Duplication Grammars [*]

Carlos MARTÍN-VIDE [†]   Gheorghe PĂUN [‡]

Abstract

Motivated by the abundance of duplication operations appearing in natural languages and in the genetic area, we introduce a generative mechanism based on duplication operations: one starts from a given finite set of strings and one produces new strings by copying certain substrings, according to a set of given rules (which specify contexts where duplicated substrings can be inserted). We mainly investigate the power of such devices, comparing the obtained families of languages to each other and with families in the Chomsky hierarchy. In this context, we also solve a problem left open in a paper by Dassow and Mitrana, [1], even giving a stronger results: the iterated duplication with rules of only one type (see formal definitions in the sequel) can generate non-context-free languages (even non-matrix languages).

1 Introduction

The duplication of strings or of parts of strings is a common operation both in natural languages and in the genetic languages. We refer to [12], [15] for discussions about this topic from the linguistical point of view; it is interesting to note that it seems that reduplication, which is a non-context-free type of operation, is more abundant in natural languages than mirror-image constructions, which can be handled by linear Chomsky grammars. This is an argument supporting the thesis that Chomsky grammars and their classification is not an adequate model of the syntax of natural languages. Details about operations appearing in the genetic area, duplication included, can be found, for instance, in [2], [10], [11], [18], or in the first chapter of [14].

We start here from the approach in [1] to duplication in the DNA area. In that paper, one considers *duplication rules* of the form (u_1, v, u_2), where u_1, v, u_2 are strings (over the DNA alphabet, but the operation can be defined in general, over any finite alphabet); the idea is that the string v can be inserted in between the strings u_1, u_2, providing that it already appears in the processed string. This

[*]Research supported by the Direcció General de Recerca, Generalitat de Catalunya (PIV)

[†]Research Group in Mathematical Linguistics and Language Engineering Rovira i Virgili University Pl. Imperial Tàrraco 1, 43005 Tarragona, Spain e-mail: cmv@astor.urv.es

[‡]Institute of Mathematics of the Romanian Academy PO Box 1 – 764, 70700 Bucureşti, Romania, e-mail: gpaun@imar.ro.

amounts to a duplication of v. Both the one-step operation of this type and the iterated operation are considered in [1], where one proves that the non-iterated operation preserves the regular and the context-free languages. It is stated as an open problem in [1] the question whether or not the iterated duplication preserves the context-freeness.

We distinguish here three types of duplications: taking v from the left of the place where a copy of it will be produced, from the right of that place, or duplicating v near an already existing copy of it; in all cases, the context (u_1, u_2) controls the operation. We prove that in each of these cases, the iterated duplication can lead finite languages to languages which cannot be generated by matrix languages (without appearance checking). This solves the problem in [1], in a stronger form. The result is not surprising, because of the context-sensitivity of the operations we use (the presence of the context (u_1, u_2) associated with each string v which can be duplicated); this also corresponds to the situation met for the so-called *insertion grammars* of [5], where the string to be inserted is not necessarily a substring of the current string (see the corresponding chapter in [13]). Somewhat unexpected is the fact that non-context-free languages can be also produced when starting from finite languages and using duplication operations in the restricted case when the copy of the duplicated string is produced adjacent to the string itself.

2 Formal Language Theory Prerequisites

We introduce a few notions and notations necessary in the sequel; for details, the reader is referred to [16].

For an alphabet V, we denote by V^* the set of all strings over V, including the empty one, denoted by λ; the set of non-empty strings over V is denoted by V^+. The length of $x \in V^*$ is denoted by $|x|$ and the number of occurrences of a symbol $a \in V$ in a string $x \in V^*$ is denoted by $|x|_a$. The left derivative of a language $L \subseteq V^*$ with respect to a string $x \in V^*$ is $\partial^l_x(L) = \{w \in V^* \mid xw \in L\}$ and the right derivative is $\partial^r_x(L) = \{w \in V^* \mid wx \in L\}$.

A *context-free grammar* is a construct $G = (N, T, S, P)$, where N is the nonterminal alphabet, T is the terminal alphabet, $S \in N$ is the axiom, and P is the set of production rules, pairs of the form (A, x) with $A \in N, x \in (N \cup T)^*$ (written in the form $A \to x$). For $x, y \in (N \cup T)^*$, we write $x \Longrightarrow y$ if and only if $x = x_1 A x_2, y = x_1 z x_2$, for some $x_1, x_2 \in (N \cup T)^*$ and $A \to z \in P$. By $\Longrightarrow^*$ we denote the reflexive and transitive closure of the relation $\Longrightarrow$. The language generated by G is defined by $L(G) = \{x \in T^* \mid S \Longrightarrow^* x\}$.

By *FIN, REG, LIN, CF, CS, RE* we denote the families of finite, regular, linear, context-free, context-sensitive, and recursively enumerable languages, respectively. This is the *Chomsky hierarchy*, the standard test bed for any new type of language generating devices.

3 Duplication Grammars

We now introduce the main object of our study, in several variants.

A *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 language over V.

The elements of the sets D_l, D_r, D_0 are called *duplication rules*, those of A are called *axioms*. Note that the duplication rules have the second element non-empty.

With respect to such a grammar, for $x, y \in V^*$ we define:

$$\begin{array}{lll} x \Longrightarrow_{D_l} y & \text{iff} & x = x_1u_1u_2x_2vx_3, y = x_1u_1vu_2x_2vx_3, \\ & & \text{for some } x_1, x_2, x_3 \in V^*, (u_1, v, u_2) \in D_l, \\ x \Longrightarrow_{D_r} y & \text{iff} & x = x_1vx_2u_1u_2x_3, y = x_1vx_2u_1vu_2x_3, \\ & & \text{for some } x_1, x_2, x_3 \in V^*, (u_1, v, u_2) \in D_r, \\ x \Longrightarrow_{D_0} y & \text{iff} & x = x_1u_1vu_2x_2, y = x_1u_1vvu_2x_2, \\ & & \text{for some } x_1, x_2 \in V^*, (u_1, v, u_2) \in D_0. \end{array}$$

We write $\Longrightarrow$ for denoting any of these relations $\Longrightarrow_\alpha$; the reflexive and transitive closure of $\Longrightarrow$ is denoted by $\Longrightarrow^*$. Then, the *language generated* by the grammar Δ is defined by

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

In words, one starts from strings in A and one iteratively duplicates subtrings as allowed by the triples in the sets D_l, D_r, D_0. In all cases, a copy of a substring of the current string is produced, to the left of its former occurrence, to the right, or adjacent to that occurrence, respectively.

Because one or two of the three sets D_l, D_r, D_0 can be empty, we obtain in this way seven classes of grammars and, correspondingly, seven families of languages. We denote by $DUPL(\alpha)$ the family of languages generated by duplication grammars containing rules of the type α, where α can be one of $l, r, 0, lr, l0, r0, lr0$; the absence of one of the symbols $l, r, 0$ means that the corresponding sets of duplication rules is empty.

Several duplication grammars and languages generated by them will be considered in the following sections, hence we do not give here examples.

4 Generative Capacity

We compare the families $DUPL(\alpha)$ to each other and to families of languages in the Chomsky hierarchy.

Note that each duplication language has the bounded growth property: for each infinite language L there is a constant k such that if $x \in L$, then there is $y \in L, y \neq x$, with $||x| - |y|| \leq k$.

Directly from the definitions, we obtain the relations in the diagram in Figure 1 (an arrow from a lower family to an upper family indicates an inclusion which is not necessarily proper). This diagram will be useful below when investigating the relationships between these families.

$$\begin{tikzcd}
 & & CS & \\
 & DUPL(lr0) \arrow[ur] & & CF \arrow[ul] \\
DUPL(lr) \arrow[ur] & DUPL(l0) \arrow[u] & DUPL(r0) \arrow[ul] & LIN \arrow[u] \\
DUPL(l) \arrow[u] \arrow[ur] & DUPL(r) \arrow[ul] \arrow[ur] & DUPL(0) \arrow[ul] \arrow[u] & REG \arrow[u] \\
 & & FIN \arrow[ull] \arrow[ul] \arrow[u] \arrow[ur] &
\end{tikzcd}$$

Figure 1

Lemma 1. *Let V be an alphabet containing at least three symbols. The language V^+ is in $DUPL(l) \cap DUPL(r)$, but not in $DUPL(0)$.*

Proof. For the duplication grammars $\Delta = (V, D_l, D_r, \emptyset, A)$, where one of D_l, D_r is empty and the other is equal to $\{(\lambda, a, \lambda) \mid a \in V\}$, and

$$A = \{x \in V^+ \mid |x|_a \leq 1 \text{ for each } a \in V\},$$

we obviously have $L(\Delta) = V^+$.

The language V^+ cannot be generated by a duplication grammar which uses only rules in D_0, because at each step of the form $x \Longrightarrow_{D_0} y$ we produce a string y

of the form $y = y_1vvy_2$ for some $v \in V^+$. However, for V containing at least three symbols, there are arbitrarily long square-free strings in V^+, [19], [17]. □

Remark 1. (Suggested to us, without a proof, by V. Mitrana) The language V^+ for $V = \{a, b\}$ is in $DUPL(0)$, that is, the previous result cannot be improved by considering V with only two symbols. Indeed, the duplication grammar

$$\begin{aligned}&\Delta = (\{a,b\}, \emptyset, \emptyset, D_0, A),\\ &A = \{a, b, ab, aba, ba, bab\},\\ &D_0 = \{(\lambda, a, \lambda),\ (\lambda, b, \lambda),\ (\lambda, ab, \lambda),\ (\lambda, ba, \lambda)\},\end{aligned}$$

generates V^+. The inclusion $L(\Delta) \subseteq V^+$ is obvious, the converse inclusion can be proved as follows. For a string $x = c_1^{i_1}c_2^{i_2}\dots c_r^{i_r}, r \geq 1, c_j \in \{a, b\}, i_j \geq 1$, for all $1 \leq j \leq r$, with $c_j \neq c_{j+1}$, $1 \leq j \leq r-1$, we denote by $red(x)$ the (*reduced*) string $c_1c_2\dots c_r$ (all blocks of symbols a and b are reduced to one symbol). Clearly, starting from strings in A and using the rules $(\lambda, ab, \lambda), (\lambda, ba, \lambda)$, we can generate all strings $w \in V^+, w = red(x)$, for some $x \in V^+$. Then, by using rules $(\lambda, a, \lambda), (\lambda, b, \lambda)$, we can pass from $red(x)$ to x, for any $x \in V^+$.

It is easy to see that $\{a, b\}^+ \in DUPL(l) \cap DUPL(r)$, too, and that $a^+ \in DUPL(\alpha)$ for all $\alpha \in \{l, r, 0\}$.

Lemma 2. (i) $\{aba^nb^n \mid n \geq 1\} \in DUPL(r) - DUPL(l)$. (ii) $\{a^nb^nab \mid n \geq 1\} \in DUPL(l) - DUPL(r)$.

Proof. (i) The duplication grammar

$$\Delta = (\{a,b\}, \emptyset, \{(a, ab, b)\}, \emptyset, \{abab\})$$

obviously generates the language $\{aba^nb^n \mid n \geq 1\}$.

This language cannot be generated by a duplication grammar which uses only rules in D_l: in order to produce strings aba^nb^n with arbitrarily large n we need to use triples of the form $(a^j, a^ib^i, b^k) \in D_l$; such a triple can be used for introducing a^ib^i in only one position of a string aba^mb^m, namely in between a^m and b^m; there is no occurrence of a^ib^i to the right of that position, as requested by the definition of the relation $\Longrightarrow_{D_l}$.

Assertion (ii) can be proved in the same way. □

Lemma 3. $\{a^nb^n \mid n \geq 1\} \notin DUPL(lr0)$.

Proof. No derivation step is possible starting from a string of the form a^nb^n because no substring of such a string can be duplicated. □

In [1] one asks whether or not the context-freeness is preserved by the iterated duplication of types $\Longrightarrow_{D_l}, \Longrightarrow_{D_r}$ (in fact, in [1] one uses only one set D of rules, applied both "to the left" and to "to the right", in the sense of D_l, D_r, respectively). We prove below that the answer is negative: even finite languages are led to non-context-free languages by iteratively duplicating substrings of them. This is one of the main results of our paper.

Lemma 4. $DUPL(0) - CF \neq \emptyset$.

Proof. Let us consider the duplication grammar

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

where D_0 contains the following rules

$$\begin{aligned} r_1 &= (ca, ab, baabbaabb), \\ r_2 &= (aababba, ab, b), \\ r_3 &= (caa, b, abbaabab), \\ r_4 &= (caabb, a, bbaabab), \\ r_5 &= (bbaabb, a, bbaabab), \\ r_6 &= (aabbaa, b, abbaabab), \\ r_7 &= (aabbaa, b, abbc), \\ r_8 &= (bbaabb, a, bbc), \end{aligned}$$

and

$$A = \{caabbaabbaabbc\}.$$

Consider also the regular language

$$R = c(aabb)^+c,$$

and examine the intersection $L(\Delta) \cap R$.

Starting from a string of the form $c(aabb)^n c$ (initially, we have $n = 3$), the rules r_1, r_2 double the substrings ab, from left to right:

$$\begin{aligned} & caabbaabbaabb \ldots aabbc \\ \Longrightarrow_{D_0} \; & caababbaabbaabb \ldots aabbc \\ \Longrightarrow_{D_0} \; & caababbaababbaabb \ldots aabbc \\ & \ldots\ldots\ldots\ldots\ldots\ldots \\ \Longrightarrow_{D_0} \; & caababbaababbaababb \ldots aababbc. \end{aligned}$$

The rules $r_3 - r_8$ can be applied to a string obtained in this way in order to double all occurrences of a and b which are not double. This is also done from left to right:

$$\begin{aligned} & caababbaababbaababb \ldots aababbc \\ \Longrightarrow_{D_0} \; & caabbabbaababbaababb \ldots aababbc \\ \Longrightarrow_{D_0} \; & caabbaabbaababbaababb \ldots aababbc \\ & \ldots\ldots\ldots\ldots\ldots\ldots \\ \Longrightarrow_{D_0} \; & caabbaabbaabbaabbaabbaabb \ldots aabbaabbc. \end{aligned}$$

In this way, each substring $aabb$ is doubled: we pass from $aabb$ to $aababb$ and then to $aabbaabb$. Thus, the obtained string is $c(aabb)^{2n}c$. The process can be repeated.

In order to obtain a string in R, the operations of doubling the substrings ab and then of doubling the symbols a, b which are not appearing in substrings aa, bb, respectively, must be completed. Indeed, consider the case of a string of the form

$$w = caababb\ldots aa\underline{ba}bbaababbaabbaabb\ldots aabbc,$$

that is, obtained after some steps where rules r_1, r_2 were applied. Start now to use the other rules. The rules r_3, r_4, r_5, r_6 can be used from the left to the right and the symbols a, b not appearing in blocks aa, bb are doubled. Assume that we perform this operation the maximal possible number of times, that is we double also the underlined symbols in the string w; we obtain the string

$$w' = caabbaabb\ldots aabbaabbaababbaabbaabb\ldots aabbc.$$

No further application of rules r_5, r_6 is possible, hence no further occurrence of a, b can be doubled. We have first to continue with the rule r_2, doubling new occurrences of ab and making possible the identification of the right contexts of rules r_5, r_6 in the current string.

Symmetrically, consider a string obtained after some steps where symbols a, b were doubled:

$$z = caabbaabb\ldots aabbaabbaababbaababb\ldots aababbc.$$

From the left, we can start doubling substrings ab; this must be done step by step, but cannot proceed ahead of the doubling of the symbols a, b. For instance, we can obtain the string

$$z' = caababbaababb\ldots aababbaababba\underline{ab}abbaababb\ldots aababbc,$$

but we cannot go further, with the underlined substring ab. Again we have to continue with the previous operation of doubling (now, that of doubling the symbols a and b appearing separately).

Consequently, in order to get a string in R we have to perform complete "translations" of the string, that is doublings of the number of occurrences of the blocks $aabb$. This implies the equality

$$L(\Delta) \cap R = \{c(aabb)^{3\cdot 2^n}c \mid n \geq 0\}.$$

Clearly, this is not a context-free language, hence $L(\Delta)$ is non-context-free either.
□

The family of languages generated by matrix (programmed, controlled, etc) grammars with context-free rules (without using appearance checking) is closed under intersection with regular languages and under morphisms, [3]. Moreover, each one-letter matrix language is regular, [9]. This proves that the language $L(\Delta)$ in the previous proof is not a matrix one (with the morphism h defined by $h(a) = h(b) = h(c) = a$ we obtain $h(L(\Delta) \cap R) = \{a^{12\cdot 2^n + 2} \mid n \geq 0\}$, which is not regular). Thus, we can conclude that $DUPL(0)$ contains non-matrix languages.

Corollary 1. $DUPL(l) - CF \neq \emptyset$, $DUPL(r) - CF \neq \emptyset$.

Proof. Let $\Delta = (\{a,b,c\}, \emptyset, \emptyset, D_0, \{caabbaabbaabbc\})$ be the duplication grammar constructed in the previous proof and consider the following grammar:

$$\Delta_r = (\{a,b,c\}, \emptyset, D_r, \emptyset, \{abcaabbaabbaabbc\}),$$
$$D_r = \{(u_1v, v, u_2) \mid (u_1, v, u_2) \in D_0\}.$$

For the regular language $R_r = abc(aabb)^+c$ we obtain

$$L(\Delta_r) \cap R_r = \{ab\}(L(\Delta) \cap R).$$

(The rules in the set D_r of Δ_r lead to duplications identical to those controlled by the rules in the set D_0 of Δ; the string ab in the left end of the strings provides the necessary strings a, b, ab for duplications.) This proves that the language $L(\Delta_r)$ is not context-free.

A similar modification of Δ leads to a grammar Δ_l (with only the set D_l non-empty) generating a non-context-free language (we take the axiom $c(aabb)^3cab$ and $R_l = c(aabb)^+cab$). □

Clearly, if we allow the use of each triple (u_1, v, u_2) in the above grammars Δ_l, Δ_r in any mode $\Longrightarrow_{D_r}, \Longrightarrow_{D_l}$, then the generated language is not modified, hence the problem in [1] is answered: the iterated duplication does not preserve the context-freeness.

We combine these results in a synthesis theorem:

Theorem 1. (i) *The families $DUPL(l), DUPL(r)$ are incomparable. The families LIN, CF are incomparable with all families $DUPL(\alpha), \alpha \in \{l, r, 0, lr, l0, r0, lr0\}$; REG is incomparable with $DUPL(0)$.*

(ii) *All families $DUPL(\alpha), \alpha \in \{l, r, 0, lr, l0, r0, lr0\}$, are strictly included in CS.*

(iii) *The inclusions $DUPL(l) \subset DUPL(lr), DUPL(r) \subset DUPL(lr), DUPL(0) \subset DUPL(l0), DUPL(0) \subset DUPL(r0)$ are proper.*

It remains as an *open problem* to clarify the relationships between the families $DUPL(0), REG$ and the families $DUPL(\alpha)$ with $\alpha \neq 0$. Is REG included in $DUPL(\alpha)$? The next result provides a partial answer to this problem.

Theorem 2. *For every regular language L there is a string w such that $\{w\}L \in DUPL(r)$ and $L\{w\} \in DUPL(l)$.*

Proof. Let L be a regular language and let $M = (K, V, q_0, F, \delta)$ be the minimal deterministic finite automaton recognizing L (K is the set of states, V is the alphabet, q_0 is the initial state, F is the set of final states, and $\delta : K \times V \longrightarrow K$ is the transition mapping).

For each $x \in V^*$, we define the mapping $\rho_x : K \longrightarrow K$ by

$$\rho_x(q) = q' \text{ iff } \delta(q, x) = q', \ q \in K.$$

Obviously, if $x_1, x_2 \in V^*$ are such that $\rho_{x_1} = \rho_{x_2}$, then for every $u, v \in V^*$, ux_1v is in L if and only if ux_2v is in L.

The set of mappings from K to K is finite. Hence the set of mappings ρ_x as above is finite. Let n_0 be their number. We construct the duplication grammar $\Delta_r = (V \cup \{c, d\}, \emptyset, D_r, \emptyset, A)$, where c, d are two symbols not in V,

$$\begin{aligned} D_r \; = \; & \{(zu_1, v, \lambda) \mid \rho_{u_1} = \rho_{u_1 v}, \text{ for } u_1, v \in V^*, |u_1|, |v| \le n_0, \\ & \text{and either } z = dz', z' \in V^*, |z'u_1| \le n_0, \text{ or } z \in V^*, |zu_1| = n_0 + 1\}, \end{aligned}$$

and A is constructed as follows. Take all strings $x \in V^*$ of length at most n_0, concatenate each of them with c at both ends, then concatenate all the obtained strings, in any given order; denote by w' the obtained string and consider $w = w'd$. Then,

$$A = \{wx \mid x \in L, |x| \le n_0 + 1\}.$$

Therefore, the string w provides substrings v for duplication, as requested by the rules in D_r. These rules cannot be applied to the substrings of w, because of the presence of symbols c: the left context of each rule in D_r is of a length greater than n_0, or it begins with d, hence this context cannot be found in w.

From the definition of mappings ρ_x and the definitions of A and D_r, it follows immediately that $L(\Delta) \subseteq \{w\}L$.

Assume that the converse inclusion is not true and let $wu \in \{w\}L - L(\Delta)$ be a string of minimal length with this property. Thus $wu \notin A$. Hence $|u| \ge n_0 + 2$. Let $u = zz'$ with $|z'| = n_0$ and $z \in V^*$. If $z' = a_1 a_2 \dots a_{n_0}$, then it has $n_0 + 1$ prefixes, namely $\lambda, a_1, a_1a_2, \dots, a_1 \dots a_{n_0}$. There are only n_0 different mappings ρ_x. Therefore there are two prefixes u_1, u_2 of z' such that $u_1 \ne u_2$ and $\rho_{u_1} = \rho_{u_2}$. With no loss in generality we may assume that $|u_1| < |u_2|$. By substituting u_2 by u_1 we obtain a string u' which is also in L. As $|u'| < |u|$ and wu was of minimal length in $\{w\}L - L(\Delta)$, we obtain $wu' \in L(\Delta)$. However, $|u_2| - |u_1| \le |u_2| \le n_0$, so if $u_2 = u_1 v$, then $(z_1 u_1, v, \lambda) \in D_r$, where either $z_1 = z$ and begins by d, or z_1 is a suffix of z such that $|zu_1| = n_0 + 1$. This implies that $wu' \Longrightarrow_{D_r} wu$, that is $wu \in L(\Delta)$, a contradiction. In conclusion, $\{w\}L \subseteq L(\Delta)$.

In the same way we can prove that $L\{dw'\} \in DUPL(l)$. □

Corollary 2. *Each regular language L can be written as the left derivative of a language in $DUPL(r)$ or as the right derivative of a language in $DUPL(l)$.*

Proof. With the notations in the previous proof, we have $L = \partial^l_w(L(\Delta)) = \partial^r_w(L(\Delta'))$, where Δ is the duplication grammar constructed above and Δ' is its version for the $DUPL(l)$ case. □

If we also use a projection (a morphism which erases certain symbols and leaves the other symbols unchanged), then a representation result like that in this corollary can be obtained for linear languages, too.

Theorem 3. *For each linear language L, there is a string w, a language $L' \in DUPL(r)$, and a language $L'' \in DUPL(l)$ such that $L = h(\partial^l_w(L')) = h(\partial^r_w(L''))$.*

Proof. Consider a linear grammar, $G = (N, T, S, P)$ and two new symbols, c, d. Each rule $X \to x$ in P is replaced by $X \to cxc$ (in this way, all terminal strings appearing in the right hand side of rules are non-empty and each right-hand member of a rule is bounded by c). Denote by P' the set of rules obtained in this way. For each rule $X \to uYv \in P'$ we consider the string $uYYv$. Let w' be the string obtained by concatenating these strings, for all rules in P', then concatenating also the strings appearing in the right hand side of terminal rules in P'. Moreover, each string is considered only once, even if two rules have the same right hand side. Then, the string w we look for is $w = w'd$.

We
now construct the duplication grammar $\Delta = (N \cup T \cup \{c, d\}, \emptyset, D_r, \emptyset, \{wSS\})$, with

$$\begin{aligned} D_r &= \{(X, uYYv, X) \mid X \to uYv \in P', \text{ nonterminal rule}\} \\ &\cup \{(X, x, X) \mid X \to x \in P', \text{ terminal rule}\}. \end{aligned}$$

From the previous construction, one can easily see that $L(\Delta)$ consists of strings of the form

$$w'dSu_1X_1u_2 \dots X_{n-1}u_nX_nxX_nv_nX_{n-1} \dots v_2X_1v_1S,$$

with $n \geq 1, X_i \in N, u_i \in \{c\}T^*, v_i \in T^*\{c\}$, for all i, and $x \in \{c\}T^*\{c\}$, such that $S \to u_1X_1v_1, X_i \to u_{i+1}X_{i+1}v_{i+1}$, for $1 \leq i \leq n-1$, and $X_n \to x$ are rules in P'. (For each derivation step we can find the string to be inserted as a substring of w. Moreover, no duplication rule can be applied for inserting a string in w, because of the presence of symbols c bounding the substrings to be duplicated and because of the fact that such substrings appear only once in w.) Therefore, the string $u_1u_2 \dots u_nxv_n \dots v_2v_1$ can be generated by using the rules in P'.

With the projection morphism $h : (N \cup T \cup \{c, d\})^* \longrightarrow T^*$ defind by $h(a) = a$ for $a \in T$, and $h(b) = \lambda$ for $a \notin T$, we obtain $L = h(\partial^l_w(L(\Delta))$.

The case of $DUPL(l)$ can be treated in the same way. □

5 Decidability and Complexity

The fact that the family CF is incomparable with all families $DUPL(\alpha)$ makes interesting several decidability questions. We solve here only one of them, the others remain open. Two examples: Is the regularity or the context-freeness of duplication languages decidable ? Given a regular language, can we decide whether or not it is in a family $DUPL(\alpha)$?

Theorem 4. *It is not decidable whether or not a context-free language is in a family $DUPL(\alpha)$, for any $\alpha \in \{l, r, 0, lr, l0, r0, lr0\}$.*

Proof. Let G be an arbitrary context-free grammar with the terminal alphabet $\{a, b\}$ and construct the language

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

If $L(G) = \{a,b\}^*$, then $L = \{a,b\}^*\{c,d\}^*$, therefore $L \in DUPL(\alpha), \alpha \in \{l,r,0\}$ (this can be easily seen for $\alpha \in \{l,r\}$ – see also the proof of Lemma 1 – and can be proved for $\alpha = 0$ by following the idea used in Remark 1).

If $L(G) \neq \{a,b\}^*$, then take a string $w \in \{a,b\}^* - L(G)$ and consider all strings of the form $wc^nd^n, n \geq 1$. They are in $\{a,b\}^*\{c^id^i \mid i \geq 1\}$, hence in L. Assume that, in these circumstances, $L = L(\Delta)$, for some duplication grammar $\Delta = (\{a,b,c,d\}, D_l, D_r, D_0, A)$. Consider a derivation step $z_1z_2 \Longrightarrow wc^nd^n$, where $z_1 \in \{a,b\}^*, z_2 \in \{c,d\}^*$, and the applied rule introduces a string in z_2. That is, $z_1 = w$ and $z_2 \Longrightarrow c^nd^n$. There are such derivation steps, with $z_2 \neq c^nd^n$, because A is finite and the set of strings as above is infinite. However, no string z_2 with this property can exist: we must have $z_2 = c^md^p$ with one of m,p equal to n and the other one strictly smaller than n, and such a string wz_2 is not in L (on the one hand, $w \notin L(G)$, on the other hand, $c^md^p \notin \{c^id^i \mid i \geq 1\}$).

It follows that $L \notin DUPL(\alpha)$, for all values of α.

Consequently, $L \in DUPL(\alpha), \alpha \in \{l,r,0,lr,l0,r0,lr0\}$, if and only if $L(G) = \{a,b\}^*$, which is undecidable. □

The complexity of a duplication grammar can be estimated from several points of view. We consider here some of them.

For a duplication grammar $\Delta = (V, D_l, D_r, D_0, A)$ we denote

$$\begin{aligned}
ax(\Delta) &= card(A),\\
axm(\Delta) &= \max\{|x| \mid x \in A\},\\
rul(\Delta) &= card(D_l) + card(D_r) + card(D_0),\\
ins(\Delta) &= \max\{|v| \mid (u_1,v,u_2) \in D_l \cup D_r \cup D_0\},\\
rad(\Delta) &= \max\{|u| \mid (u_1,v,u_2) \in D_l \cup D_r \cup D_0, u = u_1 \text{ or } u = u_2\}.
\end{aligned}$$

(These parameters count the number of axioms, the maximum length of an axiom, the number of rules, the maximum length of a string to be inserted, the maximum length of a context string – the *radius* –, respectively.)

For a language L in a family $DUPL(\alpha)$ and a measure $M \in \{ax, axm, rul, ins, rad\}$ we define

$$M_\alpha(L) = \min\{M(\Delta) \mid L = L(\Delta), \Delta \text{ is of type } \alpha\},$$

where $\alpha \in \{l,r,0,lr,l0,r0,lr0\}$.

As it is expected (as it is customary in the descriptional complexity area, see [8]), each of these parameters defines an infinite hierarchy of languages, for each α.

Theorem 5. *For each measure $M \in \{ax, axm, rul, ins, rad\}$, for each $n \geq 0$, and for each α, there is a language L_n such that $M_\alpha(L_n) = n$.*

Proof. For $n \geq 0$, consider the languages

$$L_{ax} = L_{axm} = \{a, a^2, \ldots, a^n\},$$

$$\begin{aligned} L_{rul} &= \bigcup_{i=1}^{n} (ab^i a)(ab^i a)^+, \\ L_{ins} &= a^n(a^n)^+, \\ L_{rad} &= \{a^{2n+1}\} \cup \{a^{2(n+1)+2i} \mid i \geq 0\}. \end{aligned}$$

Clearly, no rule can be applied to a string in the first language (otherwise, an infinite language is produced), hence each string must be introduced as an axiom. Thus, we need n axioms, the longest one being a^n.

In order to generate the second language, we need rules containing as strings to be inserted strings of the form $(ab^i a)^j$ with $j \geq 1$, for each $i = 1, 2, \ldots, n$ (we cannot modify a block b^i in a string $(ab^i a)(ab^i a)^r, r \geq 1$, hence we can only introduce new blocks $ab^i a$). That is, we need at least n duplication rules.

In the case of the third language it is also clear that the inserted strings must be of the form $a^{nk}, k \geq 1$, that is, at least n symbols must be simultaneously inserted.

Grammars with n rules and with a string of length n to be inserted, respectively, can generate these languages.

For the fourth language, starting from the axioms

$$A = \{a^{2n+1}, a^{2(n+1)}, a^{2(n+1)+2}\},$$

and using the rule (a^{n+1}, a^2, a^{n+1}) in any mode l, r, or 0, we can generate this language. However, we cannot generate this language by using rules (a^m, a^p, a^q) with $m, q \leq n$: any such a rule must have p even, $p = 2k, k \geq 1$; applying it to the string a^{2n+1} we produce the string $a^{2n+1+2k}$, which is not in L_{rad}, a contradiction.

These remarks are valid for all variants of duplication grammars. Consequently, for each M we have $M_\alpha(L_M) = n$. □

A natural problem concerns the closure properties of families $DUPL(\alpha)$. We do not consider it here, but we only point out that the families $DUPL(l), DUPL(r)$ are not closed under mirror image – a consequence of Lemma 2.

6 Final Remarks

We have considered here duplication operations suggested by similar operations met in linguistics and in the genetic area. Some other variants can be defined, for instance, with a transformation applied to the copy of the duplicated string (point mutations, reversal, etc). Also variants of applying the replication rules can be of interest: leftmost use of rules, parallel application, priority relations among rules and so on. The area deserves further investigations, expecially for those variants of the replication operation which are met in DNA transformation/evolution.

In general, the operations on strings inspired from biochemistry were successfully extended to various bi- or multi-dimensional structures, such as trees, graphs in general, arrays. (The reader is referred to [4], [6], [7] for modern accounts on these areas.) This happens, for instance, with the splicing operation ([10]) and it is probably true also for the duplication operations considered here. The case of trees is particularly important, because the DNA molecules are known to often take a branching structure.

References

[1] J. Dassow, V. Mitrana, On some operations suggested by the genome evolution, *Proc. of Pacific Symp. on Biocomputing*, Hawaii 97 (R. Altmann, et al, eds.), World Scientific, Singapore, 1997, 97 – 108.

[2] J. Dassow, V. Mitrana, A. Salomaa, Context-free evolutionary grammars and the structural language of nucleic acids, *Bio Systems*, 43 (1997), 169 – 177.

[3] J. Dassow, Gh. Păun, *Regulated Rewriting in Formal Language Theory*, Springer-Verlag, Berlin, Heidelberg, 1989.

[4] J. Engelfriet, Context-free graph grammars, chapter 3 in vol. 3 of [16], 125 – 213.

[5] B. S. Galiukschov, Semicontextual grammars (in Russian), *Mat. logica i mat. ling.*, Talinin Univ., 1981, 38 – 50.

[6] F. Gécseg, M. Steinby, Tree languages, chapter 1 in vol. 3 of [16], 1 – 68.

[7] D. Giammarresi, A. Restivo, Two-dimensional languages, chapter 4 in vol. 3 of [16], 215 – 267.

[8] J. Gruska, Descriptional complexity of context-free languages, *Proc. Math. Found. Computer Sci. Conf.*, High Tatras, 1973, 71 – 83.

[9] D. Hauschild, M. Jantzen, Petri nets algorithms in the theory of matrix grammars, *Acta Inform.*, 31 (1994), 719 – 728.

[10] T. Head, Gh. Păun, D. Pixton, Language theory and molecular genetics. Generative mechanisms suggested by DNA recombination, chapter 7 in vol. 2 of [16], 295 – 360.

[11] L. Hunter, Molecular biology for computer scientists, in *Artificial Intelligence and Molecular Biology* (L. Hunter, ed.), AAAI Press/The MIT Press, Menlo Park, CA, 1993, 1 – 46.

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

[13] Gh. Păun, *Marcus Contextual Grammars*, Kluwer Academic Publ., Boston, Dordrecht, 1997.

[14] Gh. Păun, G. Rozenberg, A. Salomaa, *DNA Computing. New Computing Paradigms*, Springer-Verlag, Heidelberg, 1998.

[15] W. C. Rounds, A. Manaster Ramer, 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.

[16] G. Rozenberg, A. Salomaa, eds., *Handbook of Formal Languages*, 3 volumes, Springer-Verlag, Berlin, Heidelberg, 1997.

[17] A. Salomaa, *Jewels of Formal Languages*, Computer Science Press, Rockville, 1981.

[18] D. B. Searls, The linguistics of DNA, *American Scientist*, 80 (1992), 579 - 591.

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