Duplications and Pseudo-Duplications

Da-Jung Cho[1], Yo-Sub Han[1](✉), Hwee Kim[1],
Alexandros Palioudakis[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,kimhwee,alex}@cs.yonsei.ac.kr

[2] School of Computing, Queen's University, Kingston, ON K7L 3N6, Canada
ksalomaa@cs.queensu.ca

Abstract. A duplication is basic phenomenon that occurs through molecular evolution on a biological sequence. A duplication on a string copies any substring of the string. We define k-pseudo-duplication of a string w that consists, roughly speaking, of all strings obtained from w by inserting after a substring u another substring obtained from u by at most k edit operations. We consider three variants of duplication operations, duplication, k-pseudo-duplication and reverse-duplication. First, we give the necessary and sufficient number of states that a nondeterministic finite automaton needs to recognize duplications on a string. Then, we show that regular languages and context-free languages are not closed under the duplication, k-pseudo-duplication and reverse-duplication operations. Furthermore, we show that the class of context-sensitive languages is closed under duplication, pseudo-duplication and reverse-duplication.

Keywords: Bio-inspired operations · State complexity · Finite automata · Context-free grammars · Context-sensitive grammars

1 Introduction

A DNA sequence undergoes various transformations from the primitive sequence through several biological operations such as insertion, deletion, substitution, inversion, translocation and duplication. This phenomena on DNA sequence lead many researchers to study theoretical properties of them [4,5,15,16,18]. Some researchers considered string matching problems with bio-inspired operations [3,5,25]. Moreover, one of the important problems in biology is sequence comparison and there are several tools for sequence searching such that BLAST and FASTA [22,23]. This leads some researchers to consider finite state methods that are useful for the sequence searching problems to improve search times in the face of exponentially increasing size of DNA sequences [2,13].

A duplicated segment of a chromosome occurs as a result of genetic recombination named *chromosomal crossover* [8] (see Fig. 1 for an example). Depending on the position of cutting somewhere within two chromosomes, the first segment of the first sequence and the last segment of the second sequence combine and

© Springer International Publishing Switzerland 2015

DOI: 10.1007/978-3-319-21819-9_11

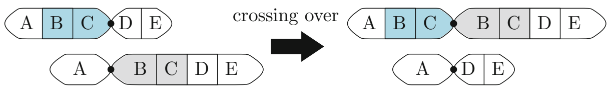

Fig. 1. An example of chromosomal crossover between two sequences. The last segment BCDE of the second sequence is attached to the first segment ABC of the first sequence. As a consequence of this crossing over, a subsequence BC occurs twice.

a new sequence with duplicated region may be obtained. Moreover, duplication of gene sequence may cause *replication slippage* during DNA replication and this phenomenon is closely linked with hereditary human diseases [12,28]. Kong et al. [19] indeed considered 736 complete chromosomes and showed that inverse segmental duplications are an important mechanism in the growth and evolution of genomes.

From a formal language theoretic framework, duplication leads a string $w_1w_2w_3$ to transform to the string $w_1w_2w_2w_3$, and this is one of the well-studied operations in both DNA computing and formal language theory. Many researchers have considered a variant of duplication operations. For other variants of duplication we refer the reader to the literature [9,10,17,20,26,30]. Searls [26] introduced linguistic formulations of rearrangement that occur in evolution such as duplication, inversion, transposition and deletion. Yokomori and Kobayashi [30] showed new representation for duplication using a set of basic operations, primitive language operation and mapping operations. Dassow et al. [9] defined an iterated duplication and considered closure properties of iterated version of duplication languages in the Chomsky hierarchy. Moreover, Dassow et al. [10] considered several operations arising from the genome evolution and noticed the result that a family of languages in the Chomsky hierarchy is closed under duplication. Leupold et al. [20] considered two types of languages defined by a string through iterated duplications and showed the formal language theoretical properties concerning two types of iterated duplications. Ito et al. [17] showed closure properties for bounded iterative duplication over alphabets of several sizes. Furthermore, some researchers considered duplication grammars [11,21].

For the DNA evolutionary analysis, an iterated version of duplication is regarded as multiple steps of evolutions, thus concerning the operation is natural to study their linguistic properties. We consider general duplication operation that occurs only once within a generation. Moreover, we introduce a new operation *k-pseudo-duplication* that copies any part of an input sequence allowing a certain amount of errors. We also consider *reverse-duplication*, and establish their properties. Note that Dassow et al. [10] presented similar closure properties for inversion, transposition and duplication: They considered a pre-specified set, and a operation works when a language contains a string in the pre-specified set. Recently, Cho et al. [4,6] showed similar results for the pseudo-inversion operation defined as an incomplete inversion, and estimated state complexity of inversion operations.

In Sect. 2, we briefly recall several notations. Then, we introduce the definitions of duplication and reverse-duplication and we define the k-pseudo-duplication in Sect. 3. Moreover, we give tight upper and lower bounds for the number of states that finite automaton needs to recognize the set of (pseudo-) duplications of a given string in Sect. 3.1. We establish some closure properties for three variants of duplication operation in Sect. 3.2. We mention a possible future direction and conclude the paper in Sect. 4.

2 Preliminaries

We briefly give definitions and notations used throughout the paper. The reader may refer to the textbooks [14,27,29] for more details on formal language theory.

Let Σ be a finite alphabet and Σ^* be the set of all strings over Σ. For any positive integer n we use $[n]$ to denote the set $\{1, 2, \ldots, n\}$. 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 and the length of w by $|w|$.

A *nondeterministic finite automaton* (NFA) is a five tuple $A = (Q, \Sigma, \delta, S, F)$, 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, $S \subseteq Q$ is the set of initial states and $F \subseteq Q$ is the set of final states. Our definition of NFAs allows the use of λ-transitions. It is well known [29] that an NFA with λ-transitions can be converted to an equivalent NFA without λ-transitions and having the same number of states. The automaton A is *deterministic* (DFA) if S is a singleton set and δ is a single-valued transition function from $Q \times \Sigma \to Q$. It is well known the NFAs and DFAs recognize the regular language [24,27].

A *context-free grammar* (CFG) is four tuple $G = (V, T, P, S)$, where V a set of non-terminal symbols, Σ is a set of final symbols, P is a set of production rules of the form $N \to \alpha$ for $N \in V$ and $\alpha \in (V \cup T)^*$, and S is the initial symbol. A language L generated by CFG is known as *context-free language.*

A grammar $G = (V, T, P, S)$ is *context-sensitive* (CSG) if P has a set of production rules of the form $\alpha N \beta \to \alpha\gamma\beta$ for $\alpha, \beta \in (V \cup \Sigma)^*$, $\gamma \in (V \cup T)^+$ and $N \in V$. A language L generated by CSG is said to be *context-sensitive language.*

The edit-distance between two strings x and y is the smallest number of basic operations that transform x to y [1,7]. We use three operations insertion, deletion and substitution: Given an alphabet Σ, an insertion operation that inserts $a \in \Sigma$ is denoted as $(\lambda \to a)$, a deletion operation that deletes $a \in \Sigma$ is denoted as $(a \to \lambda)$ and a substitution operation that substitutes b for a is denoted as $(a \to b)$. We denote the edit-distance between two string x and y by $d(x, y)$. The Hamming distance is the number of positions in which two strings of same length differ. Note that we use only a substitution operation for computing Hamming distance. We denote the Hamming distance between two strings x and y by $d_H(x, y)$.

For finding the nondeterministic state complexity of given languages, we use a technique called the *extended fooling set technique.* This technique gives us a lower bound on the size of any NFA recognizing a given language.

Proposition 1 (Extended Fooling set technique [27]). *Let $L \subseteq \Sigma^*$ be a regular language. Suppose that there exists a set $P = \{(x_i, w_i) \mid 1 \leq i \leq n\}$ of pairs such that*

(i) $x_i w_i \in L$ for $1 \leq i \leq n$,
(ii) if $i \neq j$, then $x_i w_j \notin L$ or $x_j w_i \notin L, 1 \leq i, j \leq n$.

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

Note that a set P satisfying the conditions (i) and (ii) of Proposition 1 is called *fooling set* for the language L.

3 Duplication and Pseudo-duplication

The duplication operation occurs in a bio sequence w when a substring of w is copied abnormally. We give the formal definition of the duplication operation.

Definition 1 (Searls [26]). Let $w \in \Sigma^*$ be a string over the alphabet Σ, the *duplication* of w is the set

$$\mathbb{D}(w) = \{x_1 x_2 x_2 x_3 \mid w = x_1 x_2 x_3, x_1, x_2, x_3 \in \Sigma^*.\}$$

Furthermore, Dassow et al. [9] considered the iterated duplication operation

$$\mathbb{D}^*(L) = \bigcup_{i \geq 0} \mathbb{D}^i(L).$$

Note that Dassow et al. [9] considered a duplication operation (defined by a duplication scheme) that is, roughly speaking, as in Definition 1 except that the repeated substring is restricted to belong to a pre-specified finite set. The language theoretic properties for the operation defined by a duplication scheme as in Dassow et al. [9] are significantly different from our results.

We define the *k-pseudo-duplication* that allows k errors on the resulting sequence. Note that during the process of DNA replication in practice some mutations such as insertion, deletion and substitution may occur.

Definition 2. Let $w \in \Sigma^*$ be a string and $k \geq 0$ a non-negative integer, we define the *k-pseudo-duplication* of w to be

$$\mathbb{PD}_k(w) = \{x_1 x_2 x_2' x_3 \mid w = x_1 x_2 x_3, x_1, x_2, x_3 \in \Sigma^*, d(x_2, x_2') \leq k\}.$$

When the value of k is known from the context, or is not important, we sometimes call the operation simply pseudo-duplication.

We also consider the *reverse-duplication* operation.

Definition 3 (Dassow et al. [9]). Let $w \in \Sigma^*$ be a string, we define the *reverse-duplication* of w to be

$$\mathbb{RD}(w) = \{x_1 x_2 x_2^R x_3 \mid w = x_1 x_2 x_3, x_1, x_2, x_3 \in \Sigma^*\}.$$

The *duplication* operation, the *pseudo-duplication* operation and the *reverse-duplication* operation are extended to languages in the following way:

(i) $\mathbb{D}(L) = \bigcup_{w \in L} \mathbb{D}(w)$,

(ii) $\mathbb{PD}_k(L) = \bigcup_{w \in L} \mathbb{PD}_k(w)$,

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

3.1 State Complexity of Duplication Operations

As we will see that the regular languages are not closed under the duplication, pseudo-duplication or reverse-duplication operation, here we consider the state complexity of the sets of (pseudo-) duplications of an individual string. For a string $w \in \Sigma^*$, it is obvious that the languages $\mathbb{D}(w)$, $\mathbb{PD}_k(w)$, and $\mathbb{RD}(w)$ are regular, since all are finite. Thus we focus on the nondeterministic state complexity of duplication, pseudo-duplication, and reverse-duplication operations. We give a matching upper and lower bound for the duplication operation of a string w.

Theorem 1. *Let $w \in \Sigma^*$ be a string of length n, for a positive integer n. Then, $\mathbb{D}(w)$ is recognized by a DFA with $2n+1$ states.*

Moreover, any NFA recognizing the language $\mathbb{D}(w)$ needs at least $2n+1$ states.

Proof. Let a string $w = w_1 \dots w_n$, where $w_i \in \Sigma$, for $1 \leq i \leq n$. We can also assume than $n \geq 2$, since, if $n = 1$ we can check easily that the claim is true. Then, we can construct the NFA $A_w = (Q, \Sigma, \delta, p_1, F)$ that recognizes the language $\mathbb{D}(w)$. We first define the set of states Q.

$$Q = \{p'_i \mid 1 \leq i \leq n\} \cup \{p_i \mid 1 \leq i \leq n+1\}$$

Now the transitions of the NFA A_w are as follows; $\delta(p_i, w_i) = p_{i+1}$ for all $1 \leq i \leq n$, $\delta(p'_i, w_{i+1}) = p'_{i+1}$ for all $1 \leq i \leq n-1$, and $\delta(p_i, w_j) = p'_j$ for all $1 \leq j < i \leq n$. The final states of A_w are the states p_{n+1} and p'_n. We give an example in Fig. 2, in the case where $w = x_1x_2x_3x_4$.

Now, we easily verify the lower bound for the state complexity of duplication. We note that $\mathbb{D}(w)$ is a finite language where the length of the longest string is $2n$. This implies that any NFA recognizing the language $\mathbb{D}(w)$ needs at least $2n+1$ states. □

With similar ideas we can find the state complexity bounds for the pseudo-duplication and reverse-duplication of a given word. We formalize these bounds in the next two theorems.

Theorem 2. *Let $w \in \Sigma^*$ be a string of length n, for a positive integer n. Then, the language $\mathbb{PD}_k(w)$, for $k \geq 1$, is recognized by an NFA with $k \cdot \frac{(n+1)(n+2)}{2} + n + 1$ states.*

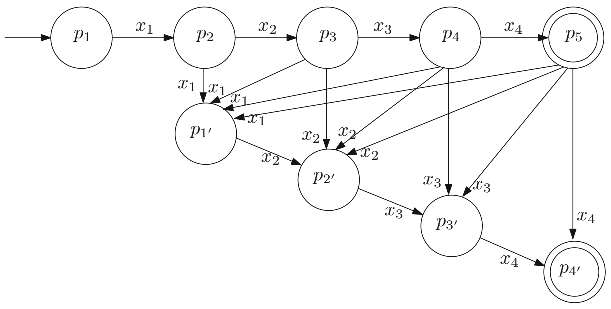

Fig. 2. An illustrative example of constructing an NFA recognizing $\mathbb{D}(w)$, where $w = x_1x_2x_3x_4$ for $x_1, x_2, x_3, x_4 \in \Sigma$.

Moreover, for every n_0 positive integer, there is a word w_0 over an alphabet Σ with $|w_0| = n_0$ and $|\Sigma| = |w_0|$, such that any NFA recognizing the language $\mathbb{PD}_k(w_0)$, needs at least $k \cdot \frac{(|w_0|+1)(|w_0|+2)}{2} + |w_0| + 1$ states.

Proof. Let a string $w = x_1 \dots x_n$, where $x_i \in \Sigma$, for $1 \leq i \leq n$. We can also assume than $n \geq 2$, since, if $n = 1$ we can check easily that the claim is true. Then, we can construct the NFA, with λ-transitions, $B_w = (Q, \Sigma, \delta, p_{(0,0,k)}, F)$ that recognizes the language $\mathbb{PD}_k(w)$. We first define the set of states Q.

$$Q = \{p_i \mid 0 \leq i \leq n\} \cup \{p_{(j,i,h)} \mid 0 \leq i \leq n,\, 0 \leq j \leq i, \text{ and } 1 \leq h \leq k\}$$

Now the transitions of the NFA B_w are as follows;

(i) $p_{(0,i,k)} \in \delta(p_{(0,i-1,k)}, x_i)$, for all $1 \leq i \leq n$,
(ii) $p_{(j,i,k)} \in \delta(p_{(j-1,i,k)}, x_j)$, for all $1 \leq i \leq n$, $1 \leq j \leq i$, and $1 \leq h \leq k$,
(iii) $p_{(j,i,k)} \in \delta(p_{(0,i,k)}, x_j)$, for $2 \leq i \leq n$, and $2 \leq j \leq i$,
(iv) $p_{(j,i,k-1)} \in \delta(p_{(0,i,k)}, *)$, for $* \in \Sigma \cup \{\lambda\}$, $1 \leq i \leq n$, and $1 \leq j \leq i$, if $k \geq 2$,
(if $k = 1$, then we have $p_j \in \delta(p_{(0,i,k)}, *)$, for $* \in \Sigma \cup \{\lambda\}$, $1 \leq i \leq n$, and $1 \leq j \leq i$)
(v) $p_{(j,i,h-1)} \in \delta(p_{(j,i,h)}, *)$, for $* \in \Sigma \cup \{\lambda\}$, $0 \leq i \leq n$, $0 \leq j \leq i$, and $2 \leq h \leq k$,
(vi) $p_{(j+1,i,h-1)} \in \delta(p_{(j,i,h)}, *)$, for $* \in \Sigma \cup \{\lambda\}$, $1 \leq i \leq n$, $0 \leq j < i$, and $2 \leq h \leq k$,
(vii) $p_j \in \delta(p_{(j,i,1)}, *)$, for $* \in \Sigma \cup \{\lambda\}$, $0 \leq i \leq n$, $0 \leq j \leq i$, and $2 \leq h \leq k$,
(viii) $p_{j+1} \in \delta(p_{(j,i,1)}, *)$, for $* \in \Sigma \cup \{\lambda\}$, $1 \leq i \leq n$, $0 \leq j < i$, and $2 \leq h \leq k$.

The final state of B_w is the states p_n. Additionally, it is not hard to transform the λ-NFA B_w to an equivalent NFA B'_w without λ transitions which has the same states as B_w. We give an example of the NFA B_w in Fig. 3, in the case where $w = x_1x_2x_3$ and $k = 2$.

Now we will give an informal explanation of how the transitions of the NFA B_w work. A state $p_{(j,i,h)}$ represents that the pseudo-duplication appears in the

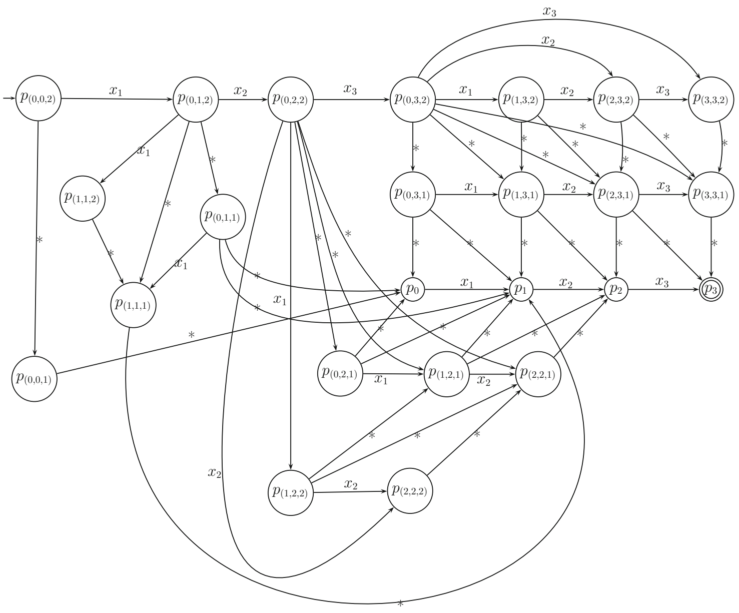

Fig. 3. An illustrative example of constructing an NFA recognizing $\mathbb{PD}_k(w)$, where $w = x_1x_2x_3$ for $x_1, x_2, x_3 \in \Sigma$ and $k = 2$.

i-th position of w, there are $i-j$ characters left from the inserted word, and that h errors remain. In more details from the definition of pseudo-duplication of w, we have all the words $x_1x_2x_2'x_3$ where $w = x_1x_2x_3$, for some $x_1, x_2, x_3 \in \Sigma^*$ and $d(x_2, x_2') \leq k$. The transitions that appear in (i) read the prefix of w x_1x_2. The transitions in (iii) and (iv) nondeterministically split the string x_1x_2 to the strings x_1 and x_2. The transitions in (ii) read the parts of the word x_2' that do not differ from the word x_2. The transitions in (v) and (vii) introduce an inserted character, in (vi) and (viii) substitute or delete a character.

Similar we can work for the correctness of the above construction. We can prove that every word $x \in \mathbb{PD}_k(w)$, it is also in $L(B_w)$ from the construction of B_w. Moreover, we can easily prove that for every word $x \in L(B_w)$ we have that w also belongs in $\mathbb{PD}_k(w)$.

The number of states of the NFA B_w are $n{+}1$ from the states p_i, $0 \leq i \leq n$, and there are $k \cdot (1{+}2{+}\ldots{+}(n{+}1))$, then it has $k\frac{(n+1)(n+2)}{2}{+}n{+}1$ states.

For the lower bound, let as have a word w_0 over an alphabet Σ_0 with $|w_0|$ letters. Let assume also that every letter of the alphabet Σ_0 appears in the word w_0. Then, let w_0 be the word $x_1x_2\ldots x_n$, for $|w_0| = n$. Now, for defining the extended fooling set, first let us have the following $n + 1$ pairs:

$$P' = \{(x_1x_2\ldots x_n(x_2)^k x_0x_1\ldots x_i, x_{i+1}\ldots x_n) \mid 0 \leq i \leq n \text{ and } x_0 = \lambda\}$$

Now, we want to transform any triple of numbers (j, i, h) to a pair of strings, for $0 \leq i \leq n$, $0 \leq j \leq i$, and $1 \leq h \leq k$. We associate the triple (j, i, h) to the pair $(x_1x_2 \dots x_i(x_n)^{k-h}x_1 \dots x_j,\ (x_n)^h x_{j+1} \dots x_n)$. The P'' be the set of pairs that we get by all triples (j, i, h) for $0 \le i \le n$, $0 \le j \le i$, and $1 \le h \le k$. Now, the fooling set will be the set of pairs $P = P' \cup P''$. We notice that for any two distinct pairs $(x, y), (x', y') \in P$ the string xy' or the string $x'y$ does not belong in $\mathbb{PD}_k(w_0)$, since the pseudo-duplication will appear in a different position or one of these strings will have more than k errors. □

Theorem 3. *Let $w \in \Sigma^*$ be a string of length n, for a positive integer n. Then, the language $\mathbb{RD}(w)$ is recognized by an NFA with $\frac{n^2+3n+2}{2}$ states.*

Moreover, for every $n \in \mathbb{N}$, there exists a string w of length n over an alphabet of size n such that any NFA recognizing the language $\mathbb{RD}(w)$ needs at least $\frac{n^2+3n+2}{2}$ states.

We omit the proof of Theorem 3 due to the page limitation, but Fig. 4 gives an insight on how we compute the nondeterministic state complexity for reverse-duplication of a string.

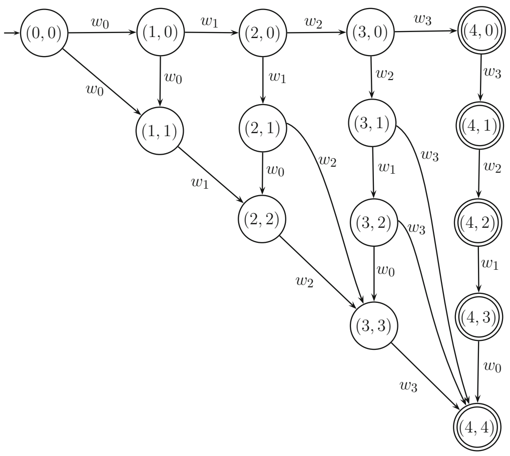

Fig. 4. An illustrative example of constructing an NFA recognizing $\mathbb{RD}(w)$, where $w = w_0w_1w_2w_3$ for $w_i \in \Sigma$, $0 \le i \le 3$.

3.2 Closure Properties of Duplication Operations

Next, we study the closure properties of duplication and pseudo-duplication for regular and context-free languages. We first show that regular and context-free languages are not closed under the duplication operation.

Theorem 4. *Regular languages are not closed under the duplication operation.*

Theorem 5. *Context-free languages are not closed under the duplication operation.*

Before we consider the closure properties of the pseudo-duplication operation, we mention that regular languages and context-free languages over unary alphabet are closed under the duplication, pseudo-duplication and reverse-duplication operations.

Proposition 2. *The unary regular languages are closed under the duplication, pseudo-duplication and reverse-duplication operations.*

Theorem 6. *Regular languages are not closed under the pseudo-duplication operation.*

Theorem 7. *Context-free languages are not closed under the pseudo-duplication operation.*

Theorems 6 and 7 show that regular and context-free languages are not closed under the pseudo-duplication operation. On the other hand, regular and context-free languages are closed under the pseudo-duplication operation when Σ is a unary alphabet.

Theorem 8. *Regular languages are not closed under the reverse-duplication operation.*

Theorem 9. *Context-free languages are not closed under the reverse-duplication operation.*

Now, we show that the class of context-sensitive languages is closed under the operations of duplication, pseudo-duplication and reverse-duplication. For clarity, we start with the following example.

Example 1. There is a context-sensitive grammar G recognizing the copy language $L = \{ww \mid w \in \Sigma^+\}$ over the alphabet $\Sigma = \{a, b\}$. We now give a grammar with the following rules:

$$\begin{array}{rlrl}
S & \to A_1A_2^SS^E \mid B_1B_2^SS^E \mid aa \mid bb & B_2B_1^E & \to B_1^EB_2 \\
S^E & \to A_1A_2S^E \mid B_1B_2S^E \mid A_1^EA_2 \mid B_1^EB_2 & A_2^SA_1^E & \to A_1^EA_2^S \\
A_2A_1 & \to A_1A_2 & B_2^SA_1^E & \to A_1^EB_2^S \\
B_2A_1 & \to A_1B_2 & A_2^SB_1^E & \to B_1^EA_2^S \\
A_2B_1 & \to B_1A_2 & B_2^SB_1^E & \to B_1^EB_2^S \\
B_2B_1 & \to B_1B_2 & A_1^EA_2^S & \to aa \\
A_2^SA_1 & \to A_1A_2^S & B_1^EA_2^S & \to ba \\
B_2^SA_1 & \to A_1B_2^S & A_1^EB_2^S & \to ab \\
A_2^SB_1 & \to B_1A_2^S & B_1^EB_2^S & \to bb \\
B_2^SB_1 & \to B_1B_2^S & \gamma A_2 & \to \gamma a,\ \gamma \in \{a, b\} \\
A_2A_1^E & \to A_1^EA_2 & \gamma B_2 & \to \gamma b,\ \gamma \in \{a, b\} \\
B_2A_1^E & \to A_1^EB_2 & A_1\gamma & \to a\gamma,\ \gamma \in \{a, b\} \\
A_2B_1^E & \to B_1^EA_2 & B_1\gamma & \to b\gamma,\ \gamma \in \{a, b\}
\end{array}$$

Where the symbols $S, S^E, A_1, A_1^S, A_1^E, B_1, B_1^S$, and B_1^E are the non-terminal symbols of the grammar.

Now for the correctness of the grammar, we notice that in the grammar the non-terminal symbol A corresponds to the final symbol a and the non-terminal symbol B corresponds to the final symbol b. Every time that the grammar produces a non-terminal symbol A for the first string, represented by A_1, A_1^S, or A_1^E, it also produces a non-terminal symbol A for the second string, A_2, A_2^S, or A_2^E, and vice versa. From the third rule of the first column up to the fifth rule of the second column, the grammar makes sure that the symbols corresponding to letters of the first string to be followed be symbols corresponding to letters of the second string. Notice that the grammar does not change the order of symbols which correspond to the same string. Finally, before any non-terminal transforms into a final symbol the grammar makes sure that all non-terminal symbols are in their correct positions. We do that by checking that the symbol corresponding to the last letter of the first string to be before the symbol corresponding to the first letter of the second string. This happens with the sixth to ninth rules of the second column and by these rules being the only rules who can start the transformation of non-terminals to final symbols.

The careful reader may notice that in Example 1, occasionally, we use rules of the form $AB \rightarrow BA$ which strictly speaking, they do not follow the definition of context-sensitive grammars. Such rules could be replaced with the rules $AB \rightarrow NB$, $NB \rightarrow NA$, and $NA \rightarrow BA$ by adding a new non-terminal symbol N. We allow rules of the above form in order to keep the number of rules low and increase readability.

Proposition 3. *Let G be a context-sensitive grammar. There is a context-sensitive grammar G' recognizing the copy language $L = \{ww \mid w \in L(G)\}$.*

Proposition 3 shows that given a context-sensitive grammar G we build a new context-sensitive grammar that recognizes the copy language $L = \{ww \mid w \in L(G)\}$. Based on the grammar of Proposition 3 we can construct a context-sensitive grammar that recognizes the language $\mathbb{D}(L(G))$.

Theorem 10. *The class of context-sensitive languages is closed under the duplication operation.*

With a similar technique with Theorem 10, we have the following theorem.

Theorem 11. *The class of context-sensitive languages is closed under the reverse-duplication and pseudo-duplication operations.*

4 Conclusions

We have considered biologically inspired operations called the duplication and reverse-duplication. The duplication operation on a string copies a substring and the reverse-duplication operation copies a substring in reverse. We have defined the pseudo-duplication operation as an extended variant of duplication where a substring can be repeated with some errors, and the number of errors is specified by an integer parameter. Then, we have estimated state complexity for these operations of a string and showed closure properties.

We have shown that the state complexity for duplication, pseudo-duplication and reverse-duplication of a string are $2n+1$, $k \cdot \frac{(n+1)(n+2)}{2}+n+1$ and $\frac{n^2+3n+2}{2}$ respectively, where n is length of a string and $k \geq 1$. Moreover, we have obtained the closure properties of the families of languages in the Chomsky hierarchy under three variants of duplication: Regular languages and context-free languages are not closed under duplication, pseudo-duplication and reverse-duplication whereas context-sensitive languages are closed under these operations.

A problem for further research is the complexity of determining whether or not a given language L has a string that belongs to the duplication, pseudo-duplication and reverse-duplication of another string in L. Moreover, it is also our future work to look for deterministic, nondeterministic state complexity of duplication and pseudo-duplication on unary regular languages.

Acknowledgements. We wish to thank the referees for valuable suggestions that improve proofs for several results.

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

References

1. Calude, C., Salomaa, K., Yu, S.: Additive distances and quasi-distances between words. Univ. Comput. Sci. **8**(2), 141–152 (2002)
2. Cameron, M., Williams, H.E., Cannane, A.: A deterministic finite automaton for faster protein hit detection in blast. Comput. Biol. **13**(4), 965–978 (2006)
3. Cantone, D., Cristofaro, S., Faro, S.: Efficient string-matching allowing for non-overlapping inversions. Theor. Comput. Sci. **483**, 85–95 (2013)
4. Cho, D.J., Han, Y.S., Kang, S.D., Kim, H., Ko, S.K., Salomaa, K.: Pseudo-inversion on formal languages. In: Ibarra, O.H., Kari, L., Kopecki, S. (eds.) UCNC 2014. LNCS, vol. 8553, pp. 93–104. Springer, Heidelberg (2014)
5. Cho, D.J., Han, Y.S., Kim, H.: Alignment with non-overlapping inversions and translocations on two strings. Theor. Comput. Sci. **575**, 90–101 (2015)
6. Cho, D.J., Han, Y.S., Ko, S.K., Salomaa, K.: State complexity of inversion operations. Theor. Comput. Sci. (in press)
7. Choffrut, C., Pighizzini, G.: Distances between languages and reflexivity of relations. Theor. Comput. Sci. **286**(1), 117–138 (2002)
8. Creighton, H.B., McClintock, B.: A correlation of cytological and genetical crossing-over in zea mays. Nat. Acad. Sci. U.S.A. **17**(8), 492–497 (1931)
9. Dassow, J., Mitrana, V., Paun, G.: On the regularity of duplication closure. Bull. EATCS **69**, 133–136 (1999)

10. Dassow, J., Mitrana, V., Salomaa, A.: Operations and language generating devices suggested by the genome evolution. Theor. Comput. Sci. **270**(1), 701–738 (2002)
11. Dassow, J., Mitrana, V., Salomaa, A.: Context-free evolutionary grammars and the structural language of nucleic acids. Biosystems **43**(3), 169–177 (1997)
12. Djian, P.: Evolution of simple repeats in dna and their relation to human disease. Cell **94**(2), 155–160 (1998)
13. Herrmannsfeldt, G.: A highly parallel finite state automaton processor for biological pattern matching. In: Stringology, pp. 58–72 (1998)
14. Hopcroft, J., Ullman, J.: Introduction to Automata Theory, Languages, and Computation, 2nd edn. Addison-Wesley, Reading (1979)
15. Hussini, S., Kari, L., Konstantinidis, S.: Coding properties of DNA languages. Theor. Comput. Sci. **290**(3), 1557–1579 (2003)
16. Ibarra, O.H.: On decidability and closure properties of language classes with respect to bio-operations. In: Murata, S., Kobayashi, S. (eds.) DNA 2014. LNCS, vol. 8727, pp. 148–160. Springer, Heidelberg (2014)
17. Ito, M., Leupold, P., Shikishima-Tsuji, K.: Closure of language classes under bounded duplication. In: Ibarra, O.H., Dang, Z. (eds.) DLT 2006. LNCS, vol. 4036, pp. 238–247. Springer, Heidelberg (2006)
18. 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)
19. Kong, S.G., Fan, W.L., Chen, H.D., Hsu, Z.T., Zhou, N., Zheng, B., Lee, H.C.: Inverse symmetry in complete genomes and whole-genome inverse duplication. PLoS One **4**(11), e7553 (2009)
20. Leupold, P., Mitrana, V., Sempere, J.M.: Formal languages arising from gene repeated duplication. In: Jonoska, N., Păun, G., Rozenberg, G. (eds.) Aspects of Molecular Computing. LNCS, vol. 2950, pp. 297–308. Springer, Heidelberg (2003)
21. Mitrana, V., Rozenberg, G.: Some properties of duplication grammars. Acta Cybernetica **14**(1), 165–177 (1999)
22. Mount, D.W.: Using the basic local alignment search tool (blast). Cold Spring Harbor Protoc. **2007**(7), pdb-top17 (2007)
23. Pearson, W.R., Lipman, D.J.: Improved tools for biological sequence comparison. Nat. Acad. Sci. **85**(8), 2444–2448 (1988)
24. Rozenberg, G., Salomaa, A. (eds.): Handbook of Formal Languages. Beyond Words, vol. 3. Springer-Verlag New York Inc., New York (1997)
25. Schöniger, M., Waterman, M.S.: A local algorithm for DNA sequence alignment with inversions. Bull. Math. Biol. **54**(4), 521–536 (1992)
26. Searls, D.B.: The computational linguistics of biological sequences. Artif. Intell. Mol. Biol. **2**, 47–120 (1993)
27. Shallit, J.: A Second Course in Formal Languages and Automata Theory. Cambridge University Press, Cambridge (2009)
28. Viguera, E., Canceill, D., Ehrlich, S.D.: Replication slippage involves DNA polymerase pausing and dissociation. EMBO J. **20**(10), 2587–2595 (2001)
29. Wood, D.: Theory of Computation. Harper & Row, New York (1987)
30. Yokomori, T., Kobayashi, S.: DNA evolutionary linguistics and RNA structure modeling: a computational approach. In: Neural and Biological Systems, pp. 38–45 (1995)