P Systems for Social Networks

Erzsébet Csuhaj-Varjú[1], Marian Gheorghe[2], György Vaszil[1], Marion Oswald[3]

[1] Computer and Automation Research Institute
Hungarian Academy of Sciences
H-1111, Budapest, Kende u. 13-17, Hungary
{csuhaj,marion,vaszil}@sztaki.hu
[2] Department of Computer Science
University of Sheffield
Regent Court, Portobello Street, Sheffield S1 4DP, United Kingdom
m.gheorghe@dcs.shef.ac.uk
[3] Institute for Computer Languages
Vienna University of Technology
Favoritenstr. 9-11, 1040 Vienna, Austria
marion@logic.at

Summary. We introduce some variants of P systems that mimic the behaviour of social networks and illustrate some of the characteristics of them. Other concepts related to social networks are discussed and suitable classes of P systems are suggested. A simple example shows the capabilities of such a P system where the intensity of the communication is modelled with complementary alphabets.

1 Introduction

Membrane computing (also called P systems theory) is a new computing paradigm, which in its initial variants was inspired by the structure and functionality of the living cell [19]. Later on some other biological entities have been considered in order to extend the capabilities of this computational model - tissues or special types of cells, like neurons, or colonies of cells, like bacteria. So far, concepts and methods of P systems theory have been successfully employed in solving important problems of computer science and describing various biological phenomena, but, except promising applications in linguistics and natural language processing, only a limited amount of attention has been paid to the suitability of membrane systems in modelling social phenomena.

In this paper, we attempt to build a bridge between membrane computing and the theory of *social networks*, an area of great interest in contemporary computer science and practice. For this reason, we define certain classes of P systems which are suitable for modelling features of social networks and which can be derived from problems in this field, and we formulate various research topics related to

connections between P systems theory and the theory of social interactions and networks. The underlying mathematical tool set is the theory of formal languages, i.e., our approach to social networks is a purely syntactic one. We note that social phenomena have already been described by different frameworks in formal language theory, our recent aim is to formulate models which combine tools of both membrane computing and formal language theory. In terms of formal grammars, communities of agents interacting with each other and with their dynamically changing environment were modelled by eco-grammar systems, a research field launched in [9]. Population of agents, called networks of (parallel) language processors, with biological and social background were described by rewriting systems in [12, 7]. Another formal language-theoretic model for communities of evolving agents, called evolutionary systems, was introduced in [10]. Multi-agent systems in terms of formal language theory and membrane computing were discussed in [4], [5], [3], and [15].

2 Social Networks

In various formalisms related to the study of social phenomena, interpersonal relationships between individuals are defined as information-carrying connections [14]. These relationships come in various forms, two of them are *strong* and *weak ties*. Weak ties seem to be responsible for the embeddedness and structure of social networks and for the communication within these systems [14]. There are other measures that characterise connections between nodes (individuals). *Centrality* gives an indication of the social power of a node and the strength of its connections. It relies on other measures, *betweenness, closeness, degree*. Betweenness measures to what extent a node is connected to nodes that have a significant number of neighbours (direct connections). Closeness is the degree describing that a node is close to all other nodes in the network: it counts the number of connections. For the above concepts as well as for other measures of the connections existing in social networks, we refer to [23].

3 P Systems Capturing Communication Aspects

We are focusing now on identifying some classes of P systems that capture communication aspects in social networks. We can consider various types of nodes: *ordinary* or *popular* nodes - those that host individuals and allow communication between them; *new-born* nodes - those that are dynamically created and linked to the existing network; *non-visible* or *extinct* nodes - the nodes that are no longer connected to the network or have disappeared; nodes with one way communication, only allowing information to go into, *blackholes* or allowing only to exit from, *whiteholes*. Some of these aspects have been already considered in the current research framework of membrane computing; for instance population P systems allow nodes

to be dynamically connected and disconnected [6]; the one-way communication for communicating accepting P systems has been considered in [13]. We can also take into account connections between nodes and look at the *volume of communication* - the amount of (new) information generated or sent-received by various nodes or groups of nodes; *frequency of communicated messages* - the number of communication steps related to the evolution (computation) steps; *communication motifs* - patterns of communication identified throughout the network evolution. In order to capture these phenomena we aim to formally define a generic and flexible framework whereby these concepts can be appropriately accommodated. In this respect we provide the following general definition of a *population P system governed by communication*, a *pgcP system*, for short.

4 Preliminaries and Definitions

Throughout the paper we assume that the reader is familiar with the basics of membrane computing and formal language theory; for details we refer to [17, 21] and [22]. For an alphabet V, we denote by $|V|$ the cardinality of V, and by V^* the set of all finite words over V. If λ, the empty word is excluded, then we use the notation V^+. As usual in membrane computing, we represent the finite multisets over V by strings over V as well, that is, a string and all its permutations correspond to the same multiset. We denote by $|w|_a$ the number of occurrences of a symbol $a \in V$ in a string $w \in V^*$ which is equal to the multiplicity of that object in the represented multiset. We use $\emptyset$ to denote the empty multiset, and also use V^* to denote the set of finite multisets over an alphabet V.

In the following we consider P systems with static underlying graph structure; the notion can easily be extended to a construct capturing dynamically changing underlying graph architecture as well.

Definition 1. *A population P system governed by communication (a pgcP system, for short), is a construct*

$$(\Sigma, E, \omega_1, \ldots, \omega_n, (\rho_1, R_1), \ldots, (\rho_k, R_k)),\ n, k \geq 1,$$

where

- $\Sigma = \Sigma_1 \cup \Sigma_2$ *is a finite alphabet of objects,* Σ_1 *is the alphabet of cellular objects, i.e., the objects in the nodes, and* Σ_2 *is the set of communication symbols;*
- $E \subseteq \{1, \ldots, n\} \times \{1, \ldots, n\}$ *is the set of (directed) links between the nodes;*
- $\omega_i \in \Sigma_1^*,\ 1 \leq i \leq n$, *is a multiset of cellular objects, the initial content of the node* i *of the system; and*
- $(\rho_i, R_i),\ 1 \leq i \leq k$, *are predicate based rule-sets governing the transitions of the system, with*
  - $\rho_i : \Sigma_2^* \rightarrow \{true, false\}$, *a predicate over the multisets of communication symbols, and*

– $R_i = (R_{i,1}, \ldots, R_{i,n})$, *an* n*-tuple of sets of rewriting rules, where* $R_{i,j}$, $1 \leq j \leq n$, *is the set of rules that are allowed to be applied at node* j, *and any rule is of the form* $u \to v$ *for* $u \in \Sigma_1^*$, $v \in (\Sigma_1 \cup (\Sigma_1 \times \Sigma_2 \times Tar))^*$ *where* $Tar = \{1, \ldots n\}$ *is a set of target indicators.*

Definition 2. *A configuration of a pgcP system*

$$\Pi = (\Sigma, E, \omega_1, \ldots, \omega_n, (\rho_1, R_1), \ldots, (\rho_k, R_k)),\ n, k \geq 1,$$

is an $n + s$*-tuple for* $s = |E|$,

$$(w_1, ..., w_n; u_1, ..., u_s),\ w_i \in \Sigma_1^*,\ u_j \in \Sigma_2^*,\ 1 \leq i \leq n,\ 1 \leq j \leq s,$$

where multiset w_i *is the multiset of cellular objects at the* i*-th node, i.e., the current content of node* i, $1 \leq i \leq n$, *and* u_j *is the multiset of communication symbols associated to the communication link* $e_j \in E$, $1 \leq j \leq s$.

The initial configuration of Π *is* $(\omega_1, \ldots, \omega_n; \emptyset, \ldots, \emptyset)$.

The pgcP system works by changing its configurations. In the following we describe the transition or configuration change: it takes place by rewriting and communication of the cellular objects and recording the performed communication. The rewriting rules are applied to the cellular objects in the maximally parallel manner, i.e., any object can be involved in at most one rule application, and as many rules are applied simultaneously to the cellular objects at the nodes as possible.

Definition 3. *Let* $\Pi = (\Sigma, E, \omega_1, \ldots, \omega_n, (\rho_1, R_1), \ldots, (\rho_k, R_k))$, $n, k \geq 1$, *be a pgcP system and let* $c_1 = (w_1, ..., w_n; u_1, ..., u_s)$ *and* $c_2 = (w'_1, ..., w'_n; u'_1, ..., u'_s)$ *be two configurations of* Π.

We say that c_1 *changes directly to* c_2 *(or* c_2 *is obtained from* c_1 *with a transition), denoted by*

$$c_1 = (w_1, ..., w_n; u_1, ..., u_s) \Rightarrow^{(\rho_i, R_i)} c_2 = (w'_1, ..., w'_n; u'_1, ..., u'_s)$$

for some $i \in \{1, \ldots, k\}$, *if the following hold:*

- $\rho_i(u_1, \ldots, u_s) = true$,
- c_1 *is changed to* c_2 *by using the rules of* $R_i = (R_{i,1}, \ldots, R_{i,n})$ *as follows:*
  - *the rules of* $R_{i,j}$ *are applied in the maximal parallel manner in the node* j *to the multiset* w_j, $1 \leq j \leq n$; *and*
  - *if a rule of the form* $u \to v$ *where* $v = v_1 v_2$ *for* $v_1 \in \Sigma_1^*$ *and* $v_2 \in (\Sigma_1 \times \Sigma_2 \times Tar)^*$ *is applied in a node* j, *then the following holds:*
    - u *is changed to* $v_1 v_2$ *and all objects in* v_1 *remain in node* j;
    - *all symbols of* $(a, a', l) \in v_2$ *are processed by sending the cellular object* $a \in \Sigma_1$ *to node* l *and adding the communication object* $a' \in \Sigma_2$ *to the multiset associated to the link* $(j, l) \in E$, $1 \leq j \neq l \leq n$.

The pgcP system may record information on the communication performed during the whole computation or only on communication during the last configuration change. This is captured in the following definition.

Definition 4. *Let $\Pi = (\Sigma, E, \omega_1, \ldots, \omega_n, (\rho_1, R_1), \ldots, (\rho_k, R_k))$, $n, k \geq 1$, be a pgcP system. We say that Π works in the*

- *history preserving mode, if for any transition*

$$(w_1, ..., w_n; u_1, ..., u_s) \Rightarrow^{(\rho_i, R_i)} (w'_1, ..., w'_n; u'_1, ..., u'_s)$$

  it holds that $u'_j = u_j u''_j$, for $1 \leq j \leq s$, where u''_j is the multiset of communication symbols sent to link j during the transition,
- *non-history preserving mode, if u'_j consists of the communication symbols sent to link j during the transition (the communication symbols in u_j are forgotten).*

Definition 5. *Let $\Pi = (\Sigma, E, \omega_1, \ldots, \omega_n, (\rho_1, R_1), \ldots, (\rho_k, R_k))$, $n, k \geq 1$, be a pgcP system. A derivation (or computation) in Π is a sequence of transitions starting in the initial configuration and ending in some final (possibly halting) configuration.*

The *result* of a computation in a pgcP system Π can be defined in various manners. We may consider the number (vector) of (certain) *communication objects* going through (certain) communication links or the number (vector) of (certain) *cellular objects* in (certain) nodes. If we assume distinguished, so called output link(s) or node(s), then we indicate this fact in the notation for the accepted languages of the pgcP system. As usual in membrane computing we consider only halting computations.

Definition 6. *Let $\Pi = (\Sigma, E, \omega_1, \ldots, \omega_n, (\rho_1, R_1), \ldots, (\rho_k, R_k))$, where $n, k \geq 1$, and let $T_i \subseteq \Sigma_i$, $1 \leq i \leq 2$, be the sets of terminal cellular and communication objects; $Out_1 \subseteq \{1, \ldots, n\}$ and $Out_2 \subseteq E$ be the sets of output nodes and output links, respectively.*

- *We define $N_{cell}(\Pi, T_1, Out_1)$ ($Ps_{cell}(\Pi, T_1, Out_1)$) as the number (vector) of terminal cellular objects in the output nodes in a final configuration.*
- *We define $N_{com}(\Pi, T_2, Out_2)$ ($Ps_{com}(\Pi, T_2, Out_2)$) as the number (vector) of terminal communication objects associated to the output links in a final configuration.*

5 Complementary Alphabets in pgcP Systems

We are focusing now on communication in these networks. In order to characterize its intensity and the fact that the importance of a connection might evolve in time by either increasing or decreasing its value, we would need some sort of symbols that act in this respect. One way to model this is by considering complementary

communication symbols, whereby the customary (or positive) symbols strengthen a connection, whereas the complementary (negative) ones weaken it. Formally this is achieved by splitting the alphabets Σ_1 and Σ_2 as follows:

$$\Sigma_i = \Sigma_i' \cup \bar{\Sigma}_i' \cup \Sigma_i''$$

where Σ_i' and $\bar{\Sigma}_i'$, $i = 1, 2$ are *dual alphabets*. Σ_i' consists of normal (positive) elements and $\bar{\Sigma}_i'$ contains complementary symbols.

The idea of complementary alphabets is not new in the field of natural computing. *DNA computing* has as a core data structure a double-strand structure consisting of dual elements, the DNA nucleotides, adenine, thymine, cytosine and guanine represented by the four letter alphabet, $\{A, T, C, G\}$, respectively; the pairs (A, T) and (C, G) are known as *complementary base pairs* [20]. In the context of *networks of Watson-Crick D0L systems*, networks of such systems over DNA-like alphabets are introduced and operations relying on complementarity properties are utilised [8, 11].

Active and passive objects have been considered in a similar way with complementary elements. Two types of such objects have been introduced and studied: within components [1] and on membranes [2]. Other membrane systems using complementary features are *spiking neural P systems with anti-spikes* where a neuron receiving s spikes and t anti-spikes is left with $s - t$ if $s \geq t$ or $t - s$ when $t \geq s$ objects [18], and membrane systems with bi-stable catalysts, where the system may switch between two states [17].

To demonstrate the above ideas, we present a simulation of an n-register machine M where the communication is controlled by "Watson-Crick-like" predicates.

An n-register machine $M = (Q, R, q_0, q_f, P)$, $n \geq 1$, is defined as usual, that is, with internal state set Q, registers $R = (A_1, \ldots, A_n)$, initial and final states $q_0, q_f \in Q$, respectively, and a set of instructions P of the form (q, A_i+, r, s) or (q, A_i-, r, s), $q, r, s \in Q$, $q \neq q_f$, $A_i \in R$. When an instruction of the first type is performed, then M is in state q, increases the value of register A_i by one, and enters a state r or s, chosen nondeterministically. When an instruction of the second type is performed, then M is in state q and it subtracts one from the value of register A_i if it stores a positive number and then enters state r, or it leaves the value of A_i unchanged if it stores zero and then enters state s. There are no instructions for the final state, therefore the machine halts after entering state q_f. M starts its work in the initial state, q_0, with empty registers. Then it performs a sequence of instructions; if the sequence is finite (it ends with halting), then we speak of a computation by M. The result of the computation is the number stored in the output register A_1 after halting.

It is known that 2-register machines are able to compute any recursively enumerable set of numbers [16].

Example 1. Let $M = (Q, R, q_0, q_f, P)$, $n \geq 1$, be an n-register machine. We construct the pgcP system Π with $n + 1$ nodes simulating M as follows. Let

$$\Pi = (\Sigma, T_1, E, w_0, w_1, \ldots, w_n, (\rho_1, R_1), (\rho_2, R_2), Out)$$

where $\Sigma = \Sigma_1 \cup \Sigma_2$ with $\Sigma_1 = \{a, \bar{a}\} \cup \{q, [q, r, s] \mid q, r, s \in Q\}$, $\Sigma_2 = \{a, \bar{a}\}$, and $E = \{(0, i) \mid 1 \leq i \leq n\}$, $T_1 = \{a\}$, and $Out = 1$. The initial configuration and the rule sets are defined as follows. Let

$$w_0 = q_0, \text{ and } w_i = \emptyset, \ 1 \leq i \leq n.$$

Moreover, let $R_1 = (R_{1,0}, \ldots, R_{1,n})$, $R_2 = (R_{2,0}, \ldots, R_{2,n})$, and let

$$\begin{aligned} \rho_1(u_1, \ldots, u_n) \ : \ & |u_i|_a \geq |u_i|_{\bar{a}} \text{ for all } 1 \leq i \leq n, \\ R_{1,0} = \ & \{q \to [q, r, s](a, a, j) \mid (q, A_j+, r, s) \in P\} \cup \\ & \{q \to [q, r, s](\bar{a}, \bar{a}, j) \mid (q, A_j-, r, s) \in P\} \cup \\ & \{[q, r, s] \to r, [q, r, s] \to s \mid q, r, s \in Q\}, \end{aligned}$$

$$\begin{aligned} \rho_2(u_1, \ldots, u_n) \ : \ & |u_i|_a < |u_i|_{\bar{a}} \text{ for some } i, \ 1 \leq i \leq n, \\ R_{2,0} = \ & \{[q, r, s] \to s(a, a, j) \mid (q, A_j-, r, s) \in P\}, \text{ and finally} \end{aligned}$$

$$R_{j,k} = \{a\bar{a} \to \varepsilon\} \text{ for all } 1 \leq j \leq 2, \ 1 \leq k \leq n.$$

If Π works in the history preserving mode, then the difference between the number of symbols a and $\bar{a}$ on a link $(0, j)$ for some $1 \leq j \leq n$ corresponds to the value of register j.

The result of the computation can be found in node 1 corresponding to the output register A_1 of M if the system introduces the symbol q_f in node 0 and halts (because there is no rule for the final state in M). Thus, these variants of pgcP systems are computationally complete.

The reader may observe that in the above example we have not only complementary alphabets, but through various global predicates, the sets of rules are split into "normal" rules in R_1 and "recovery" rules in R_2.

We note that it is possible to simulate a register machine with one active component and n others receiving values a or $\bar{a}$. In a very similar way, we can consider a distributed model where the n components corresponding to registers are used in such a way that each one has its own addition and subtraction rules, similar to $R_{1,0}, R_{2,0}$. In this case we have to communicate not only a or $\bar{a}$ but also the label of the next register to the component simulating this register; some other variants of rules can be considered. Regarding these two models, one immediate question, perhaps not difficult to be addressed, is which one is simpler, more efficient - with respect to the number of rules, symbols etc; or are they just the same? What about simulating one with the other one?

Furthermore, the example of simulating a register machine suggests the use of *dual sets of rules*, triggered by predicates, i.e., to have for each rule, r, its dual rule, $\bar{r}$, defined in such a way that the complementary rules introduce complementary symbols (only).

6 Further Research Topics

In the previous sections we introduced the concept of a pgcP system inspired by social networks and made some steps towards identifying the necessary abstract elements related to measuring the intensity of the communication in these systems. In this respect, complementary alphabets and particular variants of pgcP systems have been considered. In the following, we define some further concepts regarding some specific types of pgcP systems, and list some preliminary results for these so-called *deterministic* and *non-cooperative* pgcP systems.

Definition 7. *Let* $\Pi = (\Sigma, E, \omega_1, \ldots, \omega_n, (\rho_1, R_1), \ldots, (\rho_s, R_s))$, $n, s \geq 1$, *be a non-cooperative deterministic pgcP system.*

Let $c(t) = (w_1(t), ..., w_n(t); u_1(t), ..., u_s(t))$, $t \geq 0$, *be a computation in* Π *in the history preserving mode.*

We define the

1. *growth of communication volume on link* i *at derivation step* t, $t \geq 1$, *by* $f_i : \mathbb{N} \to \mathbb{N}$ *where* $f_i(t) = |u_i(t)| - |u_i(t-1)|$.
2. *frequency of communication on link* i*:* $h_i : \mathbb{N} \to \{0, 1\}$ *where* $h_i(t) = 0$ *if* $|u_i(t)| - |u_i(t-1)| = 0$ *and* $h_i(t) = 1$ *if* $|u_i(t)| - |u_i(t-1)| \geq 1$*;*
3. *intensity of communication on link* i*:* $g_i : \mathbb{N} \to \mathbb{R}$, *where* $g_i(t) = \frac{f_i(t)}{t}$.

If $\Sigma_2 = \Sigma_2' \cup \bar{\Sigma}_2' \cup \Sigma_2''$, i.e., complementary symbols are considered, then $\bar{f}_i(t)$ defined over $\bar{\Sigma}_2'$, and difference functions as $f_i(t) - \bar{f}_i(t)$ can also be defined and examined.

We note that the above concepts can be extended with suitable modifications to gcpP systems in the general sense; obviously, in this case we speak of relations, instead of functions.

These notions have their roots in concepts related to networks of parallel language processors [12] and evolutionary systems [10]. A network of parallel language processors with D0L systems as components (an NLP-D0L system) consists of D0L systems located in nodes of a finite virtual graph (each node has at most one D0L system) which rewrite and communicate multisets of strings present in the nodes according to their own rule sets. A D0L system $G = (V, P, \omega)$ is a triplet, where V is an alphabet, P is a finite set of rules of the form $a \to \alpha$ with $a \in V$, $\alpha \in V^*$ and for each $a \in V$ there exists exactly one rule in P, and, finally, $\omega \in V^+$. For any string $x = x_1 \ldots x_n$, $x_i \in V$, $1 \leq i \leq n$, we say that x directly derives $y = y_1 \ldots y_n$, if $x_i \to y_i \in P$ holds for $1 \leq i \leq n$. (For more details on D0L systems, we refer to [21]). The NLP-D0L system functions with alternating rewriting and communication steps. By rewriting, each string at every node is rewritten in parallel; the D0L systems work in a synchronized manner. By communication, a copy of each string at a node is sent to each other node, given that the string satisfies the sender node's output context condition (predicate) and the receiver node's input context condition (predicate). Communication is performed in a parallel and synchronized

manner as well. In [12] it was shown that if the conditions for communication are random context conditions, i.e., they check the presence and /or the absence of certain symbols in the strings to be communicated, then the growth of the number of the strings in the network can be described by a growth function of a D0L system. The growth function of a D0L system orders to the number of derivation steps the length of the string obtained at that step. It was also shown, that the number of strings at specific nodes and the number of communicated strings between nodes can also be obtained from $D0L$ growth functions with suitable homomorphisms. The idea of the proofs comes from the property that in the case of $D0L$ systems any string generates only one string and the alphabet of the successor string can be calculated from the alphabet of the predecessor string.

The reader may easily notice the close relation between NLP-D0L systems and non-cooperative deterministic pgcP systems: The multisets of different symbols communicated from a node to some other one by an NLP-D0L system at any computation step corresponds to multisets of communication symbols added to the links of an appropriate pgcP system (where the predicates checks the presence/absence of types of objects in the multiset). Therefore, we may describe the growth of the communication volume, the frequency, and the intensity of communication on the links by tools of Lindenmayer systems, in particular the theory of D0L systems. Since D0L systems demonstrate several nice decidability properties, the theory provides efficient tools for characterizing the behaviour of particular types of pgcP systems. The detailed comparison is a topic for future research.

In context of social networks and pgcP systems, a number of other general problems can also be formulated. For example, how to describe and characterize other *concepts and measures* from social networks and how to define and model problems like *leaders and clusters emergence*. Or, how to dynamically restructure the links and distinguish between *good and bad or strong and weak links*; what about *breaking the links*. Finally, how to *solve various problems or compute functions* with such systems. These and similar questions form the basis of challenging future research.

7 Acknowledgement

The work of Erzsébet Csuhaj-Varjú and György Vaszil was supported in part by the Hungarian Scientific Research Fund, OTKA, Grant no. K75952. The work of Marian Gheorghe was partially done during his visit to the Computer and Automation Research Institute, Hungarian Academy of Sciences, in June 2010, and partially was supported by the grant K75952, Hungarian Scientific Research Fund, OTKA.

References

1. B. Aman, G. Ciobanu. Turing completeness using three mobile membranes. In *Unconventional Computing 2009*, LNCS, 5715, 42–55, 2009.
2. B. Aman, G. Ciobanu. Mutual mobile membranes systems with surface objects. In *7-th Brainstorming Week of Membrane Computing*, 29–39, 2009.
3. G. Bel-Enguix. A Multi-agent Model for Simulating the Impact of Social Structure in Linguistic Convergence. In*ICAART(2)* (J. Filipe et. al, Eds.), INSTICC Press, 367–372, 2009.
4. G. Bel-Enguix, M. A. Grando, M. D. Jiménez López. A Grammatical Framework for Modelling Multi-Agent Dialogues. In *PRIMA 2006* (Z.-Z. Shi, R. Sadananda, Eds.), LNAI 4088, Springer Verlag, Berlin Heidelberg, 10–21, 2009.
5. G. Bel-Enguix, M. D. Jiménez López. Membranes as Multi-agent Systems: an Application for Dialogue Modelling. In *IFIP PPAI 2006* (J.K. Debenham, Ed.), Springer, 31–40, 2006.
6. F. Bernardini, M. Gheorghe. Population P systems. *Intern J of Universal Comp Sci*, 10, 509–539, 2004.
7. E. Csuhaj-Varjú. Networks of Language Processors. *EATCS Bulletin* 63, 120–134, 1997.
8. E. Csuhaj-Varjú. Computing by networks of Watson-Crick *D0L* systems. In *Proc. Algebraic Systems, Formal Languages and Computation* (M. Ito, Ed.) RIMS Kokyuroku 1166, August 2000, Research Institute for Mathematical Sciences, Kyoto University, Kyoto, 43–51, 2000.
9. E. Csuhaj-Varjú, J. Kelemen, A. Kelemenová, Gh. Păun. Eco-Grammar Systems: A Grammatical Framework for Studying Lifelike Interactions. *Artificial Life*, 3(3), 1–28, 1997.
10. E. Csuhaj-Varjú, V. Mitrana. Evolutionary systems: a language generating device inspired by evolving communities of cells. *Acta Informatica*, 36(11), 913–926, 2000.
11. E. Csuhaj-Varjú, A. Salomaa. Networks of Watson-Crick *D0L* systems. In *Words, Languages & Combinatorics III. Proceedings of the International Colloquium, Kyoto, Japan, March 14-21, 2000.* (M. Ito, T. Imaoka, Eds.), World Scientific Publishing Co., Singapore, 134–149, 2003.
12. E. Csuhaj-Varjú, A. Salomaa. Networks of Parallel Language Processors. In *New Trends in Formal Languages. Control, Cooperation, and Combinatorics* (Gh. Păun, A. Salomaa, Eds.), LNCS 1218, Springer Verlag, Berlin Heidelberg, 299-318, 1997.
13. E. Csuhaj-Varjú, G. Vaszil. P automata or purely communicating accepting P systems. In *Membrane Computing*(Gh. Păun, G. Rozenberg, A. Salomaa, C. Zandron, Eds.), LNCS 2597, Springer Verlag, Berlin Heidelberg, 219–233, 2003.
14. M.D. Granovetter. The Impact of Social Structures on Economic Development. *Journal of Economic Perspectives*, 19, 33–50, 2004.
15. M. D. Jiménez López. Agents in Formal Language Theory: An Overview. In *Highlights in Practical Applications of Agents and Multiagent Systems. 9th International Conference on Practical Applications of Agents and Multiagent Systems* (J. Bajo Pérez et. al, Eds.) Advances in Intelligent and Soft Computing 89, Springer, 283–290, 2011.
16. M. Minsky. *Finite and Infinite Machines.* Prentice Hall, Englewood Cliffs, New Jersey, 1967.
17. Gh. Păun. *Membrane Computing. An Introduction.* Springer, 2002.

18. L. Pan, Gh. Păun. Spiking neural P systems with anti-spikes, *Int J Computers Comms Control*, 4, 273–282, 2009.
19. Gh. Păun. Computing with Membranes. *J. of Comput. Syst. Sci.*, 61, 108–143, 2000.
20. Gh. Păun, G. Rozenberg, A. Salomaa. *DNA Computing - New Computing Paradigms*. Springer Verlag, 1998.
21. G. Rozenberg, A. Salomaa. (Eds). *Handbook of Formal Languages I-III*. Springer, 1997.
22. Gh. Păun, G. Rozenberg, A. Salomaa. (Eds). *The Handbook of Membrane Computing*. Oxford University Press, 2009.
23. S. Wasserman, K. Faust. *Social Networks Analysis: Methods and Applications*. Cambridge University Press, 1994.