Pseudo-inversion on Formal Languages

Da-Jung Cho[1], Yo-Sub Han[1], Shin-Dong Kang[1], Hwee Kim[1], Sang-Ki Ko[1], and Kai Salomaa[2]

[1] Department of Computer Science, Yonsei University
50, Yonsei-Ro, Seodaemum-Gu, Seoul 120–749, Republic of Korea
{dajung,emmous,shindong1992,kimhwee,narame7}@cs.yonsei.ac.kr
[2] School of Computing, Queen's University
Kingston, Ontario K7L 3N6, Canada
ksalomaa@cs.queensu.ca

Abstract. We consider the pseudo-inversion operation inspired by a biological event as a result of the partial inversion. We define the pseudo-inversion of a string $w = uxv$ to consist of all strings $v^R x u^R$, where $uv \neq \lambda$ and consider the operation from a formal language theoretic viewpoint. We show that regular languages are closed under the pseudo-inversion operation whereas context-free languages are not. Furthermore, we consider the iterated pseudo-inversion operation and establish the basic properties. Finally, we introduce the pseudo-inversion-freeness and examine closure properties and decidability problems for regular and context-free languages. We establish that pseudo-inversion-freeness is decidable in polynomial time for regular languages and undecidable for context-free languages.

1 Introduction

There have been many approaches that relate biological phenomena to formal languages. This makes it possible to study biological phenomena using tools of formal language theory [7, 8]. Several researchers investigated the algebraic and code-theoretic properties of DNA encoding based on formal language theory [11, 13, 14, 17]. Jonoska et al. [13] introduced involution codes based on the Watson-Crick complementarity, and Kari and Mahalingam [17] investigated the algebraic properties of DNA languages that avoid intermolecular cross hybridization. Kari et al. [16] also studied the DNA hairpin-free structure with respect to algebraic and decision properties.

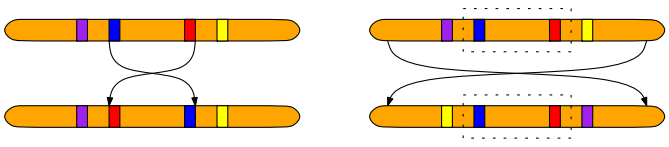
A DNA sequence undergoes various transformations from the primitive sequence through the biological operations such as insertions, deletions, substitutions, inversions, translocations and duplications. This motivates researchers to investigate the genetic operations for tracing the evolution process on a DNA sequence [1, 3–6, 18, 21, 23, 25]. For the DNA evolutionary analysis, an *inversion*—an operation to reverse an infix (substring) of a sequence—is one of the well-studied operations in both DNA computing and formal language theory. Yokomori and Kobayashi [25] showed that the inversion can be simulated by the

O.H. Ibarra et al. (Eds.): UCNC 2014, LNCS 8553, pp. 93–104, 2014.
DOI: 10.1007/978-3-319-08123-6_8,

set of primitive operations and languages using GSM mapping. Dassow et al. [6] noticed that regular and context-free languages are closed under the inversion. They also proved that regular and context-free languages are not closed under the iterated inversion. Daley et al. [4, 5] investigated the closure and decidability properties of some language classes with respect to biological operations including the *hairpin inversion*, which is an extended variant of the inversion. Since the inversion is an important operation in biology, researchers investigated the string matching and alignment problems considering inversions [1, 18, 21, 23].

Here we define a new operation called a *pseudo-inversion* operation. While the inversion operation reverses an infix of an input sequence, a pseudo-inversion operation reverses only the outermost parts of the sequence and the middle part of the sequence is not reversed. See Fig. 1 for an example.

We notice that there are two possible situations where a pseudo-inversion occurs in practice. The first case is—an inversion operation itself is a mutational process—that the inversion process may not be completed in the sense that the sequence of the central part is not fully reversed in the process. The second case is that an inversion is carried out once and the central part of the reversed part is reversed once again; this makes the sequence of the central part where the inversion is applied twice the same as the original sequence. Given two strings of the same length, we design a linear-time algorithm that determines whether or not one string is a pseudo-inversion of the other string.



(a) Inversion **(b)** Pseudo-inversion

Fig. 1. The left figure describes the inversion operation and the right figure describes the pseudo-inversion operation. Note that the sequence in the dotted box is not reversed in pseudo-inversion.

We also introduce an *iterated pseudo-inversion* operation based on the pseudo-inversion. We establish some closure properties of the pseudo-inversion and the iterated pseudo-inversion on regular languages and context-free languages. Moreover, we demonstrate that the iterated pseudo-inversion of a context-free language is recognized by a nondeterministic reversal-bounded multicounter machine. Furthermore, we investigate the decision problems regarding the proposed operations. In particular, we study the question whether a given language L is *pseudo-inversion-free,* that is, no string of L contains a pseudo-inversion of another string of L as a substring. Analogous properties have been studied in the theory of codes [15] and pseudo-inversion-free languages have potential applications in DNA encoding.

We give basic definitions and notations in Section 2. We define the pseudo-inversion operation and the iterated pseudo-inversion in Section 3. Some closure

properties of the proposed operations are also studied in Section 3. Then, we consider the decision problems—whether or not a given language is pseudo-inversion-free—and the closure properties of pseudo-inversion-free languages in Section 4 and conclude the paper in Section 5.

2 Preliminaries

We briefly present definitions and notations. Let $\mathbb{N}$ be the set of positive integers and $\mathbb{N}_0$ be the set of non-negative integers. Let S be a set and k be a positive integer. We use $[S]^k$ to denote the set of all k-tuples $(s_1, s_2, \ldots, s_k)$, where $s_i \in S$.

Let Σ be a finite alphabet and Σ^* be the set of all strings over Σ. A language over Σ is any subset of Σ^*. The symbol $\emptyset$ denotes the empty language, the symbol λ denotes the null string and Σ^+ denotes $\Sigma^* \setminus \{\lambda\}$. Given a string w, we denote the reversal of w by w^R. Let $|w|$ be the length of w. For each $a \in \Sigma$, we denote the number of occurrences of a in w by $|w|_a$. Given a language $L \in \Sigma^*$, $\bar{L}$ denotes the complement of L—$\Sigma^* \setminus L$. Given an alphabet $\Sigma = \{a_1, a_2, \ldots, a_k\}$, let $\Psi : \Sigma^* \to [\mathbb{N}_0]^k$ be a mapping defined by $\Psi(w) = (|w|_{a_1}, |w|_{a_2}, \ldots, |w|_{a_k})$. This function is called a *Parikh mapping* and $\Psi(w)$ is called the *Parikh vector* of w. We denote the symbol of the string w at position i by $w[i]$ and the substring $w[i]w[i+1]\cdots w[j]$ of w by $w[i \cdots j]$, where $1 \leq i \leq j \leq |w|$. We say that languages L_1 and L_2 are *letter-equivalent* if $\{\Psi(w) \mid w \in L_1\} = \{\Psi(w) \mid w \in L_2\}$.

A *nondeterministic finite automaton with λ-transitions* (λ-NFA) is a five-tuple $A = (Q, \Sigma, \delta, Q_0, F)$ where Q is a finite set of states, Σ is a finite alphabet, δ is a multi-valued transition function from $Q \times (\Sigma \cup \lambda)$ into 2^Q, $Q_0 \subseteq Q$ is the set of initial states and $F \subseteq Q$ is the set of final states. 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 (single-valued) function $Q \times \Sigma \to Q$. The language $L(A)$ recognized by A is the set of strings w such that some sequence of transitions spelling out w takes an initial state of A to a final state.

It is well known that λ-NFAs, NFAs and DFAs all recognize the regular languages [22, 24]. Note that any regular language recognized by a λ-NFA of size n can be also recognized by an NFA with the same number of states [24].

Proposition 1 (Wood [24]). *The language recognized by a λ-NFA A can be also recognized by an NFA (without λ-transitions) of the same number of states as A.*

A *context-free grammar* (CFG) G is a four-tuple $G = (V, \Sigma, R, S)$, where V is a set of variables, Σ is a set of terminals, $R \subseteq V \times (V \cup \Sigma)^*$ is a finite set of productions and $S \in V$ is the start variable. Let $\alpha A \beta$ be a string over $V \cup \Sigma$, where $A \in V$ and $A \to \gamma \in R$. Then, we say that A can be rewritten as γ and the corresponding derivation step is denoted by $\alpha A \beta \Rightarrow \alpha \gamma \beta$. The reflexive, transitive closure of $\Rightarrow$ is denoted by $\overset{*}{\Rightarrow}$ and the context-free language generated by G is $L(G) = \{w \in \Sigma^* \mid S \overset{*}{\Rightarrow} w\}$.

A *context-sensitive grammar* (CSG) G is a four-tuple $G = (V, \Sigma, R, S)$, where V is a set of variables, Σ is a set of terminals, $R \subseteq (V \cup \Sigma)^* V (V \cup \Sigma)^* \times (V \cup \Sigma)^*$ is a finite set of productions and $S \in V$ is the start variable.

A *nondeterministic reversal-bounded multicounter machine* (NCM) [2, 12] consists of a finite state control that reads input one-way from the input tape and a finite number of counters, that is a pushdown store over a one-letter alphabet. Furthermore, the counters are reversal-bounded, that is, the number of alternations between the non-decreasing and the non-increasing mode for each counter is bounded by a constant.[1] Thus, an NCM is a λ-NFA equipped with a finite number of reversal-bounded counters.

The reader may refer to the textbooks [10, 22, 24] for complete knowledge of formal language theory.

3 Pseudo-inversion

The pseudo-inversion reverses a given string, but the central part of the string may not be reversed. This is the reason why we call the operation the pseudo-inversion. Fig. 2 depicts an example of a pseudo-inversion of a string.

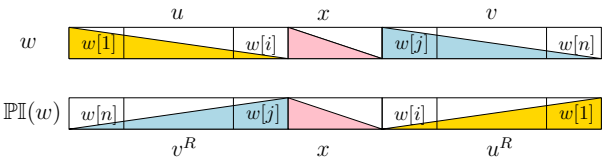


Fig. 2. Given a string $w = uxv$, the pseudo-inversion $\mathbb{PI}(w)$ of w is defined as $v^R x u^R$, where $vu \neq \lambda$

Formally, we define the pseudo-inversion as follows:

Definition 1. *For a string $w \in \Sigma^*$, we define the* pseudo-inversion *of w to be*

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

As a special case, the pseudo-inversion of λ is λ. We can extend the *pseudo-inversion* of strings to languages. Given a language L, $\mathbb{PI}(L) = \bigcup_{w \in L} \mathbb{PI}(w)$. We also define an *iterated pseudo-inversion* operation, which is an iterated version of the pseudo-inversion. First, we set $\mathbb{PI}^1(w) = \mathbb{PI}(w)$. Given a string w, we define $\mathbb{PI}^{i+1}(w) = \mathbb{PI}(\mathbb{PI}^i(w))$ for a positive integer $i > 0$.

Definition 2. *Given a string w, we define the* iterated pseudo-inversion $\mathbb{PI}^*(w)$ *of w to be* $\mathbb{PI}^*(w) = \bigcup_{i=1}^{\infty} \mathbb{PI}^i(w)$.

[1] Unrestricted two-counter machines accept all recursively enumerable languages [9].

Furthermore, given a language L, we define the iterated pseudo-inversion $\mathbb{PI}^*(L)$ of L to be $\mathbb{PI}^*(L) = \bigcup_{w \in L} \mathbb{PI}^*(w)$.

Next we define a *pseudo-inversion-free* language L (or code) where there is no pair of strings in L such that a string is a pseudo-inversion substring of the other string.

Definition 3. *Let $L \subseteq \Sigma^*$ be a language. We define L to be* pseudo-inversion-free *if no string in L is a pseudo-inversion substring of any other string in L. In other words, L is pseudo-inversion-free if $\Sigma^* \cdot \mathbb{PI}(L) \cdot \Sigma^* \cap L = \emptyset$.*

3.1 Closure Properties of Pseudo-inversion

It is well-known that regular languages are closed under the reversal operation. Given an NFA recognizing a regular language L, we can easily obtain an NFA of the same size for the reversal of L by flipping the transition directions and exchanging the set of initial states and the final states [10, 24]. We may need one more state if we do not allow the multiple initial states.

We show that regular languages are also closed under the pseudo-inversion operation.

Theorem 1. *If L is a regular language, then $\mathbb{PI}(L)$ is also regular.*

Theorem 1 shows that regular languages are closed under the pseudo-inversion operation. Based on the result, we have the following observation.

Observation 2. *Given a regular language L, $\mathbb{PI}^n(L)$ is regular for any integer $n \geq 1$.*

Notice that context-free languages are closed under reversal operation [10]. However, we demonstrate that context-free languages are not closed under the pseudo-inversion operation.

Theorem 3. *Context-free languages are not closed under the pseudo-inversion.*

Proof. We prove the statement by the context-free pumping lemma [10, 24]. Consider the context-free language $L = \{a^i b^j c^j d^i \mid i, j \geq 1\}$.

We pick a string $w = d^{2n}c^{2n}a^n b^{2n}a^n \in \mathbb{PI}(L)$, where n is the pumping constant, see Fig. 3. According to the pumping lemma we can split w into five parts, $w = uvxyz$, where the parts u, v, x, y and z satisfy the conditions of the pumping lemma. By the pumping lemma $|vxy| \leq n$, and hence vxy cannot contain both a's and d's and vxy cannot contain both b's and c's.

Remember that if a string w is in $\mathbb{PI}(L)$, then $|w|_a = |w|_d$ and $|w|_b = |w|_c$ should hold. However, since $vy \neq \lambda$, the string uv^2xy^2z does not satisfy this condition. Therefore, $uv^2xy^2z \notin \mathbb{PI}(L)$ and we conclude that $\mathbb{PI}(L)$ is not context-free. □

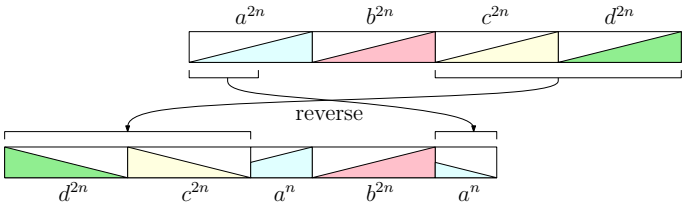


Fig. 3. For a language $L = \{a^i b^j c^j d^i \mid i, j \geq 1\}$, we pick a string $d^{2n}c^{2n}a^n b^{2n}a^n \in \mathbb{PI}(z)$, where $z = a^{2n}b^{2n}c^{2n}d^{2n} \in L$

3.2 Iterated Pseudo-inversion

We investigate the closure properties of the iterated pseudo-inversion operation. It turns out that the iterated pseudo-inversion is equivalent to the permutation operation. Given a string w, let $\pi(w)$ be the set of all permutations of w, that is, $\pi(w) = \{u \in \Sigma^* \mid (\forall a \in \Sigma)|u|_a = |w|_a\}$.

We establish the following result:

Theorem 4. *Given a string w over Σ, the iterated pseudo-inversion of w is the same as the set of all possible permutations of w; namely, $\mathbb{PI}^*(w) = \pi(w)$.*

Based on Theorem 4, we show that regular and context-free languages are not closed under the iterated pseudo-inversion.

Lemma 1. *Regular languages and context-free languages are not closed under the iterated pseudo-inversion operation. Furthermore, the iterated pseudo-inversion of a regular language need not be context-free.*

Proof. Consider a regular language $L = \{(abc)^*\}$. For L, the iterated pseudo-inversion $\mathbb{PI}^*(L)$ of L is

$$\mathbb{PI}^*(L) = \{w \in \{a, b, c\}^* \mid |w|_a = |w|_b = |w|_c\}.$$

We note that

$$\mathbb{PI}^*(L) \cap a^*b^*c^* = \{a^i b^i c^i \mid i \geq 0\}$$

is not context-free. Since the regular languages and the context-free languages are closed under intersection with regular languages, the claim follows. □

Below in Proposition 2 we see that the family of context-sensitive languages is closed under the iterated pseudo-inversion, and consequently it follows that the iterated pseudo-inversion of a regular or a context-free language is always context-sensitive.

In fact, as a consequence of Theorem 4 we see that the iterated pseudo-inversion of a context-free language can be recognized by a reversal-bounded multicounter machine NCM that defines a considerably more restricted language family than the context-sensitive languages. The Parikh set of any language recognized by an NCM is semi-linear and the emptiness problem for NCMs is decidable [12]. Furthermore, the NCMs cannot recognize, for example, the set of marked palindromes $\{w\#w^R \mid w \in \{0, 1\}^*\}$ [2].

Corollary 1. *If L is a context-free language over an alphabet Σ, $\mathbb{PI}^*(L)$ can be recognized by an NCM with $|\Sigma|$ counters that each makes only one reversal.*

Proof. There exists a regular language L' that is letter equivalent to L ([20], part I, Theorem 7.2) and let A be an NFA for L'. On an input w, the NCM stores the value $|w|_a$ for each $a \in \Sigma$ in the available counters. After that, using λ-transitions, the NCM simulates the NFA A for L'. For a transition of A on input $b \in \Sigma$, the counter corresponding to symbol b is decremented and at the end of the computation the NCM checks that all the counters are empty.

By Theorem 4, the NCM recognizes the language $\mathbb{PI}^*(L)$. □

Corollary 1 uses only Theorem 4 and the observation that the Parikh set of a context-free language is semi-linear, which means that the corollary can be stated as:

Corollary 2. *If the Parikh set of L is semi-linear, then $\mathbb{PI}^*(L)$ can be recognized by a reversal-bounded multicounter machine.*

Corollary 2 implies, in particular, that the family of languages recognized by reversal-bounded multicounter machines is closed under iterated pseudo-inversion. To conclude this section we examine the closure properties for context-sensitive languages and establish the following result. A similar result for inversion of context-sensitive languages is known from Dassow et al. [6].

Proposition 2. *Given a context-sensitive language L, $\mathbb{PI}^*(L)$ is context-sensitive.*

4 Pseudo-inversion-Freeness

We investigate the decidability problem for pseudo-inversion-freeness and establish the closure properties of pseudo-inversion-free languages.

4.1 Decidability of Pseudo-inversion-freeness

We say that a language L is pseudo-inversion-free if no string in L is a pseudo-inversion substring of any other string in L. We consider the decidability problem of pseudo-inversion-freeness when L is regular or context-free.

We first consider a simple case when we are given two strings of the same length. We determine whether or not a string is not a pseudo-inversion of the other string. In other words, given two strings u and v, is u in $\mathbb{PI}(v)$? We present a linear-time algorithm in the size of u for the question. We rely on the following observation to simplify the presentation of the algorithm.

Observation 5. *Let u and v be two strings of the same length. Then, $u \in \mathbb{PI}(v)$ if and only if $u = wxy$ and $v^R = wx^Ry$, where $wy \neq \lambda$.*

The main idea of the linear-time algorithm is to scan two strings v^R and u from both end-sides until we find an index where two strings have different

Algorithm 1. A linear-time algorithm for deciding $v \in \mathbb{PI}(u)$

```
Input: Two strings u and v of the same length n
1 i ← 0
2 j ← n
3 while i ≤ n ∧ u[i] = v^R[i] do  i ← i + 1
4 while 1 ≤ j ∧ u[j] = v^R[j] do  j ← j − 1
5 if i ≥ j then  return false
6 else
7     for k = i to j do
8         if u[k] ≠ v^R[i + j − k] then  return false
9     return true
```

characters. Let $\mathbb{M}_L$ denote the *left maximum matching index*, where the first discrepancy occurs and $\mathbb{M}_R$ denote the *right maximum matching index* where the last discrepancy occurs. Lastly, we check whether or not $u[\mathbb{M}_L \cdots \mathbb{M}_R]^R = v^R[\mathbb{M}_L \cdots \mathbb{M}_R]$. See Algorithm 1 for the whole procedure.

Theorem 6. *Given two strings u and v of length n, we can determine whether or not $v \in \mathbb{PI}(u)$ in $O(n)$ time.*

We can also determine if $v \in \mathbb{PI}^*(u)$ by checking whether or not the two Parikh vectors $\Psi(u)$ and $\Psi(v)$ are the same.

Corollary 3. *Given two strings u and v of length n, we can determine whether or not $v \in \mathbb{PI}^*(u)$ in $O(n)$ time.*

Next, we consider the regular language case. Recalling from Definition 3 the notion of pseudo-inversion-freeness, we can decide whether or not a regular language L is pseudo-inversion-free by checking whether or not $\Sigma^* \cdot \mathbb{PI}(L) \cdot \Sigma^* \cap L = \emptyset$.

Theorem 7. *Given an FA of size n recognizing a regular language L, we can determine whether or not L is pseudo-inversion-free in $O(n^4)$ time.*

Proof. Based on the NFA construction in Theorem 1, we can construct an NFA of size $O(n^3)$ recognizing $\mathbb{PI}(L)$. Since we can check the intersection emptiness of two NFAs of size m and n in $O(mn)$ time [24], we can determine whether or not L is pseudo-inversion-free in $O(n^3 \times n) = O(n^4)$ time. □

Theorem 7 shows that it is decidable whether or not a given language L is pseudo-inversion-free in polynomial time when L is regular. We prove that pseudo-inversion-freeness is undecidable for context-free languages.

First we recall the following undecidability result. An instance of the *Post's Correspondence Problem* (PCP) [19] consists of $n \in \mathbb{N}$ and two ordered n-tuples of strings (U, V), where $U = (u_0, u_1, \ldots, u_{n-1})$ and $V = (v_0, v_1, \ldots, v_{n-1})$, $u_i, v_i \in \Sigma^*$, $0 \leq i \leq n-1$. A solution for the PCP instance (U, V) is a sequence of integers $i_1, \ldots, i_k$, $0 \leq i_j \leq n-1$, $j = 1, \ldots, k$, $k \geq 1$, such that

$$u_{i_1} u_{i_2} \cdots u_{i_k} = v_{i_1} v_{i_2} \cdots v_{i_k}.$$

Proposition 3 (E. Post [19]). *The decision problem of determining whether or not a given PCP instance has a solution is unsolvable.*

Now we can prove that deciding the pseudo-inversion-freeness of a given context-free language is undecidable by reducing PCP to this problem.

Theorem 8. *It is undecidable to determine whether or not a given context-free language L is pseudo-inversion-free.*

Proof. Let Σ be an alphabet and (U, V) be an instance of Post's Correspondence Problem, where $U = (u_0, u_1, \ldots, u_{n-1})$ and $V = (v_0, v_1, \ldots, v_{n-1})$. Assume that the symbols $0, 1, \#, \$, \%, \phi, \natural$ and $\flat$ are not in Σ. Let $\Sigma' = \Sigma \cup \{0, 1, \#, \$, \%, \phi, \natural, \flat\}$. For any nonnegative integer i, let β_i be the shortest binary representation of i.

We define a linear grammar $G = (N, \Sigma', R, S)$, where

- $N = \{S, T_U, T_V\}$ is a nonterminal alphabet,
- Σ' is a terminal alphabet,
- S is the sentence symbol, and
- R has the following rules:
  - $S \to \beta_i \phi T_U u_i \#\#\%\flat\natural\natural \mid \natural\natural\%\flat\#\# v_i^R T_V \phi \beta_i$,
  - $T_U \to \beta_i \phi T_U u_i \mid \beta_i \$ u_i$, and
  - $T_V \to v_i^R T_V \phi \beta_i \mid v_i^R \$ \beta_i$

  for $i \in \{0, 1, \ldots, n-1\}$.

Then $L(G)$ consists of the following strings:

$$\beta_{i_{n-1}} \phi \cdots \phi \beta_{i_0} \$ u_{i_0} \cdots u_{i_{n-2}} u_{i_{n-1}} \#\#\%\flat\natural\natural \tag{1}$$

and

$$\natural\natural\%\flat\#\# v_{i_{n-1}}^R v_{i_{n-2}}^R \cdots v_{i_0}^R \$ \beta_{i_0} \phi \cdots \phi \beta_{i_{n-1}}. \tag{2}$$

We now show that $L(G)$ is not pseudo-inversion-free if and only if the PCP instance (U, V) has a solution.

($\Longleftarrow$) We prove that $L(G)$ is not pseudo-inversion-free if the PCP instance (U, V) has a solution. Assume that the PCP instance (U, V) has a solution. Let $z = vwx$ and $z' = ux^R wv^R y$, where $xv \neq \lambda$. Then, L is not pseudo-inversion-free if both z' and z exist in L. Since the PCP instance has a solution by the assumption, there should be a sequence $i_0, i_1, \ldots, i_{n-2}, i_{n-1}$ satisfying

$$u_{i_0} \cdots u_{i_{n-2}} u_{i_{n-1}} = v_{i_0} \cdots v_{i_{n-2}} v_{i_{n-1}}.$$

Now we decompose (1) into $uvwxy$ such that

- $v = \beta_{i_{n-1}} \phi \cdots \phi \beta_{i_0} \$ u_{i_0} \cdots u_{i_{n-2}} u_{i_{n-1}} \#\#$,
- $w = \%\flat$,
- $x = \natural\natural$, and
- $u, y = \lambda$.

Then, $x^R w v^R = \natural\natural\%\flat\#\#u_{i_{n-1}}u_{i_{n-2}}\cdots u_{i_0}\$\beta_{i_0}\phi\cdots\phi\beta_{i_{n-1}} \in L(G)$. Therefore, $L(G)$ is not pseudo-inversion-free.
($\Longrightarrow$) If $L(G)$ is not pseudo-inversion-free, then there exist two strings $z = vwx$ and $z' = ux^R w v^R y$ in $L(G)$, where $xv \neq \lambda$. Then, there are two possible decompositions as follows:

C1. $u = \lambda, v = \beta_{i_{n-1}}\phi\cdots\phi\beta_{i_0}\$u_{i_0}\cdots u_{n_{i-1}}\#\#, w = \%\flat, x = \natural\natural$, and $y = \lambda$.
C2. $u = \lambda, v = \natural\natural, w = \%\flat, x = \#\#v^R_{i_{n-1}}v^R_{i_{n-2}}\cdots v^R_{i_0}\$\beta_{i_0}\phi\cdots\phi\beta_{i_{n-1}}$, and $y = \lambda$.

It implies that the PCP instance (U, V) has a solution since

$$v = \beta_{i_{n-1}}\phi\cdots\phi\beta_{i_0}\$u_{i_0}\cdots u_{n_{i-1}}\#\#$$

should be equal to

$$x^R = \beta_{i_{n-1}}\phi\cdots\phi\beta_{i_0}\$v_{i_0}\cdots v_{n_{i-1}}\#\#.$$

Thus, $L(G)$ is not pseudo-inversion-free if and only if the PCP instance (U, V) has a solution. Since PCP is undecidable [19], it is also undecidable whether or not L is pseudo-inversion-free when L is context-free. □

We summarize the results for decision properties of pseudo-inversion-freeness:

(i) It can be decided in polynomial time whether or not a given regular language is pseudo-inversion-free (Theorem 7).
(ii) It is undecidable whether or not a given linear context-free language is pseudo-inversion-free (Theorem 8).

4.2 Closure Properties of Pseudo-inversion-free Languages

We first consider closure properties of the pseudo-inversion-free languages under the basic operations.

Theorem 9. *Pseudo-inversion-free languages are closed under intersection but not under catenation or union.*

We note that the pseudo-inversion free languages are not closed under complementation nor Kleene star. Moreover, the complementation or the Kleene star of any pseudo-inversion-free language is not pseudo-inversion-free.

Theorem 10. *For any pseudo-inversion-free language $L \subseteq \Sigma^*$, $\bar{L}$ is not pseudo-inversion-free.*

Theorem 11. *For a nonempty language $L \subseteq \Sigma^* \setminus \{\lambda\}$, $L^m \cup L^n$ is not pseudo-inversion-free, for $1 \leq m < n$. Moreover, L^* is not pseudo-inversion-free, either.*

Proof. Let $w = au$ be a string in L, where $a \in \Sigma$ and $u \in \Sigma^*$. Then, we have $w^m \in L^m$ and $w^n \in L^n$. Then, $w^m = av$ and $w^n = avw^{n-m}$, where $v = uw^{m-1}$. Since $w = au$, $w^n = avauw^{n-m-1}$ in which va appears as a substring. Since $va \in \mathbb{PI}(w^m)$, $L^m \cup L^n$ is not pseudo-inversion-free. It is easy to see that L^* is not pseudo-inversion-free since $L^* = L^0 \cup L^1 \cup L^2 \cup \cdots$. □

5 Conclusions

We have defined a biologically inspired operation called the pseudo-inversion. Informally, the pseudo-inversion incompletely reverses the order of strings while the inversion operation reverses the order of infix of strings. Given a string $w = uxv$, we define the pseudo-inversion of w to be the set of strings $v^R x u^R$, where $uv \neq \lambda$.

We have investigated the closure properties of the pseudo-inversion operation and the iterated pseudo-inversion operation. While regular languages are closed under the pseudo-inversion, context-free languages are not closed. Moreover, we have established that the iterated pseudo-inversion is equivalent to the permutation operation. We also have considered the problem of deciding whether or not a given language is pseudo-inversion-free. We have designed a polynomial-time algorithm for regular languages and established an undecidability result for linear context-free languages.

Acknowledgements. Cho, Han, Kang and Ko were supported by the Basic Science Research Program through NRF funded by MEST (2012R1A1A2044562), Kim was supported by NRF-2013-Global Ph.D. Fellowship Program and Salomaa was supported by the Natural Sciences and Engineering Research Council of Canada Grant OGP0147224.

References

1. Cantone, D., Cristofaro, S., Faro, S.: Efficient string-matching allowing for non-overlapping inversions. Theoretical Computer Science 483, 85–95 (2013)
2. Chiniforooshan, E., Daley, M., Ibarra, O.H., Kari, L., Seki, S.: One-reversal counter machines and multihead automata: Revisited. Theoretical Computer Science 454, 81–87 (2012)
3. Cho, D.-J., Han, Y.-S., Kim, H.: Alignment with non-overlapping inversions on two strings. In: Pal, S.P., Sadakane, K. (eds.) WALCOM 2014. LNCS, vol. 8344, pp. 261–272. Springer, Heidelberg (2014)
4. Daley, M., Ibarra, O.H., Kari, L.: Closure and decidability properties of some language classes with respect to ciliate bio-operations. Theoretical Computer Science 306(1-3), 19–38 (2003)
5. Daley, M., Kari, L., McQuillan, I.: Families of languages defined by ciliate bio-operations. Theoretical Computer Science 320(1), 51–69 (2004)
6. Dassow, J., Mitrana, V., Salomaa, A.: Operations and language generating devices suggested by the genome evolution. Theoretical Computer Science 270(1), 701–738 (2002)
7. Deaton, R., Garzon, M., Murphy, R.C., Rose, J.A., Franceschetti, D.R., Stevens Jr., S.E.: Genetic search of reliable encodings for DNA-based computation. In: First Conference on Genetic Programming, pp. 9–15 (1996)
8. Garzon, M., Deaton, R., Nino, L.F., Stevens, E., Wittner, M.: Encoding genomes for DNA computing. In: Proceedings of the Third Annual Conference on Genetic Programming 1998, pp. 684–690 (1998)

9. Ginsburg, S.: Algebraic and automata-theoretic properties of formal languages. North-Holland Publishing Company (1975)
10. Hopcroft, J., Ullman, J.: Introduction to Automata Theory, Languages, and Computation, 2nd edn. Addison-Wesley, Reading (1979)
11. Hussini, S., Kari, L., Konstantinidis, S.: Coding properties of DNA languages. Theoretical Computer Science 290(3), 1557–1579 (2003)
12. Ibarra, O.H.: Reversal bounded multicounter machines and their decision problems. Journal of the ACM 25, 116–133 (1978)
13. Jonoska, N., Kari, L., Mahalingam, K.: Involution solid and join codes. Fundamenta Informaticae 86(1-2), 127–142 (2008)
14. Jonoska, N., Mahalingam, K., Chen, J.: Involution codes: With application to DNA coded languages. Natural Computing 4(2), 141–162 (2005)
15. Jürgensen, H., Konstantinidis, S.: Codes. In: Handbook of Formal Languages. I, pp. 511–607. Springer (1997)
16. Kari, L., Losseva, E., Konstantinidis, S., Sosík, P., Thierrin, G.: A formal language analysis of DNA hairpin structures. Fundamenta Informaticae 71(4), 453–475 (2006)
17. Kari, L., Mahalingam, K.: DNA codes and their properties. In: Mao, C., Yokomori, T. (eds.) DNA12. LNCS, vol. 4287, pp. 127–142. Springer, Heidelberg (2006)
18. Kececioglu, J., Sankoff, D.: Exact and approximation algorithms for the inversion distance between two chromosomes. In: Apostolico, A., Crochemore, M., Galil, Z., Manber, U. (eds.) CPM 1993. LNCS, vol. 684, pp. 87–105. Springer, Heidelberg (1993)
19. Post, E.L.: A variant of a recursively unsolvable problem. Bulletin of the American Mathematical Society 52(4), 264–268 (1946)
20. Salomaa, A.: Formal Languages. Academic Press (1973)
21. Schniger, M., Waterman, M.S.: A local algorithm for DNA sequence alignment with inversions. Bulletin of Mathematical Biology 54(4), 521–536 (1992)
22. Shallit, J.: A Second Course in Formal Languages and Automata Theory. Cambridge University Press (2009)
23. Vellozo, A.F., Alves, C.E.R., do Lago, A.P.: Alignment with non-overlapping inversions in $O(n^3)$-time. In: Bücher, P., Moret, B.M.E. (eds.) WABI 2006. LNCS (LNBI), vol. 4175, pp. 186–196. Springer, Heidelberg (2006)
24. Wood, D.: Theory of Computation. Harper & Row (1986)
25. Yokomori, T., Kobayashi, S.: DNA evolutionary linguistics and RNA structure modeling: A computational approach. In: Proceedings of the 1st Intelligence in Neural and Biological Systems, pp. 38–45. IEEE Computer Society (1995)