"Dogmatic" P Systems*

José M. Sempere

Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
`jsempere@dsic.upv.es`

Summary. In this work we propose a variant of P systems based on the Central Dogma of Molecular Biology which establishes the transformation of DNA strands into protein products by applying different string transformation such as transductions and transcriptions. We introduce a new kind of worm object rules to carry out transducion operations. Finally, we establish the universality of the proposed model by simulating Iterated finite state sequential transducers (IFTs).

1 Introduction

P systems [15] were introduced as a computational model inspired by the information and biochemical product processing of living cells through the use of membrane communication. In most of the works about P systems, information is represented as multisets of symbol/objects which can interact and evolve according to predefined rules. Nevertheless, the use of strings to represent the information and the use of rules to transform strings instead of multisets of objects have always been present in the literature of this scientific area. So, in his mostly referred book [15], Gh. Păun overviews the use of string rules in P systems. Different variants of string-based P systems have been proposed along the time. We can mention *rewriting P systems* [11], referred as membrane systems with *worm objects* [2] in the case of genomic operations, *insertion-deletion P systems* [6] and *splicing P systems* [14], among others. Observe that most of these models have been used for language generation [12]. In [5, 7], the proposal of *hybrid P systems* introduces the use of contextual rules and Chomsky rules to achieve universality by generating all the recursively enumerable languages. Recently, in [13] a variant of P systems with worm objects and evolutionary based operations has been introduced to simulate Networks of Evolutionary Processors, hence to achieve universality.

In this work, we propose a variant of P systems with worm objects and a new kind of worm rules based on the central dogma of molecular biology which sets

* Work supported by the Spanish Ministerio de Educación y Ciencia under project TIN2007-60769

the framework to obtain protein products from DNA strands by applying, among others, transduction and transcription operations.

The structure of this work is as follows: In section 2 we introduce basic concepts and notation on formal language theory, iterated transductions, P systems and molecular biology related to the *Central Dogma*. Then, we will define the dogmatic rules in regions which transduce (fragments of) worm objects into (fragments of) worm objects. We will propose a simulation of iterated transductions with the new proposed model in order to achieve universality. Finally, we will outline future research related to this work.

2 Basic Concepts

We start by summarizing the notions used throughout this work. An *alphabet* is a finite and nonempty set of symbols. Any finite sequence of symbols from an alphabet V is called *word* or *string* over V. The set of all words over V is denoted by V^*. A *language* over the alphabet V is any subset of V^*.

A grammar is a construct $G = (N, \Sigma, P, S)$ where N and Σ are the alphabets of auxiliary and terminal symbols with $N \cap \Sigma = \emptyset$, $S \in N$ is the *axiom* of the grammar and P is a finite set of productions in the form $\alpha \to \beta$, where $\alpha \in (N \cup \Sigma)^* N (N \cup \Sigma)^*$ and $\beta \in (N \cup \Sigma)^*$. The language of the grammar is denoted by $L(G)$ and it is the set of terminal strings that can be obtained from S by applying symbol substitutions according to P. Formally, $w_1 \underset{G}{\Rightarrow} w_2$ if $w_1 = u\alpha v$, $w_2 = u\beta v$ and $\alpha \to \beta \in P$. We will denote by $\overset{*}{\underset{G}{\Rightarrow}}$ the reflexive and transitive closure of $\underset{G}{\Rightarrow}$. So, the language generated by G is defined by the set $L(G) = \{w \in \Sigma^* : S \overset{*}{\underset{G}{\Rightarrow}} w\}$.

Four larger families of languages generated by grammars can be defined: REG (regular), CF (context-free), CS (context-sensitive) and RE (recursively enumerable). The definition of these families comes from the restriction over the production forms in the grammar. The well known Chomsky's hierarchy establishes the inclusions $REG \subset CF \subset CS \subset RE$.

Iterated Transductions

In the following, we will introduce *Iterated finite state sequential transducers* (IFT) as it was defined in previous works ([1, 8, 10]).

An IFT is defined by the tuple $T = (Q, \Sigma, q_0, a_0, F, P)$, where Q is a finite set of *states*, Σ is an alphabet, $q_0 \in Q$ is an *initial state*, $a_0 \in \Sigma$ is a *starting symbol*, $F \subseteq Q$ is the set of *final states* and P is a finite set of *transduction rules* in the form (q, a, p, x) with $q, p \in Q$, $a \in \Sigma$ and $x \in \Sigma^*$ which we will write as $qa \to xp$. The transduction rule $qa \to xp$ means that if the finite control is in state q and it reads the symbol a then it changes to state p and writes x. We define a *direct transition step* as follows

$$uqav \vdash uwpv \text{ iff } qa \to wp \in P$$

The reflexive and transitive closure of $\vdash$ will be denoted by $\vdash^*$. We say that w derives x, and it will be denoted by $w \Longrightarrow x$, iff $q_0 w \vdash^* xp$, for $p \in Q$ (observe that p is any state in Q not necessarily final). We will denote the reflexive and transitive closure of $\Longrightarrow$ by $\Longrightarrow^*$. If in the previous derivation the process stops in a final state we will write $\overset{f}{\Longrightarrow}$ instead of $\Longrightarrow$. That is, $w \overset{f}{\Longrightarrow} x$, iff $q_0 w \vdash^* xp$, for $p \in F$. The language generated by T is defined as follows

$$L(T) = \{x \in \Sigma^* : a_0 \Longrightarrow^* w \overset{f}{\Longrightarrow} x, w \in \Sigma^*\}$$

We denote by IFT_n the family of languages generated by IFT with at most n states. The hierarchy of families in IFT_n has been completely explored, and it has been proved that it collapses at level four. We have the following results

Lemma 1.[10] $RE = IFT_4$; [1] $CS \subset IFT_3$; [10] $CF \subset IFT_2$.

In addition, IFTs have been related to the *computing by carving* paradigm [9] as a way to generate even non-recursively enumerable languages.

The Central Dogma of Molecular Biology

The *Central Dogma* of Molecular Biology is our source of inspiration for the variant of P system which we will propose later. We follow the ideas exposed in [4]. Mainly, the central dogma of molecular biology establishes a metaphor of how DNA strands in the living cell are transformed into protein products by means of information storage and transformation.

Mainly, a section of DNA (the gene) is transcribed to a molecule of messenger RNA and the mRNA is translated by the ribosome into a protein. In the eukaryotic organisms the mRNA molecule is processed, before translation, by splicing out certain subsequences called *introns*. The DNA is replicated before the transcription. The transcription is made by complementing the single DNA strand, and by substituting the thymine nucleotide by the uracil one in the RNA molecule. The translation from (spliced) mRNA to proteins is based on a mapping of nucleotide triplets called *codons* to amino acids with the help of transfer RNA (tRNA). Under a computer science point of view, the central dogma can be viewed as a sequence of well known operations over strings such as morphisms, transductions and splicing. The main ingredients that we will consider in the subsequent P system that we will propose are the followings:

- There are different processes in different regions. DNA duplication and DNA transcription to mRNA occurs in the nucleus of the cell, while mRNA translation to amino acids occurs in some cases in the endoplasmic reticulum with the membrane ribosome.
- There are different alphabet sizes and symbols involved in the operations. The DNA strands is a sequence of four different nucleotides: adenine (A), thymine (T), cytosine (C) and guanine (G), in the RNA the thymine (T) is substituted by the uracil (U), while the proteins are sequences over a twenty-letter alphabet (the amino acids)

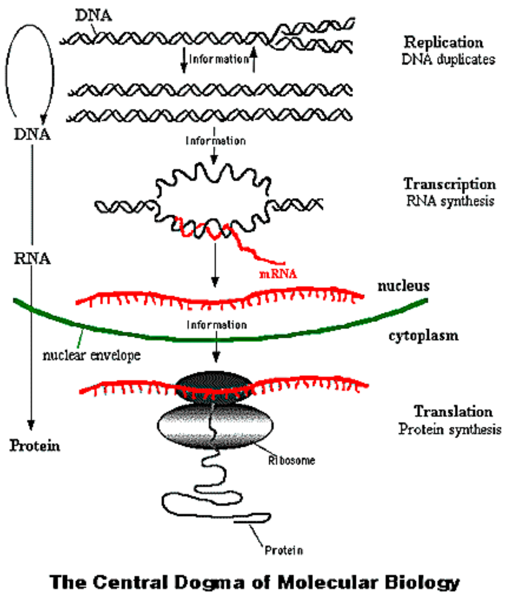


Fig. 1. The Central Dogma of Molecular Biology. (This picture has been taken from accessexcellence.org)

- Transcription and translation can be performed by alphabetic homomorphisms and finite transductions.
- There are different products at every stage which interacts into different regions. The DNA duplication, transcription and splicing needs the presence of different proteins and other molecular compounds. The proteins are the final product of the cycle DNA-RNA-protein.

3 Dogmatic P systems

In this section, we will propose a variant of P systems that work with worm objects in a transduction-like approach. First, we will introduce a new kind of region rules to work with.

A *dogmatic* rule is defined as follows

$$u : v_{pos} \longrightarrow w_{ad_1,ad_2,\cdots,ad_k}, \text{ where}$$

u, v are strings (worm objects), $pos \in \{l, r, *\}$ and for all $i : 1 \leq i \leq k$ $ad_i \in \{here, out, in_j\}$. The meaning is the following: Provided that there exist a worm object u in the region (we can omit the presence of u), all the worm objects with substring v at position pos (which means, rightmost one (r), leftmost one (l) or arbitrary position ($*$)) change substring v by w and send a copy of the new worm object at the regions defined by ad_i after eliminating the original worm object from the region.

Example 1. Let the region R have the rule r_1 defined as $eee : a_l \rightarrow bb_{here}$ and the worm objects *eee* and *abbcbaa*. Then after applying r_1 in the region, the worm objects are *eee* and *bbbbcbaa*.

If the rule r_1 is defined as $eee : a_r \rightarrow bb_{here}$, we obtain *abbcbabb* as a new worm object. Finally, if the rule is defined as $eee : a_* \rightarrow bb_{here}$ then we obtain the set of new strings $\{bbbbcbaa, abbcbbba, abbcbabb\}$. Observe that, in this case, we have previously obtained three copies of the initial string before applying the rule.

The rule $a_l \rightarrow bb_{here}$ can be applied over *baa* and it obtains the new string *bbba*. Here, we have omitted the presence of an additional string and the rule changes the leftmost appearance of a symbol a. □

The addressing label in_j, can be directly applied to contiguous regions at the same level. That is, if there exist regions j and i inside the same region, then a rule at region i can send worm objects to region j directly.

We can observe that the dogmatic rules capture the following aspects from the Central Dogma of Molecular Biology:

- The rules transform parts of a string into a new substring as in transcription and transduction.
- The rules make copies of the target string before transformation as in DNA replication.
- The rules need the presence of other objects to be applied.
- The rules can address contiguous regions (i.e. RNA moving from nucleus to ribosomes).

Now, we will define a *Dogmatic P system*[2] as the following construct

$$\Pi = (V, \mu, A_1, \cdots, A_m, (R_1, \rho_1), \cdots, (R_m, \rho_m), i_0), \text{ where:}$$

- V is an alphabet
- μ is a membrane structure consisting of m membranes
- A_i, $1 \leq i \leq m$ is a finite set of strings associated with the region i (the *axioms*)
- R_i, $1 \leq i \leq m$ is a finite set of *dogmatic rules* over V associated with the ith region and ρ_i is a partial order relation over R_i specifying a *priority*
- i_0 is a number between 1 and m and it specifies the *output* membrane of Π (in the case that it equals to ∞ the output is read outside the system).

[2] Different acronyms were candidates for naming Dogmatic P systems. Among others, *dP* systems were considered but it was previously used by other authors in a different context. Another acronym was *dogP* but the author thinks that, in such a case, catalyzers will never be used in this context given that "*dogs*" and "*cats*" could not cooperate and living in the same regions. We leave open the search for a good acronym for the proposed Dogmatic *P* systems.

Initially, the system holds the set of axioms at every region. Then, in a fully parallel manner all the rules are applied over the strings defined at every region. The system halts whenever no rule can be applied at any region.

The language generated by Π is the set of worm objects collected at region i_0. In the case that $i_0 = \infty$, the language is collected in *external mode* as the set of strings in the environment. The language generated by Π is denoted by $L(\Pi)$. Observe that if the language is infinite then the system will never halt so it will add new worm objects to the output region or the environment.

Observe that this proposal is different from [3] where the authors propose a membrane system framework with symport/antyport rules to perform different types of transductions. In that work the proposed system operates with strings by taking every symbol of the input string of the environment (outside the membrane system) and putting every symbol of the transduced string in the environment. Here, we will avoid symport/antyport rules and we will work with strings in a worm object approach.

4 A Simulation of Iterated Transductions by Dogmatic P Systems

In this section, we will show a simulation of IFTs with n states by dogmatic P Systems. Our approach will use n regions inside the skin one in order to simulate the n states of the IFT. The transitions of the IFT will be simulated by using the direct address in_j. We will need to mark some symbols in order to carry out the transduction from left to right. In addition, we will use different alphabets to avoid a wrong application of the transduction rules at different symbols, and to prevent that the simulation goes on even if the IFT cannot carry out a complete transduction.

Let $T = (Q, \Sigma, q_0, a_0, F, P)$ be an IFT with $Q = \{q_0, \cdots, q_n\}$. Then, we propose the following dogmatic P system

$$\Pi = (V, \mu, A, A_0, \cdots, A_n, (R, \rho), (R_0, \rho_0), \cdots, (R_n, \rho_n), \infty), \text{ where}$$

- $V = \Sigma \cup \hat{\Sigma} \cup \breve{\Sigma} \cup \{\#\}$, where $\hat{\Sigma} = \{\hat{a} : a \in \Sigma\}$ and $\breve{\Sigma} = \{\breve{a} : a \in \Sigma\}$
- $\mu = [[_0]_0, \cdots, [_n]_n]$ (we have omitted a label for the skin region).
- $A_0 = \{\#a_0\}$, $A = \emptyset$, and for all $i : 1 \leq i \leq n$ $A_i = \emptyset$.
- **Type (a) rules:** For every rule $q_0 a \rightarrow v q_j \in P$, we add the rule $\#a_l \rightarrow \#\hat{v}_{in_j}$ if $q_j \neq q_0$ or the rule $\#a_l \rightarrow \#\hat{v}_{here}$ if $q_j = q_0$ to R_0
- **Type (b) rules:** For every rule $q_i a \rightarrow v q_j \in P$, and for every symbol $\hat{b} \in \hat{\Sigma}$ we add the rule $\hat{b} a_l \rightarrow \hat{b}\hat{v}_{in_j}$ if $q_i \neq q_j$ or the rule $\hat{b} a_l \rightarrow \hat{b}\hat{v}_{here}$ if $q_i = q_j$ to R_i
- **Type (c) rules:** For every region R_i and for every pair of symbols $\hat{a} \in \hat{\Sigma}$ and $b \in \Sigma$ add the following rule $\hat{a} b_l \rightarrow \hat{a} b_{here}$
- **Type (d) rules:** For every region R_i such that $q_i \in F$, and for every symbol $\hat{a} \in \hat{\Sigma}$ add the following rule $\hat{a}_r \rightarrow \breve{a}_{out}$

- **Type (e) rules:** For every region R_i such that $q_i \notin F$, and for every symbol $\hat{a} \in \hat{\Sigma}$ add the following rule $\hat{a}_r \to \hat{a}_{out}$
- **Type (f) rules:** Add to R the rules $\{\hat{a}_l \to a_{here} : a \in \Sigma\}$
- **Type (g) rules:** Add to R the rules $\{\breve{a}_l \to a_{in_0,out}\}$
- **Type (h) rule:** $\#_l \to \#_{in_0}$

We will explain the rules in the system as follows: Type (a) rules start the transduction of the string from the initial state. Hence, we use the # symbol as a left delimiter of the string to be transduced. The alphabet $\hat{\Sigma}$ is used to mark the symbols that have been transduced during a derivation process. Type (b) rules simulate the transitions in the transducer. Observe that we use the address in_j to change the state in the finite control and the address *here* to simulate the transducer loops. Type (c) rules are used to block the strings that cannot be completely transduced (observe that the IFT can be non complete and it would not finish the derivation process). Type (d) rules are used to output the transduced strings that arrive to a final state. Here, we use the alphabet $\breve{\Sigma}$ to mark the strings that belong to the language generated by the transducer. Type (e) rules are used to output the transduced strings that arrive to a non final state.

The priorities of the rules in regions R_i keep the following order: Type (a) rules $>$ Type (b) rules $>$ Type (c) rules $>$ Type (d) and Type (e) rules.

The rules of the skin region are explained as follows: Type (f) rules are used to restore the string symbols of the transduced string in order to feed-back the transducer with a new input string (hence, it performs the iteration in the transduction). Type (g) rules are used to restore the symbols from those transduced strings that come from a final state (hence, they belong to the language generated by iterating the transducer). In such a case, one copy of the string is sent out the environment while another copy is sent in the region zero in order to feed-back the transducer. Finally, the rule of type (g) is used to send the transduced string into the initial region to iterate a new transduction.

If a string $w \in L(T)$, then $\#w \in L(\Pi)$. We can observe that the transitions from T are simulated by the P system by means of the rules of type (a) and (b). The iteration is carried out at the skin region by applying rules of type (g) or (h) (after restoring the symbols with rules of type (f). If the transduced string arrives to a final state, then rules of type (g) are applied and the string with the left mark # outputs the system.

Example 2. Let us consider the finite transducer defined through the following transition diagram, with a as the starting symbol

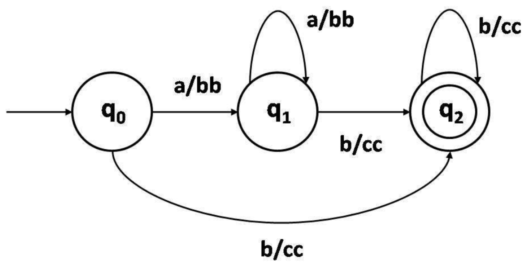
The proposed dogmatic P system is defined with a membrane structure $[[_0]_0, [_1]_1, [_2]_2]$, and the following dogmatic rules

Skin region rules

$r_1 : \hat{a}_l \to a_{here}$ $r_4 : \breve{a}_l \to a_{in_0,out}$ $r_{45} : \#l \to \#_{in_0}$

$r_2 : \hat{b}_l \to b_{here}$ $r_5 : \breve{b}_l \to b_{in_0,out}$

$r_3 : \hat{c}_l \to c_{here}$ $r_6 : \breve{c}_l \to c_{in_0,out}$



with ρ defined as $\{r_1, r_2, r_3\} > \{r_4, r_5, r_6\} > r_{45}$, and $A = \emptyset$.

Region 0 rules

$r_7 : \#a_l \to \#\hat{b}\hat{b}_{in_1}$ $r_9 : \hat{a}a_l \to \hat{a}\hat{b}\hat{b}_{in_1}$ $r_{12} : \hat{a}b_l \to \hat{a}\hat{c}\hat{c}_{in_2}$
$r_8 : \#b_l \to \#\hat{c}\hat{c}_{in_2}$ $r_{10} : \hat{b}a_l \to \hat{b}\hat{b}\hat{b}_{in_1}$ $r_{13} : \hat{b}b_l \to \hat{b}\hat{c}\hat{c}_{in_2}$
$r_{11} : \hat{c}a_l \to \hat{c}\hat{b}\hat{b}_{in_1}$ $r_{14} : \hat{c}b_l \to \hat{c}\hat{c}\hat{c}_{in_2}$

$r_{15} : \hat{a}a_l \to \hat{a}a_{here}$ $r_{18} : \hat{b}a_l \to \hat{b}a_{here}$ $r_{21} : \hat{c}a_l \to \hat{a}a_{here}$
$r_{16} : \hat{a}b_l \to \hat{a}b_{here}$ $r_{19} : \hat{b}b_l \to \hat{b}b_{here}$ $r_{22} : \hat{c}b_l \to \hat{c}b_{here}$
$r_{17} : \hat{a}c_l \to \hat{a}c_{here}$ $r_{20} : \hat{b}c_l \to \hat{b}c_{here}$ $r_{23} : \hat{c}c_l \to \hat{c}c_{here}$

$r_{24} : \hat{a}_r \to \hat{a}_{out}$
$r_{25} : \hat{b}_r \to \hat{b}_{out}$
$r_{26} : \hat{c}_r \to \hat{c}_{out}$

with ρ_0 defined as $\{r_7, r_8\} > \{r_9, r_{10}, r_{11}, r_{12}, r_{13}, r_{14}\} > \{r_{15}, r_{16}, r_{17}, r_{18}, r_{19}, r_{20}, r_{21}, r_{22}, r_{23}\} > \{r_{24}, r_{25}, r_{26}\}$, and $A_0 = \{\#a\}$

Region 1 rules

$r_{27} : \hat{a}a_l \to \hat{a}\hat{b}\hat{b}_{here}$ $r_{30} : \hat{a}b_l \to \hat{a}\hat{c}\hat{c}_{in_2}$
$r_{28} : \hat{b}a_l \to \hat{b}\hat{b}\hat{b}_{here}$ $r_{31} : \hat{b}b_l \to \hat{b}\hat{c}\hat{c}_{in_2}$
$r_{29} : \hat{c}a_l \to \hat{c}\hat{b}\hat{b}_{here}$ $r_{32} : \hat{c}b_l \to \hat{c}\hat{c}\hat{c}_{in_2}$

$r_{33} : \hat{a}a_l \to \hat{a}a_{here}$ $r_{36} : \hat{b}a_l \to \hat{b}a_{here}$ $r_{39} : \hat{c}a_l \to \hat{c}a_{here}$
$r_{34} : \hat{a}b_l \to \hat{a}b_{here}$ $r_{37} : \hat{b}b_l \to \hat{b}b_{here}$ $r_{40} : \hat{c}b_l \to \hat{c}b_{here}$
$r_{35} : \hat{a}c_l \to \hat{a}c_{here}$ $r_{38} : \hat{b}c_l \to \hat{b}c_{here}$ $r_{41} : \hat{c}c_l \to \hat{c}c_{here}$

$r_{42} : \hat{a}_r \to \hat{a}_{out}$
$r_{43} : \hat{b}_r \to \hat{b}_{out}$
$r_{44} : \hat{c}_r \to \hat{c}_{out}$

with ρ_1 defined as $\{r_{27}, r_{28}, r_{29}, r_{30}, r_{31}, r_{32}\} > \{r_{33}, r_{34}, r_{35}, r_{36}, r_{37}, r_{38}, r_{39}, r_{40}, r_{41}\} > \{r_{42}, r_{43}, r_{44}\}$, and $A_1 = \emptyset$

Region 2 rules

$r_{45} : \hat{a}b_l \rightarrow \hat{a}\hat{c}\hat{c}_{here}$
$r_{46} : \hat{b}b_l \rightarrow \hat{b}\hat{c}\hat{c}_{here}$
$r_{47} : \hat{c}b_l \rightarrow \hat{c}\hat{c}\hat{c}_{here}$

$r_{48} : \hat{a}a_l \rightarrow \hat{a}a_{here}$ $r_{51} : \hat{b}a_l \rightarrow \hat{b}a_{here}$ $r_{54} : \hat{c}a_l \rightarrow \hat{c}a_{here}$
$r_{49} : \hat{a}b_l \rightarrow \hat{a}b_{here}$ $r_{52} : \hat{b}b_l \rightarrow \hat{b}b_{here}$ $r_{55} : \hat{c}b_l \rightarrow \hat{c}b_{here}$
$r_{50} : \hat{a}c_l \rightarrow \hat{a}c_{here}$ $r_{53} : \hat{b}c_l \rightarrow \hat{b}c_{here}$ $r_{56} : \hat{c}c_l \rightarrow \hat{c}c_{here}$

$r_{57} : \hat{a}_r \rightarrow \breve{a}_{out}$
$r_{58} : \hat{b}_r \rightarrow \breve{b}_{out}$
$r_{59} : \hat{c}_r \rightarrow \breve{c}_{out}$
with ρ_2 defined as $\{r_{45}, r_{46}, r_{47}\} > \{r_{48}, r_{49}, r_{50}, r_{51}, r_{52}, r_{53}, r_{54}, r_{55}, r_{56}\} > \{r_{57}, r_{58}, r_{59}\}$, and $A_2 = \emptyset$

□

From the previous proposed P system and other works previously referred we get the following result.

Theorem 1. Every recursively enumerable language can be generated by a dogmatic P system.

Proof. The result comes from the simulation of IFTs by dogmatic P systems that we have proposed before. Given that any recursively enumerable can be generated by an IFT with four states [10] then we have the result. □

5 Conclusions and future work

In this paper we have proposed new kinds of rules for P system in which we have been inspired by the Central Dogma of Molecular Biology. The P systems that we have proposed are a suitable framework to generate languages. We think that these kind of rules will help in the construction of systems for biological simulations due to its inspiration from nature.

Our future research will focus on the power of these systems to transduce formal languages with no iteration. Hence, we will study the simulation of rational and recognizable transductions and the simulation of (restricted) *gsms*. In addition, the framework to accept languages of strings or their Parikh mappings (which is the natural framework of P systems) should be explored too. Finally, due to the relation between IFTs and *Computing by carving* we should explore the possibility of applying membrane systems to that paradigm, as a continuation of a previous work [16].

References

1. H. Bordihn, H. Fernau, M. Holzer, V. Manca, C. Martín-Vide. Iterated sequential transducers as language generating devices. *Theoretical Computer Science* 369, pp 67-81. 2006.
2. J. Castellanos, Gh. Păun, A. Rodríguez-Patón. P systems with worm-objects. In Proc. of the Seventh International Symposium on String Processing Information Retrieval (SPIRE'00), pages 64-74, A Coruña, Spain, September 2000. IEEE Computer Society.
3. G. Ciobanu, G. Păun, G. Stefănescu. P Transducers. *New Generation Computing* 24, pp 1-28, 2006.
4. W. W. Cohen. A Computer Scientist's Guide to Cell Biology. Springer, 2007.
5. S.R. Krishna, K. Lakshmanan, R. Rama. Hybrid P systems. *Romanian Journal of Information Science and Technology*, 4(1-2):111-123, 2001.
6. S.R. Krishna, R. Rama. Insertion-Deletion P systems. Proc. of Workshop on DNA-Based Computers, pp 360-370, Springer LNCS 2340. 2002.
7. M. Madhu, K. Krithivasan. A note on hybrid P systems. *Grammars*, 5(3):239-244, December 2002.
8. V. Manca. On the Generative Power of Iterated Transduction. In *Words, Semigroups, & Transductions* (M. Ito, G. Păun, S. Yu, eds.) pp 315-327. World Scientific, 2001.
9. V. Manca, C. Martín-Vide, Gh. Păun. New computing paradigms suggested by DNA computing: computing by carving. *BioSystems* 52, pp 47-54. 1999.
10. V. Manca, C. Martín-Vide, Gh. Păun. Iterated GSM Mappings: A Collapsing Hierarchy. In *Jewels are Forever* (J. Karhumäki et al., eds.) pp 182-193. Springer, 1999.
11. C. Martín-Vide, Gh. Păun. String-objects in P systems. In Proc. of Algebraic Systems, Formal Languages and Computations Workshop, pages 161-169, Kyoto, 2000. RIMS Kokyuroku, Kyoto Univ.
12. C. Martín-Vide, Gh. Păun. Language generating by means of membrane systems. *Bulletin of the EATCS*, (75):199-218, October 2001.
13. V. Mitrana, J.M. Sempere. Accepting Evolutionary P Systems. Proc. 10th Workshop on Membrane Computing WMC10. pp 552-555. Universidad de Sevilla. 2009.
14. A. Păun, M. Păun. On Membrane Computing Based on Splicing. *Words, Sequences, Languages*, pp 409-422. Kluwer Academic Publishers. 2000.
15. Gh. Păun. *Membrane Computing. An Introduction.* Springer. 2002.
16. J.M. Sempere. Computing by Carving with P Systems. A First Approach. In 6th Brainstorming Week on Membrane Computing BWMC6. pp 255-260. Fénix Editora, 2008.