Duplications and Pseudo-Duplications*

Da-Jung Cho[1], Yo-Sub Han[†‡], Hwee Kim[1], Alexandros Palioudakis[1], Kai Salomaa[2]

[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, Ontario K7L 3N6, Canada*

Received 17 December 2004; In final form 1 April 2005

A duplication is one of the basic phenomena that occur in molecular evolution on a biological sequence. A duplication on a string is a process of copying a substring of the string—Given $w = x_1x_2x_3$, a duplication of w is $x_1x_2x_2x_3$. We define k-pseudo-duplication of a string w to be a set of all strings obtained from w by inserting after a substring u of w another substring u' such that the edit distance between u and u' is at most k. We consider duplication, k-pseudo-duplication and reverse-duplication as operations on formal languages. First, we demonstrate that regular languages and context-free languages are not closed under the duplication, k-pseudo-duplication and reverse-duplication operations. Second, we show that context-sensitive languages are closed under duplication, pseudo-duplication and reverse-duplication. Last, we present the necessary and sufficient number of states that a nondeterministic finite automaton needs to recognize all duplications of a string with respect to the string length.

Key words: bio-inspired operations, duplication, formal languages, closure properties

† Corresponding Author

‡ email: emmous@yonsei.ac.kr

* A preliminary version appeared in Proceedings of Unconventional Computation & Natural Computation 2015 (UCNC 2015), LNCS 9252, Springer-Verlag, 2015, pp. 157–168.

1 INTRODUCTION

A DNA sequence undergoes various transformations from a primitive sequence through several biological events such as insertion, deletion, substitution, inversion and duplication. These biological events are useful to find crucial genomic motifs for evolutionary innovation and are closely related to mutations such as frameshift mutation and Pelizaeus-Merzbacher disease [16]. It leads researchers to formalize these events as operations, and study the algebraic and code theoretic properties with respect to the operations [5, 3, 4, 12, 14, 15, 17].

A duplicated segment of a chromosome occurs by genetic recombination—*chromosomal crossover* [6] (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 a new sequence with duplicated region may be obtained.

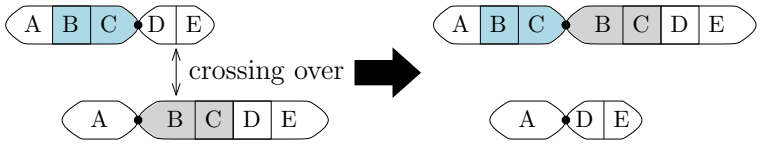

FIGURE 1: An example of chromosomal crossover between two sequences. The last segment BCDE of the second sequence is attached to the first segment ABC of the first sequence. As a consequence of this crossing over, a subsequence BC occurs twice.

In a formal language framework, we can think of duplication of a string $x = w_1w_2w_3$ as a process of generating a new string $w_1w_2w_2w_3$ from x. This operation is well-studied in both DNA computing and formal language theory. Several researchers considered duplication and several variants of duplication in the literature [7, 9, 8, 13, 17, 19, 20, 23]. Searls [20] introduced linguistic formulations of rearrangement that occur in evolution including duplication, inversion, transposition and deletion. Yokomori and Kobayashi [23] showed a new representation for duplication using a set of basic operations, primitive language operation and mapping operations. Dassow et al. [7] defined an iterative duplication and considered closure properties of iterative duplication on languages in the Chomsky hierarchy. Dassow et al. [8] considered several bio-inspired operations with a pre-specified finite set and showed that regular and context-free languages are closed under a duplication operation where the repeated substring is restricted to belong to a pre-specified finite set. Leupold

et al. [17] considered two kinds of duplication language—unbounded and bounded duplication languages. They showed that the membership, inclusion, equivalence and regularity problems are decidable for unbounded duplication languages. They also established that bounded duplication languages are context-free. Ito et al. [13] showed the closure properties for bounded iterative duplication over alphabets of several sizes. Dassow et al. [9] introduced evolutionary grammars based on the bio-inspired operations—deletion, inversion, transposition and duplication. They tackled classical problems in formal language theory such as generative power, closure properties and decidability in accordance with the operations. Mitrana and Rozenberg [19] investigated the generative power of context-free and context-sensitive duplication grammars. Both Abel et al. [1] and Hendricks et al. [10] formalized nanometer scale duplications in biology—shape replication through self-assembly model. They considered a problem of designing a self-assembly model that replicates a given shape of tiles into a specific or an infinite number of copies.

We consider the general duplication operation and introduce a new operation *k-pseudo-duplication* that copies any part of an input sequence allowing a certain amount of errors. We establish that regular languages and context-free languages are not closed under duplication, pseudo-duplication and reverse-duplication. Note that given a string there are only a finite number of strings generated from a string by these three operations and, thus, the resulting set can be recognized by a finite automaton. This leads us to examine the state complexity of the resulting sets from a string with respect to these three operations.

In Section 2, we briefly recall some basic notions. Then, we recall duplication and reverse-duplication, and introduce the k-pseudo-duplication operation, and establish closure properties for duplication, k-pseudo-duplication and reverse-duplication in Section 3. We give tight bounds for the number of states that a minimal finite automaton needs to recognize the set of (pseudo-) duplications of a given string in Section 4. We conclude the paper with possible future work in Section 5.

2 PRELIMINARIES

We briefly present some definitions and notations. The reader may refer to the textbooks [11, 22] 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 null 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 a set of initial states and $F \subseteq Q$ is a set of final states. Our definition of NFAs allows λ-transitions. It is well known 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 $Q \times \Sigma \to Q$.

A *context-free grammar* (CFG) is a four tuple $G = (V, T, P, S)$, where V is a set of nonterminal symbols, T is a set of terminal 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 is a *context-free language* if there is a context-free grammar generating L. A *context-sensitive grammar* (CSG) is a four tuple $G = (V, T, P, S)$, where V is a set of nonterminal symbols, T is a set of terminal symbols, P is a set of production rules of the form $\alpha N \beta \to \alpha\gamma\beta$ for $\alpha, \beta \in (V \cup T)^*$, $\gamma \in (V \cup T)^+$ and $N \in V$, and S is the initial symbol. A language L is a *context-sensitive language* if there is a context-sensitive grammar generating L.

The edit-distance between two strings x and y is the smallest number of basic edit operations that transform x to y [18, 21]. The basic edit operations typically used in the literature are 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 x and y by $d(x, y)$. The Hamming distance is the number of positions in which two strings of same length differ. We denote the Hamming distance between two strings x and y by $d_H(x, y)$.

For finding the nondeterministic state complexity of a regular language, we use the *fooling set technique* that gives a lower bound on the size of any NFA recognizing the language.

Proposition 1 (Fooling set technique [2])**.** *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.

We call a set P satisfying the conditions (i) and (ii) of Proposition 1 a *fooling set* for L.

3 DUPLICATION AND PSEUDO-DUPLICATION

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

Definition 1 (Dassow et al. [7])**.** *Let $w \in \Sigma^*$ be a string over the alphabet Σ. The* duplication $\mathbb{D}$ *of w is*

$$\mathbb{D}(w) = \{x_1x_2x_2x_3 \mid w = x_1x_2x_3 \text{ and } x_1, x_2, x_3 \in \Sigma^*\}.$$

The iterative duplication operation $\mathbb{D}^*$ of w is

$$\mathbb{D}^*(w) = \bigcup_{0 \leq i} \mathbb{D}^i(w).$$

Note that Dassow et al. [7] 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. Thus, the results from Dassow et al. [7] are different from our results.

In practice, it is common to have mutations such as insertion, deletion or substitution during the DNA replication process. This motives us to consider an incomplete duplication that allows a certain amount of errors—k-pseudo-duplication allows up to k errors.

Definition 2. *Let $w \in \Sigma^*$ be a string and $k \geq 0$ be a non-negative integer. We define* k-pseudo-duplication $\mathbb{PD}_k$ *of w to be*

$$\mathbb{PD}_k(w) = \{x_1x_2x_2'x_3 \mid w = x_1x_2x_3, d(x_2, x_2') \leq k \text{ and } x_1, x_2, x_3 \in \Sigma^*\}.$$

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

We also consider the *reverse-duplication* operation.

Definition 3 (Dassow et al. [7])**.** *Let $w \in \Sigma^*$ be a string. We define* reverse-duplication $\mathbb{RD}$ *of w to be*

$$\mathbb{RD}(w) = \{x_1x_2x_2^Rx_3 \mid w = x_1x_2x_3 \text{ and } x_1, x_2, x_3 \in \Sigma^*\}.$$

The *duplication* operation, the *pseudo-duplication* operation and the *reverse-duplication* operation are extended to languages as follows:

$$\mathbb{D}(L) = \bigcup_{w \in L} \mathbb{D}(w), \mathbb{PD}_k(L) = \bigcup_{w \in L} \mathbb{PD}_k(w) \text{ and } \mathbb{RD}(L) = \bigcup_{w \in L} \mathbb{RD}(w).$$

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

Theorem 1. *Regular and context-free languages are not closed under the duplication operation.*

Proof. Let $L = L(a^*b^*)$ over $\Sigma = \{a, b\}$. We show that

$$L(a^+b^+a^+b^+) \cap \mathbb{D}(L) = \{a^n b^i a^j b^m \mid n, m, i, j \geq 1, n \geq j \text{ and } m \geq i\}.$$

Let $L_1 = L(a^+b^+a^+b^+) \cap \mathbb{D}(L)$ and $L_2 = \{a^n b^i a^j b^m \mid n, m, i, j \geq 1, n \geq j \text{ and } m \geq i\}$. We prove the following two cases.

(i) $L_1 \subseteq L_2$. We prove that if a string w is in L_1, then $w \in L_2$. For a string $w \in \mathbb{D}(L)$, there are three substrings x_1, x_2, x_3 such that $w = x_1x_2x_2x_3 \in \mathbb{D}(L)$ and $u = x_1x_2x_3 \in L$. Let $u = a^n b^m$, for $n, m \geq 0$. Since $w \in L(a^+b^+a^+b^+)$, we know that $x_2 \in L(a^+b^+)$. Let $x_2 = a^j b^i$ for $j, i \geq 1$. It follows that $w = a^{n-j}a^j b^i a^j b^i b^{m-i}$. Note that a substring $a^j b^i$ occurs by duplication on u, thus, $j \leq n$ and $i \leq m$. Hence $w \in L_2$.

(ii) $L_2 \subseteq L_1$. We now prove that if a string w exists in L_2, then w also in L_1. Consider a string $w = a^n b^i a^j b^m \in L_2$ for $n, m \geq 1, n \geq j$ and $m \geq i$. Now $w = a^n b^i a^j b^m = a^{n-j}a^j b^i (a^j b^i) b^{m-i}$, in which the substring $a^j b^i$ is duplicated and, thus, $w \in L_1$.

It is easy to prove that $\{a^n b^i a^j b^m \mid n, m, i, j \geq 1, n \geq j \text{ and } m \geq i\}$ is not context-free using the context-free pumping lemma. Since $L(a^+b^+a^+b^+)$ is regular, we know that both regular and context-free languages are not closed under duplication. □

Before we consider the closure properties of the pseudo-duplication operation, we mention that unary regular (and thus context-free) languages are closed under duplication, pseudo-duplication and reverse-duplication; the proof is immediate from the operation definitions and the fact that inserting a unary string a^i is same as catenating a^i.

Theorem 2. *Regular languages are not closed under the pseudo-duplication operation.*

Proof. Let $L = L(a^*)$ over $\Sigma = \{a, b\}$. We show that

$$L(a^*b^*) \cap \mathbb{PD}_k(L) = \{a^i b^j \mid i \geq j\}.$$

Let $L_1 = L(a^*b^*) \cap \mathbb{PD}_k(L)$ and $L_2 = \{a^i b^j \mid i \geq j\}$.

(i) We first prove that $L_1 \subseteq L_2$. Consider a string $w \in L_1$. It implies that w exists in both $L(a^*b^*)$ and $\mathbb{PD}_k(L)$. Let $w = a^n b^m \in \mathbb{PD}_k(L)$ and $u = a^i \in L$ be the string such that $w \in \mathbb{PD}_k(u)$, where $i, n, m \geq 0$. Since $w \in \mathbb{PD}_k(L)$, $w = a^i a^j b^m$ for some non-negative integers j, m. Hence, it follows that $j+m \leq k$ and $n = i+j$, thus, $n \geq m$ and $w \in L_2$.

(ii) We prove that $L_2 \subseteq L_1$. We consider a string $w = a^n b^m \in L_2$, where $1 \leq m \leq n$. It implies that $w \in L(a^*b^*)$. We pick a string a^i from L and show that $a^n b^m \in \mathbb{PD}_k(a^i)$. The string $a^i(a^{i+k-m}b^m)$ can be generated from a^i with k-pseudo duplicated substring $a^{i+k-m}b^m$. Note that $d(a^i, a^{i+k-m}b^m) = k$. Hence, $w \in L(a^*b^*) \cap \mathbb{PD}_k(L)$.

It is straightforward to prove that $\{a^i b^j \mid i \geq j\}$ is not regular using the regular pumping lemma. Since $L(a^*b^*)$ is regular, $\mathbb{PD}_k(L)$ is not regular. □

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

Proof. We prove the statement by showing that there exists a context-free language L such that $\mathbb{PD}_k(L)$ is not context-free for any $k \geq 1$.

Choose $\Sigma = \{a, b, e, f, g\}$ and $L = \{fa^i fgb^i \mid i \geq 1\}$ and let $k \in \mathbb{N}$. We claim that

$$fa^+fe^kfa^+fgb^+ \cap \mathbb{PD}_k(L) = \{fa^ife^kfa^ifgb^i \mid i \geq 1\}. \tag{1}$$

First consider a string $s_i = fa^ife^kfa^ifgb^i$ belonging to the right side of Equation (1). Then choose a string $fa^ifgb^i = x_1x_2x_3 \in L$, where $x_1 = \lambda$, $x_2 = fa^if$ and $x_3 = gb^i$. Now writing $x_2' = e^kfa^if$ we note that $d(x_2, x_2') = k$ and $x_1x_2x_2'x_3 = s_i$. This means that $s_i \in \mathbb{PD}_k(L)$ (and s_i is in the left side of Equation (1)).

Conversely consider a string $z \in fa^+fe^kfa^+fgb^+ \cap \mathbb{PD}_k(L)$, that is, we can write

$$z = fa^pfe^kfa^qfgb^r = x_1x_2x_2'x_3, \quad p, q, r \geq 1, \tag{2}$$

where $x_1x_2x_3 = fa^ifgb^i \in L$ ($i \geq 1$) and $d(x_2, x_2') \leq k$. Since strings of L have no occurrences of symbol e, the only possibility is that x_2' must be obtained from x_2 by inserting and/or substituting a substring e^k of length k and performing no further modifications (since the insertion/substitution of k symbols e already uses the maximum number of allowable edit operations).

Since z contains four occurrences of the symbol f (and we know that no symbols f can be added to x_2'), we then conclude that in the decomposition $x_1x_2x_3 = fa^ifgb^i$ the string x_2 must contain at least the entire prefix fa^if. Furthermore, in the string x_2', the string e^k must be added at the beginning of x_2 since otherwise $x_1x_2x_2'x_3$ is not in $fa^+fe^kfa^+fgb^+$.

Since the string z contains only one occurrence of g, the last observation then means that g cannot be a part of x_2 because, in the edit operations transforming x_2 to x_2', we could not substitute g by e (and deletion of g, or substitution of g by some symbol other than e, would be one edit operation too many). Thus the only possibility is that, in the decomposition $x_1x_2x_3 = fa^ifgb^i$, we have $x_1 = \lambda$, $x_2 = fa^if$ and $x_3 = gb^i$, and the added string is $x_2' = e^kfa^if$. Then in order for Equation (2) to hold we must have $p = q = r = i$ and z is in the right side of Equation (1).

The language $\{fa^ife^kfa^ifgb^i \mid i \geq 1\}$ is a standard example of non-context-free languages (and this can be easily verified using the pumping lemma). Since context-free languages are closed under intersection with regular languages, by Equation (1), it follows that also $\mathbb{PD}_k(L)$ is non-context-free. □

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

Proof. Let $L = L(a^*b^*)$ over $\Sigma = \{a, b\}$. We have

$$L(a^+b^+a^+) \cap \mathbb{RD}(L) = \{a^nb^{2m}a^k \mid n, m, k \geq 1, n \geq k\}.$$

Let $L_1 = L(a^+b^+a^+) \cap \mathbb{RD}(L)$ and $L_2 = \{a^nb^{2m}a^k \mid n, m, k \geq 1, n \geq k\}$.

(i) We prove that $L_1 \subseteq L_2$. For a string $w \in L_1$, we have three substrings $x_1, x_2, x_3 \in \Sigma^*$ such that $w = x_1x_2{x_2}^Rx_3$, and we have $u = x_1x_2x_3 \in L$. Let $u = a^nb^m$ for $n, m \geq 0$. Then, we have $x_2 = a^kb^m$ for a positive integer $1 \leq k \leq n$, and $w = x_1x_2{x_2}^Rx_3 \in L_2$.

(ii) We show that $L_2 \subseteq L_1$. We consider a string $w = a^nb^{2m}a^k \in L_2$ for $n, m, k \geq 1$ and $n \geq k$. Then, we have a string $w = a^nb^m(b^ma^k)$, and it follows that a^kb^m is duplicated in reverse and $w \in L(a^+b^+a^+)$. Thus, $w \in L_1$ and we conclude that $L_1 = L_2$.

Next, the straightforward regular pumping lemma guarantees that the language $L(a^+b^+a^+) \cap \mathbb{RD}(L)$ is not regular, while $L(a^+b^+a^+)$ is regular. Hence $\mathbb{RD}(L)$ is not regular, which concludes the proof. □

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

Proof. Let $\Sigma = \{a, b, \$\}$ and $L = \{\$a^n b^n \$ \mid n \geq 0\}$. Let K be a set of all strings over Σ that have exactly 4 occurrences of the symbol \$. Now we have

$$\mathbb{RD}(L) \cap K = \{\$a^n b^n \$\$ b^n a^n \$ \mid n \geq 0\}.$$

Based on the context-free pumping lemma, it is straightforward to verify that $\mathbb{RD}(L) \cap K$ is not context-free and, hence, $\mathbb{RD}(L)$ is not context-free. □

Next, we show that context-sensitive languages are closed under duplication, pseudo-duplication and reverse-duplication. For clarity, we start with the following example.

Example 1. *There is a context-sensitive grammar $G = (V, T, P, S)$ recognizing the copy language*

$$L = \{ww \mid w \in \Sigma^+ \text{ and } \Sigma = \{a, b\}\},$$

where $V = \{S, S^E, A_1, A_1^S, A_1^E, A_2, A_2^S, A_2^E, B_1, B_1^S, B_1^E, B_2, B_2^S, B_2^E\}$ is a set of nonterminal symbols, $T = \{a, b\}$ is a set of terminal symbols, S is the initial symbol and P is a set of production rules defined as follows:

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

Now for the correctness of the grammar, we notice that the nonterminal symbol A corresponds to the terminal symbol a and the nonterminal symbol B corresponds to the terminal symbol b. When a string w is copied into ww, we call the first substring w of ww the first string and the duplicated substring w of ww the second string. Note that a nonterminal symbol with subscript 1 derives a substring of the first string and a nonterminal symbol with subscript 2 derives a substring of the second string. Whenever G produces a nonterminal symbol A for the first string, represented by A_1, A_1^S, or A_1^E, it also produces a nonterminal symbol A for the second string, A_2, A_2^S, or A_2^E, and vice versa for nonterminal symbol B. The nonterminal symbol with marker S and subscript 2 indicates that it derives the first character of the second string, and the nonterminal symbol with marker E and subscript 1 indicates that it derives the last character of the first string. From the third rule of the first column up to the fifth rule of the second column, the grammar moves a symbol corresponding to a character of the first string to the left and a symbol corresponding to a character of the second string to the right. We notice that G maintains the order of symbols that correspond to the same string. As a consequence, any corresponding nonterminal symbol with subscript 1 is on the left side and any corresponding nonterminal symbol with subscript 2 is on the right side holding the order of characters of original string. We check whether or not a nonterminal symbol with marker E and subscript 1 is followed by a nonterminal symbol with marker S and subscript 2. Note that we put marker E on a nonterminal symbol with subscript 1 that generates the last character of the first string and put marker S on a nonterminal symbol with subscript 2 that generates the first character of the second string. Then, a sequence of nonterminal symbols forms

$$X_1^1 X_1^2 \cdots X_1^{nE} X_2^{1S} X_2^2 \cdots X_2^n,$$

where $w = w_1 \cdots w_n$ and $X_i \in \{A_i, B_i \mid i \in \{1, 2\}\}$. Through the sixth to ninth rules of the second column, G makes sure that all nonterminal symbols are in their correct positions and terminates corresponding characters.

Notice that in Example 1, we occasionally use rules of the form $AB \to BA$, which, strictly speaking, do not follow the definition of context-sensitive grammars. Such rules could be replaced with the rules $AB \to NB$, $NB \to NA$, and $NA \to BA$ by adding a new nonterminal symbol N and, thus, the resulting grammar is context-sensitive. We use these rules $AB \to BA$ to keep the number of rules smaller and increase readability.

Proposition 2. *For a context-sensitive grammar G, there is a context-sensitive grammar G' recognizing the copy language $L = \{ww \mid w \in L(G)\}$.*

Proof. Let $\Sigma = \{a, b\}$. We assume that all nonterminal symbols of a given grammar G are disjoint with the nonterminal symbols of Example 1, otherwise, we just rename them. Here we consider $w \in L(G)$ while Example 1 considers $w \in \{a, b\}^+$. There is a difference between the proof of Example 1 and Proposition 2. We similarly construct G' with Example 1 but need additional rules. For every nonterminal symbol C of G, we have additional four copies C, C^S, C^E, and C^{SE}. The S marker shows that the nonterminal symbol produces the first character of the duplicated substring. Similarly, the E marker indicates the last character of the duplicated substring, and a nonterminal symbol with no marker notices that the symbol produce a character somewhere else. Let S^{SE} be a initial symbol of G'. Then, we construct G' as follows:

(i) For a rule $\alpha N \beta \to \alpha\gamma\beta$ of G, N is a nonterminal symbol and α, β, γ are strings derived by terminal or nonterminal symbols. Based on the rule, we add three rules

- $\alpha N^S \beta \to \alpha\gamma^S\beta$,
- $\alpha N^E \beta \to \alpha\gamma^E\beta$, and
- $\alpha N^{SE} \beta \to \alpha\gamma^{SE}\beta$,

where γ^S is a string with S marker on its first symbol, γ^E is a string with E marker on its last symbol, and γ^{SE} is the string with S marker on its first symbol and E marker on its last symbol. For instance, we add rules $S^S \to a^S S$, $S^E \to aS^E$ and $S^{SE} \to a^S S^E$ for a rule $S \to aS$.

(ii) Then, we replace every occurrence of a character a with the nonterminal symbols A_1A_2, every occurrence of a^S with $A_1A_2^S$, every occurrence of a^E with $A_1^E A_2$, and every occurrence of a^{SE} with $A_1^E A_2^S$. We similarly replace all the occurrences of b, b^S, b^E, and b^{SE}.

(iii) Additionally to the (i) and (ii) rules, we add all rules of Example 1 without two rules $S \to A_1A_2^S S^E \mid B_1B_2^S S^E \mid aa \mid bb$ and $S^E \to A_1A_2S^E \mid B_1B_2S^E \mid A_1^E A_2 \mid B_1^E B_2$.

We show that the grammar G' recognizes the copy language $L = \{ww \mid w \in L(G)\}$. If G generates a string $w = a_1 \dots a_n$, for $n \geq 1$, then the first two steps of (i) and (ii) produce the string $A_1^1 A_2^1 \dots A_1^n A_2^n$, where A_j^i is A_j if

and only if a_j is a and otherwise A_j^i is B_j, for $1 \leq i \leq n$, $1 \leq j \leq 2$. Note that Example 1 similarly produces the string ww. If G' produces a string w', then each character of w' derives from nonterminal symbols A or B. Since two symbols A and B are produced in the grammar with pairs A_1A_2 or B_1B_2, there are strings w_1 and w_2 such that $w' = w_1w_2$ and $|w_1| = |w_2|$. Based on the rules of Example 1, the ith character of w_1 is a if and only if the ith character of w_2 is a, for all $1 \leq i \leq |w_1|$. We conclude that $w' = w_1w_1$ and $w_1 \in L(G)$.

We similarly generate a grammar for the copy languages over non-binary alphabets. For each character c, we have nonterminal symbols C_1 and C_2. These new symbols follow similar rules as the symbols A_i, B_i work, for $i = 1, 2$. □

Proposition 2 shows that given a context-sensitive grammar G, we build a new context-sensitive grammar G' that recognizes the copy language $L = \{ww \mid w \in L(G)\}$. Based on the grammar in Proposition 2, we construct a context-sensitive grammar recognizing $\mathbb{D}(L(G))$.

Theorem 6. *Context-sensitive languages are closed under the duplication operation.*

Proof. Let L be a language generated from a context-sensitive grammar G. Based on the proof of Proposition 2, we similarly build a grammar

$$G_{\mathbb{D}} = (N, \Sigma, P, S^{S'E'})$$

for the copy language $L = \{ww \mid w \in L(G)\}$, where V is a set of nonterminal symbols, Σ is a set of terminal symbols, P is a set of production rules and $S^{S'E'}$ is the initial symbol. For any nonterminal symbol N of G, $G_{\mathbb{D}}$ has five more nonterminal symbols $N^{S'}, N^{E'}, N^{S'E'}, N^O$, and N^D. The $S'E'$ (similar for S' and E') marker works in a similarly way of the markers SE (S and E, respectively) of G' in the proof of Proposition 2. The duplicated substring derives from nonterminal symbols with marker S' to nonterminal symbols with marker E'. Note that a duplication may occur in the whole sequence of $w \in L(G)$ or in a substring w' of w. Thus, we use marker S', E' and $S'E'$ to clarify nonterminal symbols that generate a duplicated substring and others. However, a nonterminal symbol with maker S' does not guarantee that duplication occurs from the first character derived from the nonterminal symbol. Similarly, a nonterminal symbol with marker E' does not guarantee that duplication occurs to the last character derived from the nonterminal symbol. Thus we need two more markers O and D to clarify nonterminal symbols that

generate duplicated substring. A nonterminal symbol with marker O shows that the string generated by the nonterminal symbol is not duplicated, it follows the original rules of the grammar. Furthermore, a nonterminal symbol with marker D shows that the string generated by this nonterminal symbol is duplicated. Then, $G_{\mathbb{D}}$ has the following additional rules:

(i) For a rule $\alpha N\beta \to \alpha\gamma\beta$ of G and $\alpha, \beta, \gamma \in N \cup \Sigma$, $G_{\mathbb{D}}$ has three rules

- $\alpha N^{S'}\beta \to \alpha\gamma_1^{S'}\gamma_2^{D}\gamma_3^{D}\beta \mid \alpha\gamma_1^{O}\gamma_2^{S'}\gamma_3^{D}\beta \mid \alpha\gamma_1^{O}\gamma_2^{O}\gamma_3^{S'}\beta$
- $\alpha N^{E'}\beta \to \alpha\gamma_1^{E'}\gamma_2^{O}\gamma_3^{O}\beta \mid \alpha\gamma_1^{D}\gamma_2^{E'}\gamma_3^{O}\beta \mid \alpha\gamma_1^{D}\gamma_2^{D}\gamma_3^{E'}\beta$, and
- $\alpha N^{S'E'}\beta \to \alpha\gamma_1^{S'}\gamma_2^{D}\gamma_3^{E'}\beta \mid \alpha\gamma_1^{S'}\gamma_2^{E'}\gamma_3^{O}\beta \mid \alpha\gamma_1^{S'E'}\gamma_2^{O}\gamma_3^{O}\beta \mid \alpha\gamma_1^{O}\gamma_2^{S'}\gamma_3^{E'}\beta \mid \alpha\gamma_1^{O}\gamma_2^{S'E'}\gamma_3^{O}\beta \mid \alpha\gamma_1^{O}\gamma_2^{O}\gamma_3^{S'E'}\beta \mid \alpha\gamma_1^{O}\gamma_2^{O}\gamma_3^{O}\beta$.

For $1 \leq i \leq 3$, $\gamma_i^S \in N \cup \Sigma$ has the marker S on its first symbol and the marker D in other symbols. γ_i^E has the marker E on its last symbol and maker D on the previous symbols of the symbol with E. γ_i^{SE} has the marker S on its first symbol, the marker E on its last symbol and the marker D on the symbols in between. It leads us to identify where duplication occurs. Note that marker S and E indicate the starting and end position of duplicate substring and the D marker indicates the middle part of duplicated substring. For any $\gamma = \gamma_1\gamma_2\gamma_3 \in N \cup \Sigma$, $G_{\mathbb{D}}$ continues applying rules. Moreover, the markers O and D pass to all symbols in γ. For instance, we have the rules $\alpha N^{D}\beta \to \alpha\gamma^{D}\beta$ and $\alpha N^{O}\beta \to \alpha\gamma^{O}\beta$.

(ii) Then, we replace

- every occurrence of a^D with nonterminal symbols A_1A_2,
- every occurrence of a^O with the final symbol a,
- every occurrence of $a^{S'}$ with $A_1A_2^S$,
- every occurrence of $a^{E'}$ with $A_1^EA_2$ and
- every occurrence of $a^{S'E'}$ with $A_1^EA_2^S$.

We replace for all occurrences of $b^D, b^O, b^{S'}, b^{E'}$ and $b^{S'E'}$ in a similar way.

(iii) In addition, $G_{\mathbb{D}}$ contains all rules in Example 1 without two rules $S \to A_1A_2^SS^E \mid B_1B_2^SS^E \mid aa \mid bb$ and $S^E \to A_1A_2S^E \mid B_1B_2S^E \mid A_1^EA_2 \mid B_1^EB_2$.

We now prove the correctness of the construction for $G_{\mathbb{D}}$ generating

$$\mathbb{D}(L(G)) = \{w_1w_2w_2w_3 \mid w_1w_2w_3 \in L(G), \text{ where } w_1, w_2, w_3 \in \Sigma^*\}.$$

If G generates a string $w = w_1w_2w_3$, then the rules from (i) nondeterministically split w into substrings w_1, w_2, and w_3 by marking O, S', E' and D. Note that a symbol in $N \cup \Sigma$ with marker O generates a substring w_1 or w_3, and a symbol in $N \cup \Sigma$ with marker D generates a duplicated substring somewhere in w_2. A symbol in $N \cup \Sigma$ with marker S' shows information where the first character of w_2 appears, and a symbol in $N \cup \Sigma$ with marker E' indicates where the last character of w_2. Similar to Proposition 2, a duplicated substring w_2 derives from the rules in (ii) and (iii).

For the other direction, if $G_{\mathbb{D}}$ produces a string w, then each character of w derives from nonterminal symbols A or B with markers O and D. We notice that symbols with marker O produce a non-duplicated substring and symbols with marker D produce a duplicated substring. Moreover, nonterminal symbols with markers D should appear all together at only one region. Hence, the duplicated substring is unique and the resulting string belongs in $\mathbb{D}(L(G))$. □

With a similar technique of the proof in Theorem 6, we show that context-sensitive languages are closed under pseudo-duplication and reverse-duplication.

Theorem 7. *Context-sensitive languages are closed under the pseudo-duplication operation.*

Proof. We use the grammar in Theorem 6 with $k + 1$ copies of each nonterminal symbol. For a grammar G generating L, we construct a new grammar $G_{\mathbb{PD}_k}$ generating $L(G_{\mathbb{PD}_k})$. Let N be a nonterminal symbol of G and (N, i) be a set of nonterminal symbols in $G_{\mathbb{PD}_k}$, where $0 \leq i \leq k$. For a production rule $\alpha N\beta \to \alpha\gamma\beta$ of G and $0 \leq h \leq k$, we add the following rule

$$\alpha(N, h)\beta \to \alpha((\Sigma \cup N)_1, i_1)((\Sigma \cup N)_2, i_2) \cdots ((\Sigma \cup N)_t, i_t)\beta,$$

where $\gamma \in (\Sigma \cup N)^+, 1 \leq t$ and $i_1 + i_1 + \cdots i_t = h$. Note that $i_1, i_2, \ldots, i_t$ denote all possible errors that $L(G_{\mathbb{PD}_k})$ allows. For instance, G has the rule $\alpha N\beta \to \alpha N_1N_2N_3\beta$, where $N_1, N_2, N_3 \in (\Sigma \cup N)$. Then $G_{\mathbb{PD}_2}$ has the following rules:

$$\begin{aligned}\alpha(N, 2)\beta \quad \to \quad & \alpha(N_1, 2)(N_2, 0)(N_3, 0)\beta \mid \alpha(N_1, 1)(N_2, 1)(N_3, 0)\beta \mid \\ & \alpha(N_1, 1)(N_2, 0)(N_3, 1)\beta \mid \alpha(N_1, 0)(N_2, 2)(N_3, 0)\beta \mid \\ & \alpha(N_1, 0)(N_2, 1)(N_3, 1)\beta \mid \alpha(N_1, 0)(N_2, 0)(N_3, 2)\beta.\end{aligned}$$

We continue adding new rules for each $((\Sigma \cup N), i)$. For a rule $\alpha(N, h)\beta \to \alpha((\Sigma \cup N)_1, i_1)((\Sigma \cup N)_2, i_2) \cdots ((\Sigma \cup N)_t, i_t)\beta$, we add similar rules with the rule (i) of the proof in Theorem 6 using markers $S'E', S', E', O$ and D. Then, we replace

- every occurrence of $(a, i)^D$ with nonterminal symbol $(A_1, i)(A_2, i)$,
- every occurrence of $(a, i)^O$ with nonterminal symbol a,
- every occurrence of $(a, i)^{S'}$ with nonterminal symbol $(A_1, i)(A_2, i)^S$,
- every occurrence of $(a, i)^{E'}$ with nonterminal symbol $(A_1, i)^E(A_2, i)$,
- every occurrence of $(a, i)^{S'E'}$ with nonterminal symbol $(A_1, i)^E(A_2, i)^S$.

We replace for all occurrences of $(b, i)^D, (b, i)^O, (b, i)^{S'}, (b, i)^{E'}$ and $(b, i)^{S'E'}$ in a similar way. Moreover, $G_{\mathbb{PD}_k}$ contains similar rules in Example 1 to ensure that all nonterminal symbols with subscript 1 is followed by nonterminal symbols with subscript 2 and $(N_1, i)^E$ is followed by $(N_2, i)^S$. Finally, for (A_2, i), we add a rule $(A_2, i) \to aX_1 \cdots X_i$, where $X_i \in \{a, b\}$. We add a similar rule for (B_2, i).

For the correctness of the construction for $G_{\mathbb{PD}_k}$,

(i) we prove that if $w \in \mathbb{PD}_k(L(G))$ then $w \in L(G_{\mathbb{PD}_k})$. Let $u = w_1w_2w_3 \in L(G)$ and $w_1w_2w_2'w_3 \in \mathbb{PD}_k(u)$, where $d(w_2, w_2') \leq k$. We follow the proof in Theorem 6, but only one difference is that $G_{\mathbb{PD}_k}$ contains symbols $((\Sigma \cup N), i)$ to consider errors up to k. We know that symbols with markers O, S', E' and D nondeterministically split w into w_1, w_2, w_2' and w_3. Then, the new rule $(A_2, i) \to aX_1 \cdots X_i$ (vice versa for B_2) generates k-pseudo duplicated substring w_2'.

(ii) We prove that if $w \in L(G_{\mathbb{PD}_k})$, then $w \in \mathbb{PD}(L(G))$. Each character of w derives from symbols $((\Sigma \cup N), i)$ with markers O, D, S', E' and $S'E'$. Note that symbols with marker O produce a non-duplicated substring and symbols with marker D produce a duplicated substring. Furthermore, (A_2, i) and (B_2, i) generate a duplicated substring with errors up to k. Hence $w \in \mathbb{PD}(L(G))$.

□

Theorem 8. *Context-sensitive languages are closed under the reverse-duplication operation.*

Proof. We construct a grammar $G_{\mathbb{RD}}$ following the grammar of Theorem 6 with small changes. We have eight more nonterminal symbols $A_2^R, B_2^R, A_2^{SR}, B_2^{SR}, A_2^T, B_2^T, A_2^{ST}$ and B_2^{ST}. We follow the proof of Theorem 6 until $G_{\mathbb{RD}}$ derives

$$x^1x^2\cdots x^iX_1^{i+1}X_1^{i+2}\cdots X_1^{j}{}^{E}X_2^{i+1}{}^{S}X_2^{i+2}\cdots X_2^jx^{j+1}\cdots x^n,$$

where $w = w_1\cdots w_n \in L(G), x \in \{a,b\}$ and $X \in \{A,B\}$. Then, we replace $X_2{}^S$ with $X_2{}^{SR}$. Marker SR indicates where a reversed substring derives from. Marker R replace all occurrences of A_2, B_2, A_2^S, and B_2^S in reverse order. Note that it is not difficult to reverse all the appearance of the symbols with marker R. Then, we transform all symbols with marker R to the same symbols with marker T. A symbol with marker T produce corresponding nonterminal symbols a and b in the same way of the proof in Example 1.

We prove the correctness of the construction for $G_{\mathbb{RD}}$.

(i) We prove that if $w \in \mathbb{RD}(L(G))$ then $w \in L(G_{\mathbb{RD}})$. Let $u = w_1w_2w_3 \in L(G)$ and $w = w_1w_2w_2^Rw_3 \in \mathbb{RD}(L(G))$. Based on the proof of correctness check in Theorem 6, we know that if $w_1w_2w_2w_3 \in \mathbb{D}(L(G))$, then $w_1w_2w_2w_3 \in L(G_{\mathbb{D}})$. Since we follow the construction in Theorem 6, $G_{\mathbb{RD}}$ generates $w_1w_2w_2w_3$. Then, by one difference of nonterminal symbols with marker SR and R, duplicated substring w_2 is reversed. Hence, $G_{\mathbb{RD}}$ generates $w_1w_2w_2{}^Rw_3$.

(ii) We now prove that if $w \in L(G_{\mathbb{RD}})$, then $w \in \mathbb{RD}(L(G))$. Since $G_{\mathbb{RD}}$ produces w, it implies that each character of w derives from nonterminal symbols A of B with makers O and D, and we know that symbols with marker O produce a non-duplicated substring and symbols with marker D produce a duplicated substring. Furthermore, symbols with markers SR and R rearrange the order of symbols in reverse. Notice that symbols with markers SR and R appear on symbols that generate duplicated substring. Then, $w \in \mathbb{RD}(L(G))$.

□

4 NONDETERMINISTIC STATE COMPLEXITY

We consider the state complexity of the languages consisting of all (pseudo-) duplications of an individual string. Above we have seen that regular languages are not closed under duplication, pseudo-duplication or reverse-duplication. However, for a string $w \in \Sigma^*$, the languages $\mathbb{D}(w)$, $\mathbb{PD}_k(w)$ and $\mathbb{RD}(w)$ are

regular since they are all finite, and then a relevant question is how large a (nondeterministic) finite automaton is needed to recognize these languages. We establish a tight bound for the nondeterministic state complexity of the duplication operations on w as a function of the length of w.

Theorem 9. *Let $w \in \Sigma^*$ be a string of length $n \geq 1$. Then, $\mathbb{D}(w)$ is recognized by an NFA with $2n+1$ states. Moreover, any NFA recognizing $\mathbb{D}(w)$ needs at least $2n+1$ states.*

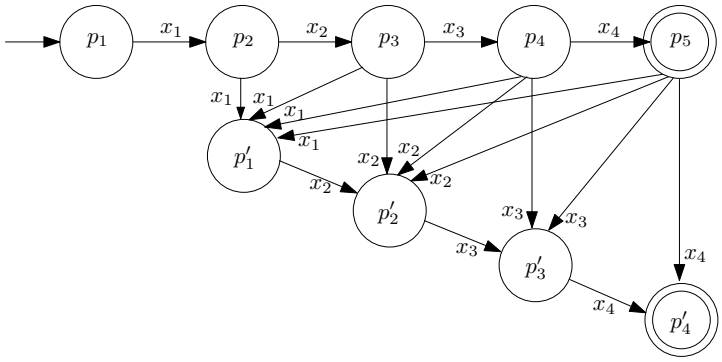

FIGURE 2: An example of constructing an NFA recognizing $\mathbb{D}(x_1x_2x_3x_4)$.

Proof. Let a string $w = w_1 \dots w_n$, where $w_i \in \Sigma$, for $1 \leq i \leq n$. We assume that $n \geq 2$. Otherwise, if $n = 1$, the claim holds immediately. We construct an NFA $A_w = (Q, \Sigma, \delta, p_1, F)$ that recognizes $\mathbb{D}(w)$ as follows:

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

The transition function is

(i) $\delta(p_i, w_i) = p_{i+1}$ for all $1 \leq i \leq n$,

(ii) $\delta(p'_i, w_{i+1}) = p'_{i+1}$ for all $1 \leq i \leq n-1$,

(iii) $\delta(p_i, w_j) = p'_j$ for all $1 \leq j < i \leq n$.

The final states of A_w are p_{n+1} and p'_n. Fig. 2 shows an example NFA for $\mathbb{D}(x_1x_2x_3x_4)$.

Now, we easily verify the lower bound for the state complexity of duplication. Note that $\mathbb{D}(w)$ is a finite language where the length of the longest string is $2n$. Therefore, any NFA recognizing $\mathbb{D}(w)$ 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 string. We formalize these bounds in the next two theorems.

Theorem 10. *Let $w \in \Sigma^*$ be a string of length 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. Moreover, for every positive integer n_0, there is a string w_0 over an alphabet Σ with $|w_0| = n_0$ and $|\Sigma| = |w_0|$ such that any NFA recognizing $\mathbb{PD}_k(w_0)$ needs at least $k \cdot \frac{(|w_0|+1)(|w_0|+2)}{2} + |w_0| + 1$ states.*

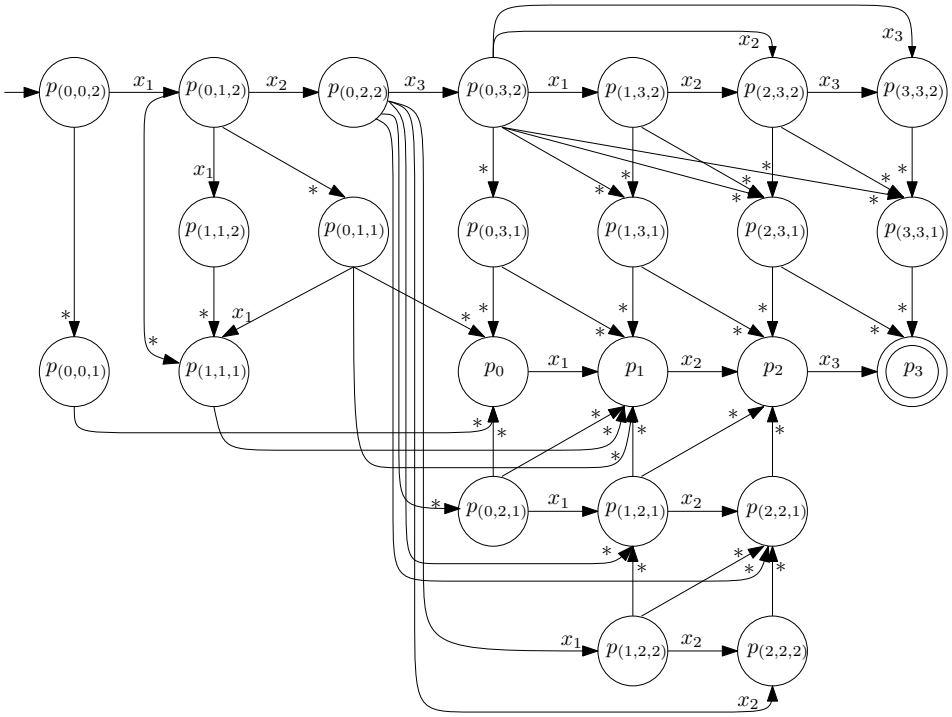

FIGURE 3: An example of constructing an NFA recognizing $\mathbb{PD}_2(x_1x_2x_3)$.

Proof. Let a string $w = x_1 \dots x_n$, where $x_i \in \Sigma$ for $1 \leq i \leq n$. We assume that $n \geq 2$. Otherwise, if $n = 1$, the claim holds immediately. we construct an NFA $B_w = (Q, \Sigma, \delta, p_{(0,0,k)}, p_n)$ recognizing $\mathbb{PD}_k(w)$. We first define the set Q of states

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

Now the transition function of B_w is defined 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 $2 \leq h \leq k$.

Now we explain how the transitions of B_w work. The transitions that appear in (i) read the prefix x_1x_2 of w. The transitions in (iii) and (iv) nondeterministically split the string x_1x_2 into x_1 and x_2. The transitions in (ii) read the parts of the string x_2' that do not differ from the string x_2. The transitions in (v) and (vii) introduce an inserted character and the transitions in (vi) and (viii) substitute or delete a character. As a consequence of the transitions, B_w recognizes all strings in $\mathbb{PD}_k(w)$. Fig. 3 shows an example NFA for $\mathbb{PD}_k(x_1x_2x_3)$. Note that a state $p_{(j,i,h)}$ represents that a pseudo-duplication appears in the i-th position of w, and there are $i-j$ characters left from the inserted string with remaining h errors. It implies that, for a string w of length n, the number of states of $\mathbb{PD}_k(w)$ is

$$k \cdot \frac{(n+1)(n+2)}{2} + n + 1.$$

Note that $\frac{(n+1)(n+2)}{2}$ states are from $p_{(j,i,h)}$ and $n+1$ states are from p_i, where $0 \leq i \leq n, 0 \leq j \leq i,$ and $1 \leq h \leq k$.

For the lower bound, let $w_0 = x_1x_2\cdots x_n$ over $\Sigma_0 = \{x_1, x_2, \ldots, x_n\}$ such that $|w_0| = n$. Now, for the fooling set, we have $n+1$ pairs:

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

Now we transform any triple (j, i, h) of numbers 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 $(x_1x_2\cdots x_i(x_n)^{k-h}x_1\cdots x_j, (x_n)^h x_{j+1}\cdots x_n)$. Let P'' be the set of pairs that we obtain by all triples (j, i, h) for $0 \leq i \leq n$, $0 \leq j \leq i$, and $1 \leq h \leq k$. Now, the fooling set is $P = P' \cup P''$. We notice that for any two distinct pairs $(x, y), (x', y') \in P$, at least one of the two strings xy' and $x'y$ does not belong in $\mathbb{PD}_k(w_0)$ since a pseudo-duplication appears in a different position or one of the two strings should have more than k errors. □

Theorem 11. *Let $w \in \Sigma^*$ be a string of length 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 and $|\Sigma| = |w|$ such that any NFA recognizing $\mathbb{RD}(w)$ needs at least $\frac{n^2+3n+2}{2}$ states.*

Proof. Let a string $w = w_0 \ldots w_{n-1}$ for $w_i \in \Sigma$ and $1 \leq i \leq n$. We assume that $n \geq 2$. Then, we construct an NFA $C_w = (Q, \Sigma, \delta, (0,0), F)$ that recognizes the language $\mathbb{RD}(w)$. We define the set Q of states

$$Q = \{(i, j) \mid 0 \leq j \leq i \leq n\}$$

and the set F of final states

$$F = \{(n, j) \mid 0 \leq j \leq n\}.$$

The transition function δ is defined as follows:

(i) $(i+1, 0) \in \delta((i, 0), w_i)$, for $0 \leq i < n$,

(ii) $(i+1, i+1) \in \delta((i, i), w_i)$, for $0 \leq i < n$,

(iii) $(i, j+1) \in \delta((i, j), w_{i-j-1})$, for $0 \leq j < i \leq n$,

(iv) $(i+1, i+1) \in \delta((i, j), w_i)$, for $0 < j < i < n$.

It is not difficult to show that $L(C_w) = \mathbb{RD}(w)$. The transitions in (i) simulate prefix $w_0 \cdots w_{i-1}$ of w. For any state $(i+1, 0)$, the transitions in (iii) allow C_w to duplicate substring of w in reverse. Note that for any state (i, j), where $0 \leq j < i \leq n$, the first element i shows that a reverse-duplication

occurs after C_w reads w_{i-1} of prefix $w_0 \cdots w_{i-1}$ of w. The second element j shows that C_2 duplicates $w_{i-j-1} \cdots w_{i-1}$ in reverse. The transitions in (ii) and (iv) allow C_w to continue the computation for $w_i \cdots w_{n-1}$ of w. Fig. 4 shows an example NFA $\mathbb{RD}(w_0w_1w_2w_3)$.

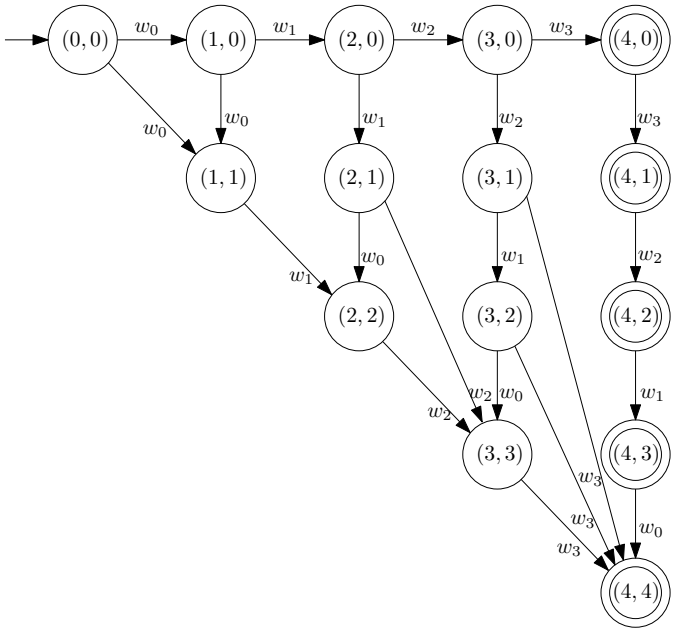

FIGURE 4: An example of constructing an NFA recognizing $\mathbb{RD}(w_0w_1w_2w_3)$.

The number of states of C_w is $\frac{n^2+3n+2}{2}$. Now we prove that for every $n \in \mathbb{N}$ there is a string w of length n such that any NFA recognizing $\mathbb{RD}(w)$ needs at least $\frac{n^2+3n+2}{2}$ states. For $0 \leq i \leq n$, let us take a set P of pairs of string as follows:

$$P_i = \{(\lambda, w_0 \dots w_i w_i \dots w_0 w_{i+1} \dots w_n), (w_0, w_1 \dots w_i w_i \dots w_0 w_{i+1} \dots w_n), \dots,$$

$$(w_0 \dots w_i w_i \dots w_0 w_{i+1} \dots w_n, \lambda)\}.$$

Each set P_i has $i+1$ pairs for all $0 \leq i \leq n$. Now let $P = \bigcup_{i=0}^{n} P_i$. We prove that P is an fooling set for the language $\mathbb{RD}(w)$. Let (x, y) and (x', y') be two pairs from P. If both pairs (x, y) and (x', y') belong in the same set P_i, then at least one of the pairs (x, y') or (x', y) is not in $\mathbb{RD}(w)$ since it has an

incorrect number of characters according to a string in $\mathbb{RD}(w)$. If $(x, y) \in P_i$ and $(x', y') \in P_{i'}$ for $i \neq i'$, then both pairs (x, y') and (x', y) are not in $\mathbb{RD}(w)$ since reverse-duplication occurs in a different position. □

5 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.

Researchers considered duplication occurrences in multiple steps of evolution and formalized it as iterative versions of duplication. We have focused on the phenomenon where duplication occurs on a gene sequence through one generation. 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 algorithmic complexity of determining whether or not a given (regular) language L has a string that belongs to the duplication, pseudo-duplication and reverse-duplication of another string in L. Another topic for future work is to determine bounds for the (non)deterministic state complexity of duplication and pseudo-duplication on finite languages. Also since unary regular languages are closed under the duplication operations, we may study the state complexity in the unary case.

ACKNOWLEDGMENTS

Cho and Han were supported by the Basic Science Research Program through NRF funded by MEST (2015R1D1A1A01060097), the International Cooperation Program managed by NRF of Korea (2014K2A1A2048512) and Yonsei University Future-leading Research Initiative of 2015. 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] Zachary Abel, Nadia Benbernou, Mirela Damian, Erik D Demaine, Martin L Demaine, Robin Flatland, Scott D Kominers, and Robert Schwelle. (2010). Shape replication through self-assembly and RNase enzymes. In *Proceedings of the 21st annual ACM-SIAM symposium on discrete algorithms*, pages 1045–1064.

[2] Jean-Camille Birget. (1993). Partial orders on words, minimal elements of regular languages and state complexity. *Theoretical Computer Science*, 119(2):267–291.

[3] Da-Jung Cho, Yo-Sub Han, Shin-Dong Kang, Hwee Kim, Sang-Ki Ko, and Kai Salomaa. Pseudo-inversion: closure properties and decidability. *Natural Computing*. To appear.

[4] Da-Jung Cho, Yo-Sub Han, and Hwee Kim. (2015). Alignment with non-overlapping inversions and translocations on two strings. *Theoretical Computer Science*, 575:90–101.

[5] Da-Jung Cho, Yo-Sub Han, Timothy Ng, and Kai Salomaa. (2016). Pseudoknot-generating operation. In *42nd International Conference on Current Trends in Theory and Practice of Computer Science*, pages 241–252.

[6] Harriet B Creighton and Barbara McClintock. (1931). A correlation of cytological and genetical crossing-over in zea mays. *Proceedings of the National Academy of Sciences of the United States of America*, 17(8):492–497.

[7] Jurgen Dassow, Victor Mitrana, and Gheorghe Paun. (1999). On the regularity of duplication closure. *Bulletin of the EATCS*, 69:133–136.

[8] Jürgen Dassow, Victor Mitrana, and Arto Salomaa. (2002). Operations and language generating devices suggested by the genome evolution. *Theoretical Computer Science*, 270(1):701–738.

[9] Jürgen Dassow, Victor Mitrana, and Arto Salomaa. (1997). Context-free evolutionary grammars and the structural language of nucleic acids. *Biosystems*, 43(3):169–177.

[10] Jacob Hendricks, MatthewJ. Patitz, and TrentA. Rogers. (2015). Replication of arbitrary hole-free shapes via self-assembly with signal-passing tiles. In *Proceedings of the 14th International Conference on Unconventional Computation and Natural Computation*, volume 9252, pages 202–214.

[11] J.E. Hopcroft and J.D. Ullman. (1979). *Introduction to Automata Theory, Languages, and Computation (2nd edition)*. Addison-Wesley, Reading, MA.

[12] Salah Hussini, Lila Kari, and Stavros Konstantinidis. (2003). Coding properties of DNA languages. *Theoretical Computer Science*, 290(3):1557–1579.

[13] Masami Ito, Peter Leupold, and Kayoko Shikishima-Tsuji. (2006). Closure of language classes under bounded duplication. In *Developments in Language Theory*, pages 238–247. Springer.

[14] Lila Kari and Kalpana Mahalingam. (2006). DNA codes and their properties. In *Proceedings of the 12th International Meeting on DNA Computing*, pages 127–142.

[15] Hwee Kim and Yo-Sub Han. (2015). Non-overlapping inversion on strings and languages. *Theoretical Computer Science*, 592:9–22.

[16] Jennifer A Lee and James R Lupski. (2006). Genomic rearrangements and gene copy-number alterations as a cause of nervous system disorders. *Neuron*, 52(1):103–121.

[17] Peter Leupold, Victor Mitrana, and José M Sempere. (2004). Formal languages arising from gene repeated duplication. In *Aspects of Molecular Computing*, pages 297–308.

[18] Vladimir I Levenshtein. (1966). Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710.

[19] Victor Mitrana and Grzegorz Rozenberg. (1999). Some properties of duplication grammars. *Acta Cybernetica*, 14(1):165–177.

[20] David B Searls. (1993). The computational linguistics of biological sequences. *Artificial intelligence and molecular biology*, 2:47–120.

[21] Robert A Wagner and Michael J Fischer. (1974). The string-to-string correction problem. *Journal of the ACM*, 21(1):168–173.

[22] Derick Wood. (1987). *Theory of Computation*. Harper & Row.

[23] Takashi Yokomori and Satoshi Kobayashi. (1995). DNA evolutionary linguistics and RNA structure modeling: A computational approach. In *Proceedings of the 1st Intelligence in Neural and Biological Systems*, pages 38–45.