Simulating Accepting Networks of Evolutionary Processors with Filtered Connections by Accepting Evolutionary P Systems (extended abstract)

Juan Castellanos[1] **Victor Mitrana**[2] **Eugenio Santos**[2]
José M. Sempere[3]

[1]Department of Artificial Intelligence, Faculty of Informatics
Polytechnic University of Madrid, 28660 Boadilla del Monte, Madrid, Spain.
E-mail: jcastellanos@fi.upm.es

[2]Department of Organization and Structure of Information,
University School of Informatics,
Polytechnic University of Madrid, Crta. de Valencia km. 7 - 28031 Madrid, Spain.
E-mail: victor.mitrana@upm.es, esantos@eui.upm.es

[3]Department of Information Systems and Computation,
Polytechnic University of Valencia
Camino de Vera, s/n, 46022 Valencia, Spain.
E-mail: jsempere@dsic.upv.es

Abstract. In this work, we propose a variant of P system based on the rewriting of string-objects by means of evolutionary rules. The membrane structure of such a P system seems to be a very natural tool for simulating the filters in accepting networks of evolutionary processors with filtered connections. We discuss an informal construction supporting this simulation. A detailed proof is to be considered in an extended version of this work.

Keywords: Network of Evolutionary Processors with Filtered Connections, Evolutionary P system.

1 Introduction

A rather informal idea of what a network of evolutionary processor is consists of a virtual (complete) graph in which each node hosts a very simple processor called an evolutionary processor. This is a pretty common architecture for parallel and distributed symbolic processing, related to the Connection Machine [5] as well as the Logic Flow paradigm [3]. By an evolutionary processor we mean a processor which is able to perform very simple operations, namely point mutations in a DNA sequence (insertion, deletion or substitution of a pair of nucleotides). More generally, each node may be viewed as a cell having genetic information encoded

in DNA sequences which may evolve by local evolutionary events, that is point mutations.

Each node processor, which is specialized just for one of these evolutionary operations, acts on the local data and then local data becomes a mobile agent which can navigate in the network following a given protocol. Only that data which is able to pass a filtering process can be communicated. This filtering process may require to satisfy some conditions imposed by the sending processor, by the receiving processor or by both of them. All the nodes send simultaneously their data and the receiving nodes handle also simultaneously all the arriving messages, according to some strategies, see [4, 5]. The filtering process that is to be considered here is regulated by some context conditions associated to the edges of the graph. This is the model introduced in [2] under the name of accepting network of evolutionary processors with filtered connections (ANEPFC). The reader interested in a survey of the main results regarding ANEPFCs is referred to [9].

P systems [13] 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 multiset objects have always been present in the literature of this scientific area. So, in his mostly referred book [13], 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* [10], referred as membrane systems with *worm objects* [1] in the case of genomic operations, *insertion-deletion P systems* [7] and *splicing P systems* [12], among others. Observe that most of these models have been used for language generation [11]. In [6, 8], 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.

In this work, we propose the use of evolutionary rules of string rewriting in all regions of a P system. The idea is not new (see for instance [7]) but our approach has two different main goals. First, the P systems considered here define languages by an accepting process in contrast with the generating variants widely considered so far in the area of membrane computing. Second, we show that the membrane structure is indispensable for simulating the filtering process in an ANEPFC. This is also in contrast with very many constructions of P systems in which the membrane structure plays actually a very minor role, most of them being reduced to just one membrane. The main part of this note is the informal construction of an evolutionary P system simulating an ANEPFC with emphasis on the membrane hierarchies of each region associated with a filtered connection.

The structure of this work is as follows: In section 2 and 3 we recall the definition of ANEPFCs and evolutionary P systems, respectively. Then, we informally describe a construction of an evolutionary P system simulating a given

ANEPFC. More precisely, we discuss the evolutionary rules in each region and the membrane structure associated with each filtered connection in the simulated network.

2 Accepting networks of evolutionary processors with filtered connections

We start by summarizing the notions used throughout this work. An *alphabet* is a finite and nonempty set of symbols. The cardinality of a finite set A is written $card(A)$. Any finite sequence of symbols from an alphabet V is called *string* over V. The set of all strings over V is denoted by V^* and the empty string is denoted by ε. The length of a string x is denoted by $|x|$ while $alph(x)$ denotes the minimal alphabet W such that $x \in W^*$.

We say that a rule $a \to b$, with $a, b \in V \cup \{\varepsilon\}$ and $ab \neq \varepsilon$ is a *substitution rule* if both a and b are not ε; it is a *deletion rule* if $a \neq \varepsilon$ and $b = \varepsilon$; it is an *insertion rule* if $a = \varepsilon$ and $b \neq \varepsilon$. The set of all substitution, deletion, and insertion rules over an alphabet V are denoted by Sub_V, Del_V, and Ins_V, respectively.
Given a rule σ as above and a string $w \in V^*$, we define the following *actions* of σ on w:

- If $\sigma \equiv a \to b \in Sub_V$, then $\sigma^*(w) = \begin{cases} \{ubv : \exists u, v \in V^* \ (w = uav)\}, \\ \{w\}, \text{ otherwise} \end{cases}$
- If $\sigma \equiv a \to \varepsilon \in Del_V$, then $\sigma^*(w) = \begin{cases} \{uv : \exists u, v \in V^* \ (w = uav)\}, \\ \{w\}, \text{ otherwise} \end{cases}$

$$\sigma^r(w) = \begin{cases} \{u : \ w = ua\}, \\ \{w\}, \text{ otherwise} \end{cases} \qquad \sigma^l(w) = \begin{cases} \{v : \ w = av\}, \\ \{w\}, \text{ otherwise} \end{cases}$$

- If $\sigma \equiv \varepsilon \to a \in Ins_V$, then

$$\sigma^*(w) = \{uav : \ \exists u, v \in V^* \ (w = uv)\}, \ \sigma^r(w) = \{wa\}, \ \sigma^l(w) = \{aw\}.$$

$\alpha \in \{*, l, r\}$ expresses the way of applying a deletion or insertion rule to a string, namely at any position ($\alpha = *$), in the left ($\alpha = l$), or in the right ($\alpha = r$) end of the string, respectively. For every rule σ, action $\alpha \in \{*, l, r\}$, and $L \subseteq V^*$, we define the *α-action of σ on L* by $\sigma^\alpha(L) = \bigcup_{w \in L} \sigma^\alpha(w)$. Given a finite set of rules M, we define the *α-action of M* on the string w and the language L by:

$$M^\alpha(w) = \bigcup_{\sigma \in M} \sigma^\alpha(w) \ \text{ and } \ M^\alpha(L) = \bigcup_{w \in L} M^\alpha(w),$$

respectively. In what follows, we shall refer to the rewriting operations defined above as *evolutionary operations* since they may be viewed as linguistic formulations of local DNA mutations.

For two disjoint subsets P and F of an alphabet V and a string w over V, we define the predicates:

$$\varphi^{(s)}(w; P, F) \equiv \ P \subseteq alph(w) \qquad \wedge \ F \cap alph(w) = \emptyset$$
$$\varphi^{(w)}(w; P, F) \equiv alph(w) \cap P \neq \emptyset \ \wedge \ F \cap alph(w) = \emptyset.$$

The construction of these predicates is based on *random-context conditions* defined by the two sets P (*permitting contexts/symbols*) and F (*forbidding contexts/symbols*). Informally, the first condition requires that all permitting symbols are present in w and no forbidding symbol is present in w, while the second

one is a weaker variant of the first, requiring that at least one permitting symbol appears in w and no forbidding symbol is present in w. For every language $L \subseteq V^*$ and $\beta \in \{(s),(w)\}$, we define:

$$\varphi^{\beta}(L,P,F) = \{w \in L \mid \varphi^{\beta}(w;P,F)\}.$$

An *evolutionary processor over* V is a tuple (M, PI, FI, PO, FO), where:
• M is a set of substitution, deletion or insertion rules over the alphabet V. Formally: $(M \subseteq Sub_V)$ or $(M \subseteq Del_V)$ or $(M \subseteq Ins_V)$. The set M represents the set of evolutionary rules of the processor. As one can see, a processor is "specialized" in one evolutionary operation, only.
• $PI, FI \subseteq V$ are the *input* permitting/forbidding contexts of the processor, while $PO, FO \subseteq V$ are the *output* permitting/forbidding contexts of the processor. Informally, the permitting input/output contexts are the set of symbols that should be present in a string, when it enters/leaves the processor, while the forbidding contexts are the set of symbols that should not be present in a string in order to enter/leave the processor.

We denote the set of evolutionary processors over V by EP_V.

An *accepting network of evolutionary processors with filtered connections* (ANEPFC for short) is a 8-tuple $\Gamma = (V, U, G, \mathcal{R}, \mathcal{N}, \alpha, \beta, x_I, x_O)$, where:

- $\diamond$ V and U are the input and network alphabet, respectively, $V \subseteq U$.
- $\diamond$ $G = (X_G, E_G)$ is an undirected graph without loops with the set of vertices X_G and the set of edges E_G. G is called the *underlying graph* of the network.
- $\diamond$ $\mathcal{R} : X_G \longrightarrow 2^{Sub_U} \cup 2^{Del_U} \cup 2^{Ins_U}$ is a mapping which associates with each node the set of evolutionary rules that can be applied in that node. Note that each node is associated only with one type of evolutionary rules, namely for every $x \in X_G$ either $\mathcal{R}(x) \subset Sub_U$ or $\mathcal{R}(x) \subset Del_U$ or $\mathcal{R}(x) \subset Ins_U$ holds.
- $\diamond$ $\mathcal{N} : E_G \longrightarrow 2^U \times 2^U$ is a mapping which associates with each edge $e \in E_G$ the disjoint sets $\mathcal{N}(e) = (P_e, F_e)$, $P_e, F_e \subset U$.
- $\diamond$ $\beta : E_G \longrightarrow \{s, w\}$ defines the *filter* type of an edge.

We say that $card(X_G)$ is the size of Γ.

A *configuration* of an ANEPFC Γ as above is a mapping $C : X_G \longrightarrow 2^{V^*}$ which associates a set of strings with every node of the graph. A configuration may be understood as the sets of strings which are present in any node at a given moment. A configuration can change either by an *evolutionary step* or by a *communication step*.

When changing by an evolutionary step each component $C(x)$ of the configuration C is changed in accordance with the set of evolutionary rules M_x associated with the node x and the way of applying these rules $\alpha(x)$. Formally, we say that the configuration C' is obtained in *one evolutionary step* from the configuration C, written as $C \Longrightarrow C'$, if and only if

$$C'(x) = M_x^{\alpha(x)}(C(x)) \text{ for all } x \in X_G.$$

When changing by a communication step, each node processor $x \in X_G$ of an ANEPFC sends one copy of each string it has to every node processor y connected to x, provided they can pass the filter of the edge between x and y. It keeps no copy of these strings but receives all the strings sent by any node

processor z connected with x providing that they can pass the filter of the edge between x and z.

Formally, we say that the configuration C' is obtained in *one communication step* from configuration C, written as $C \vdash C'$, iff

$$C'(x) = (C(x) \setminus (\bigcup_{\{x,y\} \in E_G} \varphi^{\beta(\{x,y\})}(C(x), \mathcal{N}(\{x,y\}))))$$
$$\cup (\bigcup_{\{x,y\} \in E_G} \varphi^{\beta(\{x,y\})}(C(y), \mathcal{N}(\{x,y\})))$$

for all $x \in X_G$. Note that a copy of a string remains in the sending node x only if it not able to pass the filter of any edge connected to x.

Let Γ be an ANEPFC, the computation of Γ on the input string $w \in V^*$ is a sequence of configurations $C_0^{(w)}, C_1^{(w)}, C_2^{(w)}, \ldots$, where $C_0^{(w)}$ is the initial configuration of Γ defined by $C_0^{(w)}(x_I) = \{w\}$ and $C_0^{(w)}(x) = \emptyset$ for all $x \in X_G$, $x \neq x_I$, $C_{2i}^{(w)} \Longrightarrow C_{2i+1}^{(w)}$ and $C_{2i+1}^{(w)} \vdash C_{2i+2}^{(w)}$, for all $i \geq 0$. By the previous definitions, each configuration $C_i^{(w)}$ is uniquely determined by the configuration $C_{i-1}^{(w)}$. A computation as above is said to be an *accepting computation* if there exists a configuration in which the set of strings existing in the output node x_O is non-empty. The *language accepted* by Γ is

$$L(\Gamma) = \{w \in V^* \mid \text{ the computation of } \Gamma \text{ on } w \text{ is an accepting one}\}.$$

We denote by $\mathcal{L}(ANEPFC)$ the class of languages accepted by ANEPFCs.

3 Accepting evolutionary P systems

We now informally describe the P system we are going to investigate. An *Accepting evolutionary P system* (AEPS for short) of degree m is a construct

$$\Pi = (V, U, \mu, (R_1, \rho_1), \cdots, (R_m, \rho_m)),$$

where:

- V is the input alphabet, $U \supseteq V$ is the working alphabet,
- μ is a membrane structure consisting of m membranes,
- R_i, $1 \leq i \leq m$ is a finite set of *evolutionary and/or dissolving rules* over U associated with the ith region and ρ_i is a partial order relation over R_i specifying the *priority* among the rules. An evolutionary rule is a 4-tuple (a, b, α, β) (or $a \rightarrow b_\alpha^\beta$) where $a, b \in U \cup \{\lambda\}$, $\alpha \in \{here, out, in\}$ and $\beta \in \{*, l, r\}$. A dissolving rule is 5-tuple $(a, b, \alpha, \beta, \delta)$ (or $a \rightarrow b_\alpha^\beta \delta$), where a, b, α, β have the same meaning as for evolutionary rules and $\delta \in U$ is the dissolving symbol.

The application of a rule $a \rightarrow b_\alpha^\beta$ in an arbitrary region of the system works as follows: if there exists a string w in that region, such that $w = u_1 a u_2$, then w is transformed into $u_1 b u_2$ (observe that β establishes the way of applying the evolutionary rule). Parameter α establishes where to send the new strings, namely they are sent to the outer region, to all immediate inner regions (a copy of each string is sent to all these regions), or remain in the same region, provided

that β is *out*, *in*, or *here*. If a string is to be sent to an inner region that does not exist, then it remains where it is.

If the rule is a dissolving one ($a \rightarrow b^{\beta}_{\alpha}\delta$, the membrane of the region is dissolved after the rule application, provided that the membrane is different from the skin one.

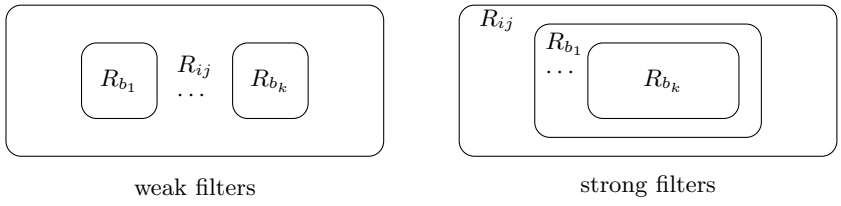
The input string is initially stored in the outmost region. Then, in a fully parallel manner all the rules are applied to the strings existing in every region according to their priorities. The system halts whenever: (1) No rule can be applied, or (2) The system is reduced to only one region, namely the outmost one.

The language accepted by Π is denoted by $L(\Pi)$. A string is in $L(\Pi)$ if and only if it being initially stored in the outmost region reduces the system to only one region.

4 A simulation of ANEPFCs by AEPSs

In this section we give just a very brief idea how an AEPS, say Π, can simulate an ANEPFC, say Γ. The membrane structure of Π consists of the skin region that includes and as many regions as connections between the processors of Γ. For every connection between processors i and j we have the regions R_{ij} and R_{ji} inside the skin membrane. Each region R_{ij} has different structure depending on the filter type. A symbol in the current string in Π indicates the fact that the corresponding string in Γ has just arrived in the node i. We discuss the simulation of one evolutionary step in the node i and the communication process of the words from the node i to the node j in Γ.

Let us suppose that the set of permitting symbols for the filter on the connection between processor i and j in Γ is defined by $P_{ij} = \{b_1, b_2, \cdots, b_k\}$. If this filter acts in the weak mode, then the membrane structure in the region R_{ij} is showed in the left part of the next figure, while if the filter acts in the strong mode, then the membrane structure is showed in the right part of the same figure.



weak filters strong filters

Fig. 1. Membrane structures for the filters

Inside the inner regions in illustrated membrane structures we apply the same evolutionary rules as those associated with the node i. In addition, the new strings are moved through the membrane structure in order to check the filter conditions. Thus evolutionary rules having the highest priority check the presence of forbidden symbols. If such a symbol is present, then the string remains blocked in an inner region. If these rules cannot be applied, then other evolutionary rules check the presence of permitting symbols. As soon as one permitting symbol is present, the string is sent to the outer region. Clearly, some special symbols are used in order to manage the string movements. It is worth noting the important role of the membrane structure in each region R_{ij} by an analogy with electronic circuits: the membranes checking in the weak mode the presence of a permitting symbol form a sort of parallel circuit, while the membranes checking in the strong mode the presence of a permitting symbol form a sort of serial circuit.

When a string is going to enter a region of the form R_{ij} where i is the output node of the ANEPFC, then a new symbol is inserted; this symbol will dissolve in turn all the membranes.

5 Conclusions and future work

In this paper we have proposed a string accepting P system based on a generalization of the evolutionary rules considered for ANEPFCs. We have intuitively described a construction of an AEPS able to naturally simulate an ANEPFC. We consider that the simulation process is natural as the membrane structure is of a great importance in the simulation of the filtering process in ANEPFC. It is worth mentioning that this is in contrast with very many constructions of P systems in which the membrane structure plays actually a very minor role, most of them being reduced to just one membrane.

A technical proof together with other computationally aspects of accepting evolutionary P systems are to be considered for an extension of this note.

References

1. 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)*, IEEE Computer Society 2000, 64–74.
2. C. Drăgoi, F. Manea, V. Mitrana, Accepting networks of evolutionary processors with filtered connections, *Journal of Universal Computer Science*, 13 pp 1598–1614 (2007).
3. Errico, L., Jesshope, C. (1994). Towards a new architecture for symbolic processing, In *Artificial Intelligence and Information-Control Systems of Robots '94*, 31–40.
4. Fahlman, S. E., Hinton, G.E., Seijnowski, T.J. (1983) Massively parallel architectures for AI: NETL, THISTLE and Boltzmann machines, In *Proc. of the National Conference on Artificial Intelligence*, 109–113.
5. Hillis, W.D. (1979). *The Connection Machine*. MIT Press, Cambridge.

6. S.R. Krishna, K. Lakshmanan, R. Rama. Hybrid P systems. *Romanian Journal of Information Science and Technology*, 4(1-2):111-123, 2001.
7. S.R. Krishna, R. Rama. Insertion-deletion P systems. In *Proc. of Workshop on DNA-Based Computers*, LNCS 2340: 360–370, 2002.
8. M. Madhu, K. Krithivasan. A note on hybrid P systems. *Grammars*, 5(3):239-244, December 2002.
9. F. Manea, C. Martín-Vide, V. Mitrana. Accepting networks of evolutionary word and picture processors: A survey. In *Scientific Applications of Language Methods*, Mathematics, Computing, Language, and Life: Frontiers in Mathematical Linguistics and Language Theory - Vol. 2, World Scientific 523–560, 2010.
10. C. Martín-Vide, Gh. Păun. String-objects in P systems. In *Proc. of Algebraic Systems, Formal Languages and Computations Workshop*, RIMS Kokyuroku, Kyoto Univ., 161–169, 2000.
11. C. Martín-Vide, Gh. Păun. Language generating by means of membrane systems. *Bulletin of the EATCS*, (75):199-218, October 2001.
12. A. Păun, M. Păun. On membrane computing based on splicing. *Words, Sequences, Languages*, Kluwer Academic Publishers, 409–422, 2000.
13. Gh. Păun. *Membrane Computing. An Introduction.* Springer. 2002.