Acta Cybernetica 14 (1999) 165-177.

Some Properties of Duplication Grammars

Victor MITRANA * Grzegorz ROZENBERG †

Abstract

This paper considers context-free variants of duplication grammars. We investigate their generative capacity, their mutual relationship, and their relationship to the context-sensitive duplication grammars. We solve some problems left open in [6], e.g., proving that all regular languages can be generated by nearly all types of context-sensitive duplication grammars. We also consider some decision problems.

1 Introduction

String duplications or duplications of segments of strings are rather frequent in both natural and genetic languages. We refer to [1], [2] and [10] for discussions of duplication, and other operations related to the language of nucleic acids. For motivations coming from linguistics, we refer to [5] and [9].

Based on [1], Martin-Vidè and Păun introduced in [6] a generative mechanism (similar to the one considered in [2]) based only on duplication: one starts with a given finite set of strings and produces new strings by copying specified substrings to certain places in a string, according to a finite set of duplication rules. This mechanism is studied in [6] from the generative power point of view. The present paper considers the context-free versions of duplication grammars - this formalizes a possible hypothesis that duplications appear more or less at random within the genome in the course of its evolution. We solve some problems left open in [6], prove new results concerning the generative power of context-sensitive and context-free duplication grammars, and compare the two classes of grammars. Finally, some decision problems are discussed.

A *context-sensitive duplication rule* is a triple whose components are strings over a given alphabet (in the case of DNA the alphabet consists of the four nucleotids), say (u, x, v), which has the following interpretation:

- the string x, which appears to the left of uv in the processed string, is inserted in between u and v;

*University of Bucharest, Faculty of Mathematics Str. Academiei 14, 70109, Bucharest, Romania mitrana@funinf.math.unibuc.ro

†Leiden Institute of Advanced Computer Science, Leiden University, PO Box 9512, 2300 RA Leiden, The Netherlands, rozenber@wi.leidenuniv.nl and Department of Computer Science, University of Colorado at Boulder, USA.

- the string x, which appears to the right of uv in the processed string, is inserted in between u and v;

- the string x which appears in between u and v is doubled.

A *context-free duplication rule* is a string over the given alphabet, say x, whose effect is the duplication of x either to the right of, or to the left of, or immediately after, an already existing copy of x. Clearly, context-free duplication rules may be viewed as context sensitive duplication rules whose contexts are empty.

In vivo, cross-over takes place just between homologous chromosomes (chromosomes of the same type and of the same length), see [4]. A model of a cross-over between a DNA molecule and its replicated version is considered in [3] - this is a model for a cross-over between "sister" chromatides. One specifies an initial finite set of strings and a finite set of cross-over rules of the form $(\alpha, \beta, \gamma, \delta)$. It is assumed that every initial string is replicated so that two identical copies of every initial string are available. The first copy is cut between the segments α and β and the other one is cut between γ and δ. Now, the last segment of the second string gets attached to the first segment of the first string, and a new string is obtained. More generally, another string is also generated, by linking the first segment of the second string with the last segment of the first string. Iterating the procedure, one gets a language. The main idea of this approach is schematically presented in the Figure 1.

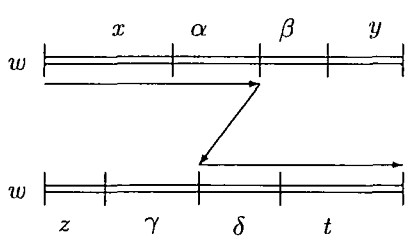

Figure 1

Hence, the splicing operation introduced by T. Head, see, e.g., [7] is performed here between identical strings. It is easily seen that one obtains the insertion of a substring of w in w; this induces a duplication of some chromosomes into genome. This type of recombination is considered to be the main way of producing tandem repeats or block deletions in chromosomes.

2 Basic definitions

In this section we give the basic notions and notations needed in the sequel. For basic formal language theory we refer to [7] or [8]. We use the following basic notation. The length of a word x is denoted by $|x|$, the empty string is denoted by ε; we have $|\varepsilon| = 0$. The mirror image of a word x is denoted by x^R. The set of all words over V is denoted by V^*, and $V^+ = V^* \setminus \{\varepsilon\}$. For sets X and Y, $X \setminus Y$ denotes the set-theoretic difference of X and Y. If X is finite, then $card(X)$ denotes its cardinality; $\emptyset$ denotes the empty set.

Here is the main notion of this paper.

A *(context-sensitive) duplication grammar* is a construct

$$\Delta = (V, D_l, D_r, D_0, A),$$

where V is an alphabet, D_l, D_r, D_0 are finite subsets of $V^* \times V^+ \times V^*$, and A is a finite subset of V^+. The elements of D_l, D_r and D_0 are context-sensitive duplication rules, and elements of A are called axioms.

Given a duplication grammar as above and two words $x, y \in V^+$, we define the following three types of direct derivation relations in Δ:

$$\begin{aligned}
x \Longrightarrow_{D_l} & \; y \text{ iff } x = x_1uvx_2zx_3,\; y = x_1uzvx_2zx_3, \\
& \text{with } x_1, x_2, x_3 \in V^*, \text{ and } (u, z, v) \in D_l, \\
x \Longrightarrow_{D_r} & \; y \text{ iff } x = x_1zx_2uvx_3,\; y = x_1zx_2uzvx_3, \\
& \text{with } x_1, x_2, x_3 \in V^*, \text{ and } (u, z, v) \in D_r, \\
x \Longrightarrow_{D_0} & \; y \text{ iff } x = x_1uzvx_2,\; y = x_1uzzvx_2, \\
& \text{with } x_1, x_2, x_3 \in V^*, \text{ and } (u, z, v) \in D_0.
\end{aligned}$$

The union of these relations is the direct derivation relation of Δ, denoted by $\Longrightarrow$, and the reflexive and transitive closure of $\Longrightarrow$ is the derivation relation of Δ, denoted by $\Longrightarrow^*$. The language generated by the duplication grammar Δ is defined by

$$L(\Delta) = \{y \in V^* \mid x \Longrightarrow^* y, \text{ for some } x \in A\}.$$

Thus, the language of Δ consists of all words obtained by beginning with strings in A, and applying iteratively duplication rules from $D_l \cup D_r \cup D_0$. The application of a rule to a string means to copy one of its substrings to the left of, or to the right of, or next to its "given" occurrence. Because each of the three sets of rules may be empty, one obtains seven families of languages denoted by $DUPL(X)$, $X \in \{l, r, 0, lr, l0, r0, lr0\}$; the presence of a letter within X means that the corresponding set of rules is non-empty, e.g., for $X = l0$, $D_l \neq \emptyset$, $D_0 \neq \emptyset$ and $D_r = \emptyset$.

Analogously, we define a context-free duplication grammar as a construct $\Delta = (V, D_l, D_r, D_0, A)$, where V and A have the same interpretation as above, but D_l, D_r, D_0 are finite subsets of V^+ whose elements are context-free duplication rules. Given a context-free duplication grammar as above and two words $x, y \in V^+$, we define three types of direct derivation relations:

$$x \models_{D_l} y \text{ iff } x = x_1x_2zx_3,\ y = x_1zx_2zx_3, \text{ with } x_1, x_2, x_3 \in V^*, \text{ and } z \in D_l,$$
$$x \models_{D_r} y \text{ iff } x = x_1zx_2x_3,\ y = x_1zx_2zx_3, \text{ with } x_1, x_2, x_3 \in V^*, \text{ and } z \in D_r,$$
$$x \models_{D_0} y \text{ iff } x = x_1zx_2,\ y = x_1zzx_2, \text{ with } x_1, x_2, x_3 \in V^*, \text{ and } z \in D_0.$$

Again, the union of these relations is the direct derivation relation, denoted by $\models$, and the reflexive and transitive closure of $\models$ is the derivation relation, denoted by $\models^*$. The language generated by the context-free duplication grammar Δ is defined by

$$L(\Delta) = \{y \in V^* \mid x \models^* y, \text{ for some } x \in A\}.$$

Again, we get seven families of languages denoted by $CFDUPL(X)$, $X \in \{l, r, 0, lr, l0, r0, lr0\}$.

3 A short comparison

We begin by settling the relationships among the seven families of context-free duplication languages.

Theorem 1. *The relations in the following diagram hold, where an arrow indicates a strict inclusion and a dotted line links two incomparable families.*

$$\begin{tikzcd}
CFDUPL(l) \arrow[drr] \arrow[ddrrrr] \arrow[ddr, dash, dashed] \arrow[dddd, dash, dashed] & & & & \\
& & CFDUPL(l0) \arrow[dr] & & \\
& CFDUPL(0) \arrow[rr] & & CFDUPL(lr0) & CFDUPL(l \arrow[l] \\
& & CFDUPL(r0) \arrow[ur] & & \\
CFDUPL(r) \arrow[urr] \arrow[uurrrr] \arrow[uur, dash, dashed] & & & &
\end{tikzcd}$$

Proof. The language $\{a^nb^ma^pb^q \mid n, m, p, q \geq 1\}$ is in $CFDUPL(0)$ (one starts with $abab$ and doubles either an occurrence of a or an occurrence of b) but not in $CFDUPL(lr)$. To see the latter, we note that each context-free duplication grammar having just left and right duplication rules generates strings in $a^+b^+a^+b^+a^+b^+$; a contradiction.

By a similar reasoning, the language $\{a^nb^m \mid n, m \geq 1\}$ belongs to $CFDUPL(l0) \cap CFDUPL(r0) \cap CFDUPL(lr)$ but not to $CFDUPL(l) \cup CFDUPL(r)$.

The language $\{a,b,c\}^+$ is in $CFDUPL(r) \cap CFDUPL(l)$ (the initial set contains all strings of length at most 3, each letter a,b,c appearing at most once; duplication rules allow copying of any letter to the right/left of one of its occurrences.) Because there are arbitrarily long square-free strings in $\{a,b,c\}^+$, [11], it follows that $\{a,b,c\}^+ \notin CFDUPL(0)$.

Finally,

$$\{a,b,c\}^+\{\$\}^+\{d,e,f\}^+ \in CFDUPL(lr0) \setminus (CFDUPL(l0) \cup CFDUPL(r0))$$

which concludes the proof. □

The following result concerning the relationships among the context-sensitive families of duplication languages has been proved in [6].

Theorem 2.[6]

1. *The families DUPL(l) and DUPL(r) are incomparable.*
2. *The following inclusions*

$$DUPL(r) \cup DUPL(l) \subset DUPL(lr)$$
$$DUPL(0) \subset (DUPL(r0) \cap DUPL(l0))$$

are proper.

It is an *open problem* whether or not $DUPL(0)$ is included in $DUPL(l)$ or in $DUPL(r)$. However, we have

Proposition 1. *CFDUPL(0) is strictly included in DUPL(lr).*

Proof. Let $\Delta = (V, \emptyset, \emptyset, D_0, A)$ be a duplication grammar with $D_0 = \{x_1, x_2, \ldots, x_n\}$. Construct a duplication grammar $\Delta' = (V, D_l, D_r, \emptyset, A')$, where

$$\begin{aligned} D_l = D_l &= \{(x_i, x_i, \varepsilon) | 1 \leq i \leq n\}, \\ A' &= \{z \in L(\Delta) | \text{ each } x_i \text{ has at most two non-overlapped occurrences} \\ &\quad \text{in } z\}. \end{aligned}$$

It is easy to see that A' is a finite set, and $L(\Delta) = L(\Delta')$. □

Along the same lines, we have

Theorem 3. $CFDUPL(X) \subset DUPL(X)$, *for all* $X \in \{0, l, r, l0, r0, lr, lr0\}$.

Proof. It suffices to provide languages that prove all inclusions to be strict.

The duplication grammar $\Delta = (\{a,b\}, \{(\varepsilon, a, a), (\varepsilon, b, b)\}, \emptyset, \emptyset, \{ab, a^2b, ab^2, a^2b^2\}$ generates $L_1 = \{a^n b^m | n, m \geq 1\}$. Hence L_1 is in $DUPL(l)$ (also in $DUPL(r)$) but not in $(CFDUPL(l) \cup CFDUPL(r)$.

Similarly, $\{a^n b^m a^p b^q | n, m, p, q \geq 1\} \in DUPL(lr) \setminus CFDUPL(lr)$.

One can show that $\{a^n b^n ab | n \geq 1\}$ cannot be generated by any context-free duplication grammar. On the other hand, $\{a^n b^n ab | n \geq 1\} \in DUPL(lr0)$ (see [6]).

Take now the language $L_2 = \{ab^nc^md^pe|1 \leq n,m,p \leq 3\}^+$. This language can be obtained by starting with the string $abcde$ and iteratively applying rules from the set

$$D_0 = \{(\varepsilon, abcde, \varepsilon), (a,b,c), (ab,b,c), (b,c,d), (bc,c,d), (c,d,e), (cd,d,e)\}.$$

Consider the homomorphism $h : \{a,b,c\}^* \longrightarrow \{a,b,c,d,e\}^*$ defined by $h(a) = ab^3cde$, $h(b) = abc^3de$, $h(c) = abcd^3e$. Let x be an arbitrarily long square-free string over $\{a,b,c\}$. The string $h(x)$ is in L_2. It is easy to notice that the adjacent identical substrings in $h(x)$ are only the letters from $\{a,b,c\}$. If L_2 were in $CFDUPL(0)$, then any context-free duplication grammar generating L_2 would generate strings containing arbitrarily many adjacent occurrences of the same letter from $\{a,b,c\}$; a contradiction. □

4 Observations on the generative power

We start by considering unary alphabets. We will prove that in this case the generative power of duplication grammars equals the accepting power of deterministic finite automata. To this end, we prove the following lemma.

Lemma 1. *Over the unary alphabet, the equality $DUPL(X) = CFDUPL(0)$ holds for any $X \in \{l, r, 0, lr, l0, r0, lr0\}$.*

Proof. Let $\Delta = (\{a\}, D_l, D_r, D_0, A)$ be a duplication grammar. Let

$$\begin{aligned}
D_l &= \{(u_l, a^{i_l}, v_l)|1 \leq l \leq n\},\\
D_r &= \{(x_l, a^{j_l}, y_l)|1 \leq l \leq m\},\\
D_0 &= \{(z_l, a^{k_l}, w_l)|1 \leq l \leq p\}.
\end{aligned}$$

Take

$$\alpha = max(\{|uxv| \ : \ (u,x,v) \in D_l \cup D_r \cup D_0\} \cup \{|x| \ : \ x \in A\}.$$

Consider now the context-free duplication grammar

$$\Delta' = (\{a\}, \emptyset, \emptyset, D_0', A'),$$

where

$$\begin{aligned}
A' &= \{x|x \in L(\Delta), |x| \leq 3\alpha\},\\
D_0' &= \{a^q|q = \sum_{s=1}^{n} \alpha_s i_s + \sum_{s=1}^{m} \beta_s j_s + \sum_{s=1}^{p} \gamma_s k_s, \ \alpha \leq q \leq 2\alpha\}.
\end{aligned}$$

We claim that $L(\Delta) = L(\Delta')$. Note that each rule in D_0' is applicable to strings of length at least α. Furthermore, each application of a rule in D_0' simulates the application of a sequence of rules from $D_l \cup D_r \cup D_0$. Consequently, $L(\Delta') \subseteq L(\Delta)$.

All strings of length at most 3α from $L(\Delta)$ are also in $L(\Delta')$. Let z be the shortest string in $L(\Delta)$ such that $|z| > 3\alpha$. Then there exists a derivation in Δ':

$$x \Longrightarrow^+ y \Longrightarrow^+ z$$

with

$$\begin{array}{ll} (i) & x \in A, \\ (ii) & \alpha \leq |y| \leq 3\alpha, \\ (iii) & \alpha \leq |z| - |y| \leq 2\alpha. \end{array}$$

Because $y \in A'$ one may write $y \Longrightarrow_{D'_0} z$, and so $z \in L(\Delta')$. Inductively, $L(\Delta) \subseteq L(\Delta')$. □

Theorem 4. *A language over a unary alphabet is regular if and only if it is generated by a duplication grammar.*

Proof. By the previous lemma, it suffices to consider duplication grammars with just context-free duplication rules whose effect is to double an occurrence of a substring. Let $L \subseteq \{a\}^*$ be a regular language. Then, there exist a finite set F and the positive integers $k_i, 1 \leq i \leq m$, and $q > max\{\#(x) | x \in F\}$ such that

$$L = F \cup \bigcup_{i=1}^{m} \{a^{k_i+nq} | n \geq 1\}$$

This can be easily seen if one considers a deterministic finite automaton accepting L, for which the transition function is defined everywhere.

Consider now the duplication grammar:

$$\Delta = (\{a\}, \emptyset, \emptyset, \{a^q\}, F \cup \{a^{k_i+q} | 1 \leq i \leq m\}.$$

Clearly, $L = L(\Delta)$. Duplications can never be carried out on words of F.

Conversely, let us consider a duplication grammar $\Delta = (\{a\}, \emptyset, \emptyset, D_0, A)$, with $D_0 = \{a^{c_1}, a^{c_2}, \ldots, a^{c_n}\}$. Let

$$p = gcd(d_1, d_2, \ldots, d_m, c_1, c_2, \ldots c_n),$$

where *gcd* means the greatest common divisor. If $L(\Delta)$ is finite, then it is obviously regular. If $L(\Delta)$ is an infinite set, then there are $t_i, 1 \leq i \leq s,\ s \leq p$, such that

$$L(\Delta) = F \cup \bigcup_{i=1}^{s} \{a^{t_i+kp} | k \geq 0\},$$

for some finite set F. Consequently, $L(\Delta)$ is regular which completes the proof. □

The next result settles a problem left open in [6].

Theorem 5. *All regular languages are in $DUPL(X)$, $X \in \{l, r, l0, r0, lr, lr0\}$.*

Proof. We present a proof for $DUPL(r)$, the proofs for other cases are analogous. Let R be a regular language recognized by the deterministic finite automaton $M = (Q, V, \delta, q_0, F)$ with the total transition function δ. Let for each state q, C_q be defined as follows:

$$C_q = \{x \in V^+ | \delta(q, x) = q \text{ by passing each state, different from } q\text{, at most once}\}.$$

For strings $x, y \in V^*$, we define the equivalence relation $\sim_R$ as follows:

$$(x \sim_R y) \text{ iff } (uxv \in R \text{ iff } uyv \in R), \text{ for any } u, v \in V^*.$$

It is well-known (see e.g. [8]) that $V^*/\sim_R$ (the quotient of V^* by $\sim_R$) is finite; let k be the cardinality of $V^*/\sim_R$ (the index of $\sim_R$).

Now, one constructs the duplication grammar $\Delta = (V, \emptyset, D_r, \emptyset, A)$, where

$$\begin{aligned} D_r &= \bigcup_{q \in Q, C_q \neq \emptyset} \{(x, y, \varepsilon) | xy \sim_R x,\ |x| < k, y \in C_q\}, \text{ and} \\ A &= \{w \in R | \text{ for each } q \in Q, \text{ each string in } C_q \\ &\qquad \text{has at most } k \text{ non-overlapping occurrences in } w\}. \end{aligned}$$

We claim that A is finite. Indeed, no word longer than $(k+1)l \cdot card(Q)$, where $l = max\{card(C_q) | q \in Q\}$, is in A. To see this, assume that such a word, say w, is in A; so $|w| = p \geq (k+1)l \cdot card(Q)$. Let $q_0, q_1, \ldots, q_p$, $q_p \in F$, be the sequence of states that accepts w. At least $(k+1)l$ states in this sequence must be the same; assume that q is such a state. But then w contains at least $k+1$ identical substrings in C_q; a contradiction.

Clearly, $L(\Delta) \subseteq R$. Let z be the shortest word in $R \setminus L(\Delta)$. Thus, there exists $x \in C_q$, for some $q \in Q$, such that x occurs more than k times in z. Let $z = wxy$, with $|w| \geq k$, where the given occurrence of x is the last (rightmost) occurrence of x in z. Let $z = uvxy$ with $|v| = k$. Thus v has $k+1$ prefixes, and so there are two prefixes v_1, v_2 of v such that $v_1 \sim_R v_2$ and $|v_1| < |v_2|$. We choose the closest pair of such prefixes. By replacing v_2 by v_1 in v we get a string $uv'xy$ which is in $L(\Delta)$ because it is in R and it is shorter than z. Moreover, $v_2 = v_1 t$, where t must be in C_q, for some $q \in Q$ (because of the choice of v_1 and v_2). Consequently, $(v_1, t, \varepsilon) \in D_r$, and so $uv'xy \Longrightarrow_{D_r} z$. Thus $z \in L(\Delta)$; a contradiction.

Analogously one proves that each regular language is in $DUPL(l)$. □

We recall that the family $DUPL(0)$ is incomparable with the family of regular languages.

The position of the class of regular languages with respect to the classes of context-free duplication languages is given by the next theorem.

Theorem 6. *The family of regular languages is incomparable with any of the families* $CFDUPL(X), X \neq 0$.

Proof. The regular language $V^+\{c\}^+V^+$, where V contains at least three symbols and $c \notin V$, cannot be generated by any context-free duplication grammar. Indeed, if a context-free duplication grammar generates all strings in $V^+\{c\}^+V^+$, then it must contain left/right duplication rules involving strings in $V^+ + c^+$. Therefore, also strings in $V^+\{c\}^+V^+\{c\}^+V^+$ can be generated.

Consider now the Dyck language over $\{a, b\}$, denoted by D_{ab}, and the non-regular language $L = \{ab\}D_{ab}$. This language is in $CFDUPL(r)$. The context-free duplication grammar $\Delta = (\{a, b\}, \emptyset, \{ab\}, \emptyset, \{abab\})$ with only right duplication rules generates L. Clearly, $L(\Delta) \subseteq L$; let z be the shortest string in $L \setminus L(\Delta)$. If $z = abxy$, with $x, y \in D_{ab}$, then ab yields z in Δ as follows:

$$ab \models^* aby \models^* abxy.$$

If $z = abaxb$, with $x \in D_{ab}$, then the derivation $ab \models abab \models^* abaxb$ is possible in Δ. Consequently, $L(\Delta) = L$. □

The relation between $CFDUPL(0)$ and the class of regular languages remains open.

Recall that a homomorphism which erases some symbols and leaves the others symbols unchanged is called a *projection*. A projection $h : (V \cup V')^* \longrightarrow V^*$ that erases the symbols in V' only is the projection of V, denoted by pr_V.

Theorem 7. *For each context-free language $L \in V^*$, there exists a language L in CFDUPL(r) (CFDUPL(l)) and a homomorphism h such that $L = pr_V(h^{-1}(L'))$.*

Proof. Let $G = (N, V, S, P)$ be a context-free grammar generating L. Assume that

$$P = \bigcup_{i=1}^{n} \{A_i \longrightarrow x_{i,j} | 1 \leq j \leq r_i\},$$

with $S = A_1$. Furthermore, we assume that $\varepsilon \notin L$. Let $V' = N \cup V \cup \{c_i | 1 \leq i \leq n\} \cup \{d\}$, where c_i, d, are new symbols. Let then Δ be the duplication grammar $(V', \emptyset, D_r, \emptyset, A)$, where

$$\begin{aligned} D_r &= \{(c_i x_{i,j} | 1 \leq i \leq n,\ 1 \leq j \leq r_i\}, \text{ and} \\ A &= \{cx_{1,1}dcx_{1,2}d \ldots dcx_{1,r_1}dcx_{2,1}d \ldots dcx_{n,r_n}dA_1\}. \end{aligned}$$

Now, let h be the homomorphism

$$h : (V \cup \{[i, j]| 1 \leq i \leq n, 1 \leq j \leq r_i\} \cup \{c_i | 1 \leq i \leq n\} \longrightarrow (V')^*$$

such that

$$\begin{aligned} h([i, j]) &= c_i x_{i,j} d, 1 \leq i \leq n,\ 1 \leq j \leq r_i, \\ h(c_i) &= A_i c_i, 1 \leq i \leq n, \text{ and} \\ h(a) &= a,\ a \in V. \end{aligned}$$

It is easy to see that $pr_V(h^{-1}(L(\Delta))) = L(G)$. Clearly, whenever a substring $c_i x_{i,j}$ is copied, this is done somewhere to the right of the last occurrence of d -

otherwise one gets a string "rejected" by applying the inverse homomorphism h. Also, all strings that contain nonterminal occurrences that are not immediately followed by some c_i, to the right of the last occurrence of d, are rejected in the same way. Moreover, every occurrence of a nonterminal A_i, situated to the right of the last occurrence of d, has to be followed by just one occurrence of c_i. In this way duplication rules simulate the application of production rules in G. □

5 Decision problems

We discuss in this section some basic decision problems. We begin by pointing out that the "totality problem" is decidable for all families of duplication languages.

Theorem 8. *Let Δ be a duplication grammar over the alphabet V. It is decidable whether or not $L(\Delta) = V^*$.*

Proof. We will consider duplication grammars having only left duplication rules - the other types of duplication grammars can be treated in a similar way. Let $\Delta = (V, D_l, \emptyset, \emptyset, A)$ be a duplication grammar. The main point of our argument is the following property

$$L(\Delta) = V^* \text{ if and only if } \{x \in V^* : |x| \leq k + 1\} \subset L(\Delta),$$

where $k = max\{|x| : x \in A\}$.

The "only if" part is obvious. For the "if" part of the proof, assume that z is a shortest word in $V^* \setminus L(\Delta)$. This word can be written as $z = ya$ with $a \in V$. Hence $y \in L(\Delta) \setminus A$, and so there exists $x \in A$ such that $x \Longrightarrow^+_{D_r} y$. Because $|xa| < |ya|$, it follows that $xa \in L(\Delta)$. But, also $xa \Longrightarrow^+_{D_r} ya = z$. To conclude, it suffices to note that the inclusion $\{x \in V^* : |x| \leq k + 1\} \subset L(\Delta)$ is decidable due to the decidabilty of the membership problem. □

It is proved in [6] that the membership of a context-free language in the family of languages $DUPL(X), X \neq 0$, is not decidable. Our next theorem extends this result to the families of context-free duplication languages, as well as to $DUPL(0)$.

Theorem 9. *It is not decidable whether or not a context-free language is in a family $CFDUPL(X), X \notin \{r, l\}$.*

Proof. The proof is similar to the one in [6]. Let G be an arbitrary context-free grammar with the terminal alphabet $\{a, b\}$, and let

$$L = L(G)\{c, d\}^* \cup \{a, b\}^*\{c^n d^n | n \geq 1\}.$$

If $L(G) = \{a, b\}^*$, then $L = \{a, b\}^*\{c, d\}^*$ which is in $CFDUPL(X)$, for all $X \notin \{r, l\}$. It is easily seen that the grammar $\Delta = (\{a, b, c, d\}, \emptyset, \emptyset, D_0, A)$, with

$$D_0 = \{a, b, c, d, ab, ba, cd, dc\}, \text{ and } A = \{a, b, c, d, ab, aba, ba, bab, cd, cdc, dc, dcd\},$$

generates $\{a, b\}^*\{c, d\}^*$. The reader my easily check this assertion.

If $L(G) \neq \{a, b\}^*$, then L cannot be generated by any context sensitive duplication grammar (see the proof of Theorem 4 in [6]). Consequently, $L \in CFDUPL(X)$ for $X \notin \{r, l\}$, if and only if $L(G) = \{a, b\}^*$, which is undecidable. □

This result can be also extended to the families $CFDUPL(r)$ and $CFDUPL(l)$.

Theorem 10. *It is not decidable whether or not a context-free language is in a family $CFDUPL(X), X \in \{r, l\}$.*

Proof. The proof is based on a reduction to the Post Correspondence Problem (PCP). Take an arbitrary instance of PCP, i.e., two arbitrary n-tuples of nonempty strings over the alphabet $\{a, b\}$:

$$x = (x_1, x_2, \ldots, x_n),$$

$$y = (y_1, y_2, \ldots, y_n).$$

Then, consider the languages

$$L_z = \{ba^{t_1}ba^{t_2}\ldots ba^{t_k}cz_{t_k}\ldots z_{t_2}z_{t_1} \mid k \geq 1\} \text{ for } z \in \{x, y\},$$

$$L_s = \{w_1cw_2cw_2^Rcw_1^R \mid w_1, w_2 \in \{a, b\}^*\}, \text{ and}$$

$$L(x, y) = \{a, b, c\}^* - (L_x\{c\}L_y^R \cap L_s).$$

It is known that $L(x, y)$ is a context-free language. For every solution $(i_1, i_2, \ldots, i_k)$ of $PCP(x, y)$ the strings

$$ba^{i_1}ba^{i_2}\ldots ba^{i_k}cx_{i_k}\ldots x_{i_2}x_{i_1}cy_{i_1}^Ry_{i_2}^R\ldots y_{i_k}^Rca^{i_k}b\ldots ba^{i_2}ba^{i_1}b$$

are not in $L(x, y)$.

Clearly, when $L(x, y) = \{a, b, c\}^*$, then $L(x, y)$ is in $CFDUPL(r) \cap CFDUPL(l)$.

Now, it is sufficient to prove that $L(x, y) \notin CFDUPL(l) \cup CFDUPL(r)$ if $L(x, y) \neq \{a, b, c\}^*$.

Let us suppose that $L(x, y) = L(\Delta)$, $\Delta = (\{a, b, c\}, \emptyset, D_r, \emptyset, A)$. We choose a solution $(i_1, i_2, \ldots i_k)$ such that

$$|x_{i_k}x_{i_{k-1}}\ldots x_{i_1}| > max\{|w| \mid w \in A\}.$$

For $\{a, b\}^* \subseteq L(\Delta)$, there exists a word $w \in A$ such that

$$w \models^* y_{i_1}^Ry_{i_2}^R\ldots y_{i_k}^R \in L(\Delta).$$

By the choice of the solution $(i_1, i_2, \ldots, i_k)$ the word

$$z = ba^{i_1}ba^{i_2}\ldots ba^{i_k}cx_{i_k}\ldots x_{i_2}x_{i_1}cwca^{i_k}b\ldots ba^{i_2}ba^{i_1}b$$

is in $L(\Delta)$.

Therefore, we get

$$z \models^* ba^{i_1}ba^{i_2}\dots ba^{i_k}cx_{i_k}\dots x_{i_2}x_{i_1}cy_{i_1}^R y_{i_2}^R\dots y_{i_k}^R ca^{i_k}b\dots ba^{i_2}ba^{i_1}b,$$

a contradiction. Hence the theorem holds. □

Finally, we consider "nonemptiness of the intersection problem" for $DUPL(X), X \neq 0$.

Theorem 11. *It is undecidable whether or not $L_1 \cap L_2 = \emptyset$? for arbitrary two duplication languages in $DUPL(X), X \neq 0$.*

Proof. Let $x = (x_1, x_2, \dots, x_n), y = (y_1, y_2, \dots, y_n)$ be an instance of PCP, and let

$$\begin{aligned} L_x &= \{w\$cd^{i_1}\$cd^{i_2}\dots\$cd^{i_k}x_{i_k}\dots x_{i_2}x_{i_1} | k \geq 1, 1 \leq i_j \leq n, 1 \leq j \leq k\} \\ &\cup \{w\$cd^{i_1}\$cd^{i_2}\dots\$cd^{i_k}\$x_{i_k}\dots x_{i_2}x_{i_1} | k \geq 1, 1 \leq i_j \leq n, 1 \leq j \leq k\}, \end{aligned}$$

where $w = cdx_1cdy_1cd^2x_2cd^2y_2\dots cd^nx_ncd^ny_n$. L_y is defined analogously.

Clearly, the duplication grammar $\Delta = (\{a, b, c, d, \$, \#\}, \emptyset, D_r, \emptyset, \{w\$\#\})$, with

$$\begin{aligned} D_r &= \{(\$, cd^ix_i, \#) | 1 \leq i \leq n\} \cup \{(\$, cd^ix_i, X) | 1 \leq i \leq n, X \in \{a, b\}\} \\ &\cup \{(d, \$, a), (d, \$, b)\} \end{aligned}$$

generates L_x.

This concludes the proof, because $L_x \cap L_y = \emptyset$ if and only if the instance (x, y) of PCP has no solution. □

Acknowledgements

The first author is grateful to Leiden Center for Natural Computing for supporting his stay at Leiden University in June 1998, during which the work on this paper was initiated.

References

[1] J. Dassow and V. Mitrana, On some operations suggested by the genome evolution. *Pacific Symposium on Biocomputing'97* (R. Altman, K. Dunker, L. Hunter, T. Klein eds.), Hawaii, 1997, 97–108.

[2] J. Dassow and V. Mitrana, Evolutionary grammars: a grammatical model for genome evolution, *Proc. German Conf. in Bioinformatics GCB'96*, (R. Hofestädt, T. Lengauer, M. Löffler, D. Schomburg eds.), LNCS 1278, Springer-Verlag, 1997, 199–209.

[3] J. Dassow and V. Mitrana, Self cross-over systems. In *Computing with biomolecules* (Gh. Paun ed.), World Scientific, 1998 (in press).

[4] D. L. Hartl, D. Freifelder and L. A. Snyder, *Basic Genetics*, Jones and Bartlett Publ., Boston, Portola Valley, 1988.

[5] A. Manaster Ramer, Uses and misuses of mathematics in linguistics, *Proc. X Congreso de Lenguajes Naturales y Lenguajes Formales*, Sevilla, 1994.

[6] C. Martin-Vidè and G. Păun, Duplication grammars, *Acta Cybernetica* (submitted).

[7] G. Păun, G. Rozenberg, A. Salomaa, *DNA Computing, New Computing Paradigms*, Springer Verlag, Berlin, Heidelberg, 1998.

[8] G. Rozenberg and A. Salomaa (eds.), *Handbook of Formal Languages*, vol. I, Springer, Berlin, 1997.

[9] W. C. Rounds, A. Manaster Ramer and J. Friedman, Finding natural languages a home in formal language theory. In *Mathematics of Language* (A. Manaster Ramer ed.), John Benjamins, Amsterdam, 1987, 349–360.

[10] D. B. Searls, The computational linguistics of biological sequences. In *Artificial Intelligence and Molecular Biology* (L. Hunter ed.), AAAI Press, The MIT Press, 1993, 47–120.

[11] A. Thue, Uber unendliche Zeitchenreihen, *Norske Vid. Selk. Skr. I. Mat. Nat. Kl. Christiania*, 7(1906), 1–22.