Research Topics in Membrane Computing: After CMC 12, Before BWMC 10

Marian Gheorghe[1], Gheorghe Păun[2,3] – Editors

[1] Department of Computer Science
University of Sheffield
Regent Court, Portobello Street, Sheffield S1 4DP, UK
m.gheorghe@dcs.shef.ac.uk

[2] Institute of Mathematics of the Romanian Academy
PO Box 1-764, 014700 Bucureşti, Romania

[3] Department of Computer Science and Artificial Intelligence
University of Sevilla
Avda. Reina Mercedes s/n, 41012 Sevilla, Spain
gpaun@us.es

Summary. What follows is a list of open problems and research topics compiled by the two editors of this "mega-paper", with the authors who contributed to this list mentioned together with their problems and research topics. The idea of such a collection occurred during CMC 2011, held in Fontainebleau, Paris, France (23 - 26 August 2011), and the result is meant to be a working material for the Tenth Brainstorming Week on Membrane Computing, Sevilla, Spain (January 30 - February 3, 2012).

Introduction

The idea of compiling a collection of open problems and research topics occurred during the Twelfth International Conference on Membrane Computing, CMC 2011, held in Fontainebleau, Paris, France, from 23 to 26 of August, 2011 (see http://cmc12.lacl.fr/). The invitation to contribute to such a collection was formulated during CMC 2011 (and after that reinforced by email) and several researchers answered this call. The result is the present "mega-paper" (*mega* because it has much more co-authors than any other paper in membrane computing...), meant to be a working material for the Tenth Brainstorming Week on Membrane Computing, Sevilla, Spain, January 30 - February 3, 2012.

The texts received from the contributors appear below as they have been submitted, with minimal editorial changes, sometimes with a short comment in the beginning. In most cases, one gives the necessary (minimal) definitions, as well as the relevant bibliography. Of course, the reader is supposed to be familiar with

basic elements of membrane computing (MC from now on) – for instance, from the sources mentioned in the end of this introduction. The authors of each "section" are mentioned, with affiliations and email addresses, so that the interested reader can contact them for further details, clarifications, cooperation in solving the problems.

The order in which the problems are given below goes, approximately, from general to theory and then to applications. The nature of questions range from local/technical open problems, to "strategic" issues, for instance, relating MC with other research areas, such as computer science, biology, ecology, economics and so on. Of course, many other precise problems or research ideas circulate in the MC community (or can be found in recent papers; see also the previous brainstorming volumes, where many problems are formulated, sometimes given in explicit lists; the "fate" of some of these open problems is recalled in the paper Gh. Păun, "Tracing Some Open Problems in Membrane Computing", *Romanian J. of Information Science and Technology*, 10, 4 (2007), 303–314). Similarly, some of the problems presented here or variants of them were already circulated in the MC community, which should raise the interest from them (as an indication of both interest and difficulty). We are aware, on the one hand, that many other authors, who have not answered our request (in time), would have other problems to propose, and, on the other hand, that many people keep for them, for their immediate research, the "juicy" topics... Anyway, we hope that this collection will both raise the interest in participating (actively) in 10th BWMC, and, perhaps, in producing a new list of open problems and research topics, either enlarging the present one or a different one.

General MC References

1. G. Ciobanu: *Membrane Computing. Biologically Inspired Process Calculi*. The Publishing House of the "Al.I. Cuza" University, Iaşi, 2010.
2. G. Ciobanu, Gh. Păun, M.J. Pérez-Jiménez, eds.: *Applications of Membrane Computing*. Springer-Verlag, Berlin, 2006.
3. P. Frisco: *Computing with Cells. Advances in Membrane Computing*. Oxford Univ. Press, 2009.
4. A. Păun: *Computability of the DNA and Cells. Splicing and Membrane Computing*. SBEB Publishing, Choudrant, Louisiana, USA, 2008.
5. Gh. Păun: *Membrane Computing. An Introduction*. Springer, Berlin, 2002 (Chinese translation in 2012).
6. Gh. Păun, G. Rozenberg, A. Salomaa, eds.: *The Oxford Handbook of Membrane Computing*. Oxford University Press, 2010.
7. The P Systems Website: `www.ppage.psystems.eu`.

Contents

1 Some General Issues

Jacob Beal

BBN Technologies, Canbridge, MA, USA
jakebeal@bbn.com

Comment. Jacob Beal was one of the invited speakers in CMC 2011 (title of talk: "Bringing Biology and Engineering Together with Spatial Computing"). After the meetings, he was asked to express his thoughts about MC, taking into account that he comes from outside the MC community, more importantly, from applied computer science. What follows is part of an e-mail message he has sent to M.Gh. in the end of August 2011.

With regards to my thoughts on directions for the membrane computing community, I think there is something very interesting and unique about the combination of chemical, compartmentalized, and tree-structured computation that P

systems gives access to. But I think that it is important to try to articulate what that is and why it is important.

In particular, the questions that I might pose would be:

- What are the most important research questions for membrane computing?
- What does membrane computing have to offer researchers who are not in the field of membrane computing?

More specifically:

- What should other computational theorists learn from the family of P systems computational models?
- What is the practical advantage of P systems models over their competitors in biological modeling or other fields?
- How might P systems models be applied to improve representations or architectures for parallel computing?
- What is quantitatively advantageous about SN P systems over other spiking models?
- How can P systems inform the theory or design of distributed algorithms?

I do not expect that any of these questions will have any one answer – in fact, I am sure that many researchers in the field will have wildly different answers. But every researcher should have clear and concise answers that they can make a good case for.

For my own part, I think that the most important research questions are:

1. How can distributed systems notions like self-stabilization be applied to P systems?
2. What consequences does the P systems model have for conventional computing?
3. What sort of complex P systems computations can be generated from high level programming languages, and what sort of languages fit best with P systems for various purposes (e.g. biological modeling, networking)?

Those priorities, however, are of course a consequence of my own research interests and biases, and I expect that others would have different answers: the important thing is the discussion of reasons.

2 The Power of Small Numbers

Artiom Alhazov

Institute of Mathematics and Computer Science
Academy of Sciences of Moldova
Chişinău, Republic of Moldova, and

Università degli Studi di Milano-Bicocca
Dipartimento di Informatica, Sistemistica e Comunicazione
Milano, Italy
artiom@math.md, aartiom@yahoo.com

Comment. Artiom Alhazov was also an invited speaker in CMC 2011; his talk, "Properties of Membrane Systems", is cited in the references below and can be helpful in clarifying some of the notions mentioned in the following problems.

Note: in case the underlying definitions are not clear, all bibliography items include URLs of the associated publications (freely accessible .PDF files or springerlink references). This made it possible to formulate the problems more concisely.

2.1 Minimal Parallelism and Number of Membrane Polarizations (2006)

It is known, [1, 2] that under minimal parallelism, P systems with active membranes can solve intractable problems in a polynomial number of steps, even without non-elementary membrane division and without membrane creation. However, the best known results are 6 (six!) polarizations, or 4 polarizations if non-standard rule types (evolution is applied sequentially and may change the polarization) are used. Are these numbers optimal?

2.2 Membrane Systems Language Class (2010)

A fundamental family of languages is still not characterized: languages generated by non-cooperative membrane systems. It is known, [5, 4] that the best known lower bound for $LOP(ncoo, tar)$ is $REG \cdot \texttt{Perm}(REG)$ (strict inclusion), while the best known upper bound is $CS \cap SLIN \cap \mathbf{P}$. An example of a difficult language in this family is

$$\{ \quad \texttt{Perm}((abc)^{2k_0})\texttt{Perm}((a'b'c')^{2k_1}) \cdots \texttt{Perm}((abc)^{2k_{2t}})\texttt{Perm}((a'b'c')^{2k_{2t+1}}) \\ \mid k_0 = 1,\ 0 \leq k_i \leq 2k_{i-1},\ 1 \leq i \leq 2t+1,\ t \geq 0\}.$$

Open questions concerning comparison of the P systems language family with particular language families and concerning particular closure properties are also formulated in the above mentioned papers.

2.3 Dynamic Properties (2011)

It is well-known, e.g., that catalytic P systems are computationally complete, while deterministic catalytic P systems are not.

In [3], an overview of a number of dynamic properties of P systems is given, the most important one being determinism. In particular, five variants are recalled where non-determinism seems an essential source of the computational power (although, as far as we know, no formal proof of power separation has been obtained), with informal justification for the word "seems":

1. P systems with active membranes, where except membrane separation, the rules are non-cooperative and the membrane structure is static (solving SAT).
2. Non-cooperative P systems with promoters or inhibitors of weight not restricted to one (universality).
3. Minimal combinations of alphabet size/number of membranes or cells (universality).
4. P systems without polarizations (universality).
5. Conditional uniport.

The open question is, for any of the variants above, to formally prove that determinism decreases the computational power of the corresponding systems (as it is in the case of catalytic systems).

The post-proceedings version (submitted) of [3] also proposes to study 6 new formal properties inspired by self-stabilization concept.

2.4 Exo-Insertion/Deletion (2011)

This is the only open problem in this list that concerns P systems with string objects. Consider P systems with string objects and operations of right or left insertion or deletion of given strings. The problem is to find a characterization of the power of P systems with exo-insertion of weight one and exo-deletion of weight one without contexts.

There exist the following partial results:

- Not computationally complete if operations (even both with weight two) are performed anywhere in the string.
- Computationally complete if insertion has weight two.
- Computationally complete if deletion has weight two.
- Computationally complete for tissue P systems.
- Computationally complete if deletion has priority over insertion (even without deletion on the right).
- The lower bound is regular languages (even with all operations on one side).

2.5 Symport-3 in One Membrane (2005)

Reaching for universality by moving objects across a single membrane lead to interesting combinatorial questions. While antiport roughly corresponds to rewriting, symport does not provide such an intuitive counterpart, although it remotely resembles insertion/deletion or vector addition.

It is well known that the minimal size of symport rules for the universality in one membrane is 3, [6]. The computational completeness is achieved there with 7 additional objects in the skin. It is not difficult to see that at least one object is necessary, or only finite sets are generated.

Indeed, the only way to increase the number of objects is to send something out, so that something comes back in, bringing something else. Generating any infinite set means that such a procedure must be iterated. Hence, sending all objects out cannot lead to halting.

Therefore, the lower bound for $LOP_1(sym_3)$ is N_7RE, while the upper bound is $N_1RE \cup NFIN$. It is an open problem to bridge (or at least decrease) the gap by investigating what sets containing numbers smaller than 7 can be generated.

References

1. A. Alhazov: Minimal Parallelism and Number of Membrane Polarizations. *The Computer Science Journal of Moldova*, **18**, 2(53), 2010, 149–170.
   `http://www.math.md/publications/csjm/issues/v18-n2/10284/`
2. A. Alhazov: Minimal Parallelism and Number of Membrane Polarizations. In: H.J. Hoogeboom, Gh. Păun, G. Rozenberg, eds.: *Preproceedings of the Seventh International Workshop on Membrane Computing*, WMC7, Lorentz Center, Leiden, 2006, 74–87.
   `http://wmc7.liacs.nl/proceedings/WMC7Alhazov.pdf`
3. A. Alhazov: Properties of Membrane Systems. In: M. Gheorghe, Gh. Păun, S. Verlan, eds.: —it Preproceedings of the Twelfth International Conference on Membrane Computing, CMC12, Fontainebleau, 2011, 3–14.
   `http://cmc12.lacl.fr/cmc12proceedings.pdf`
4. A. Alhazov, C. Ciubotaru, S. Ivanov, Yu. Rogozhin: The Family of Languages Generated by Non-cooperative Membrane Systems. *Membrane Computing. 11th International Conference*, CMC 2010, Jena, 2010. Revised Selected Papers (M. Gheorghe, Th. Hinze, Gh. Păun, G. Rozenberg, A. Salomaa, eds.) LNCS **6501**, Springer, 2011, 65–80. `http://www.springerlink.com/content/gt07j477k2020417/`
5. A. Alhazov, C. Ciubotaru, S. Ivanov, Yu. Rogozhin: Membrane Systems Languages Are Polynomial-Time Parsable. *The Computer Science Journal of Moldova*, **18**, 2(53), 2010, 139–148.
   `http://www.math.md/publications/csjm/issues/v18-n2/10282/`
6. A. Alhazov, R. Freund, Yu. Rogozhin: Computational Power of Symport/Antiport: History, Advances and Open Problems. In: R. Freund, Gh. Păun, G. Rozenberg, A. Salomaa, eds.: *Membrane Computing, 6th International Workshop*, WMC 2005, Vienna, Revised Selected and Invited Papers, LNCS **3850**, Springer, 2006, 1–30.
   `http://springerlink.metapress.com/index/f500782x34331741`

7. A. Alhazov, A. Krassovitskiy, Yu. Rogozhin: Circular Post Machines and P Systems with Exo- Insertion and Deletion. In: M. Gheorghe, Gh. Păun, S. Verlan, eds.: *Preproceedings of the Twelfth International Conference on Membrane Computing*, CMC12, Fontainebleau, 2011, 63–76.
http://cmc12.lacl.fr/cmc12proceedings.pdf

3 Polymorphic P Systems

Sergiu Ivanov[1,2], Artiom Alhazov[1,3], Yurii Rogozhin[1]

[1] Institute of Mathematics and Computer Science
Academy of Sciences of Moldova
Chişinău, Republic of Moldova
{artiom,rogozhin,sivanov}@math.md

[2] University of Academy of Sciences of Moldova
Faculty of Real Sciences
Chişinău, Republic of Moldova

[3] Università degli Studi di Milano-Bicocca
Dipartimento di Informatica, Sistemistica e Comunicazione
Milano, Italy
aartiom@yahoo.com

Polymorphic P systems introduce a new feature into membrane computing. This time the inspiration does not come from biology, but rather from conventional computing and namely from von Neumann architecture. The point is in not fixing the rules in the structural description of the P system, but rather storing them as contents of membranes.

Formally, we define a polymorphic P system as a tuple

$$P = (O, T,\ \mu,\ w_s, w_{1L}, w_{1R}, \ldots, w_{mL}, w_{mR},\ \varphi,\ i_{out}).$$

The set O is a finite alphabet, $T \subseteq O$ is the set of output objects. The object μ is a tree structure consisting of $2m+1$ membranes bijectively labeled with the elements of $H = \{s\} \cup \{iL, iR \mid 1 \leq i \leq m\}$. The skin membrane is labeled with s. It is required that the parent membrane of iL be the same as the parent membrane of iR for all $1 \leq i \leq m$. The string w_h, $h \in H$, is the initial content of the membrane with label h. The label i_{out} is the region where the output of the system will be read from. We will describe the mapping φ later on.

Observe that the description of the system does not include any rules. Instead, the contents of the membranes with labels iL and iR are interpreted as the left-hand side and right-hand side of the rule i respectively. At every step, the rules are

applied in the usual way. As a result of application of the rule i, the right-hand side of the rule (the content of iR) is injected into $\varphi(i)$. The latter mapping is defined as follows: $\varphi : \{1, \ldots, m\} \to Tar$, $Tar = \{in_j \mid j \in H\text{–}$ inner membrane of $p\} \cup \{out, here\}$, where $p \in H$ is the label of the membrane containing the rule i (the membranes iL and iR). For further information we refer the reader to [1].

Polymorphic P systems have not yet been explored sufficiently well. In the following paragraphs we will list some open problems which we find important to be considered.

- *Solve hard problems.* It has been shown that polymorphic P systems can solve certain problems faster than any other P system model (for example, generate n^2 in $O(1)$ and generate 2^{2^n} in $O(n)$). So far, only relatively simple problems were considered, but we believe that the polymorphic model has the potential to facilitate solving much harder problems. For example, possibilities to find the Gröbner basis using polymorphic P systems are currently being considered.
- *Characterize problems which may be solved faster.* A more general question, on the other hand, is to define the class of problems which can be solved more efficiently using polymorphic P systems. It has been observed that, for multiplication, a linear speed-up was introduced; a much more systematic research in this direction is necessary. In particularly, it is unclear whether it is possible to use polymorphism to construct exponential workspace for solving intractable problems in polynomial time.
- *Polymorphic P systems with active membranes.* Polymorphic P systems are a fairly simple model at the moment. This means, in particular, that certain extensions are possible. We would like to particularly stress the perspectives of considering polymorphic P systems with active membranes, where the membrane structure itself does not stay constant. Such a combination is a very powerful one, therefore it is important to establish some restrictions which will define an as simple as possible, yet sufficiently powerful, construct.
- *The power of the most restricted variant.* Another way to explore polymorphic P systems is characterizing the power of models with the minimal number of additional ingredients (non-cooperative rules, no rules with empty left-hand side, no target indications). In [1] it is shown that even this model can easily achieve superexponential growth; it is important to know how powerful polymorphism on its own is.
- *Self-assembly.* Finally, we make the observation that rules in polymorphic P systems may be treated as results of interaction of couples of initially independent membranes, which have gained additional capabilities by connecting to each other. The whole polymorphic P system may be treated as a stage in the process of interaction of membranes in a system of membranes. This brings about, in particular, the question of self-assembly of membrane structures.

References

1. A. Alhazov, S. Ivanov, Yu. Rogozhin: Polymorphic P Systems. In: M. Gheorghe, T. Hinze, Gh. Păun, G. Rozenberg, A. Salomaa, eds., *11th International Conference on Membrane Computing*, CMC 2010. LNCS 6501, pp. 81 – 94. Springer, Berlin (2010).

4 Research Directions in the Theory of P Colonies and dP Automata

Erzsébet Csuhaj-Varjú

Eötvös Loránd University
Budapest, Hungary
csuhaj@inf.elte.hu

P colonies are variants of very simple tissue-like P systems, modeling a community of very simple cells living together in a shared environment (for basic information see [8]).

In the basic model, the cells (or agents) are represented by a collection of objects and rules for processing these objects. The agents are restricted in their capabilities, i.e., only a limited number of objects, say, k objects, are allowed to be inside any cell during the function of the system. Number k is said to be the capacity of the P colony. The rules of the cells are either of the form $a \rightarrow b$, specifying that an internal object a is transformed into an internal object b, or of the form $c \leftrightarrow d$, specifying the fact that an internal object c is sent out of the cell, to the environment, in exchange of the object d, which was present in the environment. After applying these rules in parallel, a cell containing the objects a, c will contain the objects b, d. With each cell, a set of programs composed of such rules is associated. In the case of P colonies of capacity k, each program has k rules; the rules of the program must be applied in parallel to the objects in the cell.

The cells of a P colony execute a computation by synchronously applying their programs to objects inside the cells and outside in the environment. At the beginning of the computation, performed by a given P colony of capacity k, the environment contains arbitrarily many copies of a distinguished symbol e, called the environmental symbol (and no more symbols); furthermore, each cell contains k copies of e. When a halting configuration is reached, that is, when no more rules can be applied, the result of the computation is read as the number of certain types of objects present in the environment.

P colonies have been extensively examined during the years. It was shown that these simple constructs are computationally complete computing devices even with

very restricted size parameters and with other syntactical or functioning restrictions. Several extensions of the model have already been investigated as well: P colonies with dynamically varying environment (eco-P colonies) [1] or PCol automata [2], constructs where the behavior of the cells is influenced by direct impulses coming from the environment step-by-step. In the case of a PCol automaton a tape with an input string is given with the P colony, i.e., the model is augmented with a string put on an input tape to be processed by the P colony.

Except PCol automata, P colonies have been considered as generating devices, but the construct can also be considered as an accepting device (called accepting P colony or P colony acceptor), possibly working in an automaton-like fashion as well.

To define such a model, suppose that we have a P colony Π of capacity k and initialize the environment with a given finite multiset of symbols M where each symbol is different from the environmental symbol e. Let also consider an initial configuration, i.e., let us dedicate an initial state to any cell and let us distinguish a set of accepting configurations. Then, we say that M is accepted by Π, if after performing a finite computation (in some computation mode) the environment consists of only symbols e.

It is easy to see that we may consider several variants of this model: for example,

- we can limit the number of symbols in the environment (not necessarily with a finite constant, but with some function of the size of the P colony) and study the computational power of these systems with limited workspace for the computation,
- we can consider the multisets in the environment during the computation as permutations of words (or map them to words in some other way) being on the input tape of an automata and study the relation of these constructs and classical automata;
- we can map the sequences of multisets of objects entering each cell during the computation to words being on the input tape of a multitape or multihead automata and describe the correspondence between these constructs and the classical multitape or multihead automata variants.

Furthermore, by introducing double alphabets as in the case of dP automata for describing two-way multihead finite automata ([3]), automata with two-way motion of heads can also be interpreted in the framework of accepting P colonies.

In addition to demonstrate how classical automata can be represented as P colony acceptors and reversely, there are problems which would be of particular interest. These are computational complexity, communication complexity, and size complexity questions. For example, since finite automata can be represented in a natural manner in the frame of accepting P colonies, a comparative study between descriptional complexity of finite automata and P colony acceptors would certainly be useful.

The concept of accepting P colonies can be extended in some other manners as well. For example, we may not fix the number of cells in the P colony in advance

but it may be determined by the number of non-environmental symbols in the environment at the beginning. We also may define spatial P colonies where spatial parameters are added to the cells which define a neighborhood relation among the components; any cell can import only such symbols from the environment which were issued by its neighbors (is in its own environment).

Accepting P colonies can be related to cellular automata as well. One natural idea is to define P colonies corresponding to one-way cellular automata, which are linear arrays of identical copies of deterministic finite automata, called cells, working synchronously at discrete time steps. Each cell is connected to its immediate neighbors to the right. The cells are identified by positive integers. The state transition depends on the current state of a cell itself and the current state of its neighbor. An input word is accepted by a one-way cellular automata if at some step in the course of the computation the leftmost cell enters an accepting state.

A particular variant of one-way cellular automaton is the one where only a fixed number, say k, cells are given. This works similarly to the unrestricted case, but the input is processed in a different manner, namely, the input is not given at the beginning, but it is processed by the rightmost cell, symbol by symbol. Since the neighborhood can be defined in P colonies with emitting special symbols (signals) in the environment, and any cell in the P colony may have only a finite number of configurations (states), the reader may easily observe that the two computational models, the accepting P colony and the k-cell one-way cellular automaton are strongly related.

Obviously, more general cellular automata models can also be described by P colony acceptors. For example, the above extension of the concept of P colonies where the number of cells is determined by the number of initial non-environmental symbols can correspond to the unrestricted case. We can also model d-dimensional cellular automata ($d \geq 1$) by defining the neighborhood relation between cells of P colonies in an appropriate manner. Cellular automata theory has been a highly elaborated field of nature-motivated, parallel computing (see, for example, [5], [6], [7]), thus by building bridges between P colony theory and cellular automata theory, many interesting problems can also be studied.

In addition to compare accepting P colonies to variants of classical automata, we may explore the differences and similarities between these constructs and (finite) dP automata as well. A detailed study in this direction would also help in better understanding the nature of these two constructs.

P automata are variants of antiport P systems accepting strings in an automaton-like fashion (for a summary on P automata, see Chapter 6, [8]). The notion of a distributed P automaton (dP automaton in short) was introduced in [9]. Such a system consists of a finite number of component P automata which have their separate inputs and which also may communicate with each other by means of special antiport-like rules. A string accepted by a dP automaton is obtained in [9] as the concatenation of the strings accepted by the individual components during a computation performed by the system.

A dP automaton is called finite if it has only a finite number of different configurations.

The computational power of dP automata was studied in [9], [4], [10], and [11]. Among other things, it was shown that dP automata (with representing multisets as all permutations of strings) are strictly more powerful than P automata, but the language family accepted by them is strictly included in the family of context-sensitive languages.

In [3] a connection between finite dP automata (distributed P automata) and non-deterministic multi-head finite automata was explored; it was shown that the language of a non-deterministic one-way multi-head finite automaton and the language of a non-deterministic two-way multi-head finite automaton can be obtained as so-called weak agreement language or strong agreement language of a one-way, i.e., a usual finite dP automaton, and a two-way finite dP automaton.

The reader may easily observe that finite dP automata, P colony acceptors and cellular automata are closely related concepts. Their comparative study would be a promising and very useful area in P systems theory.

References

1. L. Cienciala, L. Ciencialová: Eco-P colonies. In Păun et al., eds, WMC 2009, LNCS 5957, Springer, 201-209.
2. L. Ciencala, L. Ciencialová, E. Csuhaj-Varj, Gy. Vaszil, PCol Automata: Recognizing strings with P colonies. In: Proc. BWMC 2010, Sevilla, 2010, M.A. Martnez-del-Amor et al., eds., Fénix Editora, Sevilla, 2010, 65-76.
3. E. Csuhaj-Varj, Gy. Vaszil: A Connection Between Finite dP Automata and Multi-head Finite Automata. In *Proc. Twelfth International Conference on Membrane Computing*, Fontainebleau, 23-26 August, 2011. M. Gheorghe, Gh. Păun, S. Verlan, eds., University of Paris Est, Crteil Val de Marne, 2011, 109-126.
4. R. Freund, M. Kogler, G. Păun, M.J. Pérez-Jiménez: On the power of P and dP automata. *Annals of Bucharest University. Mathematics-Informatics Series*, 63, 2009, 5–22.
5. M. Kutrib, Nature-Based Problems in Cellular Automata, CiE 2011, LNCS 6735, Springer, 2011, 171-180.
6. M. Kutrib, J. Lefevre, A. Malcher: The Size of One-Way Cellular Automata. *Automata 2010: Discrete Mathematics and Theoretical Computer Science Proceedings*, DMTCS, 2010, 71-90.
7. M. Holzer, M. Kutrib: Cellular Automata and the Quest for Nontrivial Artificial Self-Reproduction. Int. Conf. on Membrane Computing, CMC 2010, LNCS 6501, Springer, 2010, 19-36.
8. G. Păun, G. Rozenberg, A. Salomaa, eds.: *The Oxford Handbook of Membrane Computing*, Oxford University Press, 2010.
9. G. Păun, M.J. Pérez-Jiménez, Solving Problems in a Distributed Way in Membrane Computing: dP Systems, *International Journal of Computers, Communication & Control*, V(2), 2010, 238–250.
10. G. Păun, M.J. Pérez-Jiménez: P and dP automata: A survey. In: C.S. Calude, G. Rozenberg, A. Salomaa, eds., *Rainbow of Computer Science*, LNCS, 6570, pages 102–115, Springer, Berlin, 2011.

11. G. Păun, M.J. Pérez-Jiménez, An Infinite Hierarchy of Languages Defined by dP Systems. Submitted, 2010.

5 Speeding up P Automata

György Vaszil

Department of Computer Science
Faculty of Informatics
University of Debrecen, Hungary
vaszil.gyorgy@inf.unideb.hu

First we state the problem, then recall the needed definitions and discuss the necessary steps to a possible solution.

Problem 1 *Are there languages (over some finite alphabet T) accepted by P automata with object alphabet V which are (k, l, m)-efficiently parallelizable for some $k, m > 1$, $l \geq 1$, with respect to some input mapping $f : V^* \to 2^T$, such that $f \neq f_{perm}$?*

To see what this problem is about, we start with the definitions of the following: P automaton, accepted multiset sequence, input mapping, accepted language.

A *P automaton*, introduced in [2], is a system $\Pi = (V, \mu, P_1, \ldots, P_n, c_0, \mathcal{F})$ where V is an object alphabet, μ is a membrane structure, P_i, $1 \leq i \leq n$ are sets of antiport rules, $c_0 = (w_1, \ldots, w_n)$ is the initial configuration with $w_i \in V^*$, $1 \leq i \leq n$ being the initial contents of the ith region, and $\mathcal{F}$ is a set of accepting configurations of the form $(v_1, \ldots, v_n)$, $v_i \in V^*$, $1 \leq i \leq n$, where $\mathcal{F}$ is given as $E_1 \times \ldots \times E_n$, $E_i \subseteq V^*$, such that E_i is either finite, or $E_i = V^*$, $1 \leq i \leq n$.

The configurations of the P automaton are changed by applying the rules in the maximal parallel manner. This means that a (locally) maximal collection $w \in V^*$ of objects is moved according to the rules, that is, there is no rule $r \in \bigcup_{i=1}^{n} P_i$, such that r could be applied to objects which are not in w.

For two configurations $c, c' \in (V^*)^n$, we say that $c' \in \delta_\Pi(u, c)$ if Π enters configuration c' from configuration c by applying its rules while reading the input $u \in V^*$, that is, if u is the multiset that enters the system through the skin membrane from the environment while the configuration c changes to c'.

The sequence of configurations obtained this way is called a computation. If it ends in a final configuration from $\mathcal{F}$, then the sequence of multisets entering the system from the environment in each step of the computation is called an *accepted multiset sequence*. Thus, $v_1, \ldots, v_s$, $v_i \in V^*$, $1 \leq i \leq s$, is an accepted multiset sequence of Π if there are $c_0, c_1, \ldots, c_s \in (V^*)^n$, such that $c_i \in \delta_\Pi(v_i, c_{i-1})$, $1 \leq i \leq s$, and $c_s \in \mathcal{F}$.

Now we establish the correspondence between the accepted multiset sequences and the accepted languages of P automata.

Let Π be a P automaton over the object alphabet V, and let f be a mapping $f : V^* \to 2^{T^*}$ for some finite alphabet T. We call f the *input mapping* of Π. Let us assume that f is nonerasing, that is, $f(u) = \{\varepsilon\}$ for some $u \in V^*$, if and only if $u = \emptyset$. The *language* over T accepted by Π with respect to f is defined as $L(\Pi, f) = \{f(v_1) \dots f(v_s) \mid v_1, \dots, v_s$ is an accepted multiset sequence of $\Pi\}$.

It is obvious, that the choice of f in the definition above has a great influence on the accepting power of the P automaton, so we take a closer look at the mappings we can use. We define the mapping f_{perm}, the class of mappings TRANS, and discuss a little the power of the corresponding systems.

Let $f : V^* \to 2^{T^*}$, for some alphabets V and T, and let

(1) $f = f_{perm}$ if and only if $V = T$ and for all $v \in V^*$, we have $f(v) = \{a_1 a_2 \dots a_s \mid |v| = s$, and $a_1 a_2 \dots a_s$ is a permutation of the elements of $v\}$; or

(2) $f \in$ TRANS if and only if for any $v \in V^*$, we have $f(v) = \{w\}$ for some $w \in T^*$ which is obtained by applying a finite transducer to the string representation of the multiset v (as w is unique, the transducer must be constructed in such a way that all string representations of the multiset v as input result in the same $w \in T^*$ as output, and moreover, as f should be nonerasing, the transducer produces a result with $w \neq \varepsilon$ for any nonempty input).

To give a feeling about the influence of the type of the input mapping, we recall from [6] that there are simple linear languages which cannot be accepted by P automata with f_{perm}, for example $L = \{(ab)^n (ac)^n \mid n \geq 1\}$ is such a language. On the other hand, the class of languages accepted with f_{perm} also contains non-context-free context-sensitive languages ($\{a^n b^n c^n \mid n \geq 1\}$ for example), which means that it is incomparable with the class of linear and of context-free languages. (Although it contains all regular languages, see [3].)

In contrast to these results, systems with input mappings from the class TRANS characterize exactly the class of context-sensitive languages (see [1] for details).

The notion of distributed P automaton (dP automaton) was introduced in [5] to incorporate a "different kind of parallelism" into P systems: the components of a dP automaton process different parts of the input in parallel. A language is efficiently parallelizable if it can be accepted by a dP automaton in "less" (see below) computational steps then by any non-distributed P automaton.

A *distributed P automaton* is a construct $d\Pi = (V, \Pi_1, \dots, \Pi_k, R)$ where V is a finite set of objects, Π_i, $1 \leq i \leq k$, are the components of the system with $\Pi_i = (V, \mu_i, P_{i,1}, \dots, P_{i,m_i}, c_{i,0}, \mathcal{F}_i)$, $1 \leq m_i$, being P automata as defined above having the skin membranes labeled by $(i, 1)$, and R is a finite set of inter-component communication rules of the form $((i, 1), u/v, (j, 1))$ with $u, v \in V^*$, $1 \leq i, j \leq k$, $i \neq j$. The initial configuration of the dP automaton is $c_0 = (c_{1,0}, \dots, c_{k,0})$.

The language $L \subseteq T^*$ accepted by a dP automaton consists of words of the form $w_1 w_2 \dots w_k$ where $w_i \in T^*$ are strings accepted by the component Π_i, $1 \leq i \leq k$, during a successful computation, that is, one that starts in $c_{i,0}$ and ends in one

of the final configurations of $\mathcal{F}_i$ from $\mathcal{F} = (\mathcal{F}_1, \ldots, \mathcal{F}_k)$. Let $f = (f_1, \ldots, f_k)$ be a mapping $f : (V^*)^k \to (2^{T^*})^k$ with $f_i : V^* \to 2^{T^*}$, $1 \leq i \leq k$, being non-erasing, and let $L(d\Pi, f) = \{w_1 \ldots w_k \in T^* \mid w_i \in f_i(v_{i,1}) \ldots f_i(v_{i,s_i}),\ 1 \leq i \leq k,$ where $v_{i,1}, \ldots, v_{i,s_i}$ is an accepted multiset sequence of the component $\Pi_i\}$.

A language L is (k, l, m)-*efficiently parallelizable with respect to a class of mappings* F, for some $k, m > 1,\ l \geq 1$, if L can be accepted with a dP automaton $d\Pi$ with k components, such that $L = L(d\Pi, f)$ for some $f \in F$ with $\mathrm{Com}(d\Pi) \leq l$, and moreover, for all P automata Π and $f' \in F$ such that $L = L(\Pi, f')$,

$$\lim_{x \in L, |x| \to \infty} \frac{\mathrm{time}_{\Pi}(x)}{\mathrm{time}_{d\Pi}(x)} \geq m$$

where $time_X(x)$ denotes the number of computational steps that a device X needs to accept the string x, and where $\mathrm{Com}(d\Pi)$ denotes the maximal amount of communication (measured in some reasonable way, see [5]) between the components of the dP automaton $d\Pi$ during an accepting computation.

By looking at the quotient in the definition above, we might see that a language cannot satisfy the requirement of efficient parallelizability if the dividend (that is, the time that a non-distributed P automaton needs to accept the language) can be made arbitrarily small. This leads us to the problem of the possible speedup of P automata computations.

Considering the computations of Turing machines, a linear speedup is always possible by appropriately encoding the contents of the worktapes, but as the input usually has to remain in its original form on the input tape, the resulting time complexity cannot be less than the length of the input word (see for example [4]). Such a lower bound does not necessarily exist in the case of P automata, while the input itself is also "encoded" by the input mapping, so it might be possible to "read" the same word in different numbers of steps.

To demonstrate this possibility, let us recall from [8] that for any regular language L and constant $c > 0$, there exists a P automaton Π such that $L = L(\Pi, f)$ for some $f \in \mathrm{TRANS}$, and for any $w \in L$ with $|w| = n$ it holds that $\mathrm{time}_{\Pi}(w) \leq c \cdot n$. This, as we outlined above, implies that there are no efficiently parallelizable languages with respect to the class of input mappings TRANS.

The situation is different, however, if instead of an input mapping from the class TRANS, we consider f_{perm}. There are regular languages (called "frozen" in [7]) where the order of no two adjacent symbols can be exchanged, thus, each of them has to be read in different computational steps, which means that the computation of the P automaton cannot be shorter then length of the input.

After this, it is not surprising that there are efficiently parallelizable regular languages with respect to f_{perm}, as shown in [5].

So far all cases of efficient parallelizability were demonstrated with respect to the input mapping f_{perm}, thus, to state our problem at the end of these considerations one more time, it would be interesting to discover a language which is efficiently parallelizable with respect to an input mapping of some different kind.

On the other hand, it would also be interesting to prove the impossibility of efficient parallelization of not just the regular, but also of some more general language classes with respect to a class of input mappings different from f_{perm} (with respect to TRANS for example). To this aim, it would be sufficient to find a general method which (similarly to the case of Turing machines) would enable us to show that with a certain type of input mappings (TRANS for example, as it is the case for regular languages) a linear speedup of P automata is always possible.

References

1. E. Csuhaj-Varjú, M. Oswald, Gy. Vaszil: P automata. In: Gh. Păun, G. Rozenberg, A. Salomaa, eds., *The Oxford Handbook of Membrane Computing*, chapter 6, pages 144–167. Oxford University Press, 2010.
2. E. Csuhaj-Varjú, Gy. Vaszil: P automata or purely communicating accepting P systems. In: Gh. Păun, G. Rozenberg, A. Salomaa, C. Zandron, eds., *Membrane Computing, International Workshop, WMC-CdeA, Curtea de Arges, Romania, August 19-23, 2002, Revised Papers*, LNCS 2597, pages 219–233. Springer, Berlin, 2003.
3. R. Freund, M. Kogler, Gh. Păun, M.J. Pérez-Jiménez: On the power of P and dP automata. *Annals of Bucharest University Mathematical-Informatics Series*, LVIII:5–22, 2009.
4. J.E. Hopcroft, J.D. Ullman: *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, Reading, MA, 1979.
5. Gh. Păun, M.J. Pérez-Jiménez: Solving problems in a distributed way in membrane computing: dP systems. *International Journal of Computing, Communication and Control*, V(2):238–250, 2010.
6. Gh. Păun, M.J. Pérez-Jiménez: P and dP automata: A survey. In: C. Calude, G. Rozenberg, A. Salomaa, eds., *Rainbow of Computer Science*, LNCS 6570, pages 102–115. Springer, Berlin, 2011.
7. Gh. Păun, M.J. Pérez-Jiménez: An infinite hierarchy of languages defined by dP systems. *Theoretical Computer Science*, to appear.
8. Gy. Vaszil: On the parallelizability of languages accepted by P automata. In J. Kelemen and A. Kelemenová, editors, *Computation, Cooperation, and Life. Essays Dedicated to Gheorghe Păun on the Occasion of His 60th Birthday*, LNCS 6610, pages 170–178. Springer, Berlin Heidelberg, 2011.

6 Milano Open Problems

**Alberto Leporati, Giancarlo Mauri,
Antonio E. Porreca, Claudio Zandron**

Dipartimento di Informatica, Sistemistica e Comunicazione
Università di Milano-Bicocca, Italy
{leporati, mauri, porreca, zandron}@disco.unimib.it

Comment. We have chosen the previous title for this section, on the one hand, because the problems we have received from Milano falls in two categories, (state) complexity and power of energy-based P systems, and because Milano is one of the strongest research groups in MC (in particular, very active in complexity matters) – this is just an invitation to stay in contact with them...

Problem 1 (P systems with elementary active membranes). P systems with active membranes [12] are known to be able to solve computationally hard problems in polynomial time by creating exponentially many membranes via division. The most recent result in this area [10] shows that polynomial-time Turing machines having access to an oracle for a **PP** [2] problem (whose computing power includes the polynomial hierarchy [14]) can be simulated by uniform families [8] of P systems with active membranes where the only membranes subject to division are elementary (i.e., not containing further membranes), and no dissolution rules are needed. This result is stated, in symbols, as $\mathbf{P}^{\mathbf{PP}} \subseteq \mathbf{PMC}_{\mathcal{AM}(-\mathrm{d},-\mathrm{n})}$. On the other hand, this kind of P system cannot solve in polynomial time any problem outside **PSPACE** [13], in symbols $\mathbf{PMC}_{\mathcal{AM}(-\mathrm{d},-\mathrm{n})} \subseteq \mathbf{PSPACE}$. Neither inclusion is known to be proper.

Is $\mathbf{PMC}_{\mathcal{AM}(-\mathrm{d},-\mathrm{n})} = \mathbf{PSPACE}$ or, more generally, is there a precise characterization of $\mathbf{PMC}_{\mathcal{AM}(-\mathrm{d},-\mathrm{n})}$ in terms of complexity classes for Turing machines?

Problem 2 (Space complexity of P systems with active membranes). A measure of space complexity for P systems has been recently introduced [9] in order to supplement the already rich literature about computational complexity issues in membrane computing [7]. We say that the space required by a P system is the maximal size it can reach during any computation, measured as the sum of the number of membranes and the number of objects. A uniform family $\boldsymbol{\Pi}$ of recognizer P systems [8] is said to solve a problem in space $f\colon \mathbb{N} \to \mathbb{N}$ if no P systems in $\boldsymbol{\Pi}$ associated to an input string of length n requires more than $f(n)$ space. Under this notion of space complexity, the class of problems solvable in polynomial space by P systems with active membranes [12], denoted by $\mathbf{PMCSPACE}_{\mathcal{AM}}$, coincides with **PSPACE** [11].

The techniques used to prove this result do not seem to apply when the space bound is strictly less, i.e., exponential or even super-exponential. Do these kinds of P systems with active membranes also exhibit the same computing power as Turing machines working under the same space constraints?

It might also be interesting to analyze the behavior of families of P systems with active membranes working in logarithmic space. In this case, there are two complications. First of all, we must slightly change the notion of space complexity, in order to allow for a "read-only" input multiset that is not counted when the space required by the P system is measured (similarly to the input tape of a logspace Turing machine). Furthermore, the notion of uniformity used to define the families of P systems will probably need to be weakened, since polynomial-time Turing machines constructing the family might be able to solve the problems altogether by themselves. More general forms of uniformity have already been investigated [6], and that work is going to be useful when attacking this problem.

Problem 3 (The computational power of Energy-based P systems). *Energy-based P systems* have been defined in [4] as a model of membrane systems that take into account the energy manipulated during computations. Each object has an associated amount of energy, and instances of a special symbol e are used to denote free energy units occurring inside the regions of the system. The rules transform one object into another by acquiring or releasing an appropriate number of free energy units in the region in which the rule is applied. These transformations satisfy the principle of energy conservation.

In [3], the following results concerning their computational power have been obtained, assuming that rules are applied in the sequential way. If the number of free energy units is bounded in each region then only a finite number of distinct configurations can be obtained, and thus these systems can be simulated by finite state automata. On the other hand, assuming the presence of an unbounded number of free energy units does not suffice to obtain universality, since in this case energy-based P systems can be simulated by vector addition systems. However this is just an upper bound, and so the exact computational power of energy-based P systems has not yet been determined; it would be extremely interesting if they were strictly less powerful than vector addition systems. On the other hand, in [3] it has also been proved that by assigning simple local priorities to the rules we obtain universality. This behavior is similar in some respects (but different for others) to what happens with UREM P systems [1], where energy is associated to membranes rather than with the objects.

The problem of characterizing the computational power of energy-based P systems can be posed also when the rules are applied in the maximally parallel way. In such a case, can universality be reached without using priorities? Up to now, the only known result concerning maximally parallel energy-based P systems is the ability to simulate n-input/n-output circuits composed of Fredkin gates [5], that compute conservative and reversible functions.

References

1. A. Alhazov, R. Freund, A. Leporati, M. Oswald, C. Zandron: (Tissue) P systems with unit rules and energy assigned to membranes. *Fundamenta Informaticae*, 74:391–408, 2006.

2. J.T. Gill: Computational complexity of probabilistic Turing machines. In *Proceedings of the Sixth Annual ACM Symposium on Theory of Computing, STOC '74*, pages 91–95, 1974.
3. A. Leporati, D. Besozzi, P. Cazzaniga, D. Pescini, C. Ferretti: Computing with energy and chemical reactions. *Natural Computing*, 9:493–512, 2010.
4. A. Leporati, C. Zandron, G. Mauri: Simulating the Fredkin gate with energy-based P systems. *Journal of Universal Computer Science*, 10(5):600–619, 2004.
5. Alberto Leporati, Claudio Zandron, Giancarlo Mauri. Reversible P systems to simulate fredkin circuits. *Fundamenta Informaticae*, 74:529–548, 2006.
6. N. Murphy, D. Woods: The computational power of membrane systems under tight uniformity conditions. *Natural Computing*, 10(1):613–632, 2011.
7. M.J. Pérez-Jiménez: A computational complexity theory in membrane computing. In Gh. Păun, M.J. Pérez-Jiménez, A. Riscos-Núñez, G. Rozenberg, A. Salomaa, eds., *Membrane Computing, 10th International Workshop, WMC 2009*, LNCS 5957, 125–148. Springer, 2010.
8. M.J. Pérez-Jiménez, Á. Romero-Jiménez, F. Sancho-Caparrini: Complexity classes in models of cellular computing with membranes. *Natural Computing*, 2(3):265–284, 2003.
9. A.E. Porreca, A. Leporati, G. Mauri, C. Zandron: Introducing a space complexity measure for P systems. *International Journal of Computers, Communications & Control*, 4(3):301–310, 2009.
10. A.E. Porreca, A. Leporati, G. Mauri, C. Zandron: P systems simulating oracle computations. In M. Gheorghe, Gh. Păun, S. Verlan, eds., *Pre-Proceedings of the Twelfth International Conference on Membrane Computing 2011 (CMC12)*, pages 433–445, 2011.
11. A.E. Porreca, A. Leporati, G. Mauri, C. Zandron: P systems with active membranes working in polynomial space. *International Journal of Foundations of Computer Science*, 22(1):65–73, 2011.
12. Gh. Păun: P systems with active membranes: Attacking NP-complete problems. *Journal of Automata, Languages and Combinatorics*, 6(1):75–90, 2001.
13. P. Sosík, A. Rodríguez-Patón: Membrane computing and complexity theory: A characterization of PSPACE. *Journal of Computer and System Sciences*, 73(1):137–152, 2007.
14. S. Toda: PP is as hard as the polynomial-time hierarchy. *SIAM Journal on Computing*, 20(5):865–877, 1991.

7 Complexity Issues

Nial Murphy

Department of Computer Science
National University of Ireland Maynooth, Ireland
nmurphy@cs.nuim.ie

7.1 Required Definitions

- Active membrane systems, including non-elementary division triggered by objects (sometimes called weak non-elementary division).
- Recognizer P systems.
- Uniform families.

7.2 Problem: Hierarchies

Characterizations, using P systems, of each level of the **NC** and Polynomial hierarchies [6] may shed new light on the inherent parallelism and non-determinism of P systems.

- The **NC** hierarchy represent a spectrum of problems ranging from constant time, to parallel logarithmic time, up to and (it is conjectured) not including the seemingly inherently sequential **P**) [2]. To learn more about the factors that limit and permit parallelism in P systems, a characterization of the $\mathbf{NC} \stackrel{?}{=} \mathbf{P}$ problem (the so called "frontier of parallelism"), might be a good place to start.
- The Polynomial Hierarchy starts at **P**, a deterministic class. The complete problems of each successive level of the hierarchy require increasing interleaving of nondeterminism and co-nondeterminism. The whole hierarchy is finally contained in **PSPACE**. Almost all complexity results about membrane systems to date place them in the 0th level (**P**), the 1st level ($\mathbf{NP} \cup \mathbf{coNP}$) or in **PSPACE**. Characterizing each level with a single model might give us clues to the role and use of of non-determinism in P systems.

Conjecture 1. Uniform families of active membrane systems using weak non-elementary division, without charges, and with a membrane structure of depth $d + 1$ can solve exactly those problems complete for the dth level of the Polynomial Hierarchy.

7.3 P-conjecture

The original statement of the P-Conjecture [7] asked if all active membrane systems without charges can be simulated in polynomial time. The original conjecture was disproved [1] however, the case considering only elementary division has proved much more of a challenge.

Conjecture 2 (The P-conjecture). The class of all decision problems solvable in polynomial time by active membranes without charges using evolution, communication, dissolution and division rules for elementary membranes is equal to the class **P**.

To solve restricted versions of the P-conjecture have necessitated the development of some of the most powerful and useful techniques in the complexity of membrane computing, such as dependency graphs [3, 5, 4, 8].

References

1. A. Alhazov, M.J. Pérez-Jiménez: Uniform Solution to QSAT Using Polarizationless Active Membranes. In *Machines, Computations and Universality (MCU)*, J. Durand-Lose and M. Margenstern, eds., LNCS 4664, Springer, 2007, 122–133.
2. R. Greenlaw, H. James Hoover, W.L. Ruzzo: *Limits to parallel computation:P-completeness Theory.* Oxford University Press, 1995.
3. M.A. Gutiérrez-Naranjo, M.J. Pérez-Jiménez, A. Riscos-Núñez, F.J. Romero-Campero: Computational Efficiency of Dissolution Rules in Membrane Systems. *International Journal of Computer Mathematics*, 83 (2006), 593–611.
4. N. Murphy, D. Woods: The Computational Power of Membrane Systems Under Tight Uniformity Conditions. *Natural Computing*, 10 (2011), 613–??.
5. N. Murphy, D. Woods: Active Membrane Systems Without Charges and Using Only Symmetric Elementary Division Characterise P.In *Membrane Computing, 8th International Workshop, WMC 2007*, LNCS 4869, Springer, 2007, 367–384.
6. C.H. Papadimitriou: *Computational Complexity.* Addison Wesley, 1993.
7. Gh. Păun: Further Twenty Six Open Problems in Membrane Computing. In *Proceedings of the Third Brainstorming Week on Membrane Computing*, Sevilla (Spain), 2005, 249–262.
8. D. Woods, N. Murphy, M.J. Pérez-Jiménez, A. Riscos-Núñez: Membrane Dissolution and Division in P. In *Unconventional Computation*, 5715 (2009), 262–276.

8 Time-Free Solutions for Hard Computational Problems

Matteo Cavaliere

National Center for Biotechnology
CNB - CSIC, Madrid, Spain
mcavaliere@cnb.csic.es

8.1 Motivations

Programming living things cannot assume neither general restrictions on execution times nor the presence of global clocks synchronizing the execution of different parallel processes. Moreover the time of execution of certain biological processes could vary because of external uncontrollable conditions.

Therefore, in the context of membrane systems, it seems crucial to investigate the power of cellular division when such timing assumptions are not used. Can we provide "time-free" solutions to complex decision problems (i.e., where the correctness of the solution does not depend on the precise timing of the involved processes)?

Here we suggest a possible application of the notion of time-freenes ([3]) to semi-uniform solutions of computational problems ([4]). The proposed approach could be extended to more general solutions (e.g., uniform, etc.).

8.2 Decision Problems

A decision problem X is a pair (I_X, Θ_X) where I_X is a countable language over a finite alphabet (the elements are called instances), and Θ_X is a predicate (a total boolean function) over I_X. It is well-known that there exists a natural correspondence between languages and decision problems (e.g., [4]).

8.3 Timed Recognizer P Systems

From [3] we recall the notion of timed P system.

A *timed P system* $\Pi(e)$ can be constructed by adding to a (standard) P system Π a time-mapping $e : R \longrightarrow \mathbb{N}$, where R is the set of rules of Π. The time-mapping specifies the *execution times* for the rules.

A timed P system $\Pi(e)$ works in the following way. We suppose to have an external clock that marks time-units of equal length (called *steps*), starting from step 0, when the system is present in its *initial configuration*.

At each step, all the rules that can be started, in each region, and for each membrane have to be be started (maximal parallel and non-deterministic way). *When a rule r is started at step j, then its execution terminates (the rule is completed) at step $j + e(r)$, that means the rule lasts $e(r)$ steps. The objects and the*

membranes produced by the rule are available - can be subject of other rules - only starting from the step $j+e(r)+1$). When a rule r is started, then the occurrences of symbol-objects and the membrane subject by this rule cannot be anymore subject of other rules.

A computation *halts* when no rule can be started in any region and there are no rules in execution (such configuration is called *halting configuration*). We say that the computation halts in k steps, if the external clock marks step k when the last rules of the computations are completed.

From [4] we recall recognizer P systems.

A *recognizer P system* is a P system such that: (i) the working alphabet contains two distinguished elements *yes* and *no*; (ii) all computations halt; and (iii) if $\mathcal{C}$ is a computation of the system, then either object *yes* or object *no* (but not both) must have been released into the environment, and only when the last rules of the computation have been completed.

We extend recognizer P systems to their timed variant.

A *recognizer timed P system* is a *timed P system* such that: (i) the working alphabet contains two distinguished elements *yes* and *no*; (ii) all computations halt; and (iii) if $\mathcal{C}$ is a computation of the system, then either object *yes* or object *no* (but not both) must have been released into the environment, and only when the last rules of the computation have been completed.

In recognizer timed P systems, we say that a computation is an *accepting computation* (respectively, *rejecting computation*) if the object *yes* (respectively, *no*) appears in the environment associated with the corresponding halting configuration.

8.4 Time-Free Solutions to Decision Problems

Let $X = (I_X, \Theta_X)$ a decision problem. Let $\Pi = \Pi_u, u \in I_X$ a (countable) family of recognizer P systems.

We say that the family Π is *sound* (with respect to X) if for each instance of the problem $u \in I_X$ such that there exists an accepting computation of Π_u, we have $\Theta_X(u) = 1$.

We say that the family Π is *complete* (with respect to X) if for each instance of the problem $u \in I_X$ such that $\Theta_X(u) = 1$, every computation of Π_u is an accepting computation.

We say that the family Π is *polynomially bounded* if there exists a polynomial function $p(n)$ such that, for each $u \in I_X$, all computations in Π_u halts in, at most, $p(|u|)$ steps.

We can now formalize the intuition briefly discussed in the Introduction: a solution to a problem is time-free if its soundness, its completeness and its polynomial bound do not depend on the time of execution associated to the rules of the constructed systems.

We say that the family Π is *time-free sound* (with respect to X) if, for any time-mapping e, the family $\Pi_e = \Pi_u(e), u \in I_X$, is sound with respect to X.

We say that the family Π is *time-free complete* (with respect to X) if, for any time-mapping e, the family $\Pi_e = \Pi_u(e), u \in I_X$, is complete with respect to X.

We say that the family Π is *time-free polynomially bounded* if, for any time-mapping e, the family $\Pi_e = \Pi_u(e), u \in I_X$, is polynomially bounded.

We can now adapt the definition of semi-uniform solutions, as given in [4], to consider time-free semi-uniform solutions.

8.5 Time-Free Semi-Uniform Solutions

Let $X = (I_X, \Theta_X)$ a decision problem. We say that X is solvable in a *time-free polynomial time* by a family of recognizer P systems $\Pi = \Pi_u, u \in I_X$ and we denote it by $X \in tfPMC$ if the following are true:

- the family Π is polynomially uniform by a Turing machines; that is, there exists a deterministic Turing machine working in polynomial time which constructs the system Π_u from the instance $u \in I_X$.
- the family Π is time-free polynomially bounded.
- the family Π is time-free sound and time-free complete (with respect to X).

We say that the family Π is a *time-free semi-uniform solution* to the decision problem X.

The intuition of the above definition is the following one: to provide a time-free solution one must construct the family of systems Π in polynomial time (sequential time by deterministic Turing machines). However the constructed family must be "fast" (polynomially bounded), sound and complete with respect the considered problem X and these properties must not depend on the time of execution of the rules (i.e., they must be fulfilled independently of the time-mapping considered).

8.6 Questions

Find a class of P systems for which is possible to construct time-free semi-uniform solutions of hard computational problems. E.g., can the solution given in [1] become a time-free solution ?

For which solution is possible to (algorithmically) check their time-freeness? Clearly, if the solutions are obtained by using universal systems (e.g., active membranes) this is generally not possible. Therefore, interesting sub-classes should be found: powerful enough to solve complex problems, simple enough to allow checking of time-freeness.

For which class of P systems is always possible to transform a non time-free solution into an equivalent time-free one ?

References

1. Gh. Păun: P Systems with Active Membranes: Attacking NP Complete Problems. CDMTCS Technical Report 102-1999.
2. M. Cavaliere, V. Deufemia: Further Results on Time-Free P Systems. *Int. Journal of Foundations of Computer Science*, 17,1, 2006, pp. 69 – 89.
3. M. Cavaliere, D. Sburlan: Time-Independent P Systems. *Membrane Computing. Int. Workshop WMC 2004*, Milan, Italy, 2004 (G. Mauri, Gh. Păun, M.J. Pérez-Jiménez, G. Rozenberg, A. Salomaa, Eds.), LNCS 3365, Springer, 2005, pp. 239 – 258.
4. M.A. Gutiérrez-Naranjo, M.J. Pérez-Jiménez, A. Riscos-Núñez, F.J. Romero-Campero: Computational Efficiency of Dissolution Rules in Membrane Systems. *Int. J. Computer Mathematics*, 83 (2006), 593–611.

9 Numerical P Systems

Cristian Vasile[1], Ana Brânduşa Pavel[1], Ioan Dumitrache[1], Gheorghe Păun[2]

[1]Department of Automatic Control and Systems Engineering
Politehnica University of Bucharest, Romania
{cvasile, apavel, idumitrache}@ics.pub.ro

[2]Institute of Mathematics of the Romanian Academy
Bucharest, Romania, and

Research Group on Natural Computing
Department of Computer Science and Artificial Intelligence
University of Sevilla, Spain
gpaun@us.es

Numerical P systems are a class of computing models inspired both from the cell structure and economics: numerical variables evolve in the compartments of a cell-like structure by means of so-called *production–repartition programs*. The variables have a given initial value and the production function is usually a polynomial, whose value for the current values of variables is distributed among variables in the neighboring compartments according to the "repartition protocol". In this way, the values of variables evolve; all positive values taken by a specified variable are said to be computed by the P system.

These computing devices were introduced in [9], where their computational completeness was proven. The results in [9], as well as further connections between membrane computing and economics, are recalled in Chapter 23.6 of [10].

In [9] and [10] one uses the numerical P systems only in the generating mode. However, numerical P systems were recently used in a series of papers (see references in [3]) for implementing controllers for mobile robots, and in this framework the P systems work in the computing mode: an input is introduced in the form of the values of some variables and an output is produced, as the value of other variables.

Furthermore, in the robot control context, the so-called *enzymatic* numerical P systems were introduced and used, [6], [7], [8]. Such systems correspond to *catalytic P systems* in the "general" membrane computing: a reaction takes place only in the presence of a catalyst. Here, the catalyst (enzyme) is a variable attached to an evolution program; the program is used only if the value of the enzyme is strictly greater than the smallest value of any variable involved in the production polynomial.

Using enzymes introduces a checking possibility in our systems (we compare the value of the enzyme with the values of variables from the associated program), and this suggests the possibility of choosing the positive values of the output variable "inside the system". Indeed, the following result holds true (for the non-deterministic case; *enz* indicates the use of enzymes). Tissue-like numerical P systems are also considered in [12], with parallel use of programs. If in each membrane, at each step, we use a maximal set of programs (programs are selected non-deterministically, and a set of programs is applied only if it is maximal, no further program can be added to it in such a way that the new set is still applicable. Two possibilities appear: (i) a variable can appear only in *one* production function, and this is the only restriction in choosing (non-deterministically) the programs to apply in a step, and (ii) if two or more programs which are enabled at a computation step, i.e., they satisfy the condition imposed by the associated enzymes, share variables in their production functions, then they will all use the current values of those variables (we denote this with *allP*).

A large variety of classes of numerical P systems appears in this way: (1) enzymatic or non-enzymatic, (2) deterministic or non-deterministic, (3) sequential, all-parallel, one-parallel. Still, we can add: (4) generating, computing, accepting (a number is introduced in the system, as the value of a variable, and it is accepted if a certain condition is met, e.g., a specified variable gets the value 0), and (5) selecting as results only the positive values of the output variable or accepting all values, but making sure that the output variable assumes only positive values. This is a subtle difference: in the first case, we just intersect the set of values of a variable with $\mathbf{N}$, the set of natural numbers, zero excluded, in the second case we have a *property* of the system. The former case reminds the extended Chomsky grammars and Lindenmayer systems, where a terminal alphabet is used in order to squeeze the generated language from the language of sequential forms, but an important difference is that here "the terminal" sets of numbers, $\mathbf{N}$, is fixed, is not at out choice, as in Chomsky grammars and L systems.

A plethora of classes of numerical P systems appear in this framework, waiting for research efforts. A few cases were settled in [12], where two main new ideas are

introduced: (i) working in the deterministic mode even in the generative case (and this is possible, because we do not define successful computations by halting), and (ii) using register machines, [4], as the starting characterization of Turing computable sets of numbers (instead of their characterization as the sets of positive solutions of Diophantine equations).

We do not recall here the definition of numerical P systems, with or without enzyme control, but we refer the reader to the papers mentioned above.

We only mention that the family of sets of numbers $N^+(\Pi)$ computed by numerical P systems with at most m membranes, production functions which are polynomials of degree at most n, with integer coefficients, with at most r variables in each polynomial, is denoted by $N^+P_m(poly^n(r), seq)$, $m \geq 1, n \geq 0, r \geq 0$, where the fact that we work in the sequential mode (in each step, only one program is applied), is indicated by *seq*. If one of the parameters m, n, r is not bounded, then it is replaced by $*$. (Both in $N^+(\Pi)$ and in $N^+P_m(poly^n(r), seq)$, the superscript $+$ indicates the fact that as the result of a computation we only consider positive natural numbers, zero excluded. If any value of x_{j_0,i_0} is accepted, then we remove the superscript $+$.) When tissue-like systems are used, we write $NtP_m(poly^n(r), \alpha, \beta)$.

Here are a few results from [9] and [12].

Theorem 2. $NRE = N^+P_8(poly^5(5), seq) = N^+P_7(poly^5(6), seq)$.

Theorem 3. $NRE = NP_7(poly^5(5), enz, seq)$.

Theorem 4. $NRE = NtP_*(poly^1(11), enz, oneP)$.

Theorem 5. $NRE = NP_{254}(poly^2(253), enz, allP, det)$.

Whether or not the parameters appearing in these results are optimal or not is an open problem.

Only a few of the many cases mentioned above were so far investigated, the other ones wait for research efforts.

In particular, we have seen that enzymes improve the universality results in terms of the complexity of used polynomials, both in the cell-like case and the tissue-like case, provided that the evolution programs are used in a parallel manner. However, two different types of parallelism were used in the two cases; can the one-parallel mode used for tissue-like P systems be used also in the cell-like case?

Similar extensions of "general" notions in membrane computing to numerical P systems remain to examined, and this is a rich research topic. For instance, other ways of using the programs can be considered: minimally parallel, with bounded parallelism, asynchronously. Then, we can also consider rules for handling membranes, such as membrane division and membrane creation. These operations are the basic tools by which polynomial solutions to computationally hard problems, typically, **NP**-complete problems, are obtained in the framework of P systems with symbol objects. Is this possible also for numerical P systems? In the previous investigations of numerical P systems, the computations were not halting, as usual

in membrane computing; does the halting condition makes any difference? (Note that in the enzymatic case the halting is possible: no evolution program can be applied if their associated enzymes forbid it.) What about numerical P systems used in the accepting mode?

A current issue in membrane computing is to find classes of P systems which are not universal. This extends also to numerical P systems.

Of course, an important research topic is to further explore the use of numerical P systems in controlling robots.

And so on and so forth, a wealth of research ideas, which supports our belief that numerical P systems deserve further research efforts.

References

1. O. Arsene, C. Buiu, N. Popescu: *SNUPS – A simulator for numerical membrane computing*. Intern. J. of Innovative Computing, Information and Control, 7, 6 (2011), 3509–3522.
2. C. Buiu, O. Arsene, C. Cipu, M. Pătraşcu: *A software tool for modeling and simulation of numerical P systems*. BioSystems, 103, 3 (2011), 442–447.
3. C. Buiu, C. Vasile, O. Arsene: *Development of membrane controllers for mobile robots*. Information Science, accepted.
4. M. Minsky: *Computation: Finite and Infinite Machines*. Prentice-Hall, 1967.
5. A.B. Pavel: *Membrane controllers for cognitive robots*. Master Thesis, Department of Automatic Control and System Engineering, Politehnica University of Bucharest, Febr. 2011.
6. A.B. Pavel, O. Arsene, C. Buiu: *Enzymatic numerical P systems – a new class of membrane computing systems*. The IEEE Fifth Intern. Conf. on Bio-Inspired Computing. Theory and applications. BIC-TA 2010, Liverpool, Sept. 2010, 1331–1336.
7. A.B. Pavel, C. Buiu: *Using enzymatic numerical P systems for modeling mobile robot controllers*. Natural Computing, doi: 10.1007/s11047-011-9286-5, in press.
8. A.B. Pavel, C. Vasile, C. Buiu: *Robot controllers implemented with enzymatic numerical P systems*. submitted.
9. Gh. Păun, R. Păun: *Membrane computing and economics: Numerical P systems*. Fundamenta Informaticae, 73 (2006), 213–227.
10. Gh. Păun, G. Rozenberg, A. Salomaa, eds.: *The Oxford Handbook of Membrane Computing*. Oxford Univ. Press, 2010.
11. The P Systems Web Page: `http://ppage.psystems.eu`.
12. C. Vasile, A.B. Pavel, I. Dumitrache, Gh. Păun: On the power of enzymatic numerical P systems. Submitted, 2011.

10 P Systems Formal Verification and Testing

Florentin Ipate[1], **Marian Gheorghe**[1,2]

[1]University of Piteşti, Department of Computer Science
Piteşti, Romania
florentin.ipate@ifsoft.ro

[2]University of Sheffield, Department of Computer Science
Sheffield, UK
m.gheorghe@dcs.shef.ac.uk

Formal verification of P systems through model checking. Formal verification of P systems using model-checking has attracted a significant amount of research in recent years, using tools such as Maude [1], PRISM [2], NuSMV [5], Spin [6] or ProB [4]. Most research has focussed on cell-like P systems with static structure, but more recently, P systems with active membranes, in particular with division rules, have also been investigated [7]. This is a significant advance from a practical point of view since P systems with division rules are commonly used to devise efficient solutions to computationally hard problems. A new line of research is envisaged to start here by combining methods to identify various invariants using Daikon, a tool which dynamically detects program invariants based on execution traces, with the formal verification of such elements. Future work is expected to focus on developing an integrated environment for specifying and formally verifying P systems using P-lingua, Daikon and one or more model checking tools and on extending the existing approaches to other classes of P systems (e.g. tissue P systems).

Model-based testing of P systems. Testing is an essential part of software development and all software applications, irrespective of their use and purpose, are tested before being released. As in the last years there have been significant developments in using the P systems paradigm to model, simulate and formally verify various systems (biology, economics, linguistics, graphics, computer science, etc.), methods for testing systems modeled as P systems must also exist. In the last years, a number of approaches to testing P systems, based on a combination of coverage criteria, state based techniques, model checking and mutation analysis have been developed [3, 5]. Essentially, these techniques have been developed in the context of cell-like P systems with a fixed structure. The challenge for the future is to extend these to P systems with active membranes, as well as other types of membrane systems; in particular, the development of a testing approach for tissue P systems, for which the interaction with the environment is conceptually close to the input/output behavior of interactive systems, is promising.

References

1. O. Andrei, G. Ciobanu, D. Lucanu: A Rewriting Logic Framework for Operational Semantics of Membrane Systems. *Theoretical Computer Science*, 373, 2007, 163–181.
2. F. Bernardini, M. Gheorghe, F.J. Romero-Campero, N. Walkinshaw: A Hybrid Approach to Modelling Biological Systems. *Int. Conf. on Membrane Computing*, CMC 2007, LNCS 4860, Springer, 2007, 138–159.
3. M. Gheorghe, F. Ipate, C. Dragomir: Formal Verification and Testing based on P Systems. *Int. Workshop on Membrane Computing*, WMC 2009, LNCS 5957, Springer, 2009, 54–65.
4. F. Ipate, A. Ţurcanu: Modelling, verification and testing of P systems using Rodin and ProB. *Ninth Brainstorming Week on Membrane Computing*, 2011, 209–220
5. F. Ipate, M. Gheorghe, R. Lefticaru: Test generation from P systems using model checking. *Journal of Logic and Algebraic Programming*, 79(6), 2010, 350–362
6. F. Ipate, R. Lefticaru, C. Tudose: Formal verification of P systems using Spin. *International Journal of Foundations of Computer Science*, 22(1), 2011, 133–142
7. F. Ipate, R. Lefticaru, I. Pérez-Hurtado, M.J. Pérez-Jiménez, C. Tudose: Formal Verification of P Systems with Active Membranes through Model Checking. *Int. Conference on Membrane Computing*, CMC12, 2011, 241–252.

11 Iaşi Problems

Oana Agrigoroaiei, Bogdan Aman, Gabriel Ciobanu

Institute of Computer Science, Romanian Academy, Iaşi Branch, Romania
gabriel@info.uaic.ro,oanaag@iit.tuiasi.ro

11.1 Causality

Consider standard transition P systems with promoters and inhibitors and dissolution; they can be described, up to simulating one transition step with several others, by transition P systems with just one membrane (and with promoters and inhibitors).

In [3] we have defined causality at both specific and general level for transition P systems with one membrane and without any other ingredients; specific causality depends on a certain evolution step, while general causality takes into consideration all possible evolution steps. Two questions arise immediately: whether this construction is extendible to P systems involving promoters and inhibitors and whether causality can be defined in a more static manner, without involving the membrane system. One of the results of [3] (Theorem 15) indicates that the latter problem is solvable by using the more *dynamic* notion of *general causality*.

A different problem related to causality concerns the relation between various forms of evolution in transition P systems: maximal parallelism, local maximal

parallelism and unrestricted parallelism. How do causal relations change when we change the form of evolution for a given P system? We have work in progress concerning the relationship between maximal parallelism and unrestricted parallelism which we hope will also be useful in having a clearer image of what causality means for membrane systems. Moreover, we ask how are such causal relations connected with the object-based event structures we introduced in [2], where the focus was not on rule application but on the objects being produced. As always, we have to ask in what manner do results change if additional ingredients (especially promoters and inhibitors) are introduced.

Finally, the idea of "computing backwards" [1], which was also mentioned in [8], is strongly related to the notion of causality and it would be interesting to see how it can be used to clarify or even to solve the problems proposed above.

11.2 Chemical Abstract Machine and P systems

The Chemical Abstract Machine (CHAM) [5] is suited to model asynchronous concurrent computations such as algebraic process calculi. Intuitively, the state of a system is like a chemical solution in which floating molecules can interact with each other according to reaction rules; a magical mechanism stirs the solution, allowing for possible contacts between molecules. In chemistry, this is the result of Brownian motion. The solution transformation process is obviously truly parallel: any number of reactions can be performed in parallel, provided that they involve disjoint sets of molecules.

The chemical abstract machine presents molecules in a systematic way as terms of algebras and refining the classification of rules. Some molecules do not exhibit interaction capabilities; those which are ready to interact are called ions. A solution can be heated to break complex molecules into smaller ones up to ions. Conversely, a solution can be chilled to rebuild heavy molecules from components. Furthermore, to deal with abstraction and hierarchical programming, a molecule is allowed to contain a sub-solution enclosed in a membrane, which can be somewhat porous to allow communication between the encapsulated solution and its environment. The chemical abstract machines all obey a simple set of structural laws. Each particular machine is given by adding a set of simple rules that specify how to produce new molecules from old ones. Unlike the inference rules classically used in structural operational semantics, the specific rules have no premisses and are purely local.

Since P systems and CHAM start from the same premises, but use different notations and operational semantics and have different goals, it would be interesting to study the connections between these two fields.

11.3 Type Systems

Type theory is fundamental both in logic and computer science. Theory of types was introduced by B. Russell [9] in order to solve some contradictions of set theory.

In computer science, type theory refers to the design, analysis and study of type systems. Generally, a type system is used to prevent the occurrences of errors during the evolution of a system. A type inference procedure determines the minimal requirements to accept a system or a component as well-typed.

Membrane systems consider cells as mechanisms working in a maximal parallel and non-deterministic manner. However, the living cells do not work in such a way: a chemical reaction takes place only if certain quantitative constraints are fulfilled. In order to cope with such constraints, P systems should be enriched by adding a quantitative type discipline, and making use of type inference and principal typing [10]. We associate to each reduction rule a minimal set of constraints that must be satisfied in order to assure that by the application of this rule to a well-formed P system, we get a well-formed P system as well. A first step in this direction was done in [4] where a type system for P system with symport/antiport rules is given.

The type systems can be used in defining more general and simpler rules for P systems. For example, if $\mathbf{N_1}$ and $\mathbf{N_2}$ are some basic types, by considering a set of typed objects $V = \{X_1 : \mathbf{N_1}, X_2 : \mathbf{N_1}, X_3 : \mathbf{N_1}, A : \mathbf{N_2}\}$, the evolution rules of the form $X_i \to X_j$, $X_j \to A$, $1 \leq i \leq 3$, $1 \leq j \leq 3$, can be replaced by rules of a more general form:

1. $\mathbf{N_1} \to \mathbf{N_1}$ (any object of type $\mathbf{N_1}$ can evolve in any object of type $\mathbf{N_1}$);
2. $\mathbf{N_1} \to \mathbf{N_2}$ (any object of type $\mathbf{N_1}$ can evolve in any object of type $\mathbf{N_2}$).

11.4 Behavior Equivalence

Behavior equivalence is an important concept in biology needed for analyzing and comparing the organs behavior. For example, an artificial organ is the functional equivalent of the natural organ, meaning that both behave in a similar manner up to a given time; e.g. the artificial kidney has the same functional characteristics as an "in vivo" kidney. Recently, it is shown in [7] that the vas deferens' of the human, canine, and bull are equivalent in many ways, including histological similarities. In [6] are presented different methods for comparing protein structures in order to discover common patterns.

In membrane computing, two P systems are considered to be equivalent whenever they have the same input/output behavior. Such an equivalence does not take care of the evolution of the two systems. What does mean that two P systems have equivalent (timed) behavior? Defining several equivalences, we offer flexibility in selecting the right one when verifying biological systems and comparing them. When a P system can be replaced in a context with another one such that the observed behavior is the same?

References

1. O. Agrigoroaiei, G. Ciobanu: Reversing Computation in Membrane Systems. *Journal of Logic and Algebraic Programing*, vol. 79(3-5), 278-288, 2010.

2. O. Agrigoroaiei, G. Ciobanu: Rule-based and Object-based Event Structures for Membrane Systems. *Journal of Logic and Algebraic Programing*, vol. 79(6), 295–303, 2010.
3. O. Agrigoroaiei, G. Ciobanu: Quantitative Causality in Membrane Systems. *Proceedings of the Twelfth International Conference on Membrane Computing*, 53-63, 2011.
4. B. Aman, G. Ciobanu: Typed Membrane Systems. *Lecture Notes in Computer Science*, vol. 5957, 169–181, 2010.
5. G. Berry, G. Boudol: The Chemical Abstract Machine. *Theoretical Computer Science*, vol. 96, 217–248, 1992.
6. I. Eidhammer, I. Jonassen, W. Taylor: Structure Comparison and Structure Patterns, *Journal of Computational Biology*, vol.7, 685–716, 2000.
7. D.E. Leocadio, A.R. Kunselman, T. Cooper, J.H. Barrantes, J.C. Trussell: Anatomical and Histological Equivalence of the Human, Canine, and Bull Vas Deferens, *The Canadian Journal of Urology*, vol.18, 5699–5704, 2011.
8. G. Păun: Some Open Problems Collected During 7th BWMC. *Proceedings of the Seventh Brainstorming Week on Membrane Computing*, vol. 2, 197-206, 2009.
9. B. Russell: *The Principles of Mathematics*, vol.I, Cambridge University Press, 1903.
10. J. Wells: The Essence of Principal Typings. LNCS 2380, Springer, 913–925, 2002.

12 Membrane Algorithms

Gexiang Zhang

???
???
email address???

The possible interplay between membrane computing and evolutionary computation may produce three kinds of research topics:

1. Membrane-inspired evolutionary algorithms (MIEAs). MIEA concentrates on generating new evolutionary algorithms for solving optimization problems by using the hierarchical or network structures of membranes and rules of P systems, and the concepts and principles of meta-heuristic search methodologies [1, 2]. MIEAs have been studied in conjunction with cell-like membrane systems with fixed membrane structure. Further research topics might include cell-like membrane systems with active membranes, tissue-like membrane systems and population membrane systems.
2. Automated designed of membrane computing models (ADMCMs). The automated synthesis of some types of membrane computing models or of a high level specification of them is envisaged to be obtained by applying various evolutionary algorithms. ADMCMs aim to circumvent the programmability issue of membrane based models for complex systems [3].

3. Membrane evolutionary algorithms (MEAs). MEAs will focus on implementing evolutionary algorithms within a membrane system environment in order to take advantage of the parallelism and distribution of membrane computing, given that more recent investigations in membrane computing are studying the implementation of membrane systems on parallel or multi-core hardware platforms. An important challenge for any of the above research developments will be to apply them to complex real life systems.

References

1. G. Zhang, C. Liu, M. Gheorghe, F. Ipate: Solving satisfiability problems with membrane algorithms. *Proc. Fourth International Conference on Bio-lnspired Computing: Theories and Applications*, 2009, 29–36.
2. G. Zhang, C. Liu, H. Rong: Analyzing radar emitter signals with membrane algorithms. *Mathematical and Computer Modelling*, 2010,52(11-12), 1997–2010.
3. X. Huang, G. Zhang, F. Ipate: Evolutionary design of a simple membrane system. *Proceedings of CMC, 2011.*

13 Open Problems from Verona

Vincenzo Manca
University of Verona, Department of Computer Science
Verona, Italy
vincenzo.manca@univr.it

13.1 A (precise) dynamical problem

A membrane system is a form of compartmentalized rewriting structure based on two main ingredients: multisets of objects and membranes, where objects and rules, which transform and move object multisets, are placed on. Metabolic P systems, MP systems. were introduced for modeling real biochemical systems in terms of multiset rewriting. In the last years they have been widely investigated by Verona group MNC (Models of Natural Computing). A brief introduction and references can be found in the Scholarpedia page "Metabolic P Systems".

One of the most recent result about MP systems was the discovery of a methodology for solving dynamical inverse problems in a systematic way by means of MP systems [2, 3, 4].

A time series $X^T = (X[i] \mid i \leq T \in \mathbb{N})$ is a sequence of real values intended as "equally spaced" in time ($\mathbb{N}$ is the set of natural numbers).

A MP grammar G is a generator of time series, determined by the following structure ($n, m \in \mathbb{N}$):

$$G = (M, R, I, \Phi),$$

where:

1. $M = \{X_1, X_2, \ldots, X_n\}$ is a finite set of elements called *metabolites*. A *metabolic state* is given by a list of n values, each of which is associated to a metabolite (*parameters* can possibly be added to determine a metabolic state).
2. $R = \{\alpha_j \to \beta_j \mid j = 1, \ldots, m\}$ is a set of *rules*, or *reactions*, with α_j and β_j multisets over M for $j = 1, \ldots, m$;
3. I are initial values of metabolites, that is, a list $X_1[0], X_2[0], \ldots, X_n[0]$ providing the *metabolic state at step 0*;
4. $\Phi = \{\varphi_1, \ldots, \varphi_m\}$ is a list of functions, called *regulators*, one for each rule, which, for each metabolic state, provide the *fluxes* of rules in that state.

A MP graph is a natural graphical representation of G, and G becomes a MP system when values for the *time interval*, the *population unit*, and the *metabolite masses* are added. G generates the (infinite) time series $(X[i] \mid i \in \mathbb{N})$ for $X \in \{X_1, X_2, \ldots, X_n\}$, according to the following *Equational Metabolic Algorithm* (EMA), where $\gamma(X)$ denotes the multiplicity of X in the multiset γ and $s[i]$ is the metabolic state at step i:

$$X[i+1] - X[i] = \sum_{j=1,m} (\alpha(X) - \beta(X))\varphi_j(s[i]).$$

DIP formulation for MP: Given n time series

$$Y_1^T, Y_2^T, \ldots, Y_n^T,$$

corresponding to some "observed" variables, related by transformation/influence relations among them, find a MP grammar G, expressing the known relations among variables, and generating, for $i \leq T$, exactly, or even approximately enough, the time series $Y_1^T, Y_2^T, \ldots, Y_n^T$.

Many DIP problems of interest for biological/pathological phenomena were solved in terms of MP systems. The solutions obtained resulted from suitable combinations of several ingredients: i) a linear algebra formulation of EMA (to which a *stoichiometric expansion* can be applied), ii) the Least Square Evaluation method, iii) the Stepwise Linear Regression method, and iv) some suitable statistical tests based on Fischer's distributions.

A possible field of investigation could concern other classes of DIP problems, in such a way that other kinds of DIP solutions could be found, for these problems, by suitable discrete dynamical systems based on membranes.

13.2 A (vague) topological problem

Membrane computing is based on the intuition of a membrane as a spatial entity closing a subspace within an environment (inside/frontier/outside). Cells are the

most evident biological realization of this notion. However, if we want to keep this intuition close to the biological reality, the only inclusion relation of membrane containment is too weak. In fact, in the membrane computing literature, some extensions of the original notion of membrane were proposed in terms of tissue-like and neuron-like membrane structures. Even these enrichments are not expressive enough for dealing with aspects were membranes are not framed in a fixed membrane structure hosting computations, but they are subjected to topological transformations exploring and determining forms. This perspective requires a calculus on membranes rather than calculi within, or among, membranes. Some ideas along this line of investigation arose in a research [1] devoted to *multimembranes*, for translating, in a pure membrane setting, computations which can be easily expressed by MP grammars.

A possible field of investigation could concern the formulation of topological (in wide sense) operations among membranes on which calculi can be defined which resemble what happen at the level of morphogenesis in biological systems.

References

1. V. Manca, R. Lombardo: Computing with multi-membranes. In *Proc. CMC 2011*, 347–364.
2. V. Manca, L. Marchetti: Metabolic approximation of real periodical functions. *J. Logic and Algebraic Programming*, 79 (2010), 363–373.
3. V. Manca, L. Marchetti: Log-gain stoichiometricstepwise regression for MP systems. *Int. J. Foundations of Computer Science*, 22 (2011), 97–106.
4. V. Manca, L. Marchetti: Paper in progress, 2011.

14 Unraveling Oscillating Structures by Means of P Systems

Thomas Hinze[1,2]

[1]Friedrich Schiller University
Department of Bioinformatics at School of Biology and Pharmacy
Jena, Germany

[2]Saxon University of Cooperative Education
Dresden, Germany
thomas.hinze@uni-jena.de

14.1 Motivation

Endogenous oscillations have been identified to be essential for the function of numerous systems found in biology as well as engineering [1, 2, 26]. A common property of these systems lies in their necessity to synchronize and coordinate inherent chemical or physical activities based on periodically iterated trigger signals [12]. In order to sustain a stable oscillatory signal behavior, a cyclic process succession is required that is characterized by at least one positive or negative feedback loop. Here, a feedback amplifies or downregulates a process chain at its outset by means of signals obtained at its end [21]. This delayed signal evaluation enables a concerted alternation between effects caused by the process chain and counteractions initiated by the feedback. External stimuli or stochasticity might affect signal oscillations resulting from a process cycle. In complex systems, coupled oscillators can exhibit a pseudo-chaotic behavior which is hard to analyze and control [18]. Probably, numerous evolutionary origins led to oscillatory reaction systems, while independently technical attempts succeeded in construction of single clocks or clock generators [4].

Exploration of chronobiological systems emerges as a growing research field within bioinformatics focusing on various applications in medicine, agriculture, and material sciences [5]. From a systems biology perspective, the question arises whether biological control systems for regulation of oscillatory signals and their technical counterparts utilize similar mechanisms [15]. If so, modeling approaches and parameterization adopted from a strict modularization can help to identify general components for frequency control especially in *circadian clock systems* along with gaining comprehensive insight into mechanisms of clock maintenance, synchronization, and entrainment to external stimuli like the daily rhythm of sunlight and darkness.

Circadian rhythms embody an interesting biological phenomenon that can be seen as a widespread property of life. The coordination of biological activities into daily cycles provides an important advantage for the fitness of diverse organisms

[27]. Based on self-sustained biochemical oscillations, circadian clocks are characterized by a natural period close to but not exactly of 24h that persists under constant conditions (like constant darkness or constant light). Their ability for compensation of temperature variations in the physiological range enables them to maintain the period in case of environmental changes. Furthermore, circadian clocks can be entrained. This property allows a gradual reset of the underlying oscillatory system for adjustment by exposure to external stimuli like daily variations of brightness or daytime-nighttime temperature cycles.

Circadian clock systems appear to be special forms of frequency control systems [16]. Following this line, it should be possible to identify appropriate interacting modules representing elements of a dedicated control-loop model [3]. Indeed, coupling of a controllable core oscillator with a low-pass filter and a multiplicator suffices to reproduce the desired entrainment behavior of a circadian clock [17].

There are numerous types of controllable core oscillators found in circadian clocks. The majority reveals the Goodwin type [8], a cyclic gene regulatory network composed of mutual activating and inhibiting gene expressions. The most effective way to influence its frequency is modification of protein degradation rates [25]. Furthermore, core oscillators can be of post-translational type [23], exploiting a cyclic scheme of protein phosphorylation, complex formation, or decomposition. Here, the involved catalysts affect the frequency [14]. The third and most complex type of core oscillators includes *compartmental dynamics* [22] aimed to be advantageously modeled using P systems combining a representation of dynamical structures with tracing their spatiotemporal behavior.

14.2 Resulting Challenges

Within the domain of strictly continuous signals quantified by real numbers, modeling and analysis of oscillating behavior has been well-studied [24]. Chemical reaction networks assumed to reside in a homogeneous environment give a typical example: Each species is represented by its concentration which is allowed to vary continuously over time. From the static network topology together with the stoichiometry of the reactions, a corresponding ordinary differential equation system (ODE) can be derived that specifies the reaction rates for each species [6]. Inclusion of parameterized kinetic laws accomplishes a mapping between species concentration and effective reaction rate. The resulting ODE can easily be tested for its capability of undamped oscillating species concentrations. To this end, the *eigenvalues* of the *Jacobian matrix* obtained from the ODE right hand side are sufficient [10]. *Limit cycles* indicate the oscillatory behavior in detail. In case of sinusoidal or almost sinusoidal oscillatory waveforms, even properties of the entrainment behavior can be obtained analytically, for instance by the Kuramoto method [19] which estimates the expected time to synchronization subject to the range of relevant parameters.

The main advantage of analytical ODE-based methods unequivocally exploits the fact that essential characteristics and properties of a system under study can

be directly derived from the underlying mathematical model without any need for a numerical simulation of its dynamical behavior. This makes the evaluation and automated testing of candidate systems resulting from experimental data rather efficient. In contrast, there is currently a lack of corresponding methods within the field of P systems modeled in a discrete manner. Here, system properties mainly emerge from exhaustive simulation studies. Conduction of those studies still requires an extensive amount of human resources. Particularly in case of involved active membranes, compartmental plasticity, and dynamical structures, a toolbox for automated analysis would be helpful. In an ongoing project, we intend to generate sustained oscillatory systems by artificial evolution *in silico* [20]. In this context, the fitness evaluation should answer the question whether the system candidates are able to oscillate endogenously or not and how the periodicity can be controlled. Ideally, this task should be done by a piece of software [9]. Questions concerning a toolbox for systems analysis also coincide with the need to identify appropriate evolutionary operators affecting compartmental structures on the fly. Those operators can be inspired by biological processes found in living cells like division, degeneration, dissolution, creation, separation, merge, endocytosis, exocytosis, or gemmation. Some of these operators can be found in recent frameworks of P systems, but others still lack a detailed specification of its effects to sets of molecular objects and local reaction and transportation rules (configuration update schemes).

14.3 First Ideas

There are different oscillatory scenarios in biological systems. On the one hand, periodicity might also be reflected in temporal changes of the compartmental structure. On the other hand, signalling molecules are often available in low concentrations ranging from single molecules to several thousand copies. Both aforementioned scenarios have in common to prevent pure ODE-based modeling techniques due to the discrete manner of involved key entities. Motivated by the need for an appropriate toolbox covering description, simulation, and analysis of discontinuously considered biological reaction processes, we plan to extend the concept of spatiotemporal P systems with kinetic laws [7, 11] towards an underlying *backtracking mechanism* able to explore the nature of undamped oscillations beyond variations of species concentration. Following the idea of backtracking, the trace of configurations passed by a P system becomes recorded in a suitable way [13]. By monitoring the overall configurations over time, a derivation tree is obtained that provides a comprehensive data pool for further analysis by automated backtracking. Sustained oscillations are expected to appear as recurring, but nonadjacent overall configurations along a path through the derivation tree. In particular, we wish to employ this technique for identification and description of biochemically inspired computational devices equipped with clocks, counters, or frequency scalers. Moreover, we aim for gaining insight into the function of dedicated circadian clocks by reverse engineering using backtracking P systems. This approach could benefit from the flexibility regarding structural dynamics.

References

1. U. Alon. *An Introduction to Systems Biology: Design Principles of Biological Circuits.* Chapman & Hall, 2006
2. J. Aschoff. A survey on biological rhythms. *Biological Rhythms* **4**:3-10, 1981
3. B.W. Bequette. *Process control: modeling, design, and simulation.* Prentice Hall, 2003
4. R.E. Best. *Phase-locked loops: design, simulation, and applications.* McGraw-Hill, 2007
5. B. Botti, C. Youan (Eds.). *Chronopharmaceutics. Science and technology for biological rhythm guided therapy and prevention of diseases.* John Wiley & Sons, 2009
6. K.A. Connors. *Chemical Kinetics.* VCH Publishers, Weinheim, 1990
7. F. Fontana, V. Manca. Discrete solutions to differential equations by metabolic P systems. *Theoretical Computer Science* **372(2-3)**:165-182, 2007
8. B.C. Goodwin. Oscillatory behaviour in enzymatic control processes. *Advanced Enzyme Regulation* **3**:425-438, 1965
9. G. Gruenert, B. Ibrahim, T. Lenser, M. Lohel, T. Hinze, P. Dittrich. Rule-based spatial modeling with diffusing, geometrically constrained molecules. *BMC Bioinformatics* **11**:307, 2010
10. B.A. Hawkins, H.V. Cornell (Eds.). *Theoretical Approaches to Biological Control.* Cambridge University Press, 1999
11. T. Hinze, T. Lenser, P. Dittrich. A Protein Substructure Based P System for Description and Analysis of Cell Signalling Networks. In H.J. Hoogeboom et al. (Eds.). LNCS 4361: 409-423, Springer Verlag, 2006
12. T. Hinze, R. Fassler, T. Lenser, P. Dittrich. Register Machine Computations on Binary Numbers by Oscillating and Catalytic Chemical Reactions Modelled using Mass-Action Kinetics. *International Journal of Foundations of Computer Science* **20(3)**:411-426, 2009
13. T. Hinze, R. Fassler, T. Lenser, N. Matsumaru, P. Dittrich. Event-Driven Metamorphoses of P Systems. In D. Corne et al. (Eds.). LNCS 5391: 231-245, Springer Verlag, 2009
14. T. Hinze, T. Lenser, G. Escuela, I. Heiland, S. Schuster. Modelling Signalling Networks with Incomplete Information about Protein Activation States: A P System Framework of the KaiABC Oscillator. In G. Păun et al. (Eds.). LNCS 5957 :316-334, Springer Verlag, 2010
15. T. Hinze, M. Schumann, C. Bodenstein, I. Heiland, S. Schuster. Biochemical Frequency Control by Synchronisation of Coupled Repressilators: An *In-silico* Study of Modules for Circadian Clock Systems. *Computational Intelligence and Neuroscience* **2011**:e262189, 2011
16. T. Hinze, C. Bodenstein, I. Heiland, S. Schuster. Capturing Biological Frequency Control of Circadian Clocks by Reaction System Modularization. *ERCIM News* **85**:27-29, 2011
17. T. Hinze, C. Bodenstein, B. Schau, I. Heiland, S. Schuster. Chemical Analog Computers for Clock Frequency Control Based on P Modules. In M. Gheorghe et al. (Eds.). Proceedings CMC12, LNCS, Springer Verlag, 2011, to appear
18. T. Kapitaniak (Ed.). *Chaotic Oscillators: Theory and Applications.* World Scientific Pub. Co., 1992
19. Y. Kuramoto. *Chemical Oscillations, Waves, and Turbulences.* Springer, 1984

20. T. Lenser, T. Hinze, B. Ibrahim, P. Dittrich. Towards Evolutionary Network Reconstruction Tools for Systems Biology. In E. Marchiori et al. (Eds.). LNCS 4447: 132-142, Springer Verlag, 2007
21. V. Manca, L. Bianco, F. Fontana. Evolution and oscillation in P systems: Applications to biological phenomena. In G. Mauri et al. (Eds.). *Lecture Notes in Computer Science* **3365**:63-84, Springer Verlag, 2005
22. D. Morgan. *The Cell Cycle: Principles of Control.* Oxford University Press, 2006
23. T. Mori, D.R. Williams, M.O. Byrne, X. Qin, M. Egli, H.S. Mchaourab, P.L. Stewart, C.H. Johnson. Elucidating the ticking of an in vitro circadian clockwork. *PLoS Biology* **5(4)**:841-853, 2007
24. T. Pavlidis. *Biological Oscillators: Their Mathematical Analysis.* Academic Press, 1974
25. P. Ruoff, M. Vinsjevik, C. Monnerjahn, L. Rensing. The Goodwin Oscillator: On the Importance of Degradation Reactions in the Circadian Clock. *Journal of Biological Rhythms* **14(6)**:469-479, 1999
26. S. Schuster, M. Marhl, T. Hoefer. Modelling of simple and complex calcium oscillations. *European Journal of Biochemistry* **269**:1333-1355, 2002
27. V.K. Sharma, A. Joshi. Clocks, genes, and evolution. The evolution of circadian organization. In V. Kumar (Ed.). *Biological Rhythms*, p. 5-23, Springer Verlag, 2002

15 Approaching a Question of Biologically Plausible Applications of Spiking Neural P Systems for an Explanation of Brain Cognitive Functions

Adam Obtulowicz

Institute of Mathematics, Polish Academy of Sciences
Śniadeckich 8, P.O.B. 21, 00-956 Warsaw, Poland
A.Obtulowicz@impan.pl

The (hierarchical) clustering (scene segmentation in particular) and binding (feature integration) problem solution in cortical neural network together with cortical subnetworks realizing Radial Basic Functions (briefly RBFs) represent, among others, the cognitive functioning of brain. Recently, various network models of clustering, binding problem solution, and realization of RBFs in cortical network have been proposed, where spiking neural networks are the most biologically plausible models, see [16], [18], [2], [3], [12], [14], [15], and [11] for a review. The main common feature of these models is Hebbian learning which provides their biological evidence. On the other hand, a transformation of an idea of Hebbian learning from a framework of spiking neural networks to a framework of spiking neural P systems (cf. [10], [17]) has been proposed in [8]. Thus one formulates the following question:

Do spiking neural P systems provide biologically plausible mathematical models of brain cognitive functions?

We approach the question and an answer to it by the following discussion of conjectures and setting open problems.

Papers [5], [9] contain promising applications of spiking neural P systems for solving topic problems related to some cognitive brain functions. But biological evidence of these applications seems problematic because Hebbian learning procedures approach is not considered for them.

On the other hand, the Hebbian learning modeled by spiking neural P systems with only input neurons and one output neuron presented in [8] and solution of XOR problem by spiking neural networks equipped with a Hebbian learning procedure and with only three input neurons and one output neuron described in [4] gives rise to the following conjecture:

Conjecture 1. *There exists a learning problem, understood as in* [8]*, whose output is a spiking neural P system solving XOR problem.*

If we compare precise timing of spikes approach for spiking neural networks to the number of spikes approach for spiking neural P systems, then the latter seems coarse and hence less biologically plausible than the spiking neural network approach.

On the other hand the precise timing of spikes approach for spiking neural networks is less biologically plausible than probabilistic spiking neural networks because a relevant amount of noise is contained in the behavior of neurons (cf. [7]). Therefore it is worth to initiate a research of probabilistic spiking neural P systems.

The view that human mind is "massively modular" (cf. [6], [13]) argued by massively parallel functioning of brain neural network modules gives rise to a question of approaching these massive modularity and massive parallelism of mind and brain by application of a concept of a network of communicating spiking neural P systems equipped with Hebbian learning procedures, respectively. The spiking neural P systems constituting that network could correspond to brain network modules realizing simultaneously various cognitive functions, respectively.

On the other hand, since spiking neural P systems seem more coarse with respect to an approach to time than spiking neural networks with precise timing of spikes, like e.g. in [2], we propose the following conjecture.

Conjecture 2. *A biologically plausible modularity of brain could be represented (modelled) by the following hybrid constructs*:

1. *a two-level construct of a* spiking super-neural P system *which is a spiking neural P system whose neurons are superneurons, i.e., multi-layer spiking neural networks with a precise timing of spikes like, e.g., in* [2],
2. *a three-level construct of a* spiking sub-super-neural P system *which is a spiking super-neural P system as above, where the neurons of superneurons are P systems approaching neurons as cells which produce and transport copies of molecules between electrically charged membranes.*

The construct in 1) gives rise to multi-layer spiking networks which could learn themselves like in [2] their modular structure of spiking super-neural P systems and hence which could explain emergence of cognitive capabilities of brain.

It is worth to discuss the above constructs with a regard to a possibility of their molecular implementation which is suggested by recent findings outlined in [1].

References

1. Bandyopadhyay, A., Fujita, D., Pati, R., *Architecture of a Massively Parallel Processing Nano-Brain Operating 100 Billion Molecular Neurons Simultaneously*, International Journal of Nanotechnology and Molecular Computation 1 (2009), pp. 50–80.
2. Bohte, S.M., *Spiking Neural Networks*, Professorschrift, Leiden University 2003.
3. Booij, O., *Temporal Pattern Classification using Spiking Neural Networks*, M.Sc. Thesis, Amsterdam University 2004.
4. Booij, O., Hieu tat Nguyen, *A gradient descent rule for spiking neurons emitting multiple spikes*, in: Applications of Spiking Neural Networks, ed. S.M. Bohte and J.N. Kok, Information Processing Letters, Amsterdam 2005.
5. Ceterchi, R., Tomescu, I.A., *Spiking Neural P systems—a Natural Model for Sorting Networks*, in: Proceedings of Sixth Brainstorming Week on Membrane Computing, Sevilla, February 4–8, 2008, ed. D. Diaz-Perenil et al., RGNC Report 01/2008, Sevilla University Fenix Editora 2008, pp. 93–105.
6. Geary, D., *The Origin of Mind: Evolution of Brain, Cognition, and General Intelligence*, American Psychological Association 2005.
7. Gerstner, W., *Population Dynamics of Spiking Neurons: Fast Transients, Asynchronous States, and Locking*, Neural Computation 12 (2000), pp. 43–89.
8. Gutiérez-Naranjo, M.A., Pérez-Jiménez, M.J., *A spiking neural P systems based model for Hebbian learning*, in: Proceedings of 9th Workshop on Membrane Computing, Edinburgh, July 28 – July 31, 2008, ed. P. Frisco et al., Technical Report HW-MASC-TR-0061, School of Mathematical and Computer Sciences, Heriot–Watt University, Edinburgh, UK, 2008, pp. 189–207.
9. Ionescu, M., Sburlan, D., *Some Applications of Spiking Neural P Systems*, in: Proceedings of the 8th Workshop on Membrane Computing, Thessaloniki, June 25–28, 2007, ed. Eleftherakis et al., South-East European Research Centre 2007, pp. 383–394.
10. Ionescu, M., Păun, Gh., Yokomori, Y., *Spiking neural P systems*, Fund. Inform. 71 (2006), pp. 279–308.
11. Kasiński, A., Ponulak, F., *Comparison of Supervised Learning Methods for Spike Time Coding in Spiking Neural Networks*, Int. J. Appl. Math. Comput. Sci. 16 (2006), pp. 101–113.
12. Knoblauch, A., Palm. G., *Scene segmentation by spike synchronization in reciprocally connected visual areas.* II. *Global assemblies and synchronization on larger space and time scales*, Biol. Cybern. 87 (2002), pp. 168–184.
13. MacDonald, K., Chiappe, D., Review of [6] in Human Ethology Bulletin 21:2 (2006), pp. 14–18.
14. Meftah, B., Benyettou, A., Lezoray, O., Qingxiang, W., *Image Clustering with Spiking Neuron Network*, in: World Congress on Computational Intelligence, International Joint Conference on Neural Networks, Hong-Kong 2008.

15. Moore, S.C., *Back-propagation in spiking neural networks*, M.Sc. Thesis, University of Bath 2002, http://www.simonchristianmoore.co.uk/Thesis4.html.
16. Natschläger, T., Ruf, B., *Spatial and temporal pattern analysis via spiking neurons*, Network: Comp. Neural Systems 9 (1998), pp. 319–332.
17. Păun, Gh., Pérez-Jiménez, M.J., *Spiking neural P systems. Recent results, research topics*, presented at the 6th Brainstorming Week on Membrane Computing, Sevilla 2008, web page http://psystems.disco.unimib.it/download/leidenGR65.pdf
18. Ruf, B., *Computing and Learning with Spiking Neurons—Theory and Simulation*, Doctoral Thesis, Technische Universität Graz 1998.

16 Computer Vision

Daniel Díaz-Pernil[1], Miguel A. Gutiérrez-Naranjo[2]

[1]CATAM Research Group
Department of Applied Mathematics I
University of Seville, Spain
sbdani@us.es

[2]Research Group on Natural Computing
Department of Computer Science and AI
University of Seville, Spain
magutier@us.es

Computer vision [26] is probably one of the challenges for computer scientists in the next years. This flourishing research area needs contributions from many other scientific areas as artificial intelligence, pattern recognition, signal processing, neurobiology, psychology or image processing among others. It concerns with the automated processing of images from the real world to extract and interpret information on a real time basis.

From a biological point of view, vision is an extremely complex process involving the transformation of the light energy into a signal which leaves the eye by way of the optic nerve and arrives to the brain, where it is interpreted. From a computational point of view, a digital image is a function from a two dimensional surface which maps each point form the surface to a set of features as bright or color.

Since many of such features are quantitative (as the color in an RGB encoding, which is a three-dimensional vector with values in $\{0, \ldots, 255\}$), in most cases, a digital image can roughly be considered as a mapping from a subset of $\mathbb{Z} \times \mathbb{Z}$ (a subset of the integer plane) into a set of multidimensional vectors which encode their features. Let us remark that the subset of the integer plane and the set of vectors used for encoding the features of the *pixels* can be considered finite and

hence, the transformation of an image into another can be made in a *discrete* way. The different treatments of such mappings (digital images) provide a big amount of current applications in computer vision as optical character recognition (OCR), fingerprint recognition and biometrics, automotive safety, surveillance or medical imaging.

Many problems in the processing of digital images have features which make it suitable for techniques inspired by nature. One of them is that the treatment of the image can be parallelized and locally solved. Regardless how large is the picture, the process can be performed in parallel in different local areas of it. Another interesting feature is that the local information needed for a pixel transformation can also be easily encoded in the data structures used in Natural Computing.

In the literature, we can find many examples of the use of natural computing techniques for dealing with problems associated to the treatment of digital images. One of the classic examples is the use of cellular automata [24, 25]. Other efforts are related to artificial neural networks as in [14, 28].

In membrane computing, there is a large tradition in the study of information structured as two dimensional objects (see, e.g., [2, 3, 10, 19]). The main motivation for these studies is to bring together membrane computing and picture grammars. From a technical point of view, arrays are two-dimensional objects placed inside the membranes as strings are one-dimensional objects in the model of P systems with string objects [15, 23].

In [3], the model of array-rewriting P systems was presented on the basis of the transition P systems: Rules are of type $\mathcal{A} \to \mathcal{B}(tar)$ where $\mathcal{A}$ is the array to be rewritten and $\mathcal{B}$ is the new one and $tar \in \{here, in, out\}$ indicates the emplacement of the picture where the substitution has been made.

For example[4], let us consider a P system with three nested membranes $[\,[\,[\,]_3\,]_2\,]_1$, an alphabet with two symbols a and $\#$ (the blank), an initial configuration with membranes 2 and 3 empty and the array $\begin{matrix} a \\ a \end{matrix}$ placed in the membrane 1. Let us consider the sets of rules

$$R_1 = \left\{ \begin{matrix} & \# \\ \# & a \end{matrix} \to \begin{matrix} & a \\ \# & a \end{matrix} (in) \right\},$$

$$R_2 = \left\{ \begin{matrix} a & \# \\ \# & \end{matrix} \to \begin{matrix} a & a \\ \# & \end{matrix} (out), \begin{matrix} a & \# & \# \\ \# & & \end{matrix} \to \begin{matrix} a & a & a \\ \# & & \end{matrix} (in) \right\},$$

$$R_3 = \emptyset.$$

This P system generates all the L-shaped angles with equal arms, each arm being of length at least three.

Recently, a new research line has been open by applying well-known membrane computing techniques for solving problems from digital imagery. For example, a basic problem in computer vision is the *segmentation*.

[4] Adapted from the Example 1 from [3].

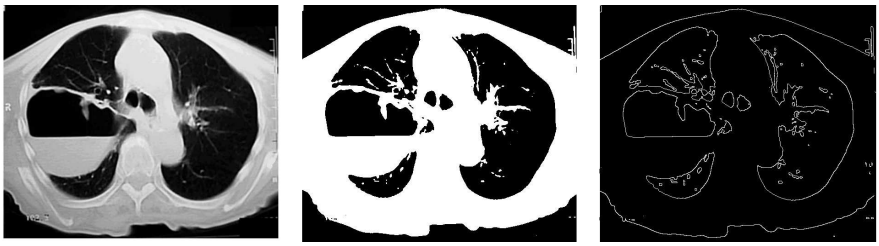

Fig. 1. Segmentation result of a 600 × 600 CT image of human lungs. On the left, the initial image, on the middle the binarized image, and on the right the segmentation result. Image taken from [12].

Segmentation is the process of assigning a label to every pixel in an image such that pixels with the same label share certain visual characteristics. The goal of segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyze. Segmentation has shown its utility in bordering tumors and other pathologies, computer-guided surgery or the study of anatomical structure, but also in techniques which are not thought to produce images but it produces positional information as electroencephalography (EEG), or electrocardiography (EKG). In [6, 9, 11, 12] we can find several approaches to this problem with membrane computing techniques. Figure 1 shows an example of segmentation result of human lungs taken from [12]. Other problems as *thresholding* [5] or *smoothing* [21] has also been considered in the framework of membrane computing.

Special attention deserves [17], where the *symmetric dynamic programming stereo* (SDPS) algorithm [18] for stereo matching was implemented by using simple P modules with duplex channels.

A different approach to computer vision can also be obtained from computational topology. In particular, algebraic topology [16] provides techniques and algorithms for dealing digital images from a topological point of view. In the literature, one can find approaches to algebraic topology by using techniques from natural computing, as in [4] where natural computing and algebraic topology are linked by using neural networks (extended Kohonen mapping).

Recently, the links between algebraic topology and membrane computing have started to be explored via *homology theory* [7, 8, 13]. *Homology theory* is a branch of algebraic topology that attempts to distinguish between spaces by constructing algebraic invariants that reflect the connectivity properties of the space. The field has its origins in the work of the French mathematician, theoretical physicist, and philosopher of science Jules Henri Poincaré. Homology groups (related to the different n-dimensional holes, connected components, tunnels, cavities, etc., of a geometric object) are invariants from algebraic topology which are frequently used in digital image analysis and structural pattern recognition. In some sense, they reflect the topological nature of the object in terms of the number and characteristics of its holes.

In a way similar to other applications of P systems, the theoretical advantages of the membrane computing techniques for computer vision need a powerful software and hardware for an effective implementation. The development of a simulators adapted to novel technologies deserves a special section in a paper devoted to current frontiers in membrane computing. We only comment here that the most recent hardware architectures as *Field Programmable Gate Arrays* (FPGAs) [27] or the *Compute Unified Device Architecture* CUDA^{TM} [29] provide the technological support for an effective parallel implementation of the algorithms. They capabilities for the parallel implementation of membrane computing techniques applied to computer vision have started to be explored with promising experimental results [1, 20, 21].

An appropriate combination of membrane computing techniques together with an efficient parallel implementation on the new hardware architectures can provide competitive algorithms to different problems from computer vision. Among them, we can cite dealing with textures, colors and/or 3D objects (or even 4D objects where the evolution of objects in *time* is also considered). From algebraic topology, the calculus of complex topological invariants of 2D and 3D objects can be a source of new open problems for membrane computing.

Acknowledgements

DDP and MAGN acknowledge the support of the projects TIN2008-04487-E and TIN-2009-13192 of the Ministerio de Ciencia e Innovación of Spain and the support of the Project of Excellence with *Investigador de Reconocida Valía* of the Junta de Andalucía, grant P08-TIC-04200.

References

1. J. Carnero, D. Díaz-Pernil, M.A. Gutiérrez-Naranjo: Designing tissue-like P systems for image segmentation on parallel architectures. In M.A. Martínez del Amor, Gh. Păun, I. Pérez-Hurtado de Mendoza, F.J. Romero-Campero, L.V. Cabrera, eds., *Ninth Brainstorming Week on Membrane Computing*, pages 43–62, Sevilla, Spain, 2011. Fénix Editora.
2. R. Ceterchi, R. Gramatovici, N. Jonoska, K.G. Subramanian: Tissue-like P systems with active membranes for picture generation. *Fundamenta Informaticae*, 56(4):311–328, 2003.
3. R. Ceterchi, M. Mutyam, Gh. Păun, K.G. Subramanian: Array-rewriting P systems. *Natural Computing*, 2(3):229–249, 2003.
4. J. Chao, J. Nakayama: Cubical singular simplex model for 3D objects and fast computation of homology groups. In *13th International Conference on Pattern Recognition (ICPR'96)*, volume IV, pages 190–194, Los Alamitos, CA, USA, 1996. IEEE Computer Society, IEEE Computer Society.
5. H.A. Christinal, D. Díaz-Pernil, M.A. Gutiérrez-Naranjo, M.J. Pérez-Jiménez: Thresholding of 2D images with cell-like P systems. *Romanian Journal of Information Science and Technology (ROMJIST)*, 13(2):131–140, 2010.

6. H.A. Christinal, D. Díaz-Pernil, P. Real: Segmentation in 2D and 3D image using tissue-like P system. In E. Bayro-Corrochano, J.-O. Eklundh, eds., *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications 14th Iberoamerican Conference on Pattern Recognition, CIARP 2009, Guadalajara, Jalisco, Mexico, November 15-18, 2009. Proceedings*, LNCS 5856, pages 169–176, Springer, Berlin, 2009.
7. H.A. Christinal, D. Díaz-Pernil, P. Real: Using membrane computing for obtaining homology groups of binary 2D digital images. In P. Wiederhold, R.P. Barneva, eds., *Combinatorial Image Analysis 13th International Workshop, IWCIA 2009, Playa del Carmen, Mexico, November 24-27, 2009. Proceedings*, LNCS 5852, pages 383–396, Berlin, 2009. Springer.
8. H.A. Christinal, D. Díaz-Pernil, P. Real: P systems and computational algebraic topology. *Journal of Mathematical and Computer Modelling*, 52(11-12):1982 – 1996, December 2010. The BIC-TA 2009 Special Issue, International Conference on Bio-Inspired Computing: Theory and Applications.
9. H.A. Christinal, D. Díaz-Pernil, P. Real: Region-based segmentation of 2D and 3D images with tissue-like P systems. *Pattern Recognition Letters*, 32(16):2206 – 2212, 2011. Advances in Theory and Applications of Pattern Recognition, Image Processing and Computer Vision.
10. K.S. Dersanambika, K. Krithivasan: Contextual array P systems. *International Journal of Computer Mathematics*, 81(8):955–969, 2004.
11. D. Díaz-Pernil, M.A. Gutiérrez-Naranjo, H. Molina-Abril, P. Real: A bio-inspired software for segmenting digital images. In A.K. Nagar, R. Thamburaj, K. Li, Zhuo Tang, R. Li, eds., *Proceedings of the 2010 IEEE Fifth International Conference on Bio-Inspired Computing: Theories and Applications BIC-TA*, volume 2, pages 1377 – 1381, Beijing, China, 2010. IEEE Computer Society.
12. D. Díaz-Pernil, M.A. Gutiérrez-Naranjo, H. Molina-Abril, P. Real: Designing a new software tool for digital imagery based on P systems. *Natural Computing*, pages 1–6, 2011. 10.1007/s11047-011-9287-4.
13. D. Díaz-Pernil, M.A. Gutiérrez-Naranjo, P. Real, V. Sánchez-Canales: Computing homology groups in binary 2D imagery by tissue-like P systems. *Romanian Journal of Information Science and Technology (ROMJIST)*, 13(2):141–152, 2010.
14. M. Egmont-Petersen, D. de Ridder, H. Handels: Image processing with neural networks - a review. *Pattern Recognition*, 35(10):2279–2301, 2002.
15. C. Ferretti, G. Mauri, C. Zandron: P systems with string objects. In Gh. Păun, G. Rozenberg, A. Salomaa, eds., *The Oxford Handbook of Membrane Computing*, pages 168 – 197. Oxford University Press, Oxford, England, 2010.
16. D. Freedman, C. Chen: Algebraic topology for computer vision. *Science And Technology*, 2009.
17. G. Gimel'farb, R. Nicolescu, S. Ragavan: P systems in stereo matching. In P. Real, D. Diaz-Pernil, H. Molina-Abril, A. Berciano, W. Kropatsch, eds., *Computer Analysis of Images and Patterns*, LNCS 6855, pages 285–292. Springer Berlin, 2011.
18. G.L. Gimel'farb: Probabilistic regularisation and symmetry in binocular dynamic programming stereo. *Pattern Recognition Letters*, 23(4):431–442, 2002.
19. S.N. Krishna, R. Rama, K. Krithivasan: P systems with picture objects. *Acta Cybernetica*, 15(1):53–74, 2001.
20. F. Peña-Cantillana, D. Díaz-Pernil, A. Berciano, M.A. Gutiérrez-Naranjo: A parallel implementation of the thresholding problem by using tissue-like P systems. In P.

Real, D. Díaz-Pernil, H. Molina-Abril, A. Berciano, W.G. Kropatsch, eds., *CAIP (2)*, LNCS 6855, pages 277–284. Springer, 2011.
21. F. Peña-Cantillana, D. Díaz-Pernil, H.A. Christinal, M.A. Gutiérrez-Naranjo: Implementation on CUDA of the smoothing problem with tissue-like P systems. *International Journal of Natural Computing Research*, 2(3):25–34, 2011.
22. Gh. Păun: Computing with membranes. Technical Report 208, Turku Centre for Computer Science, Turku, Finland, November 1998.
23. Gh. Păun: Computing with membranes. *Journal of Computer and System Sciences*, 61(1):108–143, 2000. See also [22].
24. P.L. Rosin: Training cellular automata for image processing. *IEEE Transactions on Image Processing*, 15(7):2076–2087, 2006.
25. P.J. Selvapeter, W. Hordijk: Cellular automata for image noise filtering. In *NaBIC*, pages 193–197. IEEE, 2009.
26. L.G. Shapiro, G.C. Stockman: *Computer Vision*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2001.
27. S.M. Trimberger: *Field-Programmable Gate Array Technology*. Kluwer Academic Publishers, Norwell, MA, USA, 1994.
28. Y.T. Zhou, R. Chellappa: *Artificial neural networks for computer vision*. Research notes in neural computing. Springer-Verlag, 1992.
29. NVIDIA Corporation. NVIDIA CUDA™ Programming Guide. `http://www.nvidia.com/object/cuda_home_new.html`.

17 Bridging P and R

Gheorghe Păun

Institute of Mathematics of the Romanian Academy
Bucharest, Romania, and

Department of Computer Science and Artificial Intelligence
University of Sevilla, Spain
`gpaun@us.es`

17.1 The Framework

We end this list of research topics with the issue of bridging MC with another recent branch of natural computing inspired from the biochemistry of a living cell, the *reaction systems* – see [1], [2], [3], [4], [5]. This has been already the subject of [10] and [11], but many other issues remain to be clarified.

Both areas deals with populations of reactants (molecules) which evolve by means of reactions, with several basic differences. Most of these differences are not mentioned here (e.g., the compartmental structure of models in MC versus

the missing of membranes in reaction systems – we also call them *R systems* –, the focus on evolution, not on computation, in reaction systems, the unique form of rules in reaction systems and so on), and we recall the two basic ones in the formulation from [1]:

The way that we define the result of a set of reactions on a set of elements formalizes the following two assumptions that we made about the chemistry of a cell:

(i) We assume that we have the "threshold" supply of elements (molecules) – either an element is present and then we have "enough" of it, or an element is not present. Therefore we deal with a qualitative rather than quantitative (e.g., multisets) calculus.

(ii) We do not have the "permanence" feature in our model: if nothing happens to an element, then it remains/survives (status quo approach). On the contrary, in our model, an element remains/survives only if there is a reaction sustaining it.

Passing from multisets, which are basic in P systems, to sets (actually, to multisets with an infinite multiplicity of their elements) is a fundamental assumption, which changes completely the approach; for instance, we can no longer define computations with the result expressed in terms of counting molecules: the total set of molecules is finite, any molecule is either absent or present in infinitely many copies. Moreover, the behavior of a reaction system is deterministic, from a set of symbols we precisely pass to a unique set of symbols (hence the behavior of a reaction system can be described by a graph of outdegree one, having the nodes marked with subsets of the total set of molecules). How to bridge at this level the two research areas (defining computations in reaction systems or working with multisets with infinite multiplicity of each element in P systems) remains as a research topic. Three ways to introduce nondeterminism in reaction systems, so that more interesting computation (evolution) graphs can be obtained are mentioned in the next subsection.

P systems with sets were also considered in [9], mainly from the semantics (via Petri nets) point of view.

The second assumption of the reaction systems theory is much easier to handle in terms of membrane computing. The immediate idea is to simply remove any element which does not evolve by means of a reaction; somewhat equivalently, if we want to preserve an object a which is not evolving, we may provide a dummy rule for it, of the type $a \to a$, changing nothing.

Still, many technical problems appear in this framework. The presence of such dummy rules makes the computation endless, while halting is the "standard" way to define successful computations in membrane computing. Moreover, the rules are nondeterministically chosen, hence the dummy rules can interfere with the "computing rules".

While the second difficulty is a purely technical one, the first one can be overpassed by considering other ways of defining the result of a computation in a P

system, and there are many suggestions in the literature. We mention here three possibilities: (i) the *local halting* of [7] (the computation stops when at least one membrane in the system cannot use any rule), (ii) *signal-objects* (the result consists of the number of objects in a specified membrane at the moment when a distinguished object appears in the system), (iii) *signal-events* (the result consists of the number of objects in a specified membrane at the moment when a distinguished rule is used in the system). Such signals were considered in various papers; we refer here only to [8].

Part of these possibilities are checked in [11] both for transition and for symport/antiport P systems – with some cases still remaining open (the most important one is that of catalytic P systems).

17.2 Computing R Systems

We do not give here formal definitions, but we refer the reader to the papers cited below. In particular, we only briefly mention the results from [11] and [10].

Starting from a reaction system, a "generative device" can be defined, based on passing from a configuration to another one, as usual in a reaction system (without input from the environment). This sequence of configurations is unique, because the passage from a set of molecules to the next one is deterministic, hence we either stop after a finite number of steps or we get a sequence of the the form uv^ω: after a finite path among subsets of S (the set of entities/objects in the R system), we enter a cycle which goes forever.

We do not have here too much from a computability point of view, that is why in [11] three possibilities to get a nondeterministic device are proposed: (i) working with tabled R systems, as in Lindenmayer systems, [12] (in each step, a table is used, nondeterministically chosen), (ii) considering also a finite multiplicity for some of the objects, and (iii) by introducing a general threshold on the number of rules which can use the same molecule. All these three possibilities remain to be investigated: properties of the obtained computation graphs, possible links with computing devices from formal language and automata theory, influence of the introduced parameters (number of tables, cardinality of C, threshold k), possible hierarchies.

Of course, another research topic is to find other ways of building a (string or graph) computing device in terms of reaction systems.

17.3 From R to P

Let us consider for P systems each of the two basic assumptions of R systems.

The "threshold" supply of elements was already considered in [10], where an analog of the notion of *hypercomputation* (computing beyond the Turing barrier) was introduced, under the name of *fypercomputation*. It is called so the case when a device can solve in a polynomial time problems known to be (at least) **NP**-complete (the initial **F** in "fypercomputation" comes from "fast").

Research in this framework is rather vivid in membrane computing, and the usual way to speed-up computations is to trade-off time for space, with the (exponential) space being created in a biologically inspired way: by membrane division, membrane creation, string replication, etc. Also working with "enough copies of each element which is present" leads to fypercomputations, and this is not surprising, because we have at hand an arbitrarily large working space. The P systems working with multisets with arbitrarily large multiplicity were called in [10] ωP systems.

More exactly, one considers P systems which contain certain distinguished elementary membranes, whose objects are present in arbitrarily many copies (for instance, if an object a is introduced from outside in such a membrane, then inside the membrane it becomes a^ω; it enters as a single copy, and multiplies inside to arbitrarily many, like in reaction systems).

The arbitrary multiplicity of objects introduces an important change in the functioning of a usual P systems. For instance, if we have the objects a, b, c in a distinguished membrane, together with the rules $ab \to d, ac \to e$, then both these rules can be (and should be) applied, because we have *enough* copies of a for both rules; we obtain d, e, with *all* copies of a, b, c being consumed. If also an object f is present together with a, b, c, then it remains unchanged (we do not adopt here also the second assumption from the definition of R systems, although this can be easily handled, by means of dummy rules of the form $f \to f$).

This apparently innocent observation is able to speed-up a P system to the level of fypercomputations. The proof of the following result can be found in [10]:

Theorem 6. `SAT` *can be solved (in a uniform way) in a polynomial time by an* ωP *system.*

Let us now borrow from reaction systems area the second assumption, the "non-permanence" one, saying that an object which is not involved in a rule does not pass to the next configuration. Then, we cannot define the result of a computation by halting, because in a halting step all objects vanish. Similarly, it is not enough to add dummy rules of the form $a \to a$ (in transition systems), because this time the computation never halts. Thus, we have to define successful computations by other conditions – and we consider here the three possibilities recalled in the beginning of the paper: local halting, signal-objects, signal-events. The definitions are straightforward, we pass directly to recalling from [11] a result about the power of P systems endowed with such conditions.

Theorem 7. *Transition P systems of degree 2, using cooperative rules, without the "permanence" of objects, are computationally complete when the successful computations are defined by local halting or signal-objects. The same result holds true for symport/antiport P systems (of degree 2 and of weight 2) for the case of local halting.*

An interesting *open problem* in this framework is the case of catalytic P systems, known to be universal in the "permanence" assumption (see, e.g., [6]).

The case of defining the result of a computation of symport/antiport P systems by means of signals – objects or events – remains as an *open problem*. (Considering a priority relation on each set of rules can easily solve this problem.) The symport/antiport P system used in the proof of Theorem 7 contains antiport rules of sizes (2, 1) and (1, 2), which is "large" for universality results in the case when objects are persistent. Can the size of rules be decreased also in the case discussed here?

References

1. A. Ehrenfeucht, G. Rozenberg: Basic notions of reaction systems, *Proc. DLT 2004* (C.S. Calude, E. Calude, M.J. Dinneen, eds.), LNCS 3340, Springer, 2004, 27–29.
2. A. Ehrenfeucht, G. Rozenberg: Reaction systems. *Fundamenta Informaticae*, 75 (2007), 263–280.
3. A. Ehrenfeucht, G. Rozenberg: Events and modules in reaction systems. *Theoretical Computer Sci.*, 376 (2007), 3–16.
4. A. Ehrenfeucht, G. Rozenberg: Introducing time in reaction systems. *Theoretical Computer Sci.*, 410 (2009), 310–322.
5. A. Ehrenfeucht, G. Rozenberg: Reaction systems. A model of computation inspired by biochemistry. *Proc. DLT 2010* (Y. Gao et al., eds.), LNCS 6224, Springer, 2010, 1–3.
6. R. Freund, L. Kari, P. Sosik: Computationally universal P systems without priorities: two catalysts are sufficient. *Theoretical Computer Sci.*, 330 (2005), 251–266.
7. R. Freund, M. Oswald: Partial halting in P systems. *Intern. J. Foundations of Computer Sci.*, 18 (2007), 1215–1225.
8. P. Frisco: *Computing with Cells. Advances in Membrane Computing*. Oxford University Press, 2008.
9. J. Kleijn, M. Koutny: Membrane systems with qualitative evolution rules, *Fundaenta Informaticae*, to appear.
10. Gh. Păun: Towards fypercomputations (in membrane computing). LNCS, Springer, to appear.
11. Gh. Păun, M.J. Pérez-Jiménez: Towards Bridging Two Cell-Inspired Models: P Systems and R Systems. Submitted, 2011.
12. G. Rozenberg, A. Salomaa: *The Mathematical Theory of L Systems*. Academic Press, New York, 1980.

Acknowledgements

The work of Gh. Păun was supported by Proyecto de Excelencia con Investigador de Reconocida Valía, de la Junta de Andalucía, grant P08 – TIC 04200.

Closing Remarks

As also said in the Introduction, this collection of open problems and research topics in MC is only a working material, in particular, a working material for 10th BWMC. No such list can be complete, and, as expected, some problems are local, others are very general, while the sections are not at all uniform. For instance, many further research ideas wait to be addressed in the P and dP automata area, as well as in the SN P systems area. Still, we believe that such a list is useful, on the one hand, because it can entail cooperation about the co-authors of the paper and the readers, on the other hand, because it points out active areas of MC. It remains now to see the impact of this list on the activity during 10th BWMC.

The editors are much indebted to all MC researchers who have contributed to this "mega-paper".