On compensation loops in genomic duplications

José M. Sempere
Departamento de Sistemas Informáticos y Computación, Universitat Politècnica de València,
Camino de Vera s/n 46022 Valencia, Spain
jsempere@dsic.upv.es

In this paper we investigate the compensation loops, a DNA rearrangement in chromosomes due to unequal crossing over. We study the effect of compensation loops over the gene duplication, and we formalize it as a restricted case of gene duplication in general. We study this biological process under the point of view of formal languages, and we provide some results about the languages defined in this way.

Keywords: DNA rearrangement, duplication over strings, formal languages, regular expressions.

1. Introduction

In the last years, it has been a common place for biology and computer science the study of genomic processes in a mathematical and algorithmic way. Within that approach, the formal language theory has been a fruitful framework to study different genomic processes. We can refer to [2, 3, 4, 14, 16] for discussions on different formalizations of the language of nucleic acids and related operations.

In this work, we will study a restricted way of *duplication*, that is the presence of any part of genetic material, a single locus or a large piece of a chromosome, more than once in the genome. As it is described in [8], duplications can arise as the result of an error during genetic exchanges (referred to as unequal crossing over) between synapsed chromosomes during meiosis, or through a replication error prior to meiosis. A model of duplication has been considered in [5]. 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 somewhere within it, say 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. This idea is schematically presented in the Figure 1.

Initially, Ehrenfeucht and Rozenberg [7] and Bovet and Varicchio [1] proposed different results on *copying systems* that can be considered pioneering results related to the following works on duplication languages considered here. Martín-Vide and Păun introduced in [12] a generative mechanism (similar to the one considered in [3]) 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,

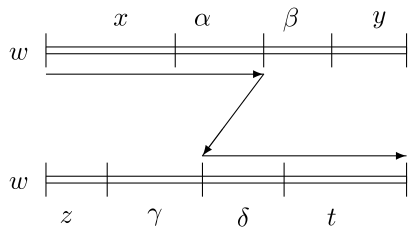

Fig. 1. A scheme for gene duplication

according to a finite set of duplication rules. This mechanism is studied in [12, 13] from the generative power point of view.

In [6] one considers a string and constructs the language obtained by iteratively duplicating any of its substrings. It has been proved that when starting from strings over two-letter alphabets, the obtained languages are regular; an answer for the case of arbitrary alphabets is given in [17], where it is proved that each string over a three-letter alphabet generates a non-regular language by duplication. Finally, in [11] it is studied several results concerning to bounded and unbounded duplication languages. The bounded duplication (i.e., the duplication where a predefined maximum length in the duplicated segment is not exceeded), is studied in [9, 10].

In this work, we will consider duplication with (dynamic) compensation loops. that is a DNA rearrangement to compensate erroneous crossing over, and we will study the effect of this process over the duplication operations, in a formal language setting. This work has the following structure: in Section 2, we provide basic definitions on formal language theory, specially we provide the formal definition of duplication operation over strings and languages. In Section 3, we formalize the concept of (static) compensation loops, and we redefine the duplication operation by introducing a variant of the shuffle operation over strings. We provide some results of this operation that relate it to the duplication languages. In Section 4, we introduce a new variant of duplication with compensation loops that models the DNA rearrangement in a dynamic way. As in Section 3, we provide different results that relate the languages obtained in this way with the languages previously referred. Finally, in Section 5, we provide some conclusions and we enumerate some open problems related to this work.

2. Basic concepts and duplication languages

We provide basic definitions and notations about formal language theory from [15]. An *alphabet* is any finite and non empty set of elements, named *symbols*. Given an alphabet V, a *string* over V is any finite and ordered sequence of symbols. The length of a string x is denoted by $|x|$, the empty string is denoted by ε (with $|\varepsilon| = 0$). Given two strings x and y, the product or concatenation of x and y is the sequence x followed by the sequence y, and it will be denoted by xy. For sets X and Y, $X \setminus Y$ denotes the set-theoretic difference of X and Y. If X is a finite set, then $card(X)$ denotes its cardinality. The set of all strings over V is denoted by V^*, and $V^+ = V^* \setminus \{\varepsilon\}$. A language over V is any subset of V^*. For any pair of languages L_1 and L_2, the product L_1L_2 is defined as the set $\{xy : x \in L_1, y \in L_2\}$. Given two alphabets V and W, a *homomorphism* is a mapping from V to W^* that can be easily extended over strings as follows: for every $x = x_1x_2 \cdots x_n \in V^+$, $h(x) = h(x_1)h(x_2) \cdots h(x_n)$, and $h(\varepsilon) = \varepsilon$.

For any alphabet V that does not contain the parentheses symbols) and (, we define *regular expressions* as follows:

(1) For every $a \in V$, a, ε and $\emptyset$ are regular expressions that denote the languages $\{a\}$, $\{\varepsilon\}$ and the empty set, respectively.
(2) If r is a regular expression, then so is (r). Both regular expressions define a language denoted by $L(r)$.
(3) For the regular expressions r and s, the following are regular expressions:
   (a) $r + s$ that denotes the language $L(r) \cup L(s)$.
   (b) rs that denotes the language $L(r)L(s)$.
   (c) r^* that denotes the language $(L(r))^*$.

Now we define the set of strings that can be obtained by duplication. Let V be an alphabet. For a string $w \in V^+$, we set

$$\begin{aligned} D(w) &= \{uxxv \mid w = uxv, u, x, v \in V^*\}, \\ D^0(w) &= w, \\ D^i(w) &= \bigcup_{x \in D^{i-1}(w)} D(x),\ i \geq 1, \\ D^*(w) &= \bigcup_{i \geq 0} D^i(w). \end{aligned}$$

From the previous definition, we can affirm that $D^*(w)$ is the smallest language $L' \subseteq V^*$ such that $w \in L'$ and whenever $uxv \in L'$, $uxxv \in L'$ holds for all $u, x, v \in V^*$. Any language $D^*(w)$ will be referred as a *duplication language*.

It has been proved that for any string w, $D^*(w)$ is regular iff w contains at most two different symbols [6, 17].

3. Gene duplication with compensation loops

As referred before, the duplication language of a string defined over an arbitrary alphabet is not regular [17]. In contrast, the duplication language of a string over a two-letter alphabet is regular [6]. Here, we consider a case where duplication closure over arbitrary alphabets is regular: gene duplication with *compensation loops*.

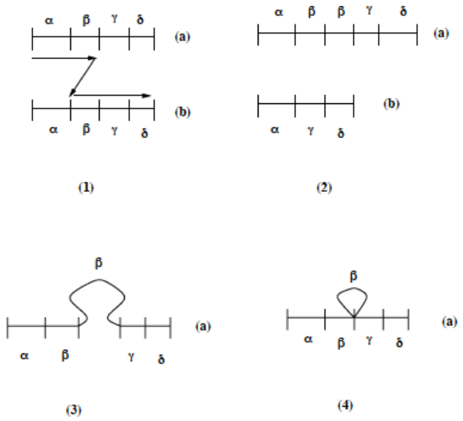

Fig. 2. A schematic illustration of the formation of a compensation loop during duplication.

Gene duplication with compensation loops can be described as follows according to [8]. We illustrate this process in Figure 2. Let us consider two pairs of homologous chromatids composed by a sequence of genes such as $\alpha\beta\gamma\delta$. Then, after a gene duplication, we obtain two different chromatids defined by the sequences $\alpha\beta\beta\gamma\delta$ (chromatid 2.(a)) and $\alpha\gamma\delta$ (chromatid 2.(b)). We center our attention to chromatid 2.(a). Here, a *compensation loop* can be arranged: chromatid 2.(a) makes a fold in gene β in order to recover its initial alignment with the original sequence (Figure 2 (3) and (4)). Then, if other duplication succeeds, it will be again on the original chromatid $\alpha\beta\gamma\delta$. This situation can be summarized by saying that duplication closure only takes into account the original sequence to repeat genes.

Obviously, the way in which duplication with compensation loops holds is a restrictive way of general duplication. We formalize this kind of duplication with an operation over strings and languages in a way similar to the general case.

First, let us consider an alphabet V such that the symbols $[,] \notin V$ (V does not contain bracket symbols). We define the *shuffle with common segments* operation over strings (in short *scs*) as follows: Let $w = x_1[w_1]x_2[w_2]\cdots x_n[w_n]$ and $z = x_1[z_1]x_2[z_2]\cdots x_n[z_n]$ with $x_1, \cdots, x_n \in V^*$, and for every integer value i, $1 \leq i \leq n$, $w_i, z_i \in V^*$. Then, $scs(w,z) = x_1[w_1 z_1]x_2[w_2 z_2]\cdots x_n[w_n z_n]$. Observe that, if w and z do not have the substrings $x_1, x_2, \cdots x_n$ then the operation is undefined. Basically, the substrings inside any bracket pair denote the gene folds in every chromatid, and $scs(w,z)$ denotes how a pair of sister chromatids with compensation loops can be recombined with equal crossover. In the rest of this work, if w_i or z_i equals to ε then the bracket symbols for them will be removed. The substrings $x_1, x_2, \ldots, x_n$ will be named *common segments.*

We define the way in which any combination of duplicated strings with compensation loops is carried out, as follows:

For a string $w \in V^+$, we set

$$
\begin{aligned}
D^0_{cl}(w) &= \{w\},\\
D^1_{cl}(w) &= \{ux[x]v \mid w = uxv, u, x, v \in V^*\},\\
D^i_{cl}(w) &= \{scs(x, y) \mid x, y \in D^{i-1}_{cl}(w)\},\ i \geq 2,\\
D^*_{cl}(w) &= \bigcup_{i \geq 0} D^i_{cl}(w).
\end{aligned}
$$

The strings in $D^*_{cl}(w)$ contain bracket symbols, with the exception of w. We can define a homomorphism that removes them from the string in the following way: for every symbol $a \in V$ $h(a) = a$ and $h(\ [\ ) = h(\ ]\ ) = \varepsilon$.

Example 1. *Let $w = abc$. Then we have*

$$D^1_{cl}(abc) = \{abc, a[a]bc, ab[b]c, abc[c], ab[ab]c, abc[bc], abc[abc]\}$$

We can obtain $scs(a[a]bc, ab[ab]c) = a[a]b[ab]c \in D^2_{cl}(abc)$ given that the string $a[a]bc$ equals to $a[a]b[\ ]c[\ ]$, and the string $ab[ab]c$ equals to $a[\ ]b[ab]c[\ ]$.

Now, we characterize the language class that defines the duplication with compensation loops.

Property 1. *For any arbitrary alphabet V and any string $w \in V^+$, $h(D^*_{cl}(w))$ is regular.*

Proof.

First, let $w = w_1 w_2 \cdots w_n$ and $w_i \in V$. We propose a regular expression r that denotes $h(D^*_{cl}(w))$. Let r be defined as follows

$$r = w_1(w_1)^* w_2(w_2 + w_1 w_2)^* w_3(w_3 + w_2 w_3 + w_1 w_2 w_3)^* \cdots w_n(w_n + \cdots + w_1 w_2 \cdots w_n)^*$$

We will prove that r denotes $h(D^*_{cl}(w))$. Obviously, w belongs to $L(r)$. Also, any string in $h(D^1_{cl}(w))$ belongs to $L(r)$ given that such string is defined as $w_1 w_2 \cdots w_{j-1}$ $w_j \cdots w_{j+k} w_j \cdots w_{j+k} w_{j+k+1} \cdots w_n$ which clearly belongs to $L(r)$.

Let us suppose that for any string $y \in D^{i-1}_{cl}(w)$, $h(y)$ belongs to $L(r)$.

Now, we take two strings x and y that belong to $D^{i-1}_{cl}(w)$, with $i \geq 2$. We take $z = scp(x, y)$ with $x, y \in D^{i-1}_{cl}(w)$. In such case, $x = w_1[x_1]w_2[x_2] \cdots\ w_n[x_n]$ and $y = w_1[y_1]w_2[y_2] \cdots w_n[y_n]$ with $x_i, y_i \in (\{w_j \cdots w_i : 1 \leq j \leq i\})^*$, given that $h(x)$ and $h(y)$ belong to $L(r)$. Then, $z = scs(x, y) = w_1[z_1]w_2[z_2] \cdots w_n[z_n]$ with $z_i = x_i y_i$ which again belongs to $(\{w_j \cdots w_i : 1 \leq j \leq i\})^*$ and, clearly, $h(z)$ belongs to r.

On the other hand, if we take any string $y \in r$, we can prove that there exists a string $z \in D^*_{cl}(w)$ such that $h(z) = y$. The proof follows from the previous induction process that we have carried out. □

Observe that duplication with compensation loops is a restrictive way of general duplication. In that case, any substring inside a compensation loop is blocked. It means that no substring into a compensation loop can be recombined to form a new duplication substring. Anyway, in some cases, this restriction does not make the duplication language smaller than in the general case. We enunciate the following property that relates general duplication with duplication with compensation loops.

Property 2. *The following two statements are true*

1. *For any alphabet V with $card(V) = 1$ and $w \in V^+$, $h(D^*_{cl}(w)) = D^*(w)$*
2. *For any alphabet V with $card(V) \geq 2$ there exists $w \in V^+$ such that $h(D^*_{cl}(w)) \subsetneq D^*(w)$.*

Proof.

First let us take the alphabet $V = \{a\}$. Then, for any string in V^+ such as a^n we have that $D^*(a^n) = a^n a^*$, and $h(D^*_{cl}(a^n)) = a^{(1)}(a^*)a^{(2)}(a + aa)^* \cdots a^{(n)}(a + aa + \cdots + a^n)^*$ which is equivalent to $a^n a^*$ (here $a^{(i)}$ denotes the ith a symbol of a^n). So, the first statement of the property is true.

For the second statement, we will prove that $h(D^*_{cl}(w)) \neq D^*(w)$ (clearly, $h(D^*_{cl}(w)) \subset D^*(w)$). Let us take $V = \{a, b\}$ and $w = aba$. We can verify that $abaabbaaba$ belongs to $D^*(w)$ (it is duplicated as $\underline{aba}$, then $a\underline{baab}a$ to obtain $abaabbaaba$). The expression for $h(D^*_{cl}(w))$, according to Property 1, is $a(a)^*b(b + ab)^*a(a + ba + aba)^*$ which clearly does not contain $abaabbaaba$, so $h(D^*_{cl}(w)) \neq D^*(w)$ and $h(D^*_{cl}(w)) \subsetneq D^*(w)$. □

4. Gene duplication with dynamic compensation loops

Once we have proved that duplication with compensation loops is regular, we will study a situation different from the previous one. Observe that, while studying duplication with compensation loops we have assumed that the gene folds occur always in the original positions, (that was the reason why the original alignment is preserved, and we were able to propose a regular expression in the proof of Property 1). Now, we propose a different way of making compensation loops. We consider that the original sequence is always preserved but the gene folds occur with respect initial and non initial gene sequences. Figure 3 illustrates this idea: First, we consider the original sequence (Figure 3.(1)) and a compensation loop is performed by duplication (Figure 3.(2)). Then, the gene fold is considered in a position different from the initial one (Figure 3.(3) and 3.(4)) and a new compensation loop holds by applying duplication (Figure 3.(5)). Again, the initial gene folds are considered (Figure 3.(6) and 3.(7)) and we can obtain a final configuration with an unique gene fold (Figure 3.8).

Here we have considered that the original sequence is always recovered and the gene folds occur in arbitrary ways by moving along the original sequence. That is the reason why we call *dynamic compensation loops* to these folds.

Now we proceed to formalize the duplication with dynamic compensation loops in a mathematical setting as we made in the previous section. First, we consider

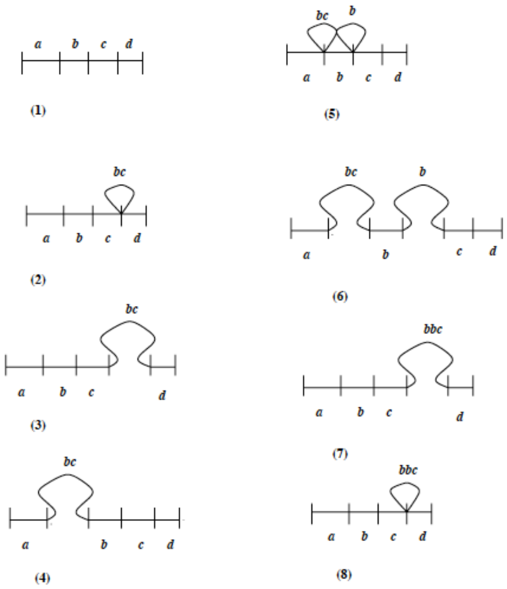

Fig. 3. Making dynamic compensation loops.

again that V is an alphabet that does not contain bracket symbols. We take two arbitrary strings $w = [w_0]x_1[w_1]x_2[w_2]\cdots x_n[w_n]$ and $z = [z_0]x_1[z_1]x_2[z_2]\cdots x_n[z_n]$ with $w_i, z_i \in V^*$ for $0 \leq i \leq n$, and $x_j \in V^*$ for $1 \leq j \leq n$. Then, we define the *generalized shuffle with common segments* (or *gscs* in short) as follows:

$$gscs(w, z) = \{[\eta_0]x_1[\eta_1]\cdots x_n[\eta_n] \mid \eta_0 x_1 \eta_1 x_2 \eta_2 \cdots x_n \eta_n = w_0 z_0 x_1 w_1 z_1 \cdots x_n w_n z_n\}$$

As it was established in Section 3, if w_i or z_i equals to ε then their corresponding bracket symbols can be removed.

We define the duplication language with dynamic compensation loops as follows: For a string $w \in V^+$, we set

$$\begin{aligned}
D^0_{dcl}(w) &= \{w\},\\
D^1_{dcl}(w) &= \{ux[x]v \mid w = uxv, u, x, v \in V^*\},\\
D^i_{dcl}(w) &= \bigcup_{x,y \in D^{i-1}_{dcl}(w)} gscs(x, y) \quad i \geq 2,\\
D^*_{dcl}(w) &= \bigcup_{i \geq 0} D^i_{dcl}(w).
\end{aligned}$$

Again, we will use the homomorphism h defined in the previous section in order to remove the bracket symbols. Observe that, if $x, y \in gscs(u, v)$ then $h(x) = h(y)$.

Example 2. *Let* $w = ab[ab]b[bb]c[bc]$ *and* $z = a[a]bbc[cc]$*, then* $gscs(w, z)$ *contains, among others, the following strings:*

$$x_1 = a[a]b[ab]b[bb]c[bccc], x_2 = [a]a[bab]bb[bcbcc]c, \text{ and } x_3 = a[aba]b[bbbc]b[cc]c$$

Then, $h(x_1) = h(x_2) = h(x_3) = aababbbbcbccc$.

We will characterize the language $D^*_{dcl}(w)$ with respect to the language $D^*_{cl}(w)$ through the following statement.

Property 3. *Let* V *be an arbitrary alphabet with at least two symbols. Then there exists* $w \in V^+$ *such that* $h(D^*_{cl}(w)) \subsetneq h(D^*_{dcl}(w))$.

Proof.

Obviously, $h(D^*_{cl}(w)) \subset h(D^*_{dcl}(w))$. We will prove that the equality between languages is not fulfilled. First, let us take $V = \{a, b\}$ and $w = ab$. We can obtain strings in $h(D^*_{dcl}(w))$ that cannot be obtained in $h(D^*_{cl}(w))$. According to Property 1, $h(D^*_{cl}(ab)) = a(a)^*b(b + ab)^*$. Then, for example, $aabaabab \notin h(D^*_{cl}(ab))$, and $aabaabab \in h(D^*_{dcl}(ab))$. Observe that $ab[ab]$ and $a[a]b$ belong to $D^1_{dcl}(ab)$. Hence, $a[a]b[ab] \in gscs(a[a]b, ab[ab])$, and, subsequently, $a[aba]b \in gscs(a[a]b, ab[ab])$, and $a[aba]b \in D^2_{dcl}(ab)$. In the following iteration, we have that $a[abaaba]b \in gsc(a[aba]b, a[aba]b)$, so $a[abaaba]b \in D^3_{dcl}(ab)$ and, subsequently, $aabaabab \in h(D^*_{dcl}(ab))$. □

Finally, we provide the following property that relates the duplication language with dynamic compensation loops with general duplication language.

Property 4. *Let* V *be an arbitrary alphabet with at least three symbols. Then there exist* $w \in V^+$ *such that* $h(D^*_{dcl}(w)) \subsetneq D^*(w)$

Proof.

Obviously, for every $w \in V^+$ $h(D^*_{dcl}(w))$ is a subset of $D^*(w)$, given that the duplication with dynamic compensation loops is a restricted case of duplication.

Let us take $w = abc$ and the string $abcacabc \in D^*(abc)$ that can be obtained by duplicating the underlined segments as $\underline{abc}$ and $ab\underline{ca}bc$. Now we will try to obtain $abcacabc$ by duplication with dynamic compensation loops from abc. First we can observe that $abc[abc], abc[c] \in D^1_{dcl}(abc)$. Then, by fixing the common segments to a, b and c, we have $ab[cab]c, ab[c]c \in D^1_{dcl}(abc)$. We can obtain $ab[ccab]c$ from $gscs(ab[c]c, ab[cab]c)$. Hence, $ab[ccab]c \in D^2_{dcl}(abc)$. It can be observed that a symbol a cannot be inserted between the two consecutive symbols c and, subsequently, $abcacabc \notin D^*_{dcl}(abc)$. A dual case can be analyzed by obtaining the string $abcaabc$ from $gscs([abc]abc, [a]abc)$, but in this case, a symbol c cannot be inserted between the two consecutive symbols a. The rest of combinations leads to similar situations as the described above. □

5. Conclusions and future works

We have proposed two new operations over strings inspired by a restricted case of genome duplication. Our proposal is inspired by biological processes that happens in nature. Hence, it is an approximation to systems biology from a formal language point of view.

This work leaves two main open questions that should be explored and answered:

(1) For any string $w \in V^+$, what is the language class for $h(D^*_{dcl}(w))$? (i.e. is it regular, context-free, ... ?)
(2) For any string w with only two different symbols, is $h(D^*_{dcl}(w)) = D^*(w)$?

We think that the first question should be investigated in detail given that the proof proposed in [17] does not seem to work in this case. In addition, our intuition is that the answer for the second question should be affirmative, given that the reasoning in Property 4 proof does not work with an alphabet with only two symbols.

Another field of study that should be explored is the case of considering not only a word and its duplication languages, as in this work, but a set of words, that is, a language, and all the combinations of duplication between its elements. Subsequently, the generalization to families of languages should be the logical step to complete a more detailed study of the languages of duplication proposed in this work.

Acknowledgements

The author really appreciates the friendship of Victor Mitrana for so many years. He is also indebted to him for his generosity and enthusiasm during all these years, sharing and discussing scientific topics such as those that the author has tried to capture in this work.

References

[1] D.P. Bovet, S. Varricchio, On the regulatrity of languages on a binary alphabet generated by copying systems. *Inf. Process. Lett.* 44(3) (1992), pp 119-123.

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

[3] J. Dassow, V. Mitrana, Evolutionary grammars: a grammatical model for genome evolution. *Proceedings of the German Conference in Bioinformatics GCB'96*, eds. R. Hofestädt, T. Lengauer, M. Löffler, D. Schomburg, (LNCS 1278, Springer, Berlin, 1997), pp. 199–209.

[4] J. Dassow, V. Mitrana, A. Salomaa, Context-free evolutionary grammars and the language of nucleic acids. *BioSystems*, 4(1997), pp. 169–177.

[5] J. Dassow, V. Mitrana, Self cross-over systems. *Computing with Bio-Molecules*, ed. Gh. Păun, (Springer, Singapore, 1998), pp. 283–294.

[6] J. Dassow, V. Mitrana, Gh. Păun, On the regularity of duplication closure, *Bull. EATCS*, 69 (October 1999), pp. 133–136.

[7] A. Ehrenfeucht, G. Rozenberg, On regularity of languages generated by copying systems. *Discrete Appl. Math.* 8 (1984), pp 313-317

[8] W. S. Klug, M. R. Cummings, *Concepts of Genetics* (Prentice Hall Inc, 1997)

[9] P. Leupold, C. Martín-Vide, V. Mitrana, Uniformly Bounded Duplication Languages. *Discrete Appl. Math.* 146 (2005), pp 301-310.

[10] P. Leupold, V. Mitrana, Uniformly Bounded Duplication Codes. *RAIRO-Theor. Inf. Appl.* 41 (2007), pp 411-424.

[11] P. Leupold, V. Mitrana, J.M. Sempere. Formal languages arising from DNA duplication. *Aspects of Molecular Computing* , eds. N. Jonoska, G. Păun, G. Rozenberg, (LNCS 2950, Springer, 2004), pp. 297–308.

[12] C. Martín-Vide, Gh. Păun, Duplication grammars. *Acta Cybernetica*, 14, 1(1999), pp. 101–113.

[13] V. Mitrana, G. Rozenberg, Some properties of duplication grammars. *Acta Cybernetica*, 14, 1(1999), pp. 165–177.

[14] Gh. Păun, G. Rozenberg and A. Salomaa, *DNA Computing. New Computing Paradigms*, (Springer, Berlin, 1998).

[15] G. Rozenberg, A. Salomaa (eds.), *Handbook of Formal Languages*, Vols. I-III (Springer, Berlin, 1997).

[16] D. B. Searls, The computational linguistics of biological sequences. *Artificial Intelligence and Molecular Biology*, ed. L. Hunter, (AAAI Press/MIT Press, Menlo Park, CA/Cambridge, MA, 1993), pp. 47–120.

[17] M. Wang, On the irregularity of the duplication closure, *Bull. EATCS*, 70 (February 2000), pp. 162–163.