Nat Comput (2016) 15:31–39
DOI 10.1007/s11047-015-9502-9

Pseudo-inversion: closure properties and decidability

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

Published online: 19 May 2015

Abstract We consider a pseudo-inversion operation inspired by biological events, such as DNA sequence transformations, where only parts of a string are reversed. We define the pseudo-inversion of a string $w = uxv$ to be the set 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 study the iterated pseudo-inversion operation and show that the iterated pseudo-inversion of a context-free language is recognized by a nondeterministic reversal-bounded multicounter machine. Finally, we introduce the notion of pseudo-inversion-freeness and examine closure properties and decidability problems for regular and context-free languages. We demonstrate that pseudo-inversion-freeness is decidable in polynomial time for regular languages and undecidable for context-free languages.

Keywords Pseudo-inversion · Bio-inspired operation · Closure properties · Decidability · Formal languages · Reversal-bounded multicounter machines

A preliminary version appeared in Proceedings of Unconventional Computation & Natural Computation 2014, UCNC 2014, LNCS, vol. 8553, Springer-Verlag, 2014, pp. 93–104.

✉ Yo-Sub Han
emmous@cs.yonsei.ac.kr

Da-Jung Cho
dajung@cs.yonsei.ac.kr

Shin-Dong Kang
shindong1992@cs.yonsei.ac.kr

Hwee Kim
kimhwee@cs.yonsei.ac.kr

Sang-Ki Ko
narame7@cs.yonsei.ac.kr

Kai Salomaa
ksalomaa@cs.queensu.ca

[1] Department of Computer Science, Yonsei University, 50, Yonsei-Ro, Seodaemun-Gu, Seoul 120–749, Republic of Korea

[2] School of Computing, Queen's University, Kingston, ON K7L 3N6, Canada

1 Introduction

There have been many studies that relate biological phenomena and formal languages (Deaton et al. 1996; Garzon et al. 1998). Several researchers investigated the algebraic and code-theoretic properties of DNA encoding based on formal language theory (Hussini et al. 2003; Jonoska et al. 2008, 2005; Kari and Mahalingam 2006). For instance, Jonoska et al. (2008) introduced involution codes based on the Watson-Crick complementarity, and Kari and Mahalingam (2006) investigated the algebraic properties of DNA languages that avoid intermolecular cross hybridization. Kari et al. (2006) 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 several 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 (Cantone et al. 2013; Cho et al. 2015a, 2015b; Daley et al. 2003, 2004; Dassow et al. 2002; Ibarra 2014; Schöniger and Waterman 1992; Yokomori and Kobayashi 1995. 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 (1995) showed that the inversion can be simulated by a set of primitive operations and languages using GSM mapping. Dassow et al. (2002) observed 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. (2003, 2004) 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. Recently, Ibarra (2014) established the closure and decidability properties of some classes of languages under hairpin-inversion, pseudo-inversion and other bio-operations using reversal-bounded counters. Researchers also considered the inversion in the string matching problems; namely an inversion occurs in a pattern or in an input text (Cantone et al. 2013; Schöniger and Waterman 1992).

We introduce a new operation called a *pseudo-inversion* operation. While the inversion operation reverses an infix of an input sequence, the pseudo-inversion operation reverses only the outermost parts of the sequence and the middle part of the sequence is not reversed. We notice that there are two possible situations where a pseudo-inversion occurs in practice. See Fig. 1 for an example.

(i) 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.
(ii) 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 occurs twice, the same as the original sequence.

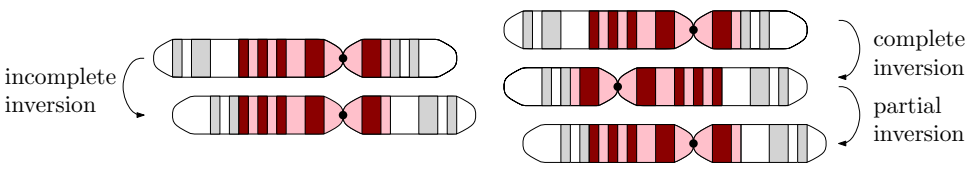
The operation leads us to investigate the problem of determining whether or not, given two strings of the same length, one string is a pseudo-inversion of the other string. We tackle the problem and obtain a linear-time algorithm.

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 various decision problems for the 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 (Jürgensen and Konstantinidis 1997) and pseudo-inversion-free languages have potential applications in DNA encoding.

We give definitions and notations in Sect. 2. We define the pseudo-inversion and the iterated pseudo-inversion, and establish some closure properties in Sect. 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 Sect. 4 and conclude the paper in Sect. 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 empty 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; $\bar{L} = \Sigma^* \setminus L$. Given an



(a) first case **(b)** second case

Fig. 1 An example of two possible situations where a pseudo-inversion occurs in practice. **a** describes the pseudo-inversion that occurs as a consequence of incomplete inversion. Note that the *middle part* is not reversed. **b** shows the pseudo-inversion resulted from two inversions, where the first inversion occurs for the whole sequence and the second inversion occurs in the middle part. Compared with the original sequence in both cases, the middle part of the resulting sequence is not reversed

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 the *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 $1 \le i \le j \le |w|$. 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. Given a transition $\delta(p, a) = q$, we say that p has an *out-transition* and q has an *in-transition*. By an NFA we mean a nondeterministic automaton without λ-transitions, that is, A is an NFA if δ is a function from $Q \times \Sigma$ into 2^Q. The automaton A is *deterministic* (a DFA) if Q_0 is a singleton set and δ is a (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.

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, $S \in V$ is the start variable and every production in R is of the form $\alpha B \gamma \to \alpha\beta\gamma$ for some $\alpha, \gamma \in (V \cup \Sigma)^*$, $\beta \in (V \cup \Sigma)^+$ and $B \in V$.

A *nondeterministic reversal-bounded multicounter machine* (NCM) (Chiniforooshan et al. 2012; Ibarra 1978) 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 (Hopcroft and Ullman 1979; Shallit 2009; Wood 1986) for more details on formal language theory.

[1] Unrestricted two-counter machines accept all recursively enumerable languages (Ginsburg 1975).

3 Pseudo-inversion

The inversion operation is one of the most common operations on biosequences. When the inversion occurs in a biosequence w, sometimes the whole w may not be completely inversed because of various reasons in practice. This gives rise to a partial inversion of w where a middle part of w is not inversed. We call this process a *pseudo-inversion*. Figure 2 depicts an example of a pseudo-inversion of a string.

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, $\mathbb{PI}(\lambda) = \emptyset$. We can extend the *pseudo-inversion* to languages. Given a language L,

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

We define an *iterated pseudo-inversion* operation as follows: Given a string w, we let $\mathbb{PI}^1(w) = \mathbb{PI}(w)$ and $\mathbb{PI}^{i+1}(w) = \mathbb{PI}(\mathbb{PI}^i(w))$ for a positive integer $i > 0$.

Definition 2 Given a string w and a language L, we define the *iterated pseudo-inversion* $\mathbb{PI}^*(w)$ and the *iterated pseudo-inversion* $\mathbb{PI}^*(L)$ to be

$$\mathbb{PI}^*(w) = \bigcup_{i=1}^{\infty} \mathbb{PI}^i(w) \text{ and } \mathbb{PI}^*(L) = \bigcup_{w \in L} \mathbb{PI}^*(w).$$

Next we introduce 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 of a 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 of a 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

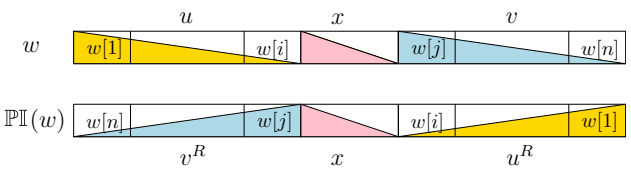


Fig. 2 Given a string $w = uxv$, the pseudo-inversion $\mathbb{PI}(w)$ of w inverses u and v (*outer parts*) from w but not x (*middle part*)

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 (Hopcroft and Ullman 1979; Wood 1986). We may need one more state if we do not allow multiple initial states. We show that regular languages are closed under pseudo-inversion.

Theorem 1 *Regular languages are closed under pseudo-inversion.*

Proof Let $A = (Q, \Sigma, \delta, Q_0, F)$ be a λ-NFA for a regular language L. We show that there is a λ-NFA that accepts $\mathbb{PI}(L)$. Without loss of generality, we assume that A has only one initial state and one final state, and the initial state has no in-transitions and the final state has no out-transitions. Thus $Q_0 = \{q_0\}$ and $F = \{q_f\}$ are singleton sets. We define a λ-NFA $B = (P, \Sigma, \gamma, P_0, F_B)$ for $\mathbb{PI}(L(A))$, where

- $P = Q^3 \cup Q \cup \tilde{Q}$, where $\tilde{Q} = \{\tilde{q} \mid q \in Q\}$,
- $P_0 = \{q_f\}$,
- $F_B = \{q_0, \tilde{q_0}\}$, and

the transition function γ is as follows:

(i) For all $q, p \in Q$ and $a \in \Sigma$, if $p \in \delta(q, a)$, then $q \in \gamma(p, a)$ and $\tilde{q} \in \gamma(\tilde{p}, a)$.
(ii) For all $q, p \in Q$, $(q, q, p) \in \gamma(p, \lambda)$, for $p \neq q_f$ or $q \notin F_B$.
(iii) For all $q, p, r_1, r_2 \in Q$ and $a \in \Sigma$, if $r_2 \in \delta(r_1, a)$, then $(q, r_2, p) \in \gamma((q, r_1, p), a)$.
(iv) For all $q, p \in Q, \tilde{q} \in \gamma((q, p, p), \lambda)$.

The automaton B operates as follows. The transition (i) simulates a computation of A in reverse, beginning from a final state of A. We choose a state $q \in Q$ nondeterministically using a λ-transition, and we reach a state (q, q, p) by the transition (ii). In the rules (ii) the conditions $p \neq q_f$ or $q \notin F_B$, together with the assumptions that A has no in-transitions to p_0 nor out-transitions from q_f guarantee that B on input string s simulates the computation of A on an input s' such that s is obtained from s' by inverting a nonempty string. The transition (iii) allows B to simulate the original computation of A from q to p. After B reaches the state (q, p, p), it can make a λ-transition to state $\tilde{q}$, and using the transition (i), B continues the reverse computation of A. Eventually, B accepts a string $v^R w u^R$ if A has an accepting computation for uwv. Figure 3 illustrates an example computation of the λ-NFA B. □

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

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

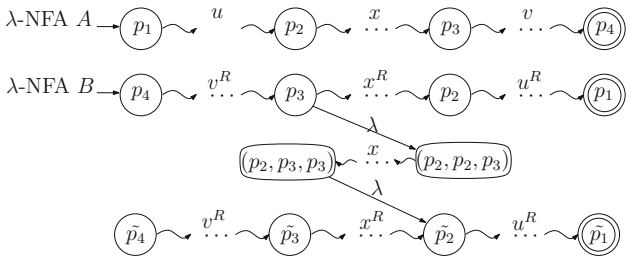


Fig. 3 An illustrative example of constructing a λ-NFA B recognizing $\mathbb{PI}(L(A))$. Note that if the λ-NFA A accepts a string uxv, then B accepts all $v^R x u^R \in \mathbb{PI}(L(A))$

Notice that context-free languages are closed under the reversal operation (Hopcroft and Ullman 1979). However, we demonstrate that context-free languages are not closed under the pseudo-inversion operation.

Theorem 2 *Context-free languages are not closed under pseudo-inversion.*

Proof We prove the statement by the context-free pumping lemma (Hopcroft and Ullman 1979; Wood 1986). Consider a 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. 4 for an illustration. By the pumping lemma, we split w into five parts, $w = uvxyz$, where substrings u, v, x, y and z satisfy the conditions of the pumping lemma. It follows that $|vxy| \leq n$, and hence, vxy cannot contain both a's and d's, and vxy cannot contain both b's and c's.

Notice that if $w \in \mathbb{PI}(L)$, 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. Thus $uv^2xy^2z \notin \mathbb{PI}(L)$ and we conclude that $\mathbb{PI}(L)$ is not context-free. □

3.2 Iterated pseudo-inversion

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

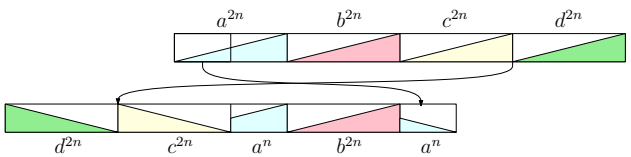


Fig. 4 For a language $L = a^i b^j c^j d^i \mid i, j \geq 1$, we pick a string $d^{2n} c^{2n} a^n b^{2n} a^n \in \mathbb{PI}(z)$, where $z = a^{2n} b^{2n} c^{2n} d^{2n} \in L$

Theorem 3 *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, $\mathbb{PI}^*(w) = \pi(w)$.*

Proof We first show that $\mathbb{PI}^*(w) \subseteq \pi(w)$. For the sake of contradiction, assume that a string $u \in \mathbb{PI}^*(w)$ is not in $\pi(w)$. This implies that there exists a character $a \in \Sigma$ such that $|u|_a \neq |w|_a$. However, the pseudo-inversion operation does not affect the number of character occurrences—it does not insert or delete a character but only relocates characters of w. Therefore, it is impossible to have a string $u \in \mathbb{PI}(w)$ such that $|u|_a \neq |w|_a$—a contradiction.

Next we show that $\pi(w) \subseteq \mathbb{PI}^*(w)$; any permutation of w is in $\mathbb{PI}^*(w)$. Let u be an arbitrary string from $\pi(w)$. We claim that w can be transformed to u by applying exactly n pseudo-inversions. Let w_i be the string obtained after i pseudo-inversions. We assume that $u[1 \cdots i-1]^R = w_{i-1}[n-i+2 \cdots n]$ if i is odd and $u[1 \cdots i-1] = w_{i-1}[1 \cdots i-1]$ otherwise. We prove that there exists a sequence of pseudo-inversions that satisfies the assumption. For the ith pseudo-inversion, assume that $w_{i-1}[j] = u[i]$.

(i) i is odd: We have the following string $w_i \in \mathbb{PI}^i(w)$

$$\underbrace{w_{i-v1}[n-i+2 \cdots n]^R}_{\text{inversed}} \underbrace{w_{i-1}[j \cdots n-i+1]}_{\text{non-inversed}} \underbrace{w_{i-1}[1 \cdots j-1]^R}_{\text{inversed}}$$
$$= u[1 \cdots i-1]u[i]w_{i-1}[j+1 \cdots n-i+1]w_{i-1}[1 \cdots j-1]^R.$$

After the pseudo-inversion, $u[1 \cdots i] = w_i[1 \cdots i]$.

(ii) i is even: We have the following string $w_i \in \mathbb{PI}^i(w)$

$$\underbrace{w_{i-1}[j+1 \cdots n]^R}_{\text{inversed}} \underbrace{w_{i-1}[i \cdots j]}_{\text{non-inversed}} \underbrace{w_{i-1}[1 \cdots i-1]^R}_{\text{inversed}}$$
$$= w_{i-1}[j+1 \cdots n]^R w_{i-1}[i \cdots j-1]u[i]u[1 \cdots i-1]^R.$$

After the pseudo-inversion, $u[1 \cdots i]^R = w_i[n-i+1 \cdots n]$.

Figure 5 shows an example of this process. From these pseudo-inversions, we can transform w to u in n steps if n is odd. When n is even, we set $w_{i-1}[j] = u[n-i+1]$ instead of $u[i]$. The rest of the process is similar to the odd case. □

Based on Theorem 3, we show that regular and context-free languages are not closed under the iterated pseudo-inversion operation.

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 is not necessarily context-free.*

Proof Let $L = L((abc)^*)$. The iterated pseudo-inversion $\mathbb{PI}^*(L)$ of L is

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

Note that

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

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

Below in Proposition 1, 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.

As a consequence of Theorem 3, we see that the iterated pseudo-inversion of a context-free language can be recognized by a nondeterministic reversal-bounded multi-counter 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 (Ibarra 1978). Furthermore, NCMs cannot recognize, for example, the set of marked palindromes $\{w\#w^R \mid w \in \{0,1\}^*\}$ (Chiniforooshan et al. 2012).

Corollary 2 *If L is a context-free language over Σ, $\mathbb{PI}^*(L)$ can be recognized by an NCM with $|\Sigma|$ counters each of which makes only one reversal.*

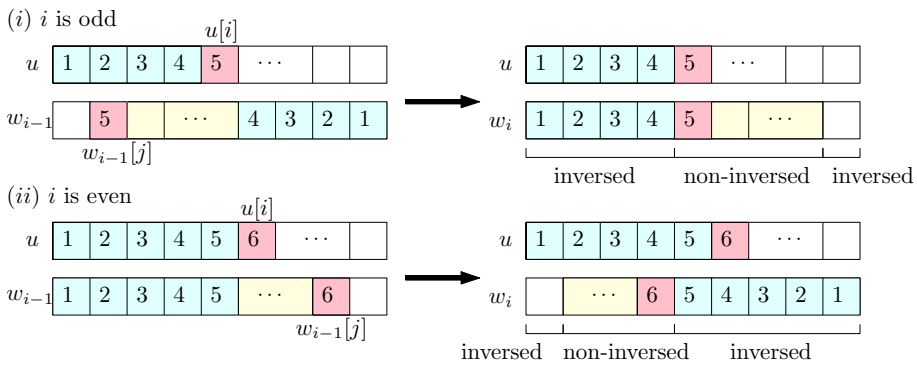


Fig. 5 The ith pseudo-inversion that transforms w to u. For better readability, we set $u = 1234 \cdots n$ in the figure

Proof There exists a regular language L' that is letter-equivalent to L (Salomaa 1973, part I, Theorem 7.2). 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 A. 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 3, the NCM recognizes the language $\mathbb{PI}^*(L)$. □

Corollary 2 uses only Theorem 3 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 3 *If the Parikh set of L is semi-linear, then $\mathbb{PI}^*(L)$ can be recognized by a nondeterministic reversal-bounded multicounter machine.*

Corollary 3 implies, in particular, that the family of languages recognized by nondeterministic reversal-bounded multicounter machines is closed under iterated pseudo-inversion. We examine the closure properties for context-sensitive languages and establish the following result. Dassow et al. (2002) established a similar result for inversion of context-sensitive languages.

Proposition 1 *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 of a 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 a pseudo-inversion of the other string. In other words, given two strings u and v, is v in $\mathbb{PI}(u)$? We present a linear-time algorithm in the size of u. We rely on the following observation to simplify the presentation of the algorithm.

Observation 1 *Let u and v be two strings of the same length. Then, $v \in \mathbb{PI}(u)$ 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 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.

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 ← 1
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
```

Lemma 2 *Given two strings u and v, $v \in \mathbb{PI}(u)$ if Algorithm 1 returns* **true**.

Proof Suppose that Algorithm 1 returns **true**, and let $\mathbb{M}_L = i$ and $\mathbb{M}_R = j$. This implies that there exist following three substrings:

(i) $u[1 \cdots i] = v^R[1 \cdots i]$,
(ii) $u[j \cdots n-1] = v^R[j \cdots n-1]$ and
(iii) $u[i+1 \cdots j-1] = v^R[i+1 \cdots j-1]^R$.

Let $u = wxy$, where $w = u[1 \cdots i], x = u[i+1 \cdots j-1]$ and $y = [j \cdots n]$. Then a string $v^R = wx^Ry$, therefore, $v \in \mathbb{PI}(u)$. □

Lemma 3 *Let $\mathbb{M}_L = i$ and $\mathbb{M}_R = j$ for $1 \leq s < i < j < k \leq n-1$. If a substring $u[s \cdots t]$ is the reversal of $v^R[s \cdots t]$, then $u[i+1 \cdots j-1]$ and $v^R[i+1 \cdots j-1]$ are the reversal of each other.*

Proof Since $\mathbb{M}_L = i$ and $\mathbb{M}_R = j$, it is immediate that $u[1\cdots i] = v^R[1\cdots i]$ and $u[j\cdots n] = v^R[j\cdots n]$. Then,

$$u[s\cdots i] = v^R[s\cdots i] = u[j\cdots t] = v^R[j\cdots t]$$

because $u[s\cdots t] = v^R[s\cdots t]^R$. It implies that $u[i+1\cdots j-1]$ and $v^R[i+1\cdots j-1]$ are the reversal of each other. □

Lemma 4 *If there exists a string $v \in \mathbb{PI}(u)$, then* Algorithm 1 *returns* **true**.

Proof Suppose that there exists a string $v \in \mathbb{PI}(u)$. Consider a string $u = wxy$, where $w = u[1\cdots i], x = u[i+1\cdots j-1]$ and $y = u[j\cdots n]$ for $1 \le i < j \le n$. This implies that $v = y^R x w^R$, $\mathbb{M}_L = i$ and $\mathbb{M}_R = j$. Therefore, Algorithm 1 returns **true** since there exist substrings $u[1\cdots i] = v^R[1\cdots i]$, $u[j\cdots n] = v^R[j\cdots n]$ and $u[i+1\cdots j-1] = v^R[i+1\cdots j-1]^R$. Note that even if a substring $u[s\cdots t]$ is a reversal of $v^R[s\cdots t]$ for $1 \le s < i < j < k \le n$, Algorithm 1 still returns **true** by Lemma 3. □

Theorem 4 *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 4 *Given two strings u and v of length n, we can determine whether or not $v \in \mathbb{PI}^*(u)$ in $O(n)$ time.*

Recalling from Definition 3 the notion of pseudo-inversion-freeness, we can also 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 5 *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 a λ-NFA of size $O(n^3)$ recognizing $\mathbb{PI}(L)$. Since we can check the intersection emptiness of two λ-NFAs of size m and n in $O(mn)$ time (Wood 1986), we can determine whether or not L is pseudo-inversion-free in $O(n^3 \times n) = O(n^4)$ time. □

Theorem 5 shows that it is decidable whether or not a regular language L is pseudo-inversion-free in polynomial time. Next, 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) (Post 1946) 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 $V = (v_0, v_1, \ldots, v_{n-1})$, $u_i, v_i \in \Sigma^*$, $0 \le i \le n-1$. A solution for the PCP instance (U, V) is a sequence of integers $i_1, \ldots, i_k$, $0 \le i_j \le n-1$, $j = 1, \ldots, k$, $k \ge 1$, such that

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

Proposition 2 (Post 1946) *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 6 *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. 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 $z' = ux^R w v^R y$, where $xv \ne \lambda$. 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. Next, we show that 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 (Post 1946), it is also undecidable whether or not L is pseudo-inversion-free when L is context-free. □

4.2 Closure properties of pseudo-inversion-free languages

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

Theorem 7 *Pseudo-inversion-free languages are closed under intersection but not under catenation or union.*

Proof We consider the three cases separately:

(i) **Intersection**: Assume that $L = L_1 \cap L_2$ is not pseudo-inversion-free whereas L_1 and L_2 are pseudo-inversion-free. Then, there are two strings $z = uvwxy$ and $z' = x^R w v^R$ in L, where $vx \neq \lambda$ and $u, w, y \in \Sigma^*$. Since two strings z, z' exist in both L_1 and L_2, this implies that L_1 and L_2 are not pseudo-inversion-free—a contradiction.

(ii) **Catenation**: Let $L_1 = \{abcc, cbcc\}$ and $L_2 = \{abbc, abba\}$ over $\Sigma = \{a, b, c\}$. Then, $abccabbc$ and $cbccabba \in L_1 \cdot L_2$, which implies that $L_1 \cdot L_2$ is not pseudo-inversion-free. Note that a string $abccabbc \in L_1 \cdot L_2$ is a pseudo-inversion of a substring of a string $cbccabba \in L_1 \cdot L_2$. Therefore, pseudo-inversion-free languages are not closed under catenation.

(iii) **Union**: Let $L_1 = \{ab, aa\}$ and $L_2 = \{ba, cc\}$ over $\Sigma = \{a, b, c\}$. Since $ab \in \mathbb{PI}(ba)$, $L_1 \cup L_2 = \{ab, aa, ba, cc\}$ is not pseudo-inversion-free. Thus pseudo-inversion-free languages are not closed under union.

□

Next, we show that the complementation or the Kleene star of any pseudo-inversion-free language is not pseudo-inversion-free.

Theorem 8 *For any pseudo-inversion-free language $L \subseteq \Sigma^*$ where $|\Sigma| \geq 1$, $\bar{L}$ is not pseudo-inversion-free.*

Proof If $L = \emptyset$, $\bar{L}$ is not pseudo-inversion-free since $\sigma, \sigma\sigma \in \bar{L}$ for some $\sigma \in \Sigma$. If L is not empty, assume that $\bar{L}$ is pseudo-inversion-free. We consider a string $z = uvw \in L$. Then, for a character $\sigma \in \Sigma$, $z' = \sigma w^R v u^R \in \bar{L}$ since $z \in \mathbb{PI}(w^R v u^R)$. Now, consider a string $z'' = w^R v u^R \sigma$. Since $z \in \mathbb{PI}(w^R v u^R)$, $z'' \notin L$. On the other hand, since $z' \in \mathbb{PI}(w^R v u^R \sigma)$, $z' \notin \bar{L}$—a contradiction. Therefore, we conclude that $\bar{L}$ is not pseudo-inversion-free. □

Theorem 9 *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 $L^* = L^0 \cup L^1 \cup L^2 \cup \cdots$.

Note that if L is $\emptyset$ or $\{\lambda\}$ (here, L is pseudo-inversion-free), then both $L^m \cup L^n$ and L^* are pseudo-inversion-free as well. □

5 Conclusions

We have defined a biologically inspired operation called the pseudo-inversion. Given a string $w = uxv$, we have defined the pseudo-inversion of w to be a 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.

In practice, various genetic errors occur frequently and they are closely related to human diseases. We have considered one of such errors—pseudo-inversion. In view of the biological implications, we hope that our results are useful for potential future research including evolution analysis or gene sequence searching.

Acknowledgments We wish to thank the referees for the careful reading of the paper and many valuable suggestions. Cho, Han, Kang 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 (2014K 2A1A2048512) and Yonsei University Future-leading Research Initiative of 2014, 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

Cantone D, Cristofaro S, Faro S (2013) Efficient string-matching allowing for non-overlapping inversions. Theor Comput Sci 483:85–95

Chiniforooshan E, Daley M, Ibarra OH, Kari L, Seki S (2012) One-reversal counter machines and multihead automata: revisited. Theor Comput Sci 454:81–87

Cho DJ, Han YS, Kim H (2015a) Alignment with non-overlapping inversions and translocations on two strings. Theor Comput Sci 575:90–101

Cho DJ, Han YS, Kim H (2015b) Frequent pattern mining with non-overlapping inversions. In: Proceedings of the 9th language and automata theory and applications, vol 8977, pp 121–132

Daley M, Ibarra OH, Kari L (2003) Closure and decidability properties of some language classes with respect to ciliate bio-operations. Theor Comput Sci 306(1–3):19–38

Daley M, Kari L, McQuillan I (2004) Families of languages defined by ciliate bio-operations. Theor Comput Sci 320(1):51–69

Dassow J, Mitrana V, Salomaa A (2002) Operations and language generating devices suggested by the genome evolution. Theor Comput Sci 270(1):701–738

Deaton R, Garzon M, Murphy RC, Rose JA, Franceschetti DR, Stevens SE Jr (1996) Genetic search of reliable encodings for DNA-based computation. In: Proceedings of the 1st annual conference on genetic programming, pp 9–15

Garzon M, Deaton R, Nino LF, Stevens E, Wittner M (1998) Encoding genomes for DNA computing. In: Proceedings of the 3rd annual conference on genetic programming, pp 684–690

Ginsburg S (1975) Algebraic and automata-theoretic properties of formal languages. North-Holland Publishing Company, Amsterdam

Hopcroft J, Ullman J (1979) Introduction to automata theory, languages, and computation, 2nd edn. Addison-Wesley, Reading

Hussini S, Kari L, Konstantinidis S (2003) Coding properties of DNA languages. Theor Comput Sci 290(3):1557–1579

Ibarra OH (1978) Reversal bounded multicounter machines and their decision problems. J ACM 25:116–133

Ibarra OH (2014) On decidability and closure properties of language classes with respect to bio-operations. In: Proceedings of 20th DNA computing and molecular programming, pp 148–160

Jonoska N, Kari L, Mahalingam K (2008) Involution solid and join codes. Fundam Inform 86(1,2):127–142

Jonoska N, Mahalingam K, Chen J (2005) Involution codes: with application to DNA coded languages. Nat Comput 4(2):141–162

Jürgensen H, Konstantinidis S (1997) Codes. In: Rozenberg G, Salomaa A (eds) Handbook of formal languages, vol I, pp 511–607. Springer

Kari L, Losseva E, Konstantinidis S, Sosík P, Thierrin G (2006) A formal language analysis of DNA hairpin structures. Fundam Inform 71(4):453–475

Kari L, Mahalingam K (2006) DNA codes and their properties. In: Proceedings of the 12th international meeting on DNA computing, pp 127–142

Post EL (1946) A variant of a recursively unsolvable problem. Bull Am Math Soc 52(4):264–268

Salomaa A (1973) Formal languages. Academic Press, Waltham

Schöniger M, Waterman MS (1992) A local algorithm for DNA sequence alignment with inversions. Bul Math Biol 54(4):521–536

Shallit J (2009) A second course in formal languages and automata theory. Cambridge University Press, Cambridge

Wood D (1986) Theory of computation. Harper & Row, New York

Yokomori T, Kobayashi S (1995) 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