $w_1 \in L_h(p)$ and $w_2 \in L_t(q)$: $w = w_1 w_2 \in L_h(p) \cdot L_t(q) \subseteq L_h(q) \cdot L_t(q) \subset L(A)$.
   (c) $w_1 \in L_h(q)$ and $w_2 \in L_t(p)$: $w = w_1 w_2 \in L_h(q) \cdot L_t(p) \subseteq L_h(q) \cdot L_t(q) \subset L(A)$.
   (d) $w_1 \in L_h(q)$ and $w_2 \in L_t(q)$: $w = w_1 w_2 \in L_h(q) \cdot L_t(q) \subset L(A)$.

Therefore, for all cases, $w \in L(A)$ and the statement holds. □

Lemma 3.7. *For two strings $x = x_h x_t$ and $y = y_h y_t$ where $|x_h| = |y_h| \geq 1$ and $|x_t| = |y_t| \geq 1$, the following statements hold:*

Statement 1. If $x = y^R$ and $x_h \in \mathbb{NOI}(y_h)$, then $x_t \in \mathbb{NOI}(y_t)$.
Statement 2. If $x \in \mathbb{NOI}(y)$ and $x_h = y_h^R$, then $x_t \in \mathbb{NOI}(y_t)$.

Proof. We prove two statements by induction on the length of x.

Base case. Suppose $x_h = \sigma_1 \in \Sigma$ and $x_t = \sigma_2 \in \Sigma$.
Statement 1. If $x = y^R$ and $x_h \in \mathbb{NOI}(y_h)$, then $\sigma_1 = \sigma_2$ and $x_t \in \mathbb{NOI}(y_t)$.
Statement 2. If $x \in \mathbb{NOI}(y)$ and $x_h = y_h^R$, then $y = \sigma_1 \sigma_2$ and $x_t \in \mathbb{NOI}(y_t)$.

Induction hypothesis. For two strings $x = x_h x_t$ and $y = y_h y_t$ where $|x| = |y| \leq n$, assume the following statements:
Statement 1. If $x = y^R$ and $x_h \in \mathbb{NOI}(y_h)$, then $x_t \in \mathbb{NOI}(y_t)$.
Statement 2. If $x \in \mathbb{NOI}(y)$ and $x_h = y_h^R$, then $x_t \in \mathbb{NOI}(y_t)$.

Inductive step. Suppose $|x| = |y| = n + 1$.
Statement 1. If $x = y^R$ and $x_h \in \mathbb{NOI}(y_h)$, there exist two cases (see Fig. 4):

(a) $|x_h| \leq |x_t|$: Let $x_t = x_m y_h^R$. Then $y_t^R = x_h x_m$ and $y_t = x_m^R x_h^R$. Since $x_m \in \mathbb{NOI}(x_m^R)$ and $y_h^R \in \mathbb{NOI}(x_h^R)$, $x_t \in \mathbb{NOI}(y_t)$.
(b) $|x_h| > |x_t|$: Let $x_h = y_t^R x_m$. Then $y_h^R = x_m x_t$. Since $x_h \in \mathbb{NOI}(y_h)$, $x_h^R \in \mathbb{NOI}(y_h^R)$. Since $x_h^R = x_m^R y_t$ and $y_R^h = x_m x_t$, $x_t \in \mathbb{NOI}(y_t)$ from the induction hypothesis for Statement 2.

Statement 2. If $x \in \mathbb{NOI}(y)$ and $x_h = y_h^R$, we can assume that $x = x_1 x_2 \cdots x_k$ and $y = x_1^R x_2^R \cdots x_k^R$ from the third statement of Observation 3.2. It is clear that $x_2 \cdots x_k \in \mathbb{NOI}(x_2^R \cdots x_k^R)$. There are three possible cases (see Fig. 5):

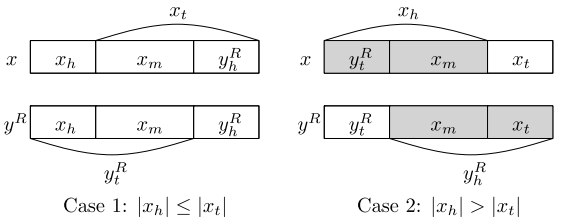

Fig. 4. Two cases for the inductive step for the first statement. Shaded boxes represent strings considering the induction hypothesis.

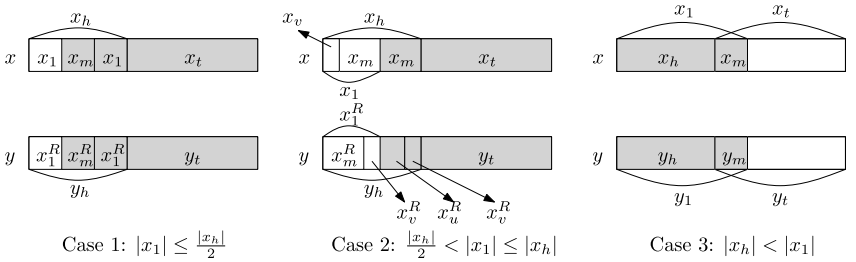

Fig. 5. Three cases for the inductive step for the second statement. Shaded boxes represent strings considering the induction hypothesis.

(a) $|x_1| \le \frac{|x_h|}{2}$: Since x_1 is a prefix of x_h and x_1^R is a prefix of y_h, let $x_h = x_1 x_m x_1$ and $y_h = x_1^R x_m^R x_1^R$. Since $x_m x_1 x_t = x_2 \cdots x_k$ and $x_m^R x_1^R y_t = x_2^R \cdots x_k^R$, $x_m x_1 x_t \in \mathbb{NOI}(x_m^R x_1^R y_t)$. Therefore, $x_1 x_t \in \mathbb{NOI}(x_1^R y_t)$ and $x_t \in \mathbb{NOI}(y_t)$ from the induction hypothesis for Statement 2.

(b) $\frac{|x_h|}{2} < |x_1| \le |x_h|$: Since x_1 is a prefix of x_h, x_1^R is a prefix of y_h. Since $x_h = y_h^R$, x_1 is both a prefix and a suffix of x_h. Let $x_h = x_1 x_m$. Since $\frac{|x_h|}{2} < |x_1|$, x_1 becomes periodic with the repeated substring x_m. Let $x_h = x_v (x_m)^j$ and $x_1 = x_v (x_m)^{j-1}$, where $x_m = x_u x_v$. Then $y_h = (x_m^R)^j x_v^R$. Since $x_m x_t = x_2 \cdots x_k$ and $x_u^R x_v^R y_t = x_2^R \cdots x_k^R$, $x_m x_t \in \mathbb{NOI}(x_u^R x_v^R y_t)$. Therefore, $x_v x_t \in \mathbb{NOI}(x_v^R y_t)$ and $x_t \in \mathbb{NOI}(y_t)$ from the induction hypothesis for Statement 2.

(c) $|x_h| < |x_1|$: Let $x_1 = x_h x_m$ and $y_1 = x_h^R y_m$. From the induction hypothesis for Statement 2, $x_m \in \mathbb{NOI}(y_m)$. Since $x_2 \cdots x_k \in \mathbb{NOI}(x_2^R \cdots x_k^R)$, $x_t \in \mathbb{NOI}(y_t)$.

Therefore, for all cases, two statements hold. □

Now we are ready to prove the correctness of the algorithm.

Lemma 3.8. *The output FA $A_{w(det)}$ of Algorithm 1 is deterministic and $L(A_{w(det)}) = L(A_w)$.*

Proof. The line 4 of Algorithm 1 guarantees that, for each state q, if there are multiple target states of q with a character σ, then we merge all these target states into a single state and, thus, remove nondeterminism from q. Once we check all states and remove their nondeterminism, the resulting FA $A_{w(det)}$ is a DFA.

Let $A'_w = (Q_{w'}, \Sigma, \delta_{w'}, s_w, F_w)$ be an intermediate NFA generated from A_w after merging some states in Algorithm 1. Suppose that $\delta(q, \sigma) = p$, $\delta(q', \sigma) = p'$ and states q and q' are merged by Algorithm 1. There are three cases to consider:

1. ($q \in Q_w^b$ or $q' \in Q_w^b$) and ($p \in Q_w^b$ or $p' \in Q_w^b$): From the construction of A_w, both q and q' cannot be backbone states and both p and p' cannot be backbone states. Without loss of generality, we assume that states q', p' are inversion states in the path from a state $r \in Q_w^b$ to a state $o \in Q_w^b$ and states q, p are backbone states in the path from r to o. Let $w[r+1:q] = x_1$, $\hat{\delta}_w(r, y_1) = \{q'\}$, $\hat{\delta}_w(p', x_2) = \{o\}$ and $w[p+1:o] = x_3$ (see Fig. 6). Since q and q' are merged, $y_1 \in \mathbb{NOI}(x_1)$. We claim that p and p' can be merged.
   From Observation 3.5 and line 4 of Algorithm 1, we know that there are only one in-transition and one out-transition of p' in A_w, and there may be multiple in-transitions and out-transitions of p in A_w. From Lemma 3.6 and the construction of A_w, we only need to show that $x_2 \in \mathbb{NOI}(x_3)$ to prove the claim that p and p' can be merged. Since $(x_1 \sigma x_3)^R = y_1 \sigma x_2$ and $y_1 \in \mathbb{NOI}(x_1)$, $x_2 \in \mathbb{NOI}(x_3)$ by Lemma 3.7.
2. $q, q' \in Q_w^v$ and ($p \in Q_w^b$ or $p' \in Q_w^b$): From the construction of A_w, both p and p' cannot be backbone states. Without loss of generality, we assume that $p \in Q_w^b$, states p', q' are inversion states in the path from a state $r' \in Q_w^b$ to a state $o \in Q_w^b$ and the state q is an inversion state in the path from a state $r \in Q_w^b$ to p. Let $\hat{\delta}_w(p', x_2) = \{o\}$ and $w[p+1:o] = x_3$. Similar to the first case, we only need to show that $x_2 \in \mathbb{NOI}(x_3)$ to prove the claim that p and p' can be merged. Since q and q' are merged, there are two cases:

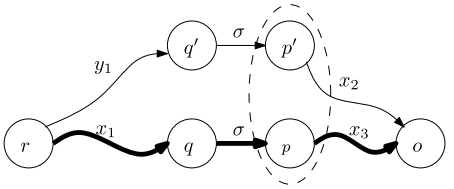

Fig. 6. A_w for the first case of the proof.

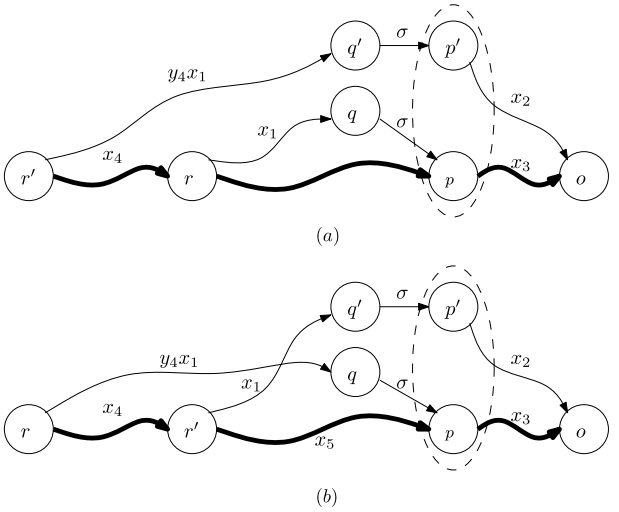

Fig. 7. A_w for the second case of the proof.

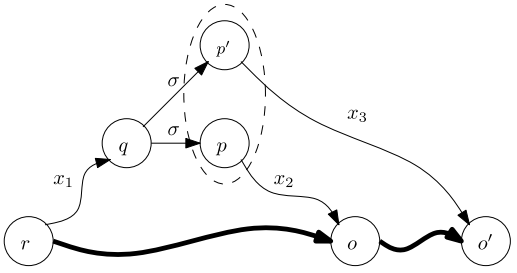

Fig. 8. $A_{w'}$ for the third case of the proof.

(a) $r' < r$: Let $w[r'+1 : r] = x_4$, $\hat{\delta}_w(r, x_1) = \{q\}$ and $\hat{\delta}_w(r', y_4x_1) = \{q\}$. Since q and q' are merged, $y_4 \in \mathbb{NOI}(x_4)$ (see (a) of Fig. 7). Since $(x_4(x_1\sigma)^R x_3)^R = y_4x_1\sigma x_2$, $x_2 \in \mathbb{NOI}(x_3)$ by Lemma 3.7.

(b) $r < r'$: Let $w[r+1 : r'] = x_4$, $\hat{\delta}_w(r', x_1) = \{q'\}$ and $\hat{\delta}_w(r, y_4x_1) = \{q\}$. Since q and q' are merged, $y_4 \in \mathbb{NOI}(x_4)$ (see (b) of Fig. 7). Let $w[r'+1 : p] = x_5$. Since $(x_4x_5)^R = y_4x_1\sigma$, $x_5 \in \mathbb{NOI}(x_1\sigma)$ by Lemma 3.7. Since $(x_5x_3)^R = x_1\sigma x_2$, $x_2 \in \mathbb{NOI}(x_3)$ by Lemma 3.7.

3. $p, p' \in Q_w^v$: Suppose that $p, p' \in \delta_{w'}(q, \sigma)$ as in Fig. 8. From Observation 3.5 and line 4 of Algorithm 1, p and p' have only one in-transition labeled with σ. Therefore, merging p and p' does not generate new strings not in $L(A_{w'})$, regardless of how many out-transitions p and p' have. Therefore, all states in $\{\delta_w(q, \sigma)\}$ can be merged.

Since we can merge p and p' by line 4 of Algorithm 1 without changing the language recognized, $L(A_{w(det)}) = L(A_w)$. □

We calculate the expected number of states that can be merged for an arbitrary string and the runtime of Algorithm 1.

Lemma 3.9. *Given an arbitrary string $w \in \Sigma^*$ of length n and its $\mathbb{NOI}$-NFA A_w accepting $\mathbb{NOI}(w)$, we can merge at least*

$$\frac{1}{2(t-1)}n^2 + \frac{3t^3 - 6t^2 + 4t - 2}{2t^2(t-1)^2}n - \frac{8t^2}{(t-1)^3}$$

expected number of inversion states to backbone states by Algorithm 1, where $t = |\Sigma| \geq 2$.

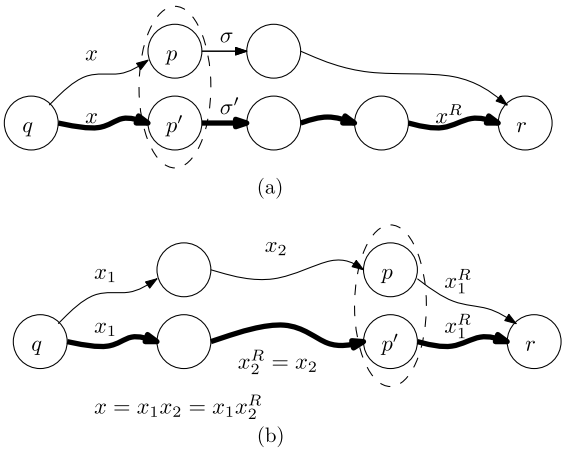

Fig. 9. Two cases that p and p' can be merged. (a) $j < \frac{i}{2}$. (b) $j \geq \frac{i}{2}$.

Proof. Given an alphabet Σ of size t, we assume that all characters of Σ have the same appearance probability. We count the expected number of inversion states that can be merged into backbone states in A_w. Let q and r be backbone states in A_w such that $lev(q) < lev(r)$. In addition, let x be the longest common prefix of $w[q+1:r]$ and $w[q+1:r]^R$, p be an inversion state such that $\hat{\delta}_w(q,x) = \{p\}$ and p' be a backbone state such that $w[q+1:p'] = x$ as depicted in Fig. 9.

Algorithm 1 merges all inversion states in a path from q to p to all backbone states in a path from q to p'. Let $i = r - q$ and $j = p' - q$. There are two possible cases:

1. $j < \frac{i}{2}$: j states are merged with the probability $(\frac{1}{t})^j$.
2. $j \geq \frac{i}{2}$: Note that in this case, $w[q+1:r]$ becomes a palindrome. Therefore, all inversion states in a path from q to r are merged with the probability $(\frac{1}{t})^{\frac{i}{2}}$.

Thus, the expected number of merged states is

$$\underbrace{\left(\left(\frac{1}{t}\right)^{\frac{i}{2}}(i-1)\right)}_{\text{palindrome case}} + \underbrace{\left(1-\left(\frac{1}{t}\right)^{\frac{i}{2}}\right)\sum_{j=1}^{\frac{i}{2}-1}\left\{\left(1-\frac{1}{t}\right)\frac{j}{t^j}\right\}}_{\text{non-palindrome case}}.$$

The expected sum $P(n)$ of all merged states is then

$$P(n) = \sum_{i=2}^{n}\left[(n+1-i)\left\{\left(\left(\frac{1}{t}\right)^{\frac{i}{2}}(i-1)\right) + \left(1-\left(\frac{1}{t}\right)^{\frac{i}{2}}\right)\sum_{j=1}^{\frac{i}{2}-1}\left\{\left(1-\frac{1}{t}\right)\frac{j}{t^j}\right\}\right\}\right]$$

Using the inequality $0 \leq \left(\frac{1}{t}\right)^{\frac{n}{2}} \leq 1$ for $n \geq 1$, we have

$$P(n) \geq \frac{1}{2(t-1)}n^2 + \frac{3t^3-6t^2+4t-2}{2t^2(t-1)^2}n - \frac{8t^2}{(t-1)^3}$$

when $t \geq 2$. □

The number of merged states decreases as t increases since the coefficients of n^2 and n decrease as t increases. Note that the bound in Lemma 3.9 only considers the first case of the proof for Lemma 3.8. It is open to calculate the expected number of merged states in the second and the third cases of the proof for Lemma 3.8.

For runtime, in the worst-case, Algorithm 1 checks all states twice and the merging step takes only constant time. Therefore, we have a linear algorithm for constructing a DFA from A_w.

Theorem 3.10. *Given a string w of length n, we can construct a DFA accepting $\mathbb{NOI}(w)$ in $O(n^3)$ time.*

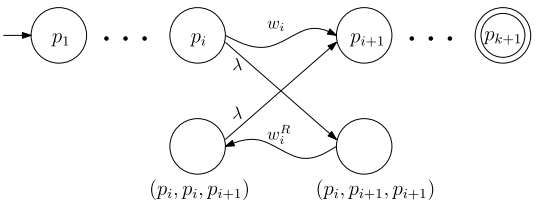

Fig. 10. A partial illustration of A_N.

4. Non-overlapping inversion operation on a language

We extend the non-overlapping inversion operation to languages. We first define the operation on a language.

Definition 4.1 *(Non-overlapping inversion operation on a language).* Given a language L, we define a non-overlapping inversion operation of L to be

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

4.1. Closure properties of non-overlapping inversion

We examine whether or not a family of languages is closed under the non-overlapping inversion operation. We first prove that regular languages are closed.

Definition 4.2. For an FA $A = (Q, \Sigma, \delta, s, F)$, we define an $\mathbb{NOI}$-NFA $A_N = (Q_N, \Sigma, \delta_N, s, F)$ recognizing $\mathbb{NOI}(L(A))$ as follows: $Q_N = Q \cup Q^3$ and δ_N is defined as follows:

- $\delta_N(p, \sigma) = q$ if $\delta(p, \sigma) = q$,
- $\delta_N((p, q, r), \sigma) = (p, s, r)$ if $\delta(s, \sigma) = q$,
- $\delta_N(p, \lambda) = (p, q, q)$ if $q \in \hat{\delta}(p, w)$ and $|w| > 1$,
- $\delta_N((p, p, q), \lambda) = q$ if $q \in \hat{\delta}(p, w)$ and $|w| > 1$.

Note that the state set Q processes substrings of $w \in L(A)$ and the state set Q^3 processes reversed substrings of w. For a state triple $(p, q, r) \in Q^3$ that processes a reversed substring w_i^R, we use states p and r to remember the fact that $r \in \hat{\delta}(p, w_i)$ and do not change while processing w_i^R.

Now we prove that $L(A_N) = \mathbb{NOI}(L(A))$. Let $w = w_1 w_2 \cdots w_k \in L$ and $w[p_i{+}1 : p_{i+1}] = w_i$ for $1 \le i \le k$. Then $(p_i, p_i, p_{i+1}) \in \hat{\delta}((p_i, p_{i+1}, p_{i+1}), w_i^R)$ from the definition of A_N. See Fig. 10 for example. Moreover, $p_1 = s$ and $p_{k+1} \in F$ since $w \in L$. Let $w' = w'_1 w'_2 \cdots w'_k \in \mathbb{NOI}(L)$, where $w'_i = w_i$ or w_i^R.

1. $w'_i = w_i$: It is immediate that $p_{i+1} \in \hat{\delta}_w(p_i, w_i)$.
2. $w'_i = w_i^R$: From the construction, we know that $\hat{\delta}(p_i, w_i^R) = \hat{\delta}((p_i, p_{i+1}, p_{i+1}), w_i^R) = \hat{\delta}((p_i, p_i, p_{i+1}), \lambda) \ni p_{i+1}$.

In both cases, we have $p_{i+1} \in \hat{\delta}_w(p_i, w_i)$. Thus, $w' \in L(A_N)$.

Observation 4.3. *Given a DFA $A = (Q, \Sigma, \delta, s, F)$, we can compute the $\mathbb{NOI}$-NFA A_N recognizing $\mathbb{NOI}(L(A))$ in $O(n^3)$ time, where $n = |Q|$. The resulting NFA, then, has $O(n^3)$ transitions and $O(n^3)$ states.*

Next, we consider the closure property for context-free languages with respect to the non-overlapping inversion operation.

Theorem 4.4. *Context-free languages are not closed under non-overlapping inversion.*

Proof. Consider the following context-free language $L = \{a^p b^q c^q d^p \mid p, q \ge 0\}$. Then, $\mathbb{NOI}(L) \cap L(a^* b^* c^* d^*) = \{a^p b^q d^p c^q \mid p, q \ge 0\}$. It is easy to verify that $\{a^p b^q d^p c^q \mid p, q \ge 0\}$ is not context-free by the context-free pumping lemma [22]. Since the intersection of a context-free language and a regular language is context-free [22], $\mathbb{NOI}(L)$ is not context-free. □

We, furthermore, consider the closure property for context-sensitive languages and show that context-sensitive languages are closed under the non-overlapping inversion operation.

Theorem 4.5. *If L is context-sensitive, then $\mathbb{NOI}(L)$ is context-sensitive.*

Proof. Let $G = (V, \Sigma, P, S)$ be a context-sensitive grammar for L. We make a context-sensitive grammar $G_N = (V_N, \Sigma, P_N, S)$, where $V_N = V \cup \{X_\sigma^O, X_\sigma^I, X_\sigma^F, X_\sigma^T, X_\sigma^D \mid \sigma \in \Sigma\}$ and P_N is the union of the following sets:

1. $P_N(1) = \{h(\alpha) \to h(\beta) \mid \alpha \to \beta \in P\}$,
2. $P_N(2) = \{X_\sigma^D \to \sigma \mid \sigma \in \Sigma\}$,
3. $P_N(3) = \{X_\sigma^O \to X_\sigma^D | X_\sigma^I | X_\sigma^F \mid \sigma \in \Sigma\}$,
4. $P_N(4) = \{X_{\sigma_1}^I X_{\sigma_2}^O \to X_{\sigma_1}^I X_{\sigma_2}^R \mid \sigma_1, \sigma_2 \in \Sigma\}$,
5. $P_N(5) = \{X_{\sigma_1}^R X_{\sigma_2}^O \to X_{\sigma_1}^R X_{\sigma_2}^R \mid \sigma_1, \sigma_2 \in \Sigma\}$,
6. $P_N(6) = \{X_{\sigma_1}^R X_{\sigma_2}^F \to X_{\sigma_2}^T X_{\sigma_1}^F \mid \sigma_1, \sigma_2 \in \Sigma\}$,
7. $P_N(7) = \{X_{\sigma_1}^R X_{\sigma_2}^T \to X_{\sigma_2}^T X_{\sigma_1}^O \mid \sigma_1, \sigma_2 \in \Sigma\}$,
8. $P_N(8) = \{X_{\sigma_1}^I X_{\sigma_2}^T \to X_{\sigma_2}^D X_{\sigma_1}^I \mid \sigma_1, \sigma_2 \in \Sigma\}$,
9. $P_N(9) = \{X_{\sigma_1}^I X_{\sigma_2}^F \to X_{\sigma_2}^D X_{\sigma_1}^D \mid \sigma_1, \sigma_2 \in \Sigma\}$,

where $h : (V \cup \Sigma)^* \to V_N$ is the homomorphism defined by

$$h(A) = A \text{ for } A \in V \text{ and } h(\sigma) = X_\sigma^O \text{ for } \sigma \in \Sigma.$$

We prove the following two claims.

Claim 1. $L(G_N) \supseteq \mathbb{NOI}(L)$: If $w \in L$, $P_N(1)$ ensures that $h(w)$ can be derived from S. The following derivations are possible for the subsequence $X_{\sigma_1}^O X_{\sigma_2}^O \cdots X_{\sigma_k}^O$ of $h(w)$ (we use boxes to denote the variables used in each derivation step):

$$\begin{aligned}
&\boxed{X_{\sigma_1}^O} X_{\sigma_2}^O \cdots \boxed{X_{\sigma_k}^O} \to \boxed{X_{\sigma_1}^I X_{\sigma_2}^O} X_{\sigma_3}^O \cdots X_{\sigma_{k-2}}^O X_{\sigma_{k-1}}^O X_{\sigma_k}^F && \text{(by } P_N(3)\text{)}\\
&\to X_{\sigma_1}^I \boxed{X_{\sigma_2}^R X_{\sigma_3}^O} \cdots X_{\sigma_{k-2}}^O X_{\sigma_{k-1}}^O X_{\sigma_k}^F && \text{(by } P_N(4)\text{)}\\
&\to X_{\sigma_1}^I X_{\sigma_2}^R \boxed{X_{\sigma_3}^R \cdots} X_{\sigma_{k-2}}^O X_{\sigma_{k-1}}^O X_{\sigma_k}^F && \text{(by } P_N(5)\text{)}\\
&\ \ \vdots\\
&\to X_{\sigma_1}^I X_{\sigma_2}^R X_{\sigma_3}^R \cdots X_{\sigma_{k-2}}^R \boxed{X_{\sigma_{k-1}}^R X_{\sigma_k}^F} && \text{(by } P_N(5)\text{)}\\
&\to X_{\sigma_1}^I X_{\sigma_2}^R X_{\sigma_3}^R \cdots \boxed{X_{\sigma_{k-2}}^R X_{\sigma_k}^T} X_{\sigma_{k-1}}^F && \text{(by } P_N(6)\text{)}\\
&\to X_{\sigma_1}^I X_{\sigma_2}^R X_{\sigma_3}^R \boxed{\cdots X_{\sigma_k}^T} X_{\sigma_{k-2}}^O X_{\sigma_{k-1}}^F && \text{(by } P_N(7)\text{)}\\
&\ \ \vdots\\
&\to \boxed{X_{\sigma_1}^I X_{\sigma_k}^T} X_{\sigma_2}^O \cdots X_{\sigma_{k-3}}^O X_{\sigma_{k-2}}^O X_{\sigma_{k-1}}^F && \text{(by } P_N(7)\text{)}\\
&\to X_{\sigma_k}^D \boxed{X_{\sigma_1}^I X_{\sigma_2}^O} \cdots X_{\sigma_{k-3}}^O X_{\sigma_{k-2}}^O X_{\sigma_{k-1}}^F && \text{(by } P_N(8)\text{)}\\
&\ \ \vdots \text{ (repeat the previous derivations)}\\
&\to X_{\sigma_k}^D X_{\sigma_{k-1}}^D X_{\sigma_{k-2}}^D \cdots X_{\sigma_3}^D \boxed{X_{\sigma_1}^I X_{\sigma_2}^F} && \text{(by } P_N(8)\text{)}\\
&\to \boxed{X_{\sigma_k}^D}\boxed{X_{\sigma_{k-1}}^D}\boxed{X_{\sigma_{k-2}}^D}\boxed{\cdots}\boxed{X_{\sigma_3}^D}\boxed{X_{\sigma_2}^D}\boxed{X_{\sigma_1}^D} && \text{(by } P_N(9)\text{)}\\
&\to \sigma_k \sigma_{k-1} \cdots \sigma_2 \sigma_1 && \text{(by } P_N(2)\text{)}
\end{aligned}$$

In other words, we can reverse arbitrary substrings of w and they do not overlap each other. Therefore, $\mathbb{NOI}(L) \subseteq L(G_N)$.

Claim 2. $L(G_N) \subseteq \mathbb{NOI}(L)$: For $w \in L$, suppose that $h(w) = X_{\sigma_1}^O X_{\sigma_2}^O \cdots X_{\sigma_k}^O$ is derived from G_N. When we apply $P_N(4)$ and $P_N(5)$, the subsequence $X_{\sigma_1}^I X_{\sigma_2}^O \cdots X_{\sigma_{j-1}}^O X_{\sigma_j}^F$ always generates the reversed substring by $P_N(6)$ to $P_N(9)$. Therefore, we need to check three other possible subsequences to ensure that they cannot derive the substring without nonterminals.

1. $X_{\sigma_1}^I X_{\sigma_2}^O \cdots X_{\sigma_{j-1}}^O X_{\sigma_j}^I$: $P_N(8)$ or $P_N(9)$ should be used to remove $X_{\sigma_j}^I$. However, the derivation rules require $X_{\sigma'}^T$ or $X_{\sigma'}^F$ adjacent to $X_{\sigma_j}^I$, which is impossible in any derivation of the given subsequence. Therefore, the subsequence cannot derive the substring without nonterminals.
2. $X_{\sigma_1}^F X_{\sigma_2}^O \cdots X_{\sigma_{j-1}}^O X_{\sigma_j}^I$: There is no production rule that can remove $X_{\sigma_1}^F$ in the derivation of the given subsequence. Therefore, the subsequence cannot derive the substring without nonterminals.

$w = w_0 =$	1	2	3	4	5
$w_1 =$	5	4	3	2	1
$w_2 =$	5	1	2	3	4
$w_3 =$	5	1	3	2	4
$w_4 =$	5	1	3	2	4
$w' = w_5 =$	5	1	3	2	4

Fig. 11. Proof for $L \supseteq L'$ in Lemma 4.7. Shaded boxes represent reversed substrings in w_j to generate w_{j+1}.

3. $X^F_{\sigma_1} X^O_{\sigma_2} \cdots X^O_{\sigma_{j-1}} X^F_{\sigma_j}$: The case is similar to the second case.

In all three cases, subsequences cannot derive the substrings without nonterminals. Therefore, there should be the pairs $(X^I_{\sigma_1}, X^F_{\sigma_j})$ of nonterminals that generate the reversed substrings to derive a string from $h(w)$. Thus, $L(G_N) \subseteq \mathbb{NOI}(L)$.

The two claims conclude the proof. □

4.2. Iterative non-overlapping inversions

We consider iterative non-overlapping inversions on strings and languages.

Definition 4.6 *(Iterative non-overlapping inversion).* For a string w and a language L, we define $\mathbb{NOI}^n(w)$ recursively as $\mathbb{NOI}^0(w) = \{w\}$ and $\mathbb{NOI}^n(w) = \mathbb{NOI}^{n-1}(\mathbb{NOI}(w))$, and $\mathbb{NOI}^n(L)$ recursively as $\mathbb{NOI}^0(L) = L$ and $\mathbb{NOI}^n(L) = \mathbb{NOI}^{n-1}(\mathbb{NOI}(L))$. Then, we define $\mathbb{NOI}^*(w) = \bigcup_{n=0}^{\infty} \mathbb{NOI}^n(w)$ and $\mathbb{NOI}^*(L) = \bigcup_{n=0}^{\infty} \mathbb{NOI}^n(L)$.

We show the equivalence between an iterative non-overlapping inversion and a permutation.

Lemma 4.7. *For a string w of length n, $\mathbb{NOI}^n(w)$ is equal to $\pi(w)$—the set of all permutations of w—and, thus, $\mathbb{NOI}^n(w) = \mathbb{NOI}^*(w)$.*

Proof. Let $L = \mathbb{NOI}^n(w)$ and $L' = \pi(w)$. Since it is trivial that for any n, $\mathbb{NOI}^n(w) \subseteq \pi(w) = L'$, we only prove the statement $L \supseteq L'$.

Suppose $w' \in L'$ and $|w'| = n$. Let p be a sequence of n integers such that $w'[i] = w[p[i]]$. Let $w_0 = w$ and $w_j \in \mathbb{NOI}(w_{j-1})$ for $1 \le j \le n$. Since $L = \mathbb{NOI}^n(w)$, $w_j \in L$ for $1 \le j \le n$. We make $w_n = w'$ by the following construction: If $p[j{+}1] = w_j[k]$, then $w_{j+1} = w_j[1:j]w_j[j{+}1:k]^R w_j[k{+}1:n]$. In other words, for each step, we move $w'[j]$ to the jth index by one inversion on the string (see Fig. 11).

Thus, $L = L'$. Moreover, it follows that $\mathbb{NOI}^n(w) = \mathbb{NOI}^*(w)$, since $\mathbb{NOI}(\pi(w)) = \pi(w)$ from Observation 3.2. □

Based on Lemma 4.7, we can establish a similar result on a finite language:

Corollary 4.8. *If L is finite, then $\mathbb{NOI}^n(L) = \mathbb{NOI}^*(L) = \pi(L)$ for $n = \max_{w \in L} |w|$.*

We notice that there may not exist such n for regular languages.

Lemma 4.9. *Regular languages and context-free languages are not closed under $\mathbb{NOI}^*$.*

Proof. Consider a regular language $L = \{(abc)^*\}$. Since $\mathbb{NOI}^*(w) = \pi(w)$, $\mathbb{NOI}^*(L) = \{w \in \{a,b,c\}^* \mid |w|_a = |w|_b = |w|_c\}$. Then, $\mathbb{NOI}^*(L) \cap L(a^*b^*c^*) = \{a^n b^n c^n \mid n \ge 0\}$, which is not context-free. Since the regular languages and the context-free languages are closed under intersection with regular languages, the claim holds. □

It follows from Lemma 4.9 that given a regular or context-free language L, there may not exist a constant n such that $\mathbb{NOI}^n(L) = \mathbb{NOI}^*(L)$.

4.3. Non-overlapping inversion-free languages

We define a non-overlapping inversion-free language L, where there are no two distinct strings in L such that one is a non-overlapping inversion of the other.

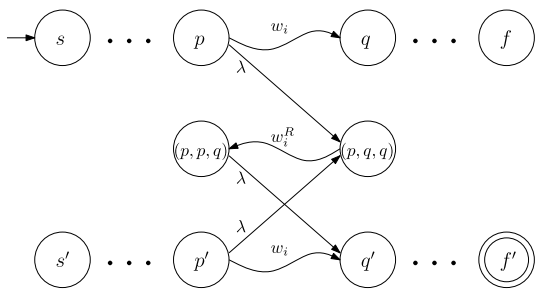

Fig. 12. A partial illustration of $A_{N'}$.

Definition 4.10 *(Non-overlapping inversion-free).* A language L is non-overlapping inversion-free ($\mathbb{NOI}$-free for short) if for all $w \in L$, $\mathbb{NOI}(w) \cap L = \{w\}$.

We investigate the decidability of $\mathbb{NOI}$-freeness for different language classes.

Theorem 4.11. *Given an FA $A = (Q, \Sigma, \delta, s, F)$ of size n, we can determine whether or not $L(A)$ is $\mathbb{NOI}$-free in $O(n^4)$ time.*

Proof. For an FA A, we can build an NFA $A_{N'} = (Q_{N'}, \Sigma, \delta_{N'}, s, F_{N'})$ as follows: $Q_{N'} = Q \cup Q^3 \cup Q'$ where $Q' = \{q' \mid q \in Q\}$, $F_{N'} = \{f' \mid f \in F\}$,

- $\delta_{N'}(p, \sigma) = q$ if $\delta(p, \sigma) = q$,
- $\delta_{N'}(p', \sigma) = q'$ if $\delta(p, \sigma) = q$,
- $\delta_{N'}((p, q, r), \sigma) = (p, s, r)$ if $\delta(s, \sigma) = q$,
- $\delta_{N'}(p, \lambda) = (p, q, q)$ if $q \in \hat{\delta}(p, w)$ and $|w| > 1$,
- $\delta_{N'}(p', \lambda) = (p, q, q)$ if $q \in \hat{\delta}(p, w)$ and $|w| > 1$,
- $\delta_{N'}((p, p, q), \lambda) = q'$ if $q \in \hat{\delta}(p, w)$ and $|w| > 1$.

The NFA $A_{N'}$ is similar to $\mathbb{NOI}$-NFA A_N, except the following two conditions:

1. There are no λ-transitions to the states in Q.
2. $F \cap F_{N'} = \emptyset$.

Two conditions prevent any string $w \in L$ from being accepted by $A_{N'}$ by only the states in Q. See Fig. 12 for example.

We claim that L is $\mathbb{NOI}$-free if and only if $L(A_{N'}) \cap L = \emptyset$.

($\Leftarrow$) Suppose that L is not $\mathbb{NOI}$-free. This implies that there exists $w' \neq w$ such that $w' \in \mathbb{NOI}(w) \cap L$. From the definition of $A_{N'}$, if $q \in \hat{\delta}(p, z)$ and $|z| > 1$, then $\hat{\delta}_{N'}(p, z^R) = \hat{\delta}_{N'}(p', z^R) \ni q'$. We can prove that $w' \in L(A_{N'})$ using a similar argument in the analysis of A_N. Since $w' \in L$ and $w' \in L(A_{N'})$, $L(A_{N'}) \cap L \neq \emptyset$.

($\Rightarrow$) Suppose that $w' \in L(A_{N'}) \cap L$. Since $w' \in L(A_{N'})$, $w' \in \mathbb{NOI}(w)$ for some $w \in L$. Moreover, since $F \cap F_{N'} = \emptyset$, $w \neq w'$. Therefore, $w' \in \mathbb{NOI}(w) \cap L$.

The construction of $A_{N'}$ takes $O(n^3)$ time and the size of $A_{N'}$ is $O(n^3)$. Since we can check the intersection emptiness of two NFAs of size n_1 and n_2 in $O(n_1 n_2)$ time [24], we can determine whether or not L is $\mathbb{NOI}$-free in $O(n^4)$ time. □

Next, we present the undecidability result for a context-free language.

Theorem 4.12. *There is no algorithm that determines whether or not a given context-free language L is $\mathbb{NOI}$-free.*

Proof. We reduce the problem from the problem of determining whether or not a machine with two pushdown stores [9] accepts the empty language. Let $M = (Q, \Sigma, \Gamma, \delta, s)$ be a machine with two pushdown stores, where $\delta \subset Q \times \Sigma \times \Gamma \times \Gamma \to Q \times (\Gamma^0 \cup \Gamma \cup \Gamma^2) \times (\Gamma^0 \cup \Gamma \cup \Gamma^2)$ and strings are accepted by empty stacks. We refer to the first stack as stack 0 and the second stack as stack 1. Let CFG $G(M) = (V, T, P, S)$ where $V = \{S\} \cup \{q_u, q_o \mid q \in Q\}$, $T = \{0, 1\}$, $S \to s_u \mid s_o$. For $\delta(p, \sigma, X_0, X_1) = (q, \gamma_1, \gamma_2)$ where $p, q \in Q$, $X_0, X_1 \in \Gamma$ and $\gamma_1, \gamma_2 \in (\Gamma^0 \cup \Gamma \cup \Gamma^2)$, the production rule for nonterminals p_u and p_o are given in Fig. 13. We claim that $L(G(M))$ is $\mathbb{NOI}$-free if and only if $L(M) = \emptyset$.

($\Rightarrow$) Suppose $L(M) \neq \emptyset$ and $w \in L(M)$. Since $L(M)$ accepts strings by empty stacks, the number of pop operations and the number of push operations are same for two stacks. Suppose that stack 0 processes c_0 pop (and push) operations and stack 1 processes c_1 pop (and push) operations. Note that $u_0 q_u u_1$ and $u_0 u_1$ can be derived from s_u, where u_0, u_1 are strings and $|u_0| = |u_1|$. Again, whenever a transition performs a push operation for stack i, i is added to u_i. Similarly, $o_1 q_o o_0$ and $o_1 o_0$ can be derived from s_o, where o_0, o_1 are strings and $|o_0| = |o_1|$. Whenever a transition performs a push operation

$p_u \rightarrow$	$\lvert\gamma_1\rvert = 0$	1	2
$\lvert\gamma_0\rvert = 0$	$q_u \mid \lambda$	$q_u \mid \lambda$	$q_u 1 \mid 1$
1	$q_u \mid \lambda$	$q_u \mid \lambda$	$q_u 1 \mid 1$
2	$0q_u \mid 0$	$0q_u \mid 0$	$0q_u 1 \mid 01$

$p_o \rightarrow$	$\lvert\gamma_1\rvert = 0$	1	2
$\lvert\gamma_0\rvert = 0$	$1q_u 0 \mid 10$	$q_o 0 \mid 0$	$q_o 0 \mid 0$
1	$1q_u \mid 1$	$q_o \mid \lambda$	$q_o \mid \lambda$
2	$1q_u \mid 1$	$q_o \mid \lambda$	$q_o \mid \lambda$

Fig. 13. The production rule for p_u, p_o when $\delta(p, \sigma, X_0, X_1) = (q, \gamma_1, \gamma_2)$.

for stack i, i is added to o_i. Therefore, there exist w_u derived from s_u and w_o derived from s_o such that $w_u = 0^{c_0}1^{c_1}$ and $w_o = 1^{c_1}0^{c_0}$. Since $w_o = w_u^R$, $L(G(M))$ is not $\mathbb{NOI}$-free.

($\Leftarrow$) Suppose $L(G(M))$ is not $\mathbb{NOI}$-free and $w_u, w_o \in L(G(M))$ where $w_o \in \mathbb{NOI}(w_u)$ and $w_u \neq w_o$. Both w_u and w_o cannot be derived from s_u, because from the conditions $|w_u| = |w_o|$ and $|w_u|_0 = |w_o|_0$, w_u becomes w_o. Similarly, w_u and w_o cannot be derived from s_o. Without loss of generality, we can assume that w_u is derived from s_u and w_o is derived from s_o. Since $w_u = 0^i1^j$ and $w_o = 1^k0^l$, the only case that $w_o \in \mathbb{NOI}(w_u)$ is $i = l$ and $j = k$. Then there exists a string w accepted by M such that stack 0 performs i pushes (and pops) and stack 1 performs j pushes (and pops). Therefore, $L(M) \neq \emptyset$.

Since a machine with two pushdown stores is equivalent to a Turing Machine [9] and the emptiness test for a Turing Machine is undecidable [19], determining whether or not L is $\mathbb{NOI}$-free is undecidable. □

5. Conclusions

Inversion is an important operation for bio sequences such as DNA or RNA sequences, and is closely related to mutations. A non-overlapping inversion is a set of inversions that do not overlap each other. We have defined the non-overlapping inversion operation to compute all possible non-overlapping inversions. We have considered the non-overlapping inversion operation on a string. Given a string w, we have suggested an $\mathbb{NOI}$-NFA A_w construction recognizing $\mathbb{NOI}(w)$ and proposed a polynomial algorithm that computes a DFA from A_w by a simple state-merging process based on the structural properties of A_w. Then, we have considered the non-overlapping inversion operation on a language and proved that regular and context-sensitive languages are closed under the operation whereas context-free languages are not. We have also defined iterative non-overlapping inversions and proved that regular and context-free languages are not closed under $\mathbb{NOI}^*$. Moreover, we have introduced non-overlapping inversion-free ($\mathbb{NOI}$-free) languages and demonstrated that we can determine whether or not a regular language is $\mathbb{NOI}$-free. We have also proved the undecidability result for $\mathbb{NOI}$-freeness when an input language is context-free.

Our result on the language properties would help to design new matching or alignment algorithms considering non-overlapping inversions on multiple strings. A possible future direction is to investigate the properties of other non-overlapping operations including transpositions, swaps and pseudoknots. Another possible future work is to examine the state complexity of regular languages with non-overlapping inversions.

Acknowledgements

We wish to tank the referees for the care they put into reading the previous version of this manuscript. Their comments were invaluable in depth and detail, and the current version owes much to their efforts.

This research was supported by the Basic Science Research Program through NRF funded by MEST (2012R1A1A2044562) and the Yonsei University Future-leading Research Initiative of 2014. Kim was supported by NRF Grant funded by the Korean Government (NRF-2013-Global Ph.D. Fellowship Program).

References

[1] D. Cantone, S. Cristofaro, S. Faro, Efficient string-matching allowing for non-overlapping inversions, Theoret. Comput. Sci. 483 (2013) 85–95.
[2] D. Cantone, S. Faro, E. Giaquinta, Approximate string matching allowing for inversions and translocations, in: Proceedings of the Prague Stringology Conference 2010, 2010, pp. 37–51.
[3] Z.-Z. Chen, Y. Gao, G. Lin, R. Niewiadomski, Y. Wang, J. Wu, A space-efficient algorithm for sequence alignment with inversions and reversals, Theoret. Comput. Sci. 325 (3) (2004) 361–372.
[4] D.-J. Cho, Y.-S. Han, S.-D. Kang, H. Kim, S.-K. Ko, K. Salomaa, Pseudo-inversion on formal languages, in: Proceedings of the Unconventional Computation & Natural Computation 2014, 2014, pp. 93–104.
[5] D.-J. Cho, Y.-S. Han, H. Kim, Alignment with non-overlapping inversions on two strings, in: Proceedings of the 8th International Workshop on Algorithms and Computation, 2014, pp. 261–272.

[6] M. Daley, L. Kari, Some properties of ciliate bio-operations, in: Proceedings of the 6th International Conference on Developments in Language Theory, 2002, pp. 116–127.
[7] J. Dassow, A ciliate bio-operation and language families, in: Proceedings of the 8th International Conference on Developments in Language Theory, 2004, pp. 151–162.
[8] J. Dassow, V. Mitrana, A. Salomaa, Operations and language generating devices suggested by the genome evolution, Theoret. Comput. Sci. 270 (1–2) (2002) 701–738.
[9] P.C. Fischer, Turing machines with restricted memory access, Inf. Control 9 (4) (1966) 364–379.
[10] S. Grabowski, S. Faro, E. Giaquinta, String matching with inversions and translocations in linear average time (most of the time), Inform. Process. Lett. 111 (11) (2011) 516–520.
[11] C. Grozea, F. Manea, M. Müller, D. Nowotka, String matching with involutions, in: Proceedings of the Unconventional Computation & Natural Computation 2012, 2012, pp. 106–117.
[12] O.H. Ibarra, On decidability and closure properties of language classes with respect to bio-operations, in: Proceedings of the 20th International Conference on DNA Computing and Molecular Programming, 2014, pp. 148–160.
[13] M. Ito, L. Kari, Z. Kincaid, S. Seki, Duplication in DNA sequences, in: Proceedings of the 12th International Conference on Developments in Language Theory, 2008, pp. 419–430.
[14] M. Ito, P. Leupold, K. Shikishima-Tsuji, Closure of language classes under bounded duplication, in: Proceedings of the 10th International Conference on Developments in Language Theory, 2006, pp. 238–247.
[15] 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.
[16] S.C. Li, Y.K. Ng, On protein structure alignment under distance constraint, in: Proceedings of the 20th International Symposium on Algorithms and Computation, 2009, pp. 65–76.
[17] O. Lipsky, B. Porat, E. Porat, B.R. Shalom, A. Tzur, Approximate string matching with swap and mismatch, in: Proceedings of the 18th International Symposium on Algorithms and Computation, 2007, pp. 869–880.
[18] T.S. Painter, A new method for the study of chromosome rearrangements and the plotting of chromosome maps, Science 78 (1933) 585–586.
[19] H.G. Rice, Classes of recursively enumerable sets and their decision problems, Trans. Amer. Math. Soc. 74 (1953) 358–366.
[20] Y. Sakai, A new algorithm for the characteristic string problem under loose similarity criteria, in: Proceedings of the 22nd International Symposium on Algorithms and Computation, 2011, pp. 663–672.
[21] M. Schöniger, M.S. Waterman, A local algorithm for DNA sequence alignment with inversions, Bull. Math. Biol. 54 (4) (1992) 521–536.
[22] J. Shallit, A Second Course in Formal Languages and Automata Theory, Cambridge University Press, 2008.
[23] 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 Workshop on Algorithms in Bioinformatics, 2006, pp. 186–196.
[24] D. Wood, Theory of Computation, Harper & Row, 1986.