Int. J. of Computers, Communications & Control, ISSN 1841-9836, E-ISSN 1841-9844
Vol. VI (2011), No. 4 (December), pp. 739-748

Computing by Folding

D. Sburlan

Dragoş Sburlan
Faculty of Mathematics and Informatics
Ovidius University of Constanta, Romania
E-mail: dsburlan@univ-ovidius.ro

Abstract: The present paper introduces a new computing paradigm based on the idea of string folding. Comparisons between the computational power of the proposed model with the classical families of languages from the Chomsky hierarchy are studied. Some preliminary results are reported and some conjectures are discussed. In this respect, the proposed model is promising not only because of the expected theoretical results, but also because of the possible indirect applications in various fields (as for instance, mathematical linguistics, DNA computing, computing using light, and so on).

1 Introduction

The computing paradigm proposed in this paper has as motivation several completely different domains that naturally converge. Whether we talk about the traditional Japanese art of paper folding (Origami), or about some in-vivo biochemical processes happening at the cellular level (for instance, the protein folding that produces particular 3-dimensional structures which are essential for their ”correct" functioning), or about the in-vitro folding of DNA (by which one can arbitrary build two and three dimensional shapes at the nanoscale), a common feature is the folding of the physical support. Aiming to design novel computational techniques, the present paper introduces a new model of computation which can be easily described by the controlled folding of the computational support.

In order to be more eloquent, one may consider the Origami perspective. Assume for the sake of argument that the paper used in Origami is actually a bi-dimensional array of symbols. Hence the paper is not only the physical support for building two or three-dimensional shapes and forms but also it might be considered as the support for computation. In this simple framework a fold of the paper corresponds to a step of computation. Even if in this informal presentation we are not interested to precisely define the outcome of performing such steps of computation, we only want to point out that, for instance, a simple fold along the vertical middle of a page corresponds actually to the simultaneous movement of symbols from one side to another. In this case, if to each symbol one associates a position in the two/three dimensional space, then this folding represents, in a certain sense, a parallel rewriting of symbols. More specifically, this simple procedure moves (rewrites) a lot of space in just one computational step. This is just one argument which indicates that, under certain conditions, the expected computational power for such systems exceeds the semi-linearity (see [1] for an operation that recalls in a certain sense the folding).

In this general framework several issues have to be settled, namely the computational support, the method by which the computational support will be folded, and the way the output is read. Here we only consider a restricted model where both the support of computation and the folding procedure are strings of symbols from some given languages. The model deals with the one-dimensional case and in our construction “computing” represents the generation of a language by means of a folding operation, starting from two “simpler” languages.

Finally, it is worth noting that the folding operation appears in various forms and contexts in formal language and natural computing theories; we quote here only [7] and [9]. In addition, the type of string operation considered in this paper recalls in a certain sense the shuffle operation as defined in [3] and [6].

2 Preliminaries

For the present work, we assume the reader be familiar with the basic notions and results of formal languages and theory of computation (for more details we indicate [2] and [8]). Here, we only present some results related to the current work.

If FL is a family of languages, then NFL denotes the family of length sets of languages in FL.

We denote by *REG*, *CF*, *CS*, *REC*, and *RE* the family of regular, context-free, context-sensitive, recursive, and recursively enumerable languages, respectively. It is known that $REG \subsetneq CF \subsetneq CS \subsetneq REC \subsetneq RE$. For regular and context-free languages there are known necessary conditions also called pumping lemmas. In case of regular languages the pumping lemma can be stated as:

Lemma 1. *Let L be an infinite regular language over a finite alphabet Σ. Then there is a positive constant n_L such that if $w \in L$, $|w| \geq n_L$, then w can be decomposed as $w = xyz$, $x, y, z \in \Sigma^*$, with $|y| \geq 1$, $|xy| \leq n_L$, and $w_i = xy^iz \in L$, for any $i \geq 0$.*

In case of context-free languages the pumping lemma can be stated as:

Lemma 2. *Let L be an infinite context-free language over a finite alphabet Σ. Then there is a positive constant n_L such that if $z \in L$, $|z| \geq n_L$, then z can be decomposed as $z = uvwxy$, $u, v, w, x, y \in \Sigma^*$, with $|vx| \geq 1$, $|vwx| \leq n_L$, and $z_i = uv^iwx^iy \in L$, for any $i \geq 0$.*

Let $I\!N$ be the set of nonnegative integers and k be a positive integer. A set $S \subseteq I\!N^k$ is a linear set if there are the vectors $v_0, v_1, \ldots, v_n \in I\!N^k$ such that $S = \{v \mid v = v_0 + a_1v_1 + \cdots + a_nv_n, a_i \in I\!N, 1 \leq i \leq n\}$. The vectors v_0 (*the constant vector*) and $v_1, v_2, \ldots, v_n$ (*the periods*) are called the *generators* of the linear set S. A set $S \subseteq I\!N^k$ is semilinear if it is a finite union of linear sets.

Let $\Sigma = \{a_1, a_2, \ldots, a_n\}$ be a finite alphabet. A mapping $\Psi : \Sigma^* \to I\!N^{|\Sigma|}$ such that $\Psi(w) = (|w|_{a_1}, \ldots, |w|_{a_n})$ is called the Parikh mapping. The mapping Ψ can be extended to languages such that if $L \subseteq \Sigma^*$ then $\Psi(L) = \bigcup\limits_{w \in L} \Psi(w)$. Two languages L_1 and L_2 are called *letter equivalent* if $\Psi(L_1) = \Psi(L_2)$. By $length(L)$ we will denote the length set of L.

A language L is called *semilinear* if $\Psi(L)$ is a semilinear set of vectors of numbers.

Semilinear sets are closed under complement, finite intersection and finite union. A permutation of symbols in a word preserves the Parikh image. All context-free languages are semilinear.

3 Folding Systems

Let Σ be a finite alphabet and $\Gamma = \{u, d\}$. Let $f : \Sigma^* \times \Sigma \times \Gamma \to \Sigma^*$ such that

$$\begin{aligned} f(w, a, u) &= aw, \\ f(w, a, d) &= wa. \end{aligned}$$

We define the *folding function* $h : \Sigma^* \times \Gamma^* \to \Sigma^*$ such that

$$h(w_1, w_2) = \begin{cases} f(\ldots f(f(\lambda, a_1, b_1), a_2, b_2) \ldots, a_k, b_k), \text{if} \\ |w_1| = |w_2|, w_1 = a_1 a_2 \ldots a_k, w_2 = b_1 b_2 \ldots b_k; \\ \text{undefined, if } |w_1| \neq |w_2|. \end{cases}$$

Remark 3.1. If $w_1 \in \Sigma^*$, $w_2 \in \Gamma^*$, and $|w_1| = |w_2|$, then by applying h we mean that w_1 is folded symbol by symbol according with the folding procedure described by the symbols of w_2 (d means to fold the "remaining" of w_1 downwards and u upwards); the result can be seen as a stack of symbols that by definition we read from the top to the bottom (see Figure 1).

Example 3. *Let $w_1 = aabb$ and $w_2 = dduu$. Then we formally have*

$$\begin{aligned} h(w_1, w_2) &= f(f(f(f(\lambda, a, d), a, d), b, u), b, u) \\ &= f(f(f(a, a, d), b, u)b, u) \\ &= f(f(aa, b, u), b, u) \\ &= f(baa, b, u) \\ &= bbaa \end{aligned}$$

or graphically, the recurrent application of f is given in Figure 1.

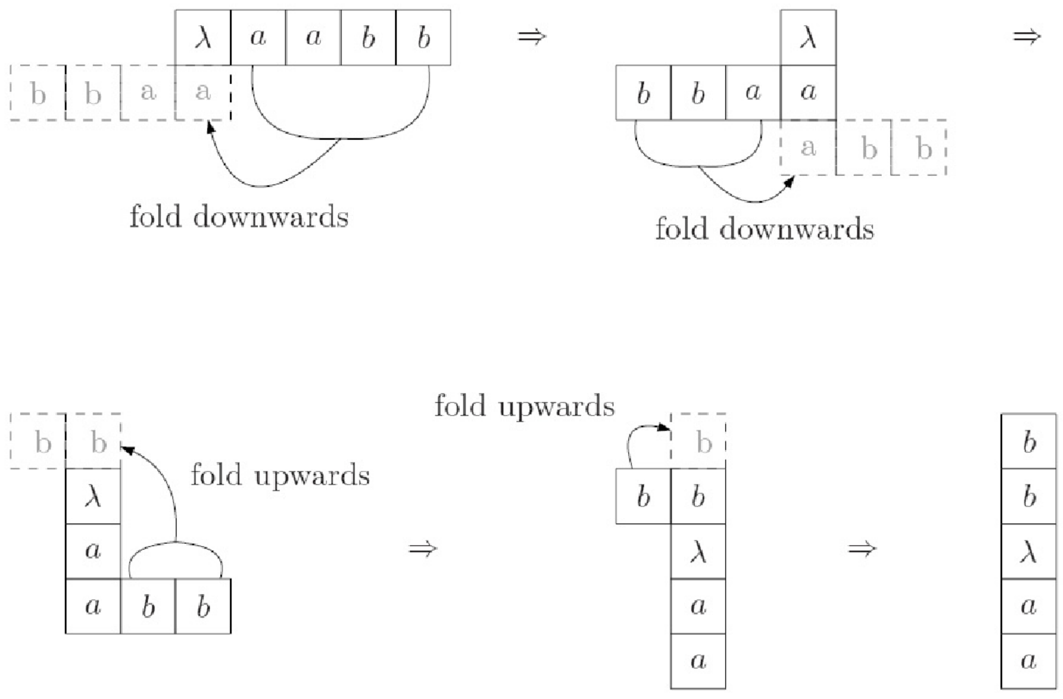

Figure 1: The computation of $h(aabb, dduu)$.

A *folding system* (in short an F-system) is a construct $\Phi = (L_1, L_2)$ where L_1 is an arbitrary language over an arbitrary alphabet Σ (called the *core language*) and L_2 is an arbitrary language over $\Gamma = \{u, d\}$ (called the *folding procedures set*). The language generated by $\Phi(L_1, L_2)$ is

$$L(\Phi) = \bigcup_{\substack{w_1 \in L_1, w_2 \in L_2 \\ |w_1| = |w_2|}} h(w_1, w_2),$$

where $h : \Sigma^* \times \Gamma^* \to \Sigma^*$ is the folding function.

For a class $\mathcal{C}$ of core languages and a class $\mathcal{H}$ of folding procedures sets, we denote by $\mathcal{F}(\mathcal{C}, \mathcal{H})$ the family of all languages generated by F-systems, using components from the respective classes, that is

$$\mathcal{F}(\mathcal{C}, \mathcal{H}) = \{L(\Phi) \mid \Phi = (L_1, L_2), L_1 \in \mathcal{C}, L_2 \in \mathcal{H}\}.$$

The following result shows the relation between the family of regular languages and the family of all languages generated by F-systems using regular languages as components.

Proposition 4. *$\mathcal{F}(REG, REG) \supsetneq REG$.*

Proof: Let L_1 be an arbitrary regular language and the regular language $L_2 = \{d\}^* \subseteq \Gamma^*$. Then we have that for any word $w_1 \in L_1$ there exists $w_2 \in L_2$ such that $|w_1| = |w_2|$. Since w_1 is "folded" always down (w_2 consists only of d symbols), then it follows that $h(w_1) = w_1$, hence $h(L_1) = L_1$ and further more $\mathcal{F}(REG, REG) \supseteq REG$.

In order to prove the strict inclusion one can construct the following F-system:

$$\Phi(L_1, L_2) \text{ where } L_1 = (ab)^* \text{ and } L_2 = (ud)^*.$$

It follows that for any even number $i = 2k$, $k > 0$, there exist exactly one word $w_1 = ab \dots ab \in L_1$ and exactly one word $w_2 = ud \dots ud \in L_2$ such that $|w_1| = |w_2| = i$. Hence $h(w_1, w_2) = a^k b^k$ and consequently $L(\Phi) = \{a^n b^n \mid n > 0\}$, the well-known non-regular but context-free language. □

By using context-free languages as core languages while preserving the regularity of folding procedure sets, one can surpass the class of context-free languages.

Proposition 5. *$\mathcal{F}(CF, REG) \supsetneq CF$.*

Proof: One can use the same argument as given in the proof of Proposition 4 to show that $\mathcal{F}(CF, REG) \supseteq CF$. In addition, consider the context-free language $L_1 = \{b^n(ac)^n \mid n \geq 0\}$ (as being generated for instance by the context-free grammar $G = (N, T, P, S)$, where $N = \{S\}$, $T = \{a, b, c\}$, and $P = \{S \to bSac, S \to \lambda\}$) and the regular language $L_2 = \{(ud)^n \mid n \geq 0\}$ indicated by the regular expression $(ud)^*$. The length of any word in L_1 is a multiple of 3, while any word from L_2 has a length which is a multiple of 2. Consequently, any word from $\Phi(L_1, L_2)$ must have a length multiple of the least common multiple of 2 and 3. It follows that $\Phi(L_1, L_2) = \{a^{2n} b^{2n} c^{2n} \mid n \geq 0\}$, which by using the pumping lemma for context-free languages can be shown not to be a context-free language. Consequently, we have that $\mathcal{F}(CF, REG) \supsetneq CF$. □

Proposition 6. *$\mathcal{F}(CF, CF) \supsetneq CF$.*

Proof: Since obviously $\mathcal{F}(CF, CF) \supseteq \mathcal{F}(CF, REG)$ then to prove the strict inclusion one can construct the following F-system, whose generated language does not belong to context-free languages family.

Let $\Phi = (L_1, L_2)$, where $L_1 = \{ww^r \mid w \in \{a, b\}^*\}$ is the context-free language of even palindromes and $L_2 = \{u^n d^n \mid n \geq 0\}$ is a context-free language as well. It follows that $L(\Phi) = \{ww \mid w \in \{a, b\}^*\}$ – a language that is not context-free.

Another interesting example which proves the strict inclusion regards the generation of the language $L = \{a^n b^n c^n \mid n \geq 0\}$. This can be achieved by an F-system $\Phi(L_1, L_2)$ where $L_1 = \{b^n(ac)^n \mid n \geq 0\}$ and $L_2 = \{u^n(ud)^n \mid n \geq 0\}$ are also context-free languages. □

Using a similar technique as the one presented in the proofs of Proposition 4 and Proposition 5 one can actually prove a more general result (that in addition suggests some interesting open problems regarding the usage of subfamilies of REG as folding procedures sets)

Lemma 7. *Let FL be a family of languages. Then $FL \subseteq \mathcal{F}(FL, REG)$.*

Moreover, one can state the following result:

Lemma 8. *Let FL_i, $1 \leq i \leq 4$, some families of languages such that $FL_1 \subseteq FL_2$ and $FL_3 \subseteq FL_4$. Then $\mathcal{F}(FL_1, FL_3) \subseteq \mathcal{F}(FL_2, FL_4)$.*

The following result states that all F-systems using context-free languages as components are equivalent to F-systems with a particular relation between the core languages and the folding procedures sets.

Lemma 9. *(Normal Form) Let $L \in \mathcal{F}(CF, CF)$. Then there exists an F-system $\Phi(L_1, L_2)$, $L_1, L_2 \in CF$, with $length(L_1) = length(L_2)$ and such that $L(\Phi) = L$.*

Proof: Let $L = L(\overline{\Phi}(\overline{L_1}, \overline{L_2}))$ such that $\overline{L_1}, \overline{L_2} \in CF$ and $length(\overline{L_1}) \neq length(\overline{L_2})$. We construct an equivalent F-system $\Phi(L_1, L_2)$, $L_1, L_2 \in CF$, such that $length(L_1) = length(L_2)$ as follows.

It is known from [5] that $NREG = NCF$. This means that there exist the languages $L_B, L_C \in REG$ such that $length(L_B) = length(\overline{L_1})$ and $length(L_C) = lenght(\overline{L_2})$. Let $\Sigma_{L_1} = \{a_1, a_2, \ldots, a_k\}$ be the alphabet of L_1, $\Sigma_B = \{b_1, b_2, \ldots, b_r\}$ be the alphabet of L_B, and $\Sigma_C = \{c_1, c_2, \ldots c_s\}$ be the alphabet of L_C. In addition, let us consider the finite substitution given by

$$\Delta_1 : \Sigma_C \to \mathcal{P}(\Sigma_B^*) \text{ such that } \Delta_1(c_i) = \Sigma_B,\ 1 \leq i \leq s.$$

Because regular languages are closed under finite substitutions and intersection then it follows that $\overline{L_B} = L_B \cap \Delta(L_C) \in REG$ ($\overline{L_B}$ contains all the words of length $t \geq 0$ from L_B for which there exists at least one word in L_C with the same length). Moreover, we have that $length(\overline{L_B}) = length(L_B) \cap length(L_C) = length(L_1) \cap length(L_2)$.

Let us consider now the substitutions:

$$\Delta_2 : \Sigma_B \to \mathcal{P}(\Sigma_A^*) \text{ such that } \Delta_2(b_i) = \Sigma_A,\ 1 \leq i \leq r,$$

and

$$\Delta_3 : \Sigma_B \to \mathcal{P}(\{u, d\}^*) \text{ such that } \Delta_3(b_i) = \{u, d\},\ 1 \leq i \leq r.$$

Then, if we take $L_1 = \overline{L_1} \cap \Delta_2(\overline{L_B})$ and $L_2 = \overline{L_2} \cap \Delta_3(\overline{L_B})$ it follows that $L_1, L_2 \in CF$ (as being the results of intersection of context-free languages with regular languages) and moreover $length(L_1) = length(L_2)$ (both substitutions Δ_2 and Δ_3 do not change the length sets of the languages on which they are applied; in particular $length(\overline{L_B}) = length(\Delta_2(\overline{L_B})) = length(\Delta_3(\overline{L_B}))$).

Similarly as before, in order to construct L_1 we get from $\overline{L_1}$ only those words which match as length at least one word in $\overline{L_B}$.

As a consequence, we have that $L(\Phi(L_1, L_2)) = L(\Phi(\overline{L_1}, \overline{L_2})) = L$. □

Remark 3.2. The language generated by an F-system $\Phi(L_1, L_2)$, where L_1 and L_2 are semilinear sets, is also semilinear.

Let us prove now a necessary condition for a language generated by a folding system using as components regular languages. Let $L_1 \subseteq \Sigma^*$ and $L_2 \subseteq \{u, d\}^*$ be two arbitrary infinite regular languages and assume that $L(\Phi(L_1, L_2))$ (the corresponding generated language) is also infinite. Let n_{L_1} and n_{L_2} be the constants from the pumping lemma for regular languages applied to L_1

and L_2, respectively. Let $n = max(n_{L_1}, n_{L_2})$. Because $L(\Phi(L_1, L_2))$ is infinite, then there are the words $r \in L_1$ and $s \in L_2$ such that $|r| = |s| \geq n$. According with the pumping lemma for regular languages, it follows that r can be written as $r = xyz$, where $x, y, z \in \Sigma^*$ and such that $|xy| \leq n$, $y \geq 1$. Similarly, $s = uvt$ where $u, v, t \in \{u, d\}^*$, and such that $|uv| \leq n$, $v \geq 1$. In both cases, the pumping lemma states that $r_i = xy^iz \in L_1$ and $s_i = uv^it \in L_2$, for any $i \geq 0$. Although in general $|r_i| \neq |s_i|$, for all $i \geq 0$, one can remark that, based on the lowest common multiple between y and v, one can define the sub-sequences of words

$$\overline{r}_i = xy^{\frac{lcm(|y|,|v|)}{|y|} \cdot i}z, \text{ for any } i \geq 0,$$

and

$$\overline{s}_i = uv^{\frac{lcm(|y|,|v|)}{|v|} \cdot i}t, \text{ for any } i \geq 0,$$

such that $|\overline{r}_i| = |\overline{s}_i|$, for any $i \geq 0$.

For simplicity of argument, let us consider now the double stranded structures $\left[\frac{\overline{r}_i}{\overline{s}_i}\right]$, for any $i \geq 0$. First of all, remark that x, z, u, t have constant lengths. Moreover, the above sub-sequences are infinite (because $|y| \geq 1$ and $|v| \geq 1$, one can remark that $|\overline{r}_{i+1}| > |\overline{r}_i|$, $|\overline{s}_{i+1}| > |\overline{s}_i|$, for any $i \geq 0$). It follows that, starting from a certain positive constant $\widetilde{n}$ (that depends on $|x|$, $|y|$, $|u|$, $|v|$, hence on L_1 and L_2), in the double stranded structure $\left[\frac{\overline{r}_j}{\overline{s}_j}\right]$, with $j \geq \widetilde{n}$, the substring from $\overline{r}_i$ composed by multiple concatenations of y overlaps (partially) the one from $\overline{s}_i$ composed by multiple concatenations of v. Moreover, assume in addition that $\widetilde{n}$ is the first rank in the sequence of double stranded structures such that the overlapping part is equal or longer than $min(\frac{lcm(|y|,|v|)}{|y|}, \frac{lcm(|y|,|v|)}{|v|})$; this is necessary for having the same "context" repeated at least once.

Then, it follows that any term in the sequence $\left[\frac{\overline{r}_i}{\overline{s}_i}\right]$, for any $i \geq \widetilde{n}$, can be written as $\left[\frac{k_1k_2^jk_3}{q_1q_2^jq_3}\right]$, where $|k_l| = |q_l|$, $1 \leq l \leq 3$, and $j \geq 1$ (see Figure 2 for a particular case).

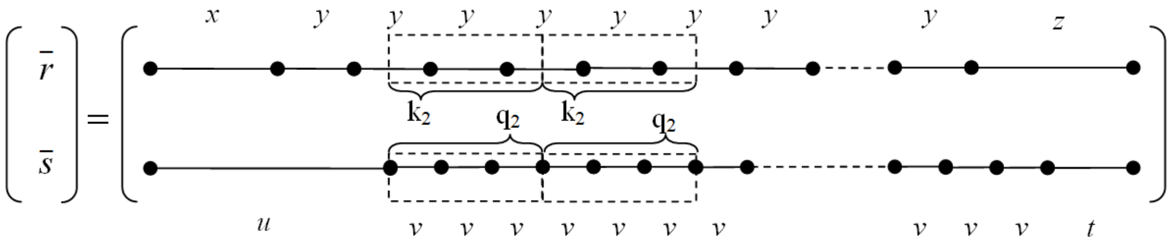

Figure 2: Here is presented a particular case of a double stranded structure (when $|u| > |x|$). For instance, here we have that $q_1 = u$ and $q_2 = v^3$.

Finally, by folding the string $k_1k_2^jk3$ with respect to $q_1q_2^jq_3$, one gets $m_1m_2^jm_3m_4^jm_5$. The string k_1 folded according with q_1 produces m_3. The string k_2 folded according with q_2 may produce a string m_2 ahead of m_3 and a string m_4 behind of m_3, but such that $|m_2m_4| \geq 1$ (because $|k_2| \geq 1$). By repeating the above procedure for all strings k_2 and q_2, one gets $m_2^jm_3m_4^j$. Eventually, by folding k_3 with respect to q_3, one obtains $m_1m_2^jm_3m_4^jm_5$.

Based on the above arguments, the following result holds true.

Lemma 10. *Let $L \in \mathcal{F}(REG, REG)$ be an infinite language. Then there exists a positive constant n such that if $w \in L$, $|w| \geq n$ then w can be decomposed as $w = uvwxy$ such that $|vx| \geq 1$, $|vwx| \leq n$, and $w_i = uv^iwx^iy \in L$, for any $i \geq 0$.*

Using Lemma 10, one can easily prove that $L = \{a^{2n}b^{2n}c^{2n} | n \geq 0\} \notin \mathcal{F}(REG, REG)$. Consequently, one can prove that

Proposition 11. $\mathcal{F}(REG, REG) \subset \mathcal{F}(CF, REG)$.

A similar result stands also in case of a infinite language $L \in \mathcal{F}(CF, REG)$. More precisely, let $L_1 \subseteq \Sigma^*$ be a infinite context-free language and $L_2 \subseteq \{u, d\}^*$ be a infinite regular languages and assume that $L = L(\Phi(L_1, L_2))$. Using the same type of arguments as in the proof of Lemma 10 it follows that there exists $r \in L_1$ and $s \in L_2$ such that $|r| = |s| \geq n = max(n_{L1}, n_{L_2})$, where n_{L_1} is the constant for L_1 given by the pumping lemma for context-free languages and n_{L_2} is the constant for L_1 given by the pumping lemma for regular languages. According to these lemmas we have

- r can be decomposed into $r = x_1x_2x_3x_4x_5$, where $x_i \in \Sigma^*$, $1 \leq i \leq 5$, such that $|x_2x_4| \geq 1$, $|x_2x_3x_4| \leq n_{L_1} \leq n$, and $r_i = x_1x_2^ix_3x_4^ix_5 \in L_1$, for $i \geq 0$;
- s can be decomposed into $s = uvt$, where $u, v, t \in \{u, d\}^*$, such that $|uv| \leq n_{L_2} \leq n$, $v \geq 1$, and $s_i = uv^it \in L_2$, for $i \geq 0$.

Based on the above relations one can define the sequences $(a_n)_n \subseteq I\!N$ and $(b_n)_n \subseteq I\!N$ such that $|r_{a_i}| = |s_{b_i}|$, for $i \in I\!N$. More precisely if we denote by $c_1 = |x_1| + |x_3| + |x_5|$, $c_2 = |x_2| + |x_4|$, and $c_3 = |u| + |t|$, $c_4 = v$, then the following relations have to be true.

$$c_1 + a_ic_2 = c_3 + b_ic_4 \text{ , hence } b_i = \frac{c_1 - c_3 + a_ic_2}{c_4}$$

Taking now

$$a_i = \begin{cases} (c_4i - 1)(c_1 - c_3) & \text{, if } c_1 \geq c_3 \\ (c_4i - 1)(c_3 - c_1) & \text{, if } c_3 > c_1 \end{cases} \quad i \in I\!N, i > 0$$

it follows that the subsequences $\overline{r}_i = x_1x_2^{a_i}x_3x_4^{a_i}x_5$, for $i > 0$, and $\overline{s}_i = uv^{b_i}t$, for $i > 0$, have equal lengths, that is $|\overline{r}_i| = |\overline{s}_i|$, for $i > 0$.

Considering the double stranded structures $\left[\frac{\overline{r}_i}{\overline{s}_i}\right]$, for any $i \geq 0$, then it follows that, starting from a certain positive constant $\widetilde{n}$ (that depends on L_1 and L_2), in the double stranded structure $\left[\frac{\overline{r}_j}{\overline{s}_j}\right]$, with $j \geq \widetilde{n}$, the substring from $\overline{s}_j$ composed by multiple concatenations of v underlays (partially) the one from $\overline{r}_j$ composed (in order) by multiple concatenations of x_2, x_3, and multiple concatenations of x_5 (because the substrings x_1 and x_3 have constant lengths). Moreover, assume that $\widetilde{n}$ is the first rank in the double stranded sequence such that the following conditions are satisfied:

- the substring from r_j composed by multiple concatenation of x_2 overlaps the substring from s_j composed by multiple concatenations of v; assume that the overlapping part is equal or longer than $min(\frac{lcm(|x_2|,|v|)}{|x_2|}, \frac{lcm(|x_2|,|v|)}{|v|})$;
- the substring from r_j composed by multiple concatenation of x_4 overlaps the substring from s_j composed by multiple concatenations of v; assume that the overlapping part is equal or longer than $min(\frac{lcm(|x_4|,|v|)}{|x_4|}, \frac{lcm(|x_4|,|v|)}{|v|})$.

Using similar arguments as in the proof of Lemma 10, it follows that any term in the sequence $\left[\frac{\overline{r}_i}{\overline{s}_i}\right]$, for any $i \geq \widetilde{n}$, can be written as $\left[\frac{k_1k_2^hk_3k_4^jk_5}{q_1q_2^hq_3q_4^jq_5}\right]$, where k_l, q_l, $1 \leq l \leq 5$, are strings (of finite lengths), $|k_l| = |q_l|$, $1 \leq l \leq 5$, and $h, j \geq 1$. Moreover, we have that $|k_2k_4| \geq 1$ (because k_2k_4

contains at least one symbol from x_2x_4 and $|x_2x_4| \geq 1$). Although in general $h \neq j$, one can decompose $\overline{r}_i$ and $\overline{s}_i$ as

$$\left[\frac{k_1k_2^hk_3k_4^{j-h}k_4^hk_5}{q_1q_2^hq_3q_4^{j-h}q_4^hq_5}\right], \text{ if } j \geq h, \text{ or } \left[\frac{k_1k_2^jk_2^{h-j}k_3k_4^jk_5}{q_1q_2^jq_2^{h-j}q_3q_4^jq_5}\right], \text{ if } h \geq j.$$

Furthermore, based on the above construction, one can actually build an infinite sequence of type

$$\left[\frac{\alpha_1\alpha_2^i\alpha_3\alpha_4^i\alpha_5}{\beta_1\beta_2^i\beta_3\beta_4^i\beta_5}\right], \text{ for } i \geq 0, \text{ where } |\alpha_l| = |\beta_l|, 1 \leq l \leq 5, |\alpha_2\alpha_4| \geq 1 \text{ and } |\alpha_2\alpha_3\alpha_4| \leq \widetilde{n}.$$

Consequently, by folding $\alpha_1\alpha_2^i\alpha_3\alpha_4^i\alpha_5$ according with $\beta_1\beta_2^i\beta_3\beta_4^i\beta_5$, for $i \geq 0$, one gets a string $w_i = w_1w_2^iw_3w_4^iw_5w_6^iw_7w_8^iw_9$, hence the following result is true.

Lemma 12. *Let $L \in \mathcal{F}(CF, REG)$ be an infinite language. Then there exists a positive constant n such that if $w \in L$, $|w| \geq n$ then w can be decomposed as $w = w_1w_2w_3w_4w_5w_6w_7w_8w_9$ such that*

$$|w_2w_4w_6w_8| \geq 1$$
$$|w_2w_3w_4w_5w_6w_7w_8| \leq n$$

and $w_i = w_1w_2^iw_3w_4^iw_5w_6^iw_7w_8^iw_9 \in L$, for any $i \geq 0$.

Using Lemma 12 one can prove that the language $L = \{ww \mid w \in \{a,b\}^*\} \notin \mathcal{F}(CF, REG)$. Consequently, the following result holds true.

Proposition 13. *$\mathcal{F}(CF, REG) \subset \mathcal{F}(CF, CF)$.*

Furthermore, one can show the following result.

Proposition 14. *$\mathcal{F}(CS, CS) = CS$.*

Proof: Assume that L_1 and L_2 are arbitrary context-sensitive languages. Then one can construct a linear bounded automaton M such that $L(M) = L(\Phi(L_1, L_2))$ and which works as follows. M uses one read only input tape and three auxiliary tapes. A word w is placed on the input tape. The first auxiliary tape is used by M to nondeterministically generate a word w_1 of the same length as the one present on the input tape, and then to check if $w_1 \in L_1$. The second auxiliary tape is used by M to nondeterministically generate a word w_2 of the same length as the one present on the input tape, and then to check if $w_2 \in L_1$. Finally, if $w_1 \in L_1$ and $w_2 \in L_2$, then M uses the third tape to simulate the folding of w_1 according with w_2 and to check whether or not the result is equal to w. Because all the above mentioned tasks can be performed in linear space with respect to the length of the input, then it follows that $L(\Phi(L_1, L_2)) \in CS$. Moreover, one can notice that all context-sensitive languages can be generated (to prove this, one can use a similar construction as in the proof of Proposition 4). Consequently, we have that $\mathcal{F}(CS, CS) = CS$. □

Because the class o context-sensitive languages CS contains non semilinear languages, then using Remark 3.2 it follows that

Proposition 15. *$\mathcal{F}(CF, CF) \subset \mathcal{F}(CS, CS) = CS$.*

Using a similar construction as in the proof of Proposition 14 one can prove that

Proposition 16. $\mathcal{F}(REC, REC) = REC$.

Proposition 17. $\mathcal{F}(RE, RE) = RE$.

Finally we can sum up all the results in the following hierarchy:

Theorem 18.

$$\begin{array}{lllllllll} REG & \subset & \mathcal{F}(REG, REG) & \subseteq & CF & \subset & \mathcal{F}(CF, REG) & \subset \\ \mathcal{F}(CF, CF) & \subset & \mathcal{F}(CS, CS) & = & CS & \subset & \mathcal{F}(REC, REC) & = \\ REC & \subset & \mathcal{F}(RE, RE) & = & RE. \end{array}$$

4 Conclusion

In this paper we introduced a new computing paradigm called *folding systems*. We investigated their computational power under certain constraints and we studied some of their properties.

During our endeavor several open problems arose. One of them regards the existence of a pumping lemma for folding systems where both the core language and the folding procedure set are context-free languages. Another open problem regards the existence of a polynomial time parsing algorithm for folding systems which use regular/context-free languages as core/folding procedure set. Future investigations will focus on variants of folding systems where other families of languages will be used (for example those generated by Lindenmayer systems), but also on defining more complex and realistic methods for performing the string folding.

We conclude with the belief that many interesting problems can be pursued in this line of research.

Acknowledgment

The work of the author was supported by the CNCSIS IDEI-PCE grant, no. 551/2009, Romanian Ministry of Education, Research and Innovation.

Bibliography

[1] T. Head, Parallel Computing by Xeroxing on Transparencies, *Algorithmic Bioprocesses*, 9, pp. 631–637, Springer-Verlag, Berlin, 2009.

[2] J. E. Hopcroft, J. D. Ullman, *Introduction to Automata Theory, Languages and Computation*, Addison-Wesley Publishing, Reading Massachusetts, 1979.

[3] A. Mateescu, G. Rozenberg, A. Salomaa, Shuffle on Trajectories: Syntactic Constraints, *Theoretical Computer Science*, 197, 1-2, pp. 1–56, 1998.

[4] M. Minsky, *Computation – Finite and Infinite Machines*. Prentice Hall, Englewood Cliffs, NJ, 1967.

[5] R. J. Parikh. On Context–free languages. *Journal of the Association for Computing Machinery*, 13 (4), pp. 570–581, 1966.

[6] G. Păun, G. Rozenberg, A. Salomaa, Grammars Based on the Shuffle Operation, *Journal of Universal Computer Science*, 1 (1), pp. 67–82, 1995.

[7] P.W.K. Rothemund, Folding DNA to create nanoscale shapes and patterns, *Nature* (440), pp. 297–302, 2006.

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

[9] G. Rozenberg, Gene Assembly in Ciliates: Computing by Folding. *A Half-Century of Automata Theory*, World Scientific Publishing Company, pp. 93–130, 2000.