Formal Languages Arising from Gene Repeated Duplication

Peter Leupold[1], Victor Mitrana[2], and José M. Sempere[3]

[1] Research Group on Mathematical Linguistics
Rovira i Virgili University
Pça. Imperial Tàrraco 1, 43005 Tarragona, Spain
pl.doc@estudiants.urv.es
[2] Faculty of Mathematics and Computer Science, Bucharest University
Str. Academiei 14, 70109 Bucureşti, Romania
and
Research Group on Mathematical Linguistics
Rovira i Virgili University
Pça. Imperial Tàrraco 1, 43005 Tarragona, Spain
vmi@fll.urv.es
[3] Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
Camino de Vera s/n, 46022 Valencia, Spain
jsempere@dsic.upv.es

Abstract. We consider two types of languages defined by a string through iterative factor duplications, inspired by the process of tandem repeats production in the evolution of DNA. We investigate some decidability matters concerning the unbounded duplication languages and then fix the place of bounded duplication languages in the Chomsky hierarchy by showing that all these languages are context-free. We give some conditions for the non-regularity of these languages. Finally, we discuss some open problems and directions for further research

1 Introduction

In the last years there have been introduced some operations and generating devices based on duplication operations, motivated by considerations from molecular genetics. It is widely accepted that DNA and RNA structures may be viewed to a certain extent as strings; for instance, a DNA strand can be presented as a string over the alphabet of the complementary pairs of symbols $(A,T),(T,A),(C,G),(G,C)$. Consequently, point mutations as well as large scale rearrangements occurring in the evolution of genomes may be modeled as operations on strings.

One of the most frequent and less well understood mutations among the genome rearrangements is the gene duplication or the duplication of a segment of a chromosome. Chromosomal rearrangements include pericentric and paracentric inversions, intrachromosomal as well as interchromosomal transpositions,

N. Jonoska et al. (Eds.): Molecular Computing (Head Festschrift), LNCS 2950, pp. 297–308, 2004.

translocations, etc. Crossover results in recombination of genes in a pair of homologous chromosomes by exchanging segments between parental chromatides. We refer to [3], [4], [5] and [19] for discussions on different formal operations on strings related to the language of nucleic acids. This feature is also known in natural languages. For motivations coming from linguistics, we refer to [11] and [17].

In the process of duplication, a stretch of DNA is duplicated to produce two or more adjacent copies, resulting in a tandem repeat. An interesting property of tandem repeats is that they make it possible to do "phylogenetic analysis" on a single sequence which might be useful to determine a minimal or most likely duplication history.

Several mathematical models have been proposed for the production of tandem repeats including replication, slippage and unequal crossing over [10,24,18]. These models have been supported by biological studies [20,?].

The so-called crossing over between "sister" chromatides is considered to be the main way of producing tandem repeats or block deletions in chromosomes. In [2], modeling and simulation suggests that very low recombination rates (unequal crossing over) can result in very large copy number and higher order repeats. A model of this type of crossing over has been considered in [6]. 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 (see Figure 1). 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 to the last segment of the first string.

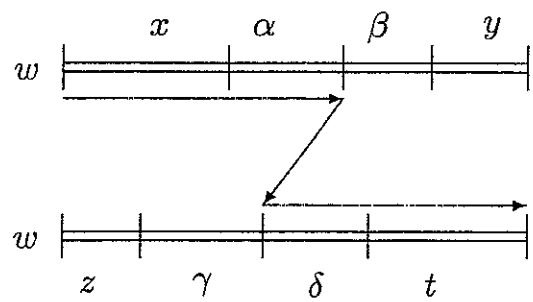

Figure 1: A scheme for gene duplication

It is easily seen that one obtains the insertion of a substring of w in w; this has the potential for inducing duplications of genes within a chromosome. We note here that, despite this model is inspired from recombination in vivo, it actually makes use of splicing rules in the sense of [8], where a computational model based on the DNA recombination under the influence of restriction enzymes and ligases essentially in vitro is introduced. This model turned out to be very attractive for computer scientists, see, e.g., the chapter [9] in [16] and [15].

Based on [3], Martín-Vide and Păun introduced in [12] a generative mechanism (similar to the one considered in [4]) 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 [12] from the generative power point of view. In [14] one considers the context-free versions of duplication grammars, solves some problems left open in [12], proves new results concerning the generative power of context-sensitive and context-free duplication grammars, and compares the two classes of grammars. Context-free duplication grammars formalize the hypothesis that duplications appear more or less at random within the genome in the course of its evolution.

In [7] one considers a string and constructs the language obtained by iteratively duplicating any of its substrings. One proves 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 [13], where it is proved that each string over a three-letter alphabet generates a non-regular language by duplication.

This paper continues this line of investigation. Many questions are still unsolved; we list some of them, which appear more attractive to us – some of them will be investigated in this work:

- Is the boundary of the duplication unique, is it confined to a few locations or it is seemingly unrestricted?
- Is the duplication unit size unique, does it vary in a small range or is it unrestricted?
- Does pattern size affect the variability of duplication unit size?
- Does duplication occur preferentially at certain sites?

In [7] the duplication unit size is considered to be unrestricted. We continue here with a few properties of the languages defined by unbounded duplication unit size and then investigate the effect of restricting this size within a given range.

The paper is organized as follows: in the next section we give the basic definitions and notations used throughout the paper. Then we present some properties of the unbounded duplication languages based essentially on [7,13]. The fourth section is dedicated to bounded duplication languages. The main results of this section are: 1. Each bounded duplication language is context-free. 2. Any square-free word over an at least three-letter alphabet defines a k-bounded duplication languages which is not regular for any $k \geq 4$. The papers ends with a discussion on several open problems and directions for further research.

2 Preliminaries

Now, we give the basic notions and notations needed in the sequel. For basic formal language theory we refer to [15] or [16]. We use the following basic notation. 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. The set

of all strings (words) over an alphabet V is denoted by V^* and $V^+ = V^* \setminus \{\varepsilon\}$, where ε denotes the empty string. The length of a string x is denoted by $|x|$, hence $|\varepsilon| = 0$, while the number of all occurrences of a letter a in x is denoted by $|x|_a$. For an alphabet $V = \{a_1, a_2, \ldots, a_k\}$ (we consider an ordering on V), the Parikh mapping associated with V is a homomorphism Ψ_V from V^* into the monoid of vector addition on $\mathbb{N}^k$, defined by $\Psi_V(s) = (|s|_{a_1}, |s|_{a_2}, \ldots, |s|_{a_k})$; moreover, given a language L over V, we define its image through the Parikh mapping as the set $\Psi_V(L) = \{\Psi_V(x) \mid x \in L\}$. A subset X of $\mathbb{N}^k$ is said to be *linear* if there are the vectors $c_0, c_1, c_2, \ldots, c_n \in \mathbb{N}^k$, for some $n \geq 0$ such that $X = \{c_0 + \sum_{i=1}^{n} x_i c_i \mid x_i \in \mathbb{N}, 1 \leq i \leq n\}$. A finite union of linear sets is called *semilinear*. For any positive integer n we write $[n]$ for the set $\{1, 2, \ldots, n\}$.

Let V be an alphabet and $X \in \{\mathbb{N}\} \cup \{[k] \mid k \geq 1\}$. For a string $w \in V^+$, we set

$$D_X(w) = \{uxxv \mid w = uxv, u, v \in V^*, x \in V^+, |x| \in X\}.$$

We now define recursively the languages:

$$D_X^0(w) = \{w\}, \qquad D_X^i(w) = \bigcup_{x \in D_X^{i-1}(w)} D_X(x), \; i \geq 1,$$

$$D_X^*(w) = \bigcup_{i \geq 0} D_X^i(w).$$

The languages $D_{\mathbb{N}}^*(w)$ and $D_{[k]}^*(w)$, $k \geq 1$, are called the *unbounded duplication language* and the *k-bounded duplication language*, respectively, defined by w. In other words, for any $X \in \{\mathbb{N}\} \cup \{[k] \mid k \geq 1\}$, $D_X^*(w)$ is the smallest language $L' \subseteq V^*$ such that $w \in L'$ and whenever $uxv \in L'$, $uxxv \in L'$ holds for all $u, v \in V^*$, $x \in V^+$, and $|x| \in X$.

A natural question concerns the place of unbounded duplication languages in the Chomsky hierarchy. In [7] it is shown that the unbounded duplication language defined by any word over a two-letter alphabet is regular, while [13] shows that these are the only cases when the unbounded language defined by a word is regular. By combining these results we have:

Theorem 1. [7,13] *The unbounded duplication language defined by a word w is regular if and only if w contains at most two different letters.*

3 Unbounded Duplication Languages

We do not know whether or not all unbounded duplication languages are context-free. A straightforward observation leads to the fact that all these languages are linear sets, that is, the image of each unbounded duplication language through the Parikh mapping is linear. Indeed, if $w \in V^+$, $V = \{a_1, a_2, \ldots, a_n\}$, then one can easily infer that

$$\Psi_V(D_{\mathbb{N}}^*(w)) = \{\Psi_V(w) + \sum_{i=1}^{n} x_i e_i^{(n)} \mid x_i \in \mathbb{N}, 1 \leq i \leq n\},$$

where $e_i^{(n)}$ is the vector of size n having the ith entry equal to 1 and all the other entries equal to 0.

Theorem 2. *Given a regular language L one can algorithmically decide whether or not L is an unbounded duplication language.*

Proof. We denote by $alph(x)$ the smallest alphabet such that $x \in (alph(x))^*$. The algorithm works as follows:

(i) We find the shortest string $z \in L$ (this can be done algorithmically). If there are more strings in L of the same length as z, then L is not an unbounded duplication language.
(ii) We now compute the cardinality of $alph(z)$.
(iii) If $card(alph(z)) \geq 3$, then there is no x such that $L = D^*_{\mathbb{N}}(x)$.
(iv) If $card(alph(z)) = 1$, then L is an unbounded duplication language if and only if $L = \{a^{|z|+m} \mid m \geq 0\}$, where $alph(z) = a$.
(v) If $k = 2$, $z = z_1 z_2 \dots z_n$, $z_i \in alph(z)$, $1 \leq i \leq n$, then L is an unbounded duplication language if and only if

$$L = z_1^+ e_1 z_2 e_2 \dots e_{n-1} z_n^+, \tag{1}$$

where

$$e_i = \begin{cases} z_{i+1}^*, & \text{if } z_i = z_{i+1} \\ \{z_i + z_{i+1}\}^*, & \text{if } z_i \neq z_{i+1} \end{cases}$$

for all $1 \leq i \leq n-1$. Note that one can easily construct a deterministic finite automaton recognizing the language in the right-hand side of equation (1).

□

Theorem 3. *1. The following problems are algorithmically decidable for unbounded duplication languages:*

Membership: Given x and y, is x in $D^*_{\mathbb{N}}(y)$?
Inclusion: Given x and y, does $D^*_{\mathbb{N}}(x) \subseteq D^*_{\mathbb{N}}(y)$ hold?

2. The following problems are algorithmically decidable in linear time:

Equivalence: Given x and y, does $D^*_{\mathbb{N}}(x) = D^*_{\mathbb{N}}(y)$ hold?
Regularity: Given x, is $D^*_{\mathbb{N}}(x)$ a regular language?

Proof. Clearly, the membership problem is decidable and

$$D^*_{\mathbb{N}}(x) \subseteq D^*_{\mathbb{N}}(y) \text{ iff } x \in D^*_{\mathbb{N}}(y).$$

For $D^*_{\mathbb{N}}(x) = D^*_{\mathbb{N}}(y)$, it follows that $|x| = |y|$, hence $x = y$. In conclusion, $x = y$ iff $D^*_{\mathbb{N}}(x) = D^*_{\mathbb{N}}(y)$. This implies that the equivalence problem is decidable in linear time.

The regularity can be decided in linear time by Theorem 1. □

4 Bounded Duplication Languages

Unlike the case of unbounded duplication languages, we are able to determine the place of bounded duplication languages in the Chomsky hierarchy. This is the main result of this section.

Theorem 4. *For any word r and any integer $n \geq 1$, the n-bounded duplication language defined by r is context-free.*

Proof. For our alphabet $V = alph(r)$ we define the extended alphabet V_e by $V_e := V \cup \{\langle a \rangle \mid a \in V\}$. Further we define $L^{\leq l} := \{w \in L \mid |w| \leq l\}$ for any language L and integer l. We now define the pushdown automaton

$$A_n^r = \left(Q, V, \Gamma, \delta, \begin{bmatrix} \varepsilon \\ \varepsilon \\ r \end{bmatrix}, \bot, \left\{ \begin{bmatrix} \varepsilon \\ \varepsilon \\ \varepsilon \end{bmatrix} \right\}\right),$$

where $Q = \left\{ \begin{bmatrix} \mu \\ v \\ w \end{bmatrix} \mid \mu \in (V_e^* \cdot V^* \cup V^* \cdot V_e^*)^{\leq n}, v \in (V^*)^{\leq n}, w \in (V^*)^{\leq |r|} \right\}$,

and $\Gamma = \{\bot\} \cup \left\{ \overline{\mu} \mid \begin{bmatrix} \mu \\ v \\ w \end{bmatrix} \in Q, v \in (V^*)^{\leq n}, w \in (V^*)^{\leq |r|} \right\}$.

Here we call the three strings occurring in a state from bottom to top *pattern*, *memory*, and *guess*, respectively. Now we proceed to define an intermediate deterministic transition function δ'. In this definition the following variables are always quantified universally over the following domains: $u, v \in (V^*)^{\leq n}$, $w \in (V^*)^{\leq |r|}$, $\mu \in (V_e^*)^{\leq n}$, $\eta \in (V_e^* \cdot V^* \cup V^* \cdot V_e^*)^{\leq n}$, $\gamma \in \Gamma$, $x \in V$ and $Y \in V_e$.

(i) $\delta'\left(\begin{bmatrix} \varepsilon \\ \varepsilon \\ xw \end{bmatrix}, x, \bot\right) = \left(\begin{bmatrix} \varepsilon \\ \varepsilon \\ w \end{bmatrix}, \bot\right)$ and $\delta'\left(\begin{bmatrix} \varepsilon \\ xu \\ xw \end{bmatrix}, \varepsilon, \bot\right) = \left(\begin{bmatrix} \varepsilon \\ u \\ w \end{bmatrix}, \bot\right)$

(ii) $\delta'\left(\begin{bmatrix} \mu x u \\ \varepsilon \\ w \end{bmatrix}, x, \gamma\right) = \left(\begin{bmatrix} \mu\langle x \rangle u \\ \varepsilon \\ w \end{bmatrix}, \gamma\right)$ and $\delta'\left(\begin{bmatrix} \mu x u \\ xv \\ w \end{bmatrix}, \varepsilon, \gamma\right) = \left(\begin{bmatrix} \mu\langle x \rangle u \\ v \\ w \end{bmatrix}, \gamma\right)$

(iii) $\delta'\left(\begin{bmatrix} u\langle x \rangle Y \mu \\ \varepsilon \\ w \end{bmatrix}, x, \gamma\right) = \left(\begin{bmatrix} uxY\mu \\ \varepsilon \\ w \end{bmatrix}, \gamma\right)$ and

$\delta'\left(\begin{bmatrix} u\langle x \rangle Y \mu \\ xv \\ w \end{bmatrix}, \varepsilon, \gamma\right) = \left(\begin{bmatrix} uxY\mu \\ v \\ w \end{bmatrix}, \gamma\right)$

(iv) $\delta'\left(\begin{bmatrix} u\langle x \rangle \\ \varepsilon \\ w \end{bmatrix}, x, \overline{\eta}\right) = \left(\begin{bmatrix} \eta \\ ux \\ w \end{bmatrix}, \varepsilon\right)$, $\delta'\left(\begin{bmatrix} u\langle x \rangle \\ xv \\ w \end{bmatrix}, \varepsilon, \overline{\eta}\right) = \left(\begin{bmatrix} \eta \\ uxv \\ w \end{bmatrix}, \varepsilon\right)$.

For all triples $(q, x, \gamma) \in Q \times (V \cup \{\varepsilon\}) \times \Gamma$ not listed above, we put $\delta'(q, x, \gamma) = \emptyset$.

To ensure that our finite state set suffices, we take a closer look at the memory of the states – since after every reduction the reduced word is put there (see transition set (iv)), there is a danger of this being unbounded. However, during any reduction of a duplication, which in the end puts $k \leq n$ letters into the memory, either $2k$ letters from the memory or all letters of the memory (provided the memory is shorter than $2k$) have been read, since reading from memory has priority (note that the tape is read in states with empty memory only). This gives us a bound on the length of words in the memory which is n. It is worth noting that the transitions of δ' actually match either the original word r or the

guess against the input or memory. To obtain the transition function δ of our automaton, we add the possibility to interrupt in any point the computation of δ' and change into a state that starts the reduction of another duplication. To this end we define for all $\begin{bmatrix} \eta \\ v \\ w \end{bmatrix} \in Q \setminus F$, and $\gamma \in \Gamma$

$$\delta\left(\begin{bmatrix} \eta \\ v \\ w \end{bmatrix}, \varepsilon, \gamma\right) = \delta'\left(\begin{bmatrix} \eta \\ v \\ w \end{bmatrix}, \varepsilon, \gamma\right) \cup \left\{\left(\begin{bmatrix} z \\ v \\ w \end{bmatrix}, \overline{\eta}\gamma\right) \mid z \in (\Sigma^+)^{\leq n}\right\}.$$

Such a transition guesses that at the current position a duplication of the form zz can be reduced (note that also here the bound of the length of the memory is not violated). For the sake of understandability we first describe also the function of the sets of transitions of δ'. Transitions (i) match the input word and r; this is only done on empty guess and stack, which ensures that every letter is matched only after all duplications affecting it have been reduced. Sets (ii) and (iii) check whether the guess, which is the segment of a guessed duplication, does indeed occur twice adjacently in the memory followed by the input. This is done by converting first guess letters from letters in V to the corresponding letters in V_e (set (ii)) exactly if the respective letter is read from either memory or input. Then set (iii) recovers the original letters. Finally the transitions in (iv) check the last letter; if it also matches, then the duplication is reduced by putting only one copy (two have been read) of the guess in the memory, and the computation is continued by putting the string encoded in the topmost stack symbol back into the guess. Now we can state an important property of $\mathcal{A}_n^r$ which allows us to conclude that it accepts exactly the language $D^*_{[n]}(r)$.

Property. *If there is an accepting computation in $\mathcal{A}_n^r$ starting from the configuration $\left(\begin{bmatrix} \eta \\ \varepsilon \\ w \end{bmatrix}, v, \alpha\right)$, then there is also an accepting computation starting from any configuration $\left(\begin{bmatrix} \eta \\ v_1 \\ w \end{bmatrix}, v_2, \alpha\right)$ where $v = v_1v_2$, $|v_1| \leq n$.*

Proof of the property. The statement is trivially true for $v_1 = \varepsilon$, therefore in the rest of the proof we consider $v_1 \neq \varepsilon$. We prove the statement by induction on the length of the computation. First we note that there exists an unique accepting configuration that is $\left(\begin{bmatrix} \varepsilon \\ \varepsilon \\ \varepsilon \end{bmatrix}, \varepsilon, \bot\right)$. Let Π be an accepting computation in $\mathcal{A}_n^r$ starting from the configuration $\left(\begin{bmatrix} \eta \\ \varepsilon \\ w \end{bmatrix}, v, \alpha\right)$.

If the length of Π is one, then $\eta = \varepsilon$, $\alpha = \bot$, and $v = w \in V$ which makes the statement obviously true. Let us assume that the statement is true for any computation of length at most p and consider a computation Π of length $p+1$ where we emphasize the first step. We distinguish three cases:

Case 1.

$$\left(\begin{bmatrix} \eta \\ \varepsilon \\ w \end{bmatrix}, v, \alpha\right) \vdash \left(\begin{bmatrix} z \\ \varepsilon \\ w \end{bmatrix}, v, \overline{\eta}\alpha\right) \vdash^* \left(\begin{bmatrix} \varepsilon \\ \varepsilon \\ \varepsilon \end{bmatrix}, \varepsilon, \bot\right).$$

Clearly, we have also $\left(\begin{bmatrix}\eta\\ v_1\\ w\end{bmatrix}, v_2, \alpha\right) \vdash \left(\begin{bmatrix}z\\ v_1\\ w\end{bmatrix}, v_2, \overline{\eta}\alpha\right)$ for any $v_1v_2 = v$, $|v_1| \leq n$. By the induction hypothesis, $\left(\begin{bmatrix}z\\ v_1\\ w\end{bmatrix}, v_2, \overline{\eta}\alpha\right) \vdash^* \left(\begin{bmatrix}\varepsilon\\ \varepsilon\\ \varepsilon\end{bmatrix}, \varepsilon, \perp\right)$ holds as well.

Case 2.

$$\left(\begin{bmatrix}\eta\\ \varepsilon\\ w\end{bmatrix}, xv', \alpha\right) \vdash \left(\begin{bmatrix}\eta'\\ \varepsilon\\ w\end{bmatrix}, v', \alpha\right) \vdash^* \left(\begin{bmatrix}\varepsilon\\ \varepsilon\\ \varepsilon\end{bmatrix}, \varepsilon, \perp\right),$$

where $v = xv'$ and the first step is based on a transition in the first part of the sets (i–iii). We have also $\left(\begin{bmatrix}\eta\\ xv_1'\\ w\end{bmatrix}, v_2, \alpha\right) \vdash \left(\begin{bmatrix}\eta'\\ v_1'\\ w\end{bmatrix}, v_2, \alpha\right)$ based on one of the corresponding transitions in the second part of the sets (i–iii). Since $v_1 = xv_1'$, by the induction hypothesis we are done in the second case.

Case 3.

$$\left(\begin{bmatrix}\eta\\ \varepsilon\\ w\end{bmatrix}, xv', \overline{\sigma}\alpha'\right) \vdash \left(\begin{bmatrix}\sigma\\ y\\ w\end{bmatrix}, v', \alpha'\right) \vdash^* \left(\begin{bmatrix}\varepsilon\\ \varepsilon\\ \varepsilon\end{bmatrix}, \varepsilon, \perp\right),$$

where the first step is based on a transition in the first part of the set (iv). Since reading from memory has priority, there exist w', τ, and β such that

$$\left(\begin{bmatrix}\sigma\\ y\\ w\end{bmatrix}, v', \alpha'\right) \vdash^* \left(\begin{bmatrix}\tau\\ \varepsilon\\ w'\end{bmatrix}, v', \beta\right) \vdash^* \left(\begin{bmatrix}\varepsilon\\ \varepsilon\\ \varepsilon\end{bmatrix}, \varepsilon, \perp\right).$$

We have $\left(\begin{bmatrix}\eta\\ xv_1'\\ w\end{bmatrix}, v_2, \overline{\sigma}\alpha'\right) \vdash \left(\begin{bmatrix}\sigma\\ yv_1'\\ w\end{bmatrix}, v_2, \alpha'\right)$ for any $v_1v_2 = v$, $|v_1| \leq n$. Further, following the same transitions as above, we get $\left(\begin{bmatrix}\sigma\\ yv_1'\\ w\end{bmatrix}, v_2, \alpha'\right) \vdash^* \left(\begin{bmatrix}\tau\\ v_1'\\ w'\end{bmatrix}, v_2, \beta\right)$, and by the induction hypothesis we conclude the proof of the third case, and thus the proof of the property.

After the elaborate definition of our pushdown automaton, the proof for $L(\mathcal{A}_n^r) = D_{[n]}^*(r)$ by induction on the number of duplications used to create a word in $D_{[n]}^*(r)$ is now rather straightforward. We start by noting that the original word r is obviously accepted. Now we suppose that all words reached from r by $m-1$ duplications are also accepted and look at a word s reached by m duplications. Clearly there is a word s' reached from r by $m-1$ duplications such that s is the result of duplicating one part of s'. By the induction hypothesis, s' is accepted by $\mathcal{A}_n^r$, which means that there is a computation, we call it Ξ, which accepts s'. Let $s'[l \ldots k]$ (the subword of s' which starts at the position l and ends at the position k in s', $l \leq k$) be the segment of s' which is duplicated at the final stage of producing s. Therefore

$$s = s'[1 \ldots k]s'[l \ldots |s'|] = s'[1 \ldots l-1]s'[l \ldots k]s'[l \ldots k]s'[k+1 \ldots |s'|].$$

Because $\mathcal{A}_n^k$ reads in an accepting computation every input letter exactly once, there is exactly one step in Ξ, where $s'[l]$ is read. Let this happen in a state $\begin{bmatrix}\mu\\ \varepsilon\\ w\end{bmatrix}$

with $s'[l \ldots |s'|] = s[k+1 \ldots |s|]$ left on the input tape and α the stack contents. Now we go without reading the input tape to state $\begin{bmatrix} s'[l \ldots k] \\ \varepsilon \\ w \end{bmatrix}$ and push $\overline{\mu}$ onto the stack. Then we reduce the duplication of the subword $s'[l \ldots k]$ of s, which is the guess of the current state, by matching it twice against the letters read on the input tape. After reducing this duplication, we arrive at a configuration having a state containing μ in the guess, $s'[l \ldots k]$ in the memory, and w in the pattern, $s[k+1 \ldots |s|]$ left on the tape, and the stack contents as before. By the property above, there exists an accepting computation starting with this configuration, hence s is accepted. Since all words accepted by $\mathcal{A}_n^r$ are clearly in $D^*_{[n]}(r)$, and we are done. □

Clearly, any language $D^*_{[1]}(w)$ is regular. The same is true for any language $D^*_{[2]}(w)$. Indeed, it is an easy exercise (we leave it to the reader) to check that

$$D^*_{[2]}(w) = (w[1]^+w[2]^+)^*(w[2]^+w[3]^+)^* \ldots (w[|w-1|]^+w[|w|]^+)^*.$$

The following question appears in a natural way: Which are the minimal k and n such that there are words w over an n-letter alphabet such that $D^*_{[k]}(w)$ is not regular?

Theorem 5. *1. $D^*_{[k]}(w)$ is always regular for any word w over a two-letter alphabet and any $k \geq 1$.*

2. For any word w of the form $w = xabcy$ with $a \neq b \neq c \neq a$, $D^*_{[k]}(w)$ is not regular for any $k \geq 4$.

Proof 1. The equality $D^*_{\mathbb{N}}(w) = D^*_{[k]}(w)$ holds for any $k \geq 2$ and any word w over a two-letter alphabet, hence the first item is proved.

2. Our reasoning for proving the second item, restricted without loss of generality to the word $w = abc$, is based on a similar idea to that used in [13]. So, let $w = abc$, $V = \{a, b, c\}$, and $k \geq 4$. First we prove that for any $u \in V^+$ such that wu is square-free, there exists $v \in V^*$ such that $wuv \in D^*_{[k]}(w)$. We give a recursive method for constructing wuv starting from w; at any moment we have a string $wu'v'$ where u' is a prefix of u and v' is a suffix of v. Initially, the current string is w which satisfies these conditions. Let us assume that we reached $x = wu[1]u[2] \ldots u[i-1]v'$ and we want to get $y = wu[1]u[2] \ldots u[i]v''$. To this end, we duplicate the shortest factor of x which begins with $u[i]$ and ends on the position $2+i$ in x. Because $wu[1]u[2] \ldots u[i-1]$ is square-free and any factor of a square-free word is square-free as well, the length of this factor which is to be duplicated is at most 4. Now we note that, given u such that wu is square free and v is the shortest word such that $wuv \in D^*_{[k]}(w)$, we have on the one hand $k|u| + 3 \geq |wuv|$ (each duplication produces at least one symbol of u), and on the other hand $(k-1)|v| \geq |u|$ (each duplication produces at least one symbol of v since wu is always square-free). Therefore,

$$(k-1)|u| \geq |v| \geq \frac{|u|}{k-1} \tag{2}$$

We are ready now to prove that $D^*_{[k]}(w)$ is not regular using the Myhill-Nerode characterization of regular languages. We construct an infinite sequence of square-free words $w_1, w_2, \ldots,$ each of them being in a different equivalence class:

$$w_1 = w \text{ and } w_{i+1} = wu \text{ such that } wu \text{ is square-free and } (k-1)|w_i| < \frac{|u|}{k-1}.$$

Clearly, we can construct such an infinite sequence of square-free words since there are arbitrarily long square-free words having the prefix abc [21,22,1]. For instance, all words $h^n(a)$, $n \geq 1$, are square-free and begin with abc, where h is an endomorphism on V^* defined by $h(a) = abcab$, $h(b) = acacb$, $h(c) = acbcacb$. Let v_i be the shortest word such that $w_i v_i \in D^*_{[k]}(w)$, $i \geq 1$. By relation (2), $w_{i+1} v_j \notin D^*_{[k]}(w)$ for any $1 \leq j \leq i$. Consequently, $D^*_{[k]}(w)$ is not regular. □

Since each square-free word over an at least three-letter alphabet has the form required by the previous theorem, the next corollary directly follows.

Corollary 1. *$D^*_{[k]}(w)$ is not regular for any square-free word w over an alphabet of at least three letters and any $k \geq 4$.*

5 Open Problems and Further Work

We list here some open problems which will be in our focus of interest in the near future:

1. Is any unbounded duplication language context-free? A way for attacking this question, in the aim of an affirmative answer, could be to prove that for any word w there exists a natural k_w such that $D^*_{\mathbb{N}}(w) = D^*_{[k_w]}(w)$. Note that a similar result holds for words w over alphabets with at most two letters.

2. What is the complexity of the membership problem for unbounded duplication languages? Does the particular case when y is square-free make any difference?

3. We define the *X-duplication distance* between two strings x and y, denoted by $Dupd_X(x, y)$, as follows:

$$Dupd_X(x, y) = \min\{k \mid x \in D^k_X(y) \text{ or } y \in D^k_X(x)\}, X \in \{\mathbb{N}\} \cup \{[n] \mid n \geq 2\}.$$

Clearly, *Dupd* is a distance. Are there polynomial algorithms for computing this distance? What about the particular case when one of the input strings is square-free?

4. An *X-duplication root*, $X \in \{\mathbb{N}\} \cup \{[n] \mid n \geq 2\}$, of a string x is an X-square-free string y such that $x \in D^*_X(y)$. A string y is X-square-free if it does not contain any factor of the form zz with $|z| \in X$. It is known that there are words having more than one $\mathbb{N}$-duplication root. Then the following question is natural: If x and y have a common duplication root and X is as above, then $D^*_X(x) \cap D^*_X(y) \neq \emptyset$? We strongly suspect an affirmative answer. Again, the case of at most two-letter alphabets is quite simple: Assume that x and y are two strings over the alphabet $V = \{a, b\}$ which have the same duplication root,

say aba (the other cases are identical). Then $x = a^{k_1}b^{p_1}a^{k_2} \ldots a^{k_n}b^{p_n}a^{k_{n+1}}$ and $y = a^{j_1}b^{q_1}a^{j_2} \ldots a^{j_m}b^{q_m}a^{j_{m+1}}$ for some $n, m \geq 1$. It is an easy exercise to get by duplication starting from x and y, respectively, the string $(a^ib^i)^sa^i$, where $s = \max(n, m)$ and $i = \max(A)$, with

$$A = \{k_t \mid 1 \leq t \leq n+1\} \cup \{p_t \mid 1 \leq t \leq n\}$$
$$\cup \{j_t \mid 1 \leq t \leq m+1\} \cup \{q_t \mid 1 \leq t \leq m\}.$$

What is the maximum number of X-duplication roots of a string? How hard is it to compute this number? Given n and $X \in \{\mathbb{N}\} \cup \{[n] \mid n \geq 2\}$, are there words having more than n X-duplication roots? (the *non-triviality* property) Given n, are there words having exactly n X-duplication roots? (the *connectivity* property) Going further, one can define the X-duplication root of a given language. What kind of language is the X-duplication root of a regular language? Clearly, if it is infinite, then it is not context-free.

References

1. Bean, D.R., Ehrenfeucht, A., Mc Nulty, G.F. (1979) Avoidable patterns in strings of symbols, Pacific J. of Math. 85:261–294.
2. Charlesworth, B., Sniegowski, P., Stephan, W. (1994) The evolutionary dynamics of repetitive DNA in eukaryotes, Nature 371:215–220.
3. Dassow, J., Mitrana, V. (1997) On some operations suggested by the genome evolution. In: Altman, R., Dunker, K., Hunter, L., Klein, T. (eds.) Pacific Symposium on Biocomputing'97, 97–108.
4. Dassow, J., Mitrana, V. (1997) Evolutionary grammars: a grammatical model for genome evolution. In: Hofestädt, R., Lengauer, T., Löffler, M., Schomburg, D. (eds.) Proceedings of the German Conference in Bioinformatics GCB'96, LNCS 1278, Springer, Berlin, 199–209.
5. Dassow, J., Mitrana, V., Salomaa, A. (1997) Context-free evolutionary grammars and the language of nucleic acids. BioSystems 4:169–177.
6. Dassow, J., Mitrana, V. (1998) Self cross-over systems. In: Păun, G. (ed.) Computing with Bio-Molecules, Springer, Singapore, 283–294.
7. Dassow, J., Mitrana, V., Păun, G. (1999) On the regularity of duplication closure, Bull. EATCS, 69:133–136.
8. Head, T. (1987) Formal language theory and DNA: an analysis of the generative capacity of specific recombinant behaviours, Bull. Math. Biology 49:737–759.
9. Head, T., Păun, G., Pixton, D. (1997) Language theory and molecular genetics. Generative mechanisms suggested by DNA recombination. In: [16]
10. Levinson, G., Gutman, G. (1987) Slipped-strand mispairing: a major mechanism for DNA sequence evolution, Molec. Biol. Evol. 4:203–221.
11. Manaster Ramer, A. (1999) Some uses and misuses of mathematics in linguistics. In: Martín-Vide, C. (ed.) Issues from Mathematical Linguistics: A Workshop, John Benjamins, Amsterdam, 70–130.
12. Martín-Vide, C., Păun, G. (1999) Duplication grammars, Acta Cybernetica 14:101–113.
13. Ming-wei, W. (2000) On the irregularity of the duplication closure, Bull. EATCS, 70:162–163.

14. Mitrana, V., Rozenberg, G. (1999) Some properties of duplication grammars, Acta Cybernetica, 14:165–177
15. Păun, G., Rozenberg, G., Salomaa, A. (1998) DNA Computing. New Computing Paradigms, Springer, Berlin
16. Rozenberg, G., Salomaa, A. (eds.) (1997) Handbook of Formal Languages, vol. I-III, Springer, Berlin.
17. Rounds, W.C., Manaster Ramer, A., Friedman, J. (1987) Finding natural languages a home in formal language theory. In: Manaster Ramer, A. (ed.) Mathematics of Language, John Benjamins, Amsterdam, 349–360.
18. Schlotterer, C., Tautz, D. (1992) Slippage synthesis of simple sequence DNA, Nucleic Acids Res. 20:211-215
19. Searls, D.B. (1993) The computational linguistics of biological sequences. In: Hunter, L. (ed.) Artificial Intelligence and Molecular Biology, AAAI Press/MIT Press, Menlo Park, CA/Cambridge, MA, 47–120.
20. Strand, M., Prolla, T., Liskay, R., Petes, T. (1993) Destabilization of tracts of simple repetitive DNA in yeast by mutations affecting DNA mismatch repair, Nature 365:274–276.
21. Thue, A. (1906) Uber unendliche Zeichenreihen, Norske Videnskabers Selskabs Skrifter Mat.-Nat. Kl. (Kristiania), 7:1–22.
22. Thue, A (1912) Uber die gegenseitige Lage gleicher Teile gewiisser Zeichenreihen, Norske Videnskabers Selskabs Skrifter Mat.-Nat. Kl. (Kristiania), 1:1–67.
23. Weitzmann, M., Woodford, K., Usdin, K. (1997) DNA secondary structures and the evolution of hyper-variable tandem arrays, J. of Biological Chemistry 272:9517–9523.
24. Wells, R. (1996) Molecular basis of genetic instability of triplet repeats, J. of Biological Chemistry 271:2875–2878.