Frontiers of Membrane Computing: Open Problems and Research Topics

Marian Gheorghe[1], Gheorghe Păun[2,3], Mario J. Pérez-Jiménez[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, marper@us.es

Summary. This is a list of open problems and research topics collected after the Twelfth Conference on Membrane Computing, CMC 2012 (Fontainebleau, France (23 - 26 August 2011), meant initially to be a working material for Tenth Brainstorming Week on Membrane Computing, Sevilla, Spain (January 30 - February 3, 2012). The result was circulated in several versions before the brainstorming and then modified according to the discussions held in Sevilla and according to the progresses made during the meeting. In the present form, the list gives an image about key research directions currently active in membrane computing.

Introduction

The idea of compiling a collection of open problems and research topics in membrane computing (MC) occurred during the Twelfth International Conference on Membrane Computing, CMC 12, 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 12 (and after that reinforced by email) and several researchers answered this call. The result was circulated under the name of "mega-paper" (*mega* because it has much more co-authors than any other paper in MC...), meant to be a working material for the Tenth Brainstorming Week on Membrane Computing, Sevilla, Spain, January 30 - February 3, 2012 (BWMC 10). During CMC 12 there were also discussions and suggestions regarding some other topics which are not developed here; for this reason we briefly mention

(some of) them: exploring more systematically hypercomputation research ideas within MC area; focussing on translations between different classes of membrane systems (P systems) and studying complexity aspects related to these translations (the goal being to import results from a branch of MC to another one); identifying the most "natural" applications of P systems in modeling biological processes and producing a set of coherent and convincing case studies (a research volume on such applications in biology is now in progress); investigate in more details the dP automata, their efficiency and connections with communication complexity; look for biological applications of spiking neural P systems.

Before presenting an overview of the paper we mention [7], [8] as key references for general MC topics. More specific MC topics, like MC and process calculi [1], interplay between MC and DNA computing [6] and conformon MC systems [3], are also well-established. Applications of MC in various areas can be found in [2].

The initial "mega-paper" was changed several times, incorporating discussions and progresses carried out during BWMC 10. The present version is considered a "closed" one (although such a project can never be closed); for further results related to the problems collected here the reader is invited to follow the MC website from [9]. In particular, one can find there the proceedings volumes, with all papers emerged in connection with the brainstorming.

The texts received from the contributors were revised by their authors after BWMC 10, and appear below in the final form they have been submitted, with minimal editorial changes. 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 MC – for instance, from the sources mentioned at the end of this introduction. A quick introduction to MC is given at the beginning of this paper, just to help the reader not familiar with this research area to have a flavor of it. At the beginning of each section there are mentioned the main notions, from MC and from computability in general, supposed to be known in order to understand the problems which follow (sometimes, part of these notions are briefly introduced together with the problems). The authors of each "section" are mentioned, with affiliations and email addresses, so that the interested reader can contact them for further details, clarifications and cooperation in solving the problems.

The order in which the problems are given below goes, approximately, from general issues to theory and then to applications. In what concerns the computability topics, there are sections devoted to both power and efficiency of P systems, considering them as numbers or strings generators or acceptors, in "old" versions (symport/antiport, catalytic, spiking neural P systems) or in recently introduced forms (polymorphic, dP systems), looking for generalizations (e.g., for "kernel P systems") or for classic notions of language theory not yet extended to MC (such as control words); computational complexity is a vivid direction of investigation, addressing both time and space complexity (defining specific complexity classes, comparing them with existing classes, looking for possibilities of solving computationally hard problems (typically, **NP**-complete problems) in a polynomial time,

by making use of the massive parallelism of P systems and trading-off space for time, with the space obtained by means of biologically inspired operations, such as membrane division and membrane creation). The term "fypercomputing" (following the model of "hypercomputing" = "passing beyond the Turing barrier", with the initial "f" coming from "fast") tries to call attention to a systematic study of "passing polynomially beyond the **NP** barrier". All classes of P systems are considered: cell-like, tissue-like, (spiking) neural, and numerical. Moving to applications, one mentions issues related to the semantics, formal verification, possible bridges with reaction systems (a younger "sister" research area of natural computing, inspired from biochemistry). The applications refer both to the simulation of biological and bio-medical processes and to (somewhat unexpected) applications in approximate optimization (basically, distributed evolutionary algorithms, with the distribution controlled by means of membranes, and borrowing ingredients from MC), robotics (mobile robots controlled by means of numerical P systems), and computer graphics, as well as to more speculative ideas, dealing, for instance, with the functioning of the brain.

The nature of questions range from local/technical open problems, asking to improve existing results, especially for a better delimitation of the borderline between universality and non-universality, between efficiency and non-efficiency (in particular, concerning the influence of some qualitative parameters, such as the number of membranes, the size of the rules, or qualitative features, such as the difference between deterministic and non-deterministic systems, using or not various types of rules), to "strategic" issues, for instance, relating MC with other research areas, such as computer science, biology, ecology, robotics and so on. Of course, many other precise problems or research ideas circulate within 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 proposed in the present paper or variants of them were already circulated within the MC community also before, a fact which should call attention to 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 of the reader in approaching MC and, maybe, in participating in the future editions of the yearly BWMC.

Because several sections below refer to the CMC 12 pre-proceedings and proceedings volumes, we also mention them below – [4] and [5], respectively.

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, Berlin, 2006.
3. P. Frisco: *Computing with Cells. Advances in Membrane Computing*. Oxford Univ. Press, 2009.
4. M. Gheorghe, Gh. Păun, S. Verlan, eds.: *Twelfth International Conference on Membrane Computing, CMC12, Fontainebleau, France, 23–26 August 2011*. LACL, Univ. Paris Est – Créteil Val de Marne, 2011.
5. M. Gheorghe, Gh. Păun, G. Rozenberg, A. Salomaa, S. Verlan, eds.: *Membrane Computing. 12th International Conference, CMC 2011, Fontainebleau, France, August 2011, Revised Selected Papers*, LNCS 7184, Springer, Berin, 2012.
6. A. Păun: *Computability of the DNA and Cells. Splicing and Membrane Computing*. SBEB Publishing, Choudrant, Louisiana, USA, 2008.
7. Gh. Păun: *Membrane Computing. An Introduction*. Springer, Berlin, 2002 (Chinese translation in 2012).
8. Gh. Păun, G. Rozenberg, A. Salomaa, eds.: *The Oxford Handbook of Membrane Computing*. Oxford Univ. Press, 2010.
9. The P Systems Website: `www.ppage.psystems.eu`.

Contents

1 A Glimpse to Membrane Computing

Membrane computing (MC) is a branch of natural computing (introduced in [1], with the report version of the paper circulated as Turku Center for Computer Science – TUCS Report 208, in November 1998, see `www.tucs.fi`) which aims to abstract computing models from the structure and the functioning of the living cell and from populations of cells (e.g., tissues, organs), including the brain. One of the basic notions is that of a *membrane*, understood as a 3D vesicle, separating "an inside" and "an outside", where *objects* can be placed and where specific biochemistries take place. The membranes can be arranged in a hierarchical structure (like in a cell, hence described by a tree) or in an arbitrary structure (like in tissues, hence described by a graph). The space between a membrane and the membranes placed immediately inside it (parent-children, in a tree) is called *region* or *compartment*. A membrane without any membrane inside is said to be *elementary*. In the case of a cell-like arrangement of membranes, the external membrane is called the *skin*. The space outside the skin membrane is called the *environment* (and similarly is called the space external to all membranes of a tissue-like membrane structure). A membrane structure can be formally represented by a rooted labeled tree (each membrane is identified by a label, which is then associated with the node of the tree associated with the membrane), or, correspondingly, by an expression of labeled parentheses, with a unique external pair of parentheses, corresponding to the skin membrane.

The objects are present in the regions of a membrane structure and in the environment in the form of *multisets*, sets with their elements present in a given number of copies (sets with multiplicities of elements). The multiplicity can be finite (expressed by a natural number) or infinite/arbitrary (we say that an object with this

property is of ω multiplicity). For the beginning, let us have in mind only atomic objects, represented by symbols of a given (finite) alphabet, and let us imagine that they correspond to the chemical compounds, from ions to macromolecules, which swim in water in the cell compartments. In this framework, it is convenient to represent the multisets by strings of symbols, with the number of occurrences of a symbol in a string corresponding to the multiplicity of that object in the multiset (that is, any permutation of a string represents the same multiset). These objects react, according to given *evolution rules*. The basic ones (often simply called "evolution rules") are the multiset rewriting rules corresponding to the biochemical reactions taking place in a cell. They are of the form $u \to v$, where u and v are multisets. Many other types of evolution rules are inspired by other biological operations. We mention here only the basic ones: *symport/antiport* correspond to the coupled passage of chemicals through (the protein channels embedded in) the cell membranes, *membrane division* corresponds to mitosis, *membrane creation* and *membrane dissolution* can also be associated with biological processes (the same with exo- and endocytosis, but we do not enter into details). There also are more complex types of rules, or rules inspired from computer science (broadcasting, communication between two membranes placed in a common environment), rules mimicking the way the neurons communicate by means of *spikes* (electrical impulses of identical shapes). Important is that both the rules and the objects are placed in compartments and that the rules act locally, on the objects in the same compartment. Objects can also pass through membranes, both in the cell-like case and in the tissue-like case, hence the compartments cooperate.

There are several ways the rules are applied (several *semantics*). The most investigated one, corresponding to the parallelism of reactions in a solution, is the *maximal parallelism*: a maximal multiset of rules is used, where maximality is defined in the sense of multiset inclusion (no rules can be added to the multiset so that the obtained multiset of rules is still applicable to the multiset of objects present in the respective compartment). When several (maximal) multisets can be applied, the one to use is chosen nondeterministically. Many other possibilities were considered: sequential, limited parallelism, minimal parallelism (the idea is that each compartment which can use a rule – hence it is "alive" – has to use at least one rule, with natural extensions to P systems whose rules are not associated with compartments – as it is the case of symport/antiport systems, where the rules are associated with the membranes). In all these cases, the system is synchronized, a universal clock exists which measures the time in the same way for all membranes and with rules used, synchronously, in each time unit. The natural counterpart is that of asynchronous systems.

Such a device, consisting of membranes, objects, evolution rules, is called a *membrane system* – currently called also a *P system*.

Starting from an *initial configuration* (membranes and objects) of a P system and using the rules according to a chosen strategy, one obtains *computations*, sequences of *transitions* among configurations. If a configuration is obtained such that no rule can be applied, we say that the system *halts*. Several *results* can be

associated with a halting computation, for instance, in the form of the number of objects present in the halting configuration in a designated elementary membrane. A P system can then be seen as a generative device, generating a set of numbers: because of nondeterminism, we have several computations, hence several numbers.

Formally, a P system of the basic form (cell-like, with symbol objects, evolving by multiset rewriting rules) can be given as follows (for an alphabet A, we denote by A^* the set of all strings over A, including the empty string λ; $A^* - \{\lambda\}$ is denoted by A^+):

$\Pi = (O, \mu, w_1, \ldots, w_m, R_1, \ldots, R_m, i_0)$, where
- m is the *degree* of the system,
- O is the alphabet of objects,
- μ is the membrane structure, with m membranes,
- $w_1, \ldots, w_m \in O^*$ are multisets associated with the m regions of μ,
- $R_1, \ldots, R_m$ are finite rules of the form $u \to v$ where u and v are multisets over O with the objects in v also having *target indications* of the form *in, out, here*; an object with indication *out* exits the membrane, one with the indication *here* remains in the same region, and one with the target *in* enters any of the membranes delimiting the region from below, nondeterministically choosing the destination,
- i_0 is the label of the output membrane, the one where the result is obtained.

A transition between two configurations C_1, C_2 of Π is denoted by $C_1 \Longrightarrow C_2$, and the set of numbers generated by Π is denoted by $N(\Pi)$.

The rules of the arbitrary form $u \to v$ are said to be *cooperative*, if $u \in O$, then the rule is called *non-cooperative* (it corresponds to context-free rules in a grammar); an intermediate case is that of *catalytic* rules, which are of the form $ca \to cv$, where $c \in O$ is a catalyst, assisting the object $a \in O$ to get transformed into $v \in O^*$. When applying a rule $u \to v$, the objects from u are consumed and those from v are produced.

An *antiport* rule is of the form $(u, out; v, in)$ with $u, v \in O^*$; using such a rule (associated with a membrane i) means to move the multiset u outside membrane i, simultaneously with bringing the multiset v inside the membrane. If one of the multisets u, v is empty, then the rule becomes a *symport* one.

We do not give here further technical definitions or notations; the interested reader can consult any of the titles indicated in the end of the Introduction, especially the Handbook [8].

However, we mention informally a series of notions and of further classes of P systems. There are many possibilities to extend the previously introduced computing device and its functioning. Instead of counting objects in a compartment, we can consider as the result of a computation the sequence of objects sent to the environment (this is the so-called, *external output*), hence a P system can then

generate a language. A language is obtained also if we follow the *trace* of a special objects across membranes. Then, we can use a (symport/antiport) P system in the accepting mode: the objects entering the system from the environment are arranged in a string and we say that the string is accepted if the computation halts. In the case of tissue P systems, the objects can evolve inside membranes by multiset rewriting rules and can pass from a membrane to another one by antiport rules. The communication channels among membranes are hence implicitly defined by the provided rules for communication; a more complex case is that of *population* P systems, where there also are rules for establishing channels between cells and for destroying them. Besides rules for handling objects, we can also have rules for changing the membrane structure. We mentioned division, creation, and dissolution rules, exo- and endocytosis, but there also are separation, budding, gemmation rules. Observe the biological inspiration, although abstracted in a way which brings us far from biology – in their initial forms, P systems were not meant to be used as models with a biological relevance. The objects can be described by symbols, as above, but they can also have a structure, for instance, described by strings (processed by string operations, such as rewriting, DNA splicing, replication, insertion-deletion), or even more complex, such as 2D arrays, trees, etc. A special case is that of *numerical* P systems, where numerical variables are placed in the regions of a cell-like membrane structure, evolving by means of *programs*, composed of a *production function* (e.g., a polynomial), and a *repartition protocol*; in each compartment, the local variables are subject of a local production function, and the value of this function is distributed among the variables in that region and in the neighboring regions according to the repartition protocol (e.g., proportionally with given numbers, part of the program). The model, somewhat inspired from economics, can both generate sets of numbers, but also compute functions of several variables, a situation which is completely different from the generative-accepting functioning of usual object-based P systems. An interesting variant is that of P systems with objects bound on membranes (as actually is the case with many chemicals in a cell), and then with the rules evolving at the same time objects which are free inside regions and these fixed objects.

Finally, let us mention the so-called *spiking neural P systems* (SN P systems), where membranes (representing neurons) are placed in the nodes of a graph, whose links represent *synapses*, holding several copies of a single object, corresponding to a *spike*; the spikes evolve by rules which first check the contents of the neuron (by means of a regular expression), consume a number of spikes and produce a number of spikes, which are sent, immediately or with a delay, to all neurons to which a synapse goes from the neuron where the rule was used. The spikes sent to the environment by a designated output neuron form the *spike train* produced by the system; numbers or strings can be associated with a spike train, hence again a generative device is obtained.

Up to now, we mentioned only the *generative* mode (corresponding to grammars) of using a P system. A dual case (corresponding to automata) is the *accepting* mode: a number is introduced in a system, e.g., as the multiplicity of a

specified object in a specified compartment, and the number is accepted if the computation halts. Strings can also be recognized, by bringing their symbols, one by one, in a system (e.g., in a symport/antiport one), with the string accepted if the computation halts.

In all cases, we can also use a P system as a *decidability* machine: a decision problem (with YES/NO answer) is introduced in the system, encoded in a specified way in the form of a multiset, and the system says whether the problem (actually, its instance introduced in the initial configuration) has an affirmative answer by halting or by sending a special object `yes` into the environment. This is the usual way of investigating the computational complexity of P systems (the time or the space needed to solve a class of decidability problems).

Most classes of P systems are computationally complete, equivalent with Turing machines (one also says that they are universal), even in restricted cases: small number of membranes, using only catalytic rules (with at least two catalysts: the power of one catalyst P systems is still open), symport/antiport rules of reduced sizes, SN P systems of restricted forms, etc. Similarly, many classes of P systems able to create an exponential working space in a linear time (the typical case is that of P systems using membrane division, also called *with active membranes*) can solve **NP**-complete problems (sometimes even **PSPACE** problems) in a polynomial time. The literature of MC abounds in results of these types.

An important part of the research in MC deals with applications. Using P systems for modeling processes taking place in a cell or in complexes of cells, such as populations of bacteria, is expected; the model starts from biology, hence it is natural to return to biology. Several features make P systems attractive for the biologist (especially in comparison with the models based on differential equations): the direct connection with the biochemistry, which also means a high understandability, the multicompartmental structure, the easy scalability, the intrinsic discrete nature of the model, the easy programmability, the possibility to attach probabilities (reaction rates, stoichiometric coefficients) to the evolution rules, the emergent behavior of a P system (the overall evolution is not at all a "sum" of the parts evolution). All these applications are based on simulation programs (there are several such programs available – see the webpage of the domain, mentioned in the bibliography of the Introduction, [9]). Most of them run on the usual sequential computers, but there also are attempt to implement P systems on dedicated hardware, clusters and grids, on parallel hardware (such as NVIDIA graphical cards). A specialized programming language, *P-lingua*, was also elaborated.

Also somewhat expected are the applications in modeling and simulating ecosystems (we have "membranes" where several agents interact, like the chemicals in a cell). Not so expected however are the applications in approximate optimization (distributed evolutionary computing), computer graphics (following the style of L systems based graphics, but also recent attempts to process images in the parallel framework of P systems), while the recent applications of numerical P systems in controlling mobile robots is completely unexpected.

Problems and research topics about most of these issues will be found in the sections below. Of course, the previous bird's eye view about MC is not enough to technically address these problems, many details were omitted or given in an approximate way, but at least the reader can have an image of this research area, of its many branches, of the richness of results and applications, but also of the fact that many issues still wait for clarifications. The frontiers of MC are still moving, after more than 13 years since this research area was initiated, it still can be considered as an "Emergent Research Front in Computer Science", as it was called already in 2003 by Thomson-Reuters Institute for Scientific Information, ISI, with [1] considered a "fast breaking paper", see `http://esi-topics.com`.

References

1. Gh. Păun: Computing with membranes. *J. Computer and System Sciences*, 61, 1 (2000), 108–143.

2 Some General Issues

Jacob Beal

BBN Technologies, Cambridge, MA, USA
`jakebeal@bbn.com`

Comment. Jacob Beal was one of the invited speakers at CMC 12 (title of talk: "Bringing Biology and Engineering Together with Spatial Computing"). After the meeting, 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 sent to M.Gh. around the end of August 2011.

Required Notions: general knowledge of MC, membrane structure, multiset processing, distribution, parallelism

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 give 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.

3 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 at CMC 12; 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.

Several problems related to the optimality of certain parameters appearing in characterization of the computing power of various classes of P systems (symport/antiport, insertion-deletion, with active membranes) and of their efficiency; in particular, questions about the languages described (in the external mode) by P systems are formulated.

Required Notions: symport/antiport, external output, active membranes, minimal parallelism, insertion-deletion

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.

3.1 Minimal Parallelism and Number of Membrane Polarizations (2006)

It is known, [1, 2] that under minimal parallelism, P systems with polarized 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 deal with P systems using 6 (six!) polarizations, or 4 polarizations if non-standard rule types (evolution rules are applied sequentially and may change the polarization) are used. Are these numbers optimal?

3.2 Membrane Systems Language Class (2010)

A fundamental family of languages is still not characterized: languages generated (in the sense of external output) 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

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

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.

3.3 Dynamical 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 dynamical properties of P systems is given, the most important one being determinism. In particular, five variants are recalled where nondeterminism 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 (universality).

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 of [3] (i.e., the version appearing the LNCS volume) also proposes to study 6 new formal properties inspired by self-stabilization concept.

3.4 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 ("exo" means leftmost or rightmost).

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 the family of regular languages (even with all operations on one side).

3.5 Symport-3 in One Membrane (2005)

Reaching for universality by moving objects across a single membrane leads 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 (2010), 149–170.
`http://www.math.md/publications/csjm/issues/v18-n2/10284/`
2. A. Alhazov: Minimal parallelism and number of membrane polarizations. *Preproc. Seventh International Workshop on Membrane Computing* (H.J. Hoogeboom et al., eds.), WMC7, Lorentz Center, Leiden, 2006, 74–87.
`http://wmc7.liacs.nl/proceedings/WMC7Alhazov.pdf`
3. A. Alhazov: Properties of membrane systems. *Preproc. Twelfth International Conference on Membrane Computing* (M. Gheorghe et al., eds.), 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 et al., 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 (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. *Membrane Computing, 6th International Workshop* (R. Freund et al., eds.), 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. *Preproc. Twelfth International Conference on Membrane Computing* (M. Gheorghe et al., eds.), CMC12, Fontainebleau, 2011, 63–76.
http://cmc12.lacl.fr/cmc12proceedings.pdf

4 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 are briefly introduced (systems where the evolution rules are produced dynamically, during the computation) and several problems about their power and efficiency are formulated.

Required Notions: cell P systems, active membrane, complexity

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. This new construction has not yet been studied properly; very little is known about the computational power of polymorphic P systems.

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, 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 is 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} indicates 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 rule. 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: $\varphi : \{1, \ldots, m\} \to Tar$, $Tar = \{in_j \mid j \in H \text{ is an inner membrane of } p\} \cup \{out, here\}$, 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 list some open problems which we find of interest.

- *Solve hard problems.* It has been shown that polymorphic P systems can solve certain problems faster than any other P system model (for example, they 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, linear speed-up was introduced; a much more systematic research in this direction is necessary. In particular, it is unclear whether it is possible to use the 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 indepen-

dent 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. *11th International Conference on Membrane Computing* (M. Gheorghe et al., eds.), CMC 2010, Jena, Germany, LNCS 6501, Springer, Berlin, 2010, 81–94.

5 P Colonies and dP Automata

Erzsébet Csuhaj-Varjú

Department of Algorithms and Their Applications
Faculty of Informatics, Eötvös Loránd University, Budapest, Hungary
csuhaj@inf.elte.hu

Required Notions: tissue-like P system, P colony, dP automaton

5.1 P Colonies

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 functioning 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 $a \to b$, 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 is 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 other 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 a (multiset) *accepting device* (called *accepting P colony* or *P colony acceptor*), possibly working in an automaton-like fashion as well. In the following we propose problems and problem areas in this direction.

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.

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.

The concept of accepting P colonies can be extended in some other manners as well. For example, we do not fix the number of cells in the P colony in advance but it is determined by the number of non-environmental symbols in the environment at the beginning. Spatial P colonies can also be defined. In this case spatial parameters are added to the cells and a neighborhood relation among the components is given; a cell can import only such symbols from the environment which were issued by its neighbors (are placed 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 automaton if at some step in the course of the computation the leftmost cell enters an accepting state.

A particular variant of one-way cellular automata 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 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.

5.2 dP Automata

In addition to comparing 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 of [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]. In [3] a connection between finite dP 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.

Acknowledgement. Work supported in part by the Hungarian Research Fund "OTKA", project K75952.

References

1. L. Cienciala, L. Ciencialová: Eco-P colonies. *Proc. WMC 2009*, LNCS 5957 (Gh. Păun et al., eds.), Springer, 201–209.
2. L. Ciencala, L. Ciencialová, E. Csuhaj-Varjú, Gy. Vaszil: PCol automata: Recognizing strings with P colonies. *Proc. BWMC 2010, Sevilla, 2010* (M.A. Martínez-del-Amor et al., eds.), Fénix Editora, Sevilla, 2010, 65–76.
3. E. Csuhaj-Varjú, Gy. Vaszil: A connection between finite dP automata and multi-head finite automata. *Proc. Twelfth International Conference on Membrane Computing*, Fontainebleau, 23-26 August, 2011 (M. Gheorghe et al., eds.), 109–126.
4. R. Freund, M. Kogler, Gh. Păun, M.J. Pérez-Jiménez: On the power of P and dP automata. *Annals of Bucharest Univ. Math.-Informatics Series*, 63, 2009, 5–22.
5. M. Kutrib: Nature-based problems in cellular automata, *Proc. CiE 2011*, LNCS 6735, Springer, 2011, 171–180.
6. M. Kutrib, J. Lefevre, A. Malcher: The size of one-way cellular automata. *Proc. Automata 2010: Discrete Mathematics and Theoretical Computer Science*, DMTCS, 2010, 71–90.
7. M. Holzer, M. Kutrib: Cellular automata and the quest for nontrivial artificial self-reproduction. *Proc. Conf. on Membrane Computing, CMC 2010*, LNCS 6501, Springer, 2010, 19–36.
8. Gh. Păun, G. Rozenberg, A. Salomaa, eds.: *The Oxford Handbook of Membrane Computing*, Oxford Univ. Press, 2010.
9. Gh. 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. Gh. Păun, M.J. Pérez-Jiménez: P and dP automata: A survey. *Rainbow of Computer Science* (C.S. Calude et al., eds.), LNCS 6570, Springer, 2011, 102–115.

11. Gh. Păun, M.J. Pérez-Jiménez: An infinite hierarchy of languages defined by dP systems. *Theoretical Computer Sci.*, 431 (2012), 4–12.

6 Spiking Neural P Systems

Linqiang Pan, Tao Song

Key Laboratory of Image Processing and Intelligent Control
Department of Control Science and Engineering
Huazhong University of Science and Technology, Wuhan, Hubei, China
lqpan@mail.hust.edu.cn, songtao608@gmail.com

Applications of spiking neural P systems are proposed and some problems related to such applications are formulated.

Required Notions: spiking neuron, SN P system, spiking neural network

Spiking neural P systems (SN P systems, for short) were introduced in [4] as a class of distributed and parallel computing models inspired by spiking neurons. In an SN P system, the *neurons* are placed in the nodes of a directed graph. The content of each neuron consists of a number of copies of a single object type, called the *spike*. Each neuron contains a number of *firing* and *forgetting rules*. Firing rules allow a neuron to send information to other neurons in the form of electrical impulses (also called spikes) which are accumulated at the target cells. The applicability of each rule is determined by checking the content of the neuron against a regular set associated with the rule. A forgetting rule removes a specified number of spikes from the neuron. In each time unit, if a neuron can use some of its rules, firing or forgetting, then one of the rules must be used. The rule to be applied is nondeterministically chosen.

One of the neurons is designated as the output neuron of the system, and its spikes are also sent to the environment; their sequence is called *the spike train* generated by the system. Several *results of a computation* can be defined associated with the spike train (strings or numbers).

SN P systems use individual spikes allowing to incorporate spatial and temporal information in computation, which corresponds to the fact that neurons use spatial and temporal information of incoming spikes to encode their message to other neurons, where the number and timing of spikes matters. In the above sense, SN P systems fall into the third generation of neural network models [6].

Many computational properties of SN P systems have been studied (but many of them raise further research topics, but we do not refer to them here). SN P systems were proved to be computationally complete as number computing devices [4], language generators [1, 2], and function computing devices [8]. SN P systems were also used to (theoretically) solve computationally hard problems in a feasible

time [5, 7]. In contrast with the relatively rich theoretical results, the practical applications of SN P systems are few (although some attempts are already made, e.g., Hebbian learning in the framework of SN P systems [3]). However, as a representative of the third generation of neural network models, spiking neural networks (SNNs) could have very hands-on applications such as speech recognition, learning, associative memory, function approximation (see, e.g., *Information Processing Letters*, 95, 2005), and have proved to be useful in neuroscience. It is interesting to move the SN P systems investigations towards applications. In the following, we list some problems which we find of interest.

- In SN P systems, the use of spike timing information is based on regular expressions, which can be considered as an integrate-and-fire scheme. The scheme of regular expressions is quite different from the traditional ones, such as the sigmoidal scheme. What is the advantage of the scheme of regular expressions from the application point of view? Can the two schemes (the regular expression and the sigmoidal one) be related?
- What ingredients can be added to SN P systems for practical applications (maybe, noise, randomness)?
- What are the specific real world problems where SN P systems have a practical advantage over other SNNs?
- How can some variants of SN P systems be designed such that they would deal with features of more biological plausibleness?

References

1. H. Chen, M. Ionescu, T.-O. Ishdorj, A. Păun, Gh. Păun, M.J. Pérez-Jiménez: Spiking neural P systems with extended rules: universality and languages. *Natural Computing*, 7 (2008), 147–166.
2. H. Chen, R. Freund, M. Ionescu, Gh. Păun, M.J. Pérez-Jiménez: On string languages generated by spiking neural P systems. *Fundamenta Informaticae*, 75 (2007), 141–162.
3. M.A. Gutiérrez-Naranjo, Mario J. Pérez-Jiménez: Hebbian learning from spiking neural P systems view. *Proc. WMC9 2008* (D. Corne et al., eds.), LNCS 5391, Springer, Berlin, 2009, 217–230.
4. M. Ionescu, G. Păun, T. Yokomori: Spiking neural P systems. *Fundamenta Informaticae*, 71 (2006), 279–308.
5. T.-O. Ishdorj, A. Leporati, L. Pan, X. Zeng, X. Zhang: Deterministic solutions to QSAT and Q3SAT by spiking neural P systems with pre-computed resources. *Theoretical Computer Science*, 411 (2010), 2345–2358.
6. W. Maass: *The Third Generation of Neural Network Models*. Technische Universitat Gräz, 1997
7. L. Pan, Gh. Păun, M.J. Pérez-Jiménez: Spiking neural P systems with neuron division and budding. *Science China Information Sciences*, 54 (2011), 1596–1607.
8. A. Păun, Gh. Păun: Small universal spiking neural P systems. *BioSystems*, 90 (2007), 48–60.

7 Control Words Associated with P Systems

Kamala Krithivasan[1], Gheorghe Păun[2], Ajeesh Ramanujan[1]

[1]Department of Computer Science and Engineering
Indian Institute of Technology Madras, Chennai, India
kamala@iitm.ac.in

[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

Ways to associate a control word with a computation in a P system are proposed and some of the problems which are natural to be investigated in this respect are mentioned.

Required Notions: Szilard language, Chomsky hierarchy, cell P system, SN P system, parallelism

Control words are almost never considered in membrane computing – actually, we know no paper dealing with this issue, although generating or recognizing languages are central research topics (with the languages identified by the sequence of symbols entering or leaving a P system, or by traces of certain symbols in their passage across membranes). The reason is the fact that in the same step of a computation several rules are used, possibly with several labels, hence the control word is not clearly defined. On the other hand, a sort of bidimensional control word was introduced already during the first BWMC, in [1], under the name of *Sevilla carpet*, as a way to describe the rules used in a computation and their multiplicity in each step, but not as a way to define a control language associated with the computations in a P system.

A possible solution to the above difficulty is to consider a sequence of multisets of labels, those labels associated with all rules applied in a given step. Then, a string of symbols can be obtained following the ideas also used for accepting P systems: take a function from multisets to strings and build the string(s) obtained by concatenating the strings associated with the multisets. For instance, all permutations of the labels in a multiset can be considered, as in [3], or only one specific string (maybe a symbol) associated with the multiset, like in [2].

Another idea was recently introduced in [4], starting from the following restriction: all rules used in a computation step should have the same label, or they can also be labeled with λ.

The definition in [4] is given for SN P systems, but it works for any type of P systems, not only for SN P systems.

Indeed, let us consider a P system Π, of any type, with the total set of rules (the union of all sets of rules associated with compartments, membranes, neurons – as it is the case) denoted with R. Consider a labeling mapping $l : R \to B \cup \{\lambda\}$, where B is an alphabet. We consider only transitions $s \Longrightarrow_b s'$, between configurations s, s' of Π, which use only rules with the same label b and rules labeled with λ. We say that such a transition is *label restricted*. With a label restricted transition we associate the symbol b if at least one rule with label b is used; if all used rules have the label λ, then we associate λ to this transition. Thus, with any computation in Π starting from the initial configuration and proceeding through label restricted transitions we associate a (control) word. Consider also a criterion $\mathtt{C}$ of the correct termination of a computation (e.g., halting or reaching a configuration from a given set F of final configurations, or both of these, etc.) The language of control words associated with all label restricted computations in Π which are correctly terminated (with respect to $\mathtt{C}$) is denoted by $Sz_{\mathtt{C}}(\Pi)$ (with Sz coming from *Szilard*, as usual in language theory).

Now, a series of natural problems can be formulated: investigate the languages of control words for (i) various classes of P systems, with (ii) various criteria $\mathtt{C}$, in particular, (iii) allow only transitions which use at least a rule labeled by $b \in B$. When λ transitions are accepted, characterizations of RE languages are expected, but when each step produces a symbol, there is no possibility for "hidden work", the computation has the same length as the control string, so that the generated language is recursive. In this latter case the comparison with language families in Chomsky hierarchy is of interest (with the conjecture that languages of the forms $\{xx \mid x \in V^*\}$, $\{xx^R \mid x \in V^*\}$, where $card(V) \geq 2$ and x^r is the mirror image of x, cannot be obtained as the language of control words of a P system.

In particular, the languages $Sz_{\mathtt{C}}(\Pi)$ can be associated with SN P systems, with or without anti-spikes. We expect interesting (language theory) results in this research area.

References

1. G. Ciobanu, Gh. Păun, Gh. Ştefănescu: Sevilla carpets associated with P systems. *Proc. Brainstorming Week on Membrane Computing* (M. Cavaliere et al., eds.), Tarragona Univ., TR 26/03, 2003, 135–140.
2. E. Csuhaj-Varjú, Gy. Vaszil: P automata or purely communicating accepting P systems. *Membrane Computing, International Workshop, WMC-CdeA, Curtea de Argeş, Romania, August 19-23, 2002, Revised Papers* (Gh. Păun et al., eds.), LNCS 2597, Springer, 2003, 219–233.
3. M. Oswald: *P Automata*, PhD Thesis, TU Viena, 2003.
4. A. Ramanujan, K. Krithivasan: Control words of spiking neural P systems. Paper in preparation.

8 Speeding Up P Automata

György Vaszil

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

The issue of efficient parallelization of languages with respect to dP automata is discussed (especially, the dependence on the multiset-to-strings functions which are used to define the input language).

Required Notions: Regular language, context-sensitive language, P automata and dP automata (accepted multiset sequence, input mapping, accepted language)

This section deals with the possibility of speeding up P automata computations (in a similar sense as a linear speedup for Turing machines is possible), a problem which is important from the point of view of the efficiency of the parallelization of P automata computations with distributed P automata.

A *P automaton*, introduced in [2], is an antiport P system placed in an environment, from where a sequence of input multisets is read during the computation. A multiset sequence is accepted, if the computation ends in an accepting configuration, and the accepted multiset sequence is interpreted as a string (a sequence of symbols) using a so called input mapping $f : V^* \to 2^{T^*}$ where T is a finite alphabet and V is the object alphabet of the P automaton. (We assume that f is nonerasing, that is, $f(u)$ is the empty word for some multiset $u \in V^*$, if and only if u is empty.) The language accepted by a P automaton Π with respect to f is defined as $L(\Pi, f) = \{f(v_1) \dots f(v_s) \mid v_1, \dots, v_s \text{ is an accepted multiset sequence of } \Pi\}$.

It is obvious that the choice of the mapping f has a great influence on the accepting power of the P automaton, so let us take a closer look at the mappings we can use.

Let $f : V^* \to 2^{T^*}$, and (1) let us denote f with f_{perm}, if and only if $V = T$, and for all $v \in V^*$, we have $f(v) = \{u \mid u \text{ is a permutation of } v\}$. Moreover, (2) we say that $f \in$ 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 $w \neq \lambda$ for any nonempty input).

Let us 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 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 are P automata which process different parts of the input in parallel. 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. Let $f = (f_1, \dots, f_k)$ 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 $L(d\Pi, f) = \{w_1 \dots w_k \in T^* \mid w_i \in f_i(v_{i,1}) \dots f_i(v_{i,s_i}),\ 1 \leq i \leq k,$ where $v_{i,1}, \dots, v_{i,s_i}$ is an accepted multiset sequence of the component $\Pi_i\}$.

A language is efficiently parallelizable, as defined in [5], if it can be accepted by a dP automaton in "less" computational steps than by any non-distributed P automaton, that is, 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 $\mathrm{time}_X(x)$ 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 possibility of speeding up P automata computations.

Recall that for 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 using different mappings, it might be possible to "read" the same word in several different ways, possibly also in different numbers of computational steps.

To demonstrate this, 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 regular 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 than the length of the input. Thus, 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}, which makes it interesting to ask the following.

Problem. *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 $f \neq f_{perm}$?*

In this context, 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.

Acknowledgement. Work supported in part by the Hungarian Research Fund "OTKA", project K75952.

References

1. E. Csuhaj-Varjú, M. Oswald, Gy. Vaszil: P automata. *The Oxford Handbook of Membrane Computing* (Gh. Păun et al., eds.), Oxford Univ. Press, 2010, chapter 6, 144–167.
2. E. Csuhaj-Varjú, Gy. Vaszil: P automata or purely communicating accepting P systems. *Membrane Computing, International Workshop, WMC-CdeA, Curtea de Argeş, Romania, August 19-23, 2002, Revised Papers* (Gh. Păun et al., eds.), LNCS 2597, Springer, 2003, 219–233.
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 (2009), 5–22.
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 (2010), 238–250.
6. Gh. Păun, M.J. Pérez-Jiménez: P and dP automata: A survey. *Rainbow of Computer Science* (C. Calude et al., eds.), LNCS 6570, Springer, 2011, 102–115.
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. *Computation, Cooperation, and Life. Essays Dedicated to Gheorghe Păun on the Occasion of His 60th Birthday* (J. Kelemen, A. Kelemenová, eds.), LNCS 6610, Springer, 2011, 170–178.

9 Space Complexity and the Power of Elementary Membrane Division

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

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

The title is self-explanatory – it should be mentioned that the Milano team has important contributions related to the (time and space) complexity issue, and mainly open problems related to these results are formulated below.

Required Notions: active membranes, computational complexity theory (including counting problems and oracle Turing machines), space complexity for P systems

Problem 1 (P systems with elementary active membranes). P systems with active membranes [8] 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 [6] shows that polynomial-time Turing machines having access to an oracle for a **PP** [1] problem (whose computing power includes the polynomial hierarchy [10]) can be simulated by uniform families [4] 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 systems cannot solve in polynomial time any problem outside **PSPACE** [9], 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 [5] in order to supplement the already rich literature about computational complexity issues in membrane computing [3]. 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 [4] is said to solve a problem in space $f\colon \mathbb{N} \to \mathbb{N}$ if no P system 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, denoted by $\mathbf{PMCSPACE}_{\mathcal{AM}}$, coincides with **PSPACE** [7]. Furthermore, during the 10th Brainstorming Week

on Membrane Computing it was also proved that the problems solvable in exponential space by that variant of P systems and by Turing machines coincide, in symbols $\mathbf{EXPMCSPACE}_{\mathcal{AM}} = \mathbf{EXPSPACE}$.

The techniques used to prove these results do not seem to apply when the space bound is less strict, i.e., 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 should be weakened, since polynomial-time Turing machines constructing the families might be able to solve the problems altogether by themselves. More general forms of uniformity have already been investigated [2], and that work is going to be useful when attacking this problem.

References

1. J.T. Gill: Computational complexity of probabilistic Turing machines. *Proc. Sixth Annual ACM Symposium on Theory of Computing, STOC '74*, 1974, 91–95.
2. N. Murphy, D. Woods: The computational power of membrane systems under tight uniformity conditions. *Natural Computing*, 10 (2011), 613–632.
3. M.J. Pérez-Jiménez: A computational complexity theory in membrane computing. *Membrane Computing, 10th International Workshop, WMC 2009* (Gh. Păun et al., eds.), LNCS 5957, Springer, 2010, 125–148.
4. A.E. Porreca, A. Leporati, G. Mauri, C. Zandron: Elementary active membranes have the power of counting. *International Journal of Natural Computing Research*, 2 (2011), 35–48.
5. A.E. Porreca, A. Leporati, G. Mauri, C. Zandron: Introducing a space complexity measure for P systems. *International Journal of Computers, Communications & Control*, 4 (2009), 301–310.
6. A.E. Porreca, A. Leporati, G. Mauri, C. Zandron: P systems simulating oracle computations. *Membrane Computing, 12th International Conference, CMC 2011* (M. Gheorghe et al., eds.), LNCS 7184, Springer, 2012, 346–358.
7. 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 (2011), 65–73.
8. Gh. Păun: P systems with active membranes: Attacking NP-complete problems. *Journal of Automata, Languages and Combinatorics*, 6 (2011), 75–90.
9. P. Sosík, A. Rodríguez-Patón: Membrane computing and complexity theory: A characterization of PSPACE. *Journal of Computer and System Sciences*, 73 (2007), 137–152.
10. S. Toda: PP is as hard as the polynomial-time hierarchy. *SIAM Journal on Computing*, 20 (1991), 865–877.

10 The P-Conjecture and Hierarchies

Niall Murphy

Facultad de Informática
Universidad Politécnica de Madrid, Madrid, Spain
niall.murphy@upm.es

The P conjecture deals with the power of polarizations associated with the membranes of a P system with active membranes (looking for the borderline between efficiency and non-efficiency in this framework).

Required Notions: active membranes (without charges), weak non-elementary membrane division, elementary membrane division, dissolution rules, recognizer P systems, uniform families

Conjecture 1 (The P-conjecture (Problem F in [8])). 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 **P**.

Attempting to resolve the P-conjecture and its restrictions [1, 3, 4, 5, 9, 10] has resulted in many interesting new techniques, such as dependency graphs [3], for proving upper-bounds on membrane systems. Hopefully solving the P-conjecture will need new tools that yield deep revelations and open new questions into the nature of P-systems.

If the P-conjecture is proved to be true, then membrane systems with elementary division rules characterize **P** while those with weak non-elementary division rules characterize **PSPACE** [1, 9]. The deterministic class **P** is also the 0th level of Polynomial Hierarchy [6]. The complete problems of each successive level of the hierarchy seem to require increasing interleaving of nondeterminism and co-nondeterminism. The union of all levels is referred to as **PH** which is contained in **PSPACE**.

Characterizing each level of the Polynomial Hierarchy with a single model might give us clues to the role of nondeterminism in P systems. For example, could it be that division of different numbers of nested membranes is the membrane computing equivalent of alternating universal and existential nondeterminism?

Conjecture 2. 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.

Some ideas for showing the lower-bound of this conjecture can be found in [7].

Continuing with the idea of hierarchies, a characterization of each level of the **NC** (or **AC**) hierarchy [6] may shed new light on the role of parallelism in

membrane systems. The **NC** hierarchy represents 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]. A good place to start learning more about the factors that limit and permit parallelism in P systems might be a membrane characterization of the $\mathbf{NC} \stackrel{?}{=} \mathbf{P}$ problem (the so called "frontier of parallelism").

Problem 3. Is it possible to parameterize a resource or rule of a membrane computing model such that when the parameter is i it characterizes the ith level of the **NC** or **AC** hierarchy.

References

1. A. Alhazov, M.J. Pérez-Jiménez: Uniform solution to QSAT using polarizationless active membranes. *Machines, Computations and Universality (MCU)* (J. Durand-Lose, 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), 613–632.
5. N. Murphy, D. Woods: Active membrane systems without charges and using only symmetric elementary division characterise P. *Membrane Computing, 8th International Workshop, WMC 2007*, LNCS 4860, Springer, 2007, 367–384.
6. C.H. Papadimitriou: *Computational Complexity*. Addison Wesley, 1993.
7. A.E. Porreca, N. Murphy: First steps towards linking membrane depth and the Polynomial Hierarchy. *Proc. 8th Brainstorming Week on Membrane Computing*, Sevilla, 2010, 255–266.
8. Gh. Păun: Further twenty six open problems in membrane computing. *Proc. Third Brainstorming Week on Membrane Computing*, Sevilla (Spain), 2005, 249–262.
9. P. Sosík, A Rodríguez-Patón: Membrane computing and complexity theory: A characterization of PSPACE. *Journal of Computer and System Sciences*, 73 (2007), 137–152.
10. D. Woods, N. Murphy, M.J. Pérez-Jiménez, A. Riscos-Núñez: Membrane dissolution and division in P. *Unconventional Computation*, 5715 (2009), 262–276.

11 Seeking Sharper Frontiers of Efficiency in Tissue P Systems

Mario J. Pérez-Jiménez[1], Agustín Riscos-Núñez[1], Miquel Rius-Font[2], Álvaro Romero-Jiménez[1]

[1] Department of Computer Science and Artificial Intelligence
University of Sevilla, Spain
marper@us.es

[2] Department of Applied Mathematics IV
Universitat Politécnica de Catalunya, Casteldefels, Spain
mrius@ma4.upc.edu

In a P system, there are several ingredients which concur to their efficiency; varying them, one can get efficient systems (able to solve computationally hard problems in polynomial time) or non-efficient systems (e.g., solving **NP**-hard problems in an exponential time). The borderline between efficiency and non-efficiency is thus a problem of a central interest. This issue is explored here for tissue P systems, where the respective research started later than for cell P systems.

Required Notions: tissue P systems, complexity classes, cell division, cell separation, symport/antiport rule

A *tissue P system with symport/antiport rules* $\Pi = (\Gamma, \mathcal{E}, \mathcal{M}_1, \ldots, \mathcal{M}_q, \mathcal{R}, i_{out})$, of degree $q \geq 1$ can be viewed as a set of q cells, labeled by $1, \ldots, q$, with an environment labeled by 0 which initially have an arbitrary number of copies of some kind of objects, and a set of rules which can be of several types: communication, division or separation (see [3, 4] for details).

For each natural number $k \geq 1$, $\mathbf{TDC}(k)$ (respectively, $\mathbf{TDS}(k)$ or $\mathbf{TDA}(k)$) is the class of recognizer tissue P systems with cell division and communication rules (allowing only **s**ymport or **a**ntiport rules, respectively) of length at most k. Similarly, by considering separation rules instead of division rules, we denote $\mathbf{TSC}(k)$, $\mathbf{TSS}(k)$ and $\mathbf{TSA}(k)$ respectively. We denote by $\mathbf{PMC_R}$ the set of all decision problems which can be solved in a uniform way and polynomial time by means of families of systems from a class $\mathbf{R}$ of recognizer tissue P systems.

(A) Tissue P systems with cell division and with cell separation

By using the dependency graph technique, it has been proved that $\mathbf{P} = \mathbf{PMC}_{TDC(1)} = \mathbf{PMC}_{TSC(1)}$ [2, 3]. Furthermore, efficient and uniform solutions to the SAT problem by using systems from $\mathbf{TDC}(3)$ [1] and from $\mathbf{TSC}(8)$ [3] have been given. Recently, the last result has been improved to $\mathtt{SAT} \in \mathbf{PMC}_{TSC(3)}$ [6].

Problem 1. Assuming $\mathbf{P} \neq \mathbf{NP}$, in the framework of tissue P systems with cell division/cell separation, a frontier of the tractability is obtained when passing from communication rules with length 1 to communication rules with length at most 3. Does passing from 1 to 2, amounts to passing from non–efficiency to efficiency?

Conjecture: $\mathbf{NP} \cup \mathbf{co}\text{-}bfNP \subseteq \mathbf{PMC}_{TDC(2)}$.

(B) The role of direction in communication rules

Next, we deal with complexity aspects of tissue P systems with cell divison/celll separation where only symport or antiport rules are allowed. We have: $\mathbf{P} = \mathbf{PMC}_{TDA(1)} = \mathbf{PMC}_{TSA(1)}$, and $\mathbf{NP} \cup \mathbf{co} - \mathbf{NP} \subseteq \mathbf{PMC}_{TDA(3)} \cap \mathbf{PMC}_{TSA(3)}$. Thus, assuming $\mathbf{P} \neq \mathbf{NP}$, a first frontier between efficiency and non-efficiency is obtained in the above framework when passing from communication rules with length 1 to communication rules with length at most 3.

Problem 2. What about the complexity classes $\mathbf{PMC}_{TDA(2)}$, $\mathbf{PMC}_{TSA(2)}$, $\mathbf{PMC}_{TDS(k)}$ and $\mathbf{PMC}_{TSS(k)}$, for all $k \geq 1$?

Conjecture: $\mathbf{P} = \mathbf{PMC}_{TSA(2)}$, and for all $k \geq 1$, $\mathbf{P} = \mathbf{PMC}_{TSS(k)}$.

If this conjecture is true, then passing from symport rules to antiport rules with length at least three, amounts to passing from non–efficiency to efficiency, in the framework of tissue P systems with cell separation.

(C) The role of the environment

Classical tissue P systems have a special alphabet associated with the environment, whose elements appear at the initial configuration of the system, in an arbitrary large amount of copies. What may happen if this property is removed, that is, if the alphabet associated to the environment were empty? We use a "hat" to indicate the case when the environment is initially empty.

Recently, have been proved that, for each $k \geq 1$, $\mathbf{PMC}_{\mathcal{TDC}(k)} = \mathbf{PMC}_{\widehat{\mathcal{TDC}}(k)}$ [5], that is, in the framework of tissue P systems with cell division the role of the environment is not relevant from the complexity point of view.

Conjecture: For each $k \geq 1$, $\mathbf{P} = \mathbf{PMC}_{\widehat{\mathcal{TSC}}(k)}$.

If this conjecture is true, then in the framework of tissue P systems with cell communication the following holds: (a) passing from separation rules to division rules (length at least three) amounts to passing from non–efficiency to efficiency; and (b) the environment provides a new borderline of efficiency.

References

1. D. Díaz, M.A. Gutiérrez, M.J. Pérez-Jiménez, A. Riscos-Núñez: A uniform family of tissue P systems with cell division solving 3-COL in a linear time. *Theoretical Computer Science*, 404 (2008), 76–87.
2. R. Gutiérrez-Escudero, M.J. Pérez-Jiménez, M. Rius-Font: Characterizing tractability by tissue-like P systems. *Membrane Computing. 10th International Workshop, WMC 2009, Curtea de Argeş, August 2009. Revised Selected and Invited Papers*, LNCS 5957, Springer, Berlin, 2010, 289–300.
3. L. Pan, M.J. Pérez-Jiménez: Computational complexity of tissue–like P systems. *Journal of Complexity*, 26 (2010), 296–315.

4. Gh. Păun, M.J. Pérez-Jiménez, A. Riscos-Núñez: Tissue P systems with cell division. *International Journal of Computers, Communications & Control*, 3 (2008), 295–303.
5. M.J. Pérez-Jiménez: The role of the environment in tissue P systems with cell division. Submitted, 2012.
6. M.J.Pérez-Jiménez, P. Sosík: Improving the efficiency of tissue P systems with cell separation. Submitted, 2012.

12 Time-Free Solutions to Hard Computational Problems

Matteo Cavaliere

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

P systems are usually synchronized, a unique clock marks the time for all components, and in each time unit each component evolves (usually, in the maximal parallel manner). In time-free (and clock-free) systems, this strong assumption is removed. Up to now, the efficiency of P systems was not investigated also for this case.

Required Notions: Time-free P system, synchronization, recognizing P system, uniform/semi-uniform solution.

12.1 Motivations

Living cells have division rates that are highly heterogeneous (even in identical environmental conditions), consequence of their stochastic gene expression, [1]. Therefore, the possibility of programming living cells should not assume the presence of uniform replication rates. Ideally, one should construct "cellular computers" whose functioning is independent of cellular division rates. We suggest that such problem can be addressed in the framework of membrane computing by extending the notion of time-freeness ([4]) to the idea of semi-uniform solutions of computational problems based on membrane divisions ([3]).

12.2 Timed Recognizer P Systems

From [4] 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 started (maximal parallel and nondeterministic use of rules). *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*). 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 [3] we recall the notion of recognizer P systems. 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.

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 by proposing the following timed variant: a *recognizer timed P system* is a *timed P system* with properties (i), (ii), (iii) above.

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.

12.3 Time-Free Solutions to Decision Problems

Let $X = (I_X, \Theta_X)$ be 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 original motivations: 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 [3], and consider 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 $\Pi = \Pi_u, u \in I_X$, if the following are true:

- the family Π is polynomially uniform by a Turing machine; 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.

In other words, to provide a time-free solution one must construct the family of systems Π in polynomial time (sequential time by deterministic Turing machines) and the constructed family must be "fast" (polynomially bounded), sound and complete with respect to the considered problem X, and these properties must be independent of the execution time of the rules (i.e., they must be fulfilled independently of the time-mapping considered).

The definition of time-free semi-uniform solution captures the problem informally discussed in the Motivations. The basic *question* consists in finding a class of membrane systems for which it is possible to construct time-free semi-uniform solutions to hard computational problems. The simplest possibility is to transform the solutions already present in literature into time-free solutions (e.g., could the solution given in [2] be adapted to become a time-free solution?).

Another interesting problem is to find classes of membrane systems that are powerful enough to solve complex problems, but simple enough to allow an automatic (i.e., algorithmic) checking of their time-freeness.

References

1. M.B. Elowitz, J. Levine, E.D. Siggia, P.S. Swain: Stochastic gene expression in a single cell. *Science*, 297 (2002), 5584.
2. Gh. Păun: P systems with active membranes: Attacking NP complete problems. *Journal of Automata, Language and Combinatorics*, 6 (2001), 75–90.
3. M.J. Pérez-Jiménez, A. Riscos-Núñez, A. Romero–Jiménez, D. Woods: Complexity – membrane division, membrane creation. Chapter 12 in [5], 302–336.

4. M. Cavaliere, D. Sburlan: Time-independent P systems. *Membrane Computing. Int. Workshop WMC 2004*, Milan, Italy, 2004 (G. Mauri et al., eds.), LNCS 3365, Springer, 2005, 239–258.
5. Gh. Păun, G. Rozenberg, A. Salomaa, eds.: *The Oxford Handbook of Membrane Computing*. Oxford Univ. Press, 2010.

13 Fypercomputations

Gheorghe Păun

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

Following the model of *hypercomputation* (computing beyond the "Turing barrier"), we propose here the term *fypercomputation* to name the research area of "solving polynomially problems which are (at least) **NP**-complete". Some ideas from/for MC are mentioned.

Required Notions: membrane division, membrane creation, hypercomputing, SN P system, reaction system, accelerated P system

Looking for ideas which would lead to computing devices able to compute "beyond the Turing barrier" is already a well established research area of computing theory; such devices are said to be able of doing *hypercomputations*. It is also a dream and a concern of computability to speed-up computing devices; a name was proposed in [7] (the idea was further elaborated in [8]) for the case when this leads to polynomial solutions to problems known to be (at least) **NP**-complete: *fypercomputing* – with the initial **F** coming from "fast".

In short: *fypercomputing means going polynomially beyond* **NP**.

The model we have in mind is that of hypercomputations, already with a large literature (we only mention the recent survey from [10]). More than a dozen of ideas were proposed and proved to reach the goal of computing "beyond Turing": oracles (already considered by Turing), introducing real numbers in the device, accelerating the functioning of machines, using ingredients of an analogical nature and so on. Many of these ideas can probably lead not only to hypercomputations, but also to fypercomputations, both in MC and in other frameworks.

Although not clustered under a good name, such as hypercomputation (there are periodical meetings dedicated to this research direction), there also are many

papers which can be placed under the flag of fypercomputing. They exploit ideas from physics, such as [9], propose analogical computations, such as [1]. Also the area of DNA computing is full of such ideas.

The literature of membrane computing abounds in papers dealing with fypercomputations. In most cases, polynomial solutions to **NP**-complete problems – often, also of **PSPACE**-complete problems – are obtained, by making use of a space-time trade-off, with the space obtained during the computation, by means of operations inspired from biology. The most investigated operations of this kind are *membrane division* (with variants: separation, budding, etc.) and *membrane creation*.

Further two similar ideas were also explored. The first one is based on string replication (see [3] for details), the second one is that of considering arbitrarily large *pre-computed resources* (see, e.g., [6]), but the last idea is only briefly investigated so far. Issues related to the conditions to be imposed to the given pre-computed resources should be further considered.

Three more ideas, essentially different from the previous ones, were proposed in [8] and need additional research efforts.

(1) The first candidate is the *acceleration*, an old one in computer science: a "clever" computing device learns from its own functioning; after performing a step in a time unit, it performs better for the second step, which is completed in half of the time necessary for the first step – and so on, at each step halving the time with respect to the previous step. If the first step takes one time unit, then the second one takes 1/2 time units, the third one 1/4 and so on, hence in two time units the computation ends.

Important: we have here two clocks, an internal one, of the machine, and an external one, of the observer. The internal clock is faster and faster, so that the computation ends in two time units *measured by the external clock*, that of the observer/user.

Accelerated Turing machines can solve the halting problem, hence they compute what usual Turing machines cannot. See references in [2], where the idea is extended to P systems: starting from the biological observation that "smaller is faster" and using membrane creation rules to create "faster reactors" (inner membranes), in an unbounded hierarchy, one can obtain P systems which "compute the uncomputable".

This trick can be used also in complexity, but we have to be cautious: we accelerate in order to get a speed-up... In two (external) time units we solve any problem, whatever complex it is. A way to make the things interesting is to accelerate only parts of a P system, thus having several levels of time speed. For instance, we can accelerate only (i) some elementary membranes, or (ii) only some rules (a given rule takes one time unit for the first application, half for the second application, and so on), or (ii) to have "accelerated objects" (the descendants of an object react faster than the father object, irrespective which are the rules which act on them and irrespective of the membranes where they are). Precise definitions should be found and their usefulness explored.

(2) The previous ideas suggest the following speculation. We mentioned that we have (at least) two clocks, an external one, of the observer (or of the higher membranes in the structure) and the local clock(s), of the accelerated element, membrane, rule, object. Always, the inner clock is (much) faster than the external one, it performs sometimes an exponential number of steps while the external one only ticks once. We can then imagine that the inner time is orthogonal to the external time, hence the time has a 2D structure: the observer only senses one dimension of time, but certain "processors" can run along the other dimension, doing computations at-no-time for the observer. This looks very much as using oracles. Again, good definitions have to be found and explored.

(3) One further idea, proved in [8] to lead to fypercomputations comes from the recently introduced *reaction systems* (we call them *R systems*) – see [4], [5]. One of the crucial postulates of R systems concerns the fact that one works with ω multisets: an object either is not present, or it is present in arbitrarily many copies. This assumption can be extended also to P systems. More exactly, we consider 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 immediately becomes a^ω). In [8], such a system is called ωP system and it is proved that `SAT` can be solved (in a uniform way) in a polynomial time by an ωP system.

The construction in [8] uses cooperative rules; we do not know whether the result can be improved by imposing the restriction to use only non-cooperative rules.

References

1. J.J. Arulanandham, C.S. Calude, M.J. Dinneen: Balance machines: computing = balancing. *Aspects of Molecular Computing, 2004*, LNCS 2950, Springer, 2004, 148–161.
2. C.S. Calude, Gh. Păun: Bio-steps beyond Turing. *BioSystems*, 77 (2004), 175–194.
3. J. Castellanos, Gh. Păun, A. Rodriguez-Patón: Computing with membranes: P systems with worm-objects. *Proc. IEEE 7th. Intern. Conf. on String Processing and Information Retrieval, SPIRE 2000*, La Coruna, Spain, 2000, 64–74.
4. A. Ehrenfeucht, G. Rozenberg: Basic notions of reaction systems, *Proc. DLT 2004*, LNCS 3340, Springer, 2004, 27–29.
5. A. Ehrenfeucht, G. Rozenberg: Reaction systems. *Fundamenta Informaticae*, 75 (2007), 263–280.
6. T.-O. Ishdorj, A. Leporati: Uniform solutions to SAT and 3-SAT by spiking neural P systems with pre-computed resources. *Natural Computing*, 7 (2008), 519–534.
7. Gh. Păun: Membrane computing at twelve years. Back to Turku. *Proc. UC 2011*, Turku, Finland, June 2011, LNCS 6714, Springer, 2011, 36–37.
8. Gh. Păun: Towards "fypercomputations" (in membrane computing), LNCS 6714, Springer, 2011, 36–37.

9. V. Putz, K. Svozil: Can a computer be "pushed" to perform faster-than-light? *Proc. UC10 Hypercomputation Workshop "HyperNet 10"*, Univ. of Tokyo, June 22, 2010.
10. A. Syropoulos: *Hypercomputation: Computing Beyond the Church-Turing Barrier.* Springer, 2008.

14 Numerical P Systems

Cristian Ioan 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

Extensions of numerical P systems motivated by using such systems in robot controlling are mentioned and problems occurring in this framework are formulated.

Required Notions: numerical P system, complexity, promoters-inhibitors, catalyst

Numerical P systems are a class of computing models (introduced in [5]; see also Chapter 23.6 of [6]) 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 systems were recently used in a series of papers (see references in [1]) for implementing controllers for mobile robots; 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, [2], [3], [4]. Such systems correspond to *catalytic P systems*

in the "general" membrane computing: a program is applied only if the value of the associated enzyme is strictly greater than the smallest value of any variable involved in the production polynomial. Enzyme variables are not consumed or produced by the rules which they catalyze, but can be changed by the rules for which they do not act as catalysts. Therefore, their values can evolve during the computational process.

Tissue numerical P systems are also considered in [8], with parallel use of programs. If in each membrane, at each step, we use a maximal set of programs (programs are selected nondeterministically, 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 (nondeterministically) 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 nondeterministic, (3) sequential, all-parallel, one-parallel, (4) used in the generating, computing, accepting mode; further variants can be added. By combining all these, a plethora of classes of numerical P systems appear.

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 Π 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 is accepted, then we remove the superscript $+$.) When tissue systems are used, we write $NtP_m(poly^n(r), \alpha, \beta)$.

Here are a few results from [5] and [8].

Theorem 1. $NRE = N^+P_8(poly^5(5), seq) = N^+P_7(poly^5(6), seq) = NP_7(poly^5(5), enz, seq) = NtP_*(poly^1(11), enz, oneP) = NP_{254}(poly^2(253), enz, allP, det)$.

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 P systems) be used also in the cell-like case?

Similar extensions of "general" notions in membrane computing to numerical P systems remain to be 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?

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, a natural research topic is to further explore the use of numerical P systems in controlling robots. In this framework, an important question is to develop a complexity theory based on numerical P systems: define specific complexity classes, compare them with existing classes, look for ways to speed-up computations (see also the previous suggestion, to bring to numerical P systems further ideas investigated for symbol object P systems, in particular, tools to create an exponential working space in polynomial time).

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. C. Buiu, C.I. Vasile, O. Arsene: Development of membrane controllers for mobile robots. *Information Sciences*, 187 (2012), 33–51.
2. A.B. Pavel, C. Buiu: Using enzymatic numerical P systems for modeling mobile robot controllers. *Natural Computing*, in press, DOI: 10.1007/s11047-011-9286-5, 2011
3. 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.
4. A.B. Pavel, C.I. Vasile, I. Dumitrache: Robot localization implemented with enzymatic numerical P systems. Submitted, 2012
5. Gh. Păun, R. Păun: Membrane computing and economics: Numerical P systems. *Fundamenta Informaticae*, 73 (2006), 213–227.
6. Gh. Păun, G. Rozenberg, A. Salomaa, eds.: *The Oxford Handbook of Membrane Computing*. Oxford Univ. Press, 2010.
7. The P Systems Web Page: `http://ppage.psystems.eu`.
8. C.I. Vasile, A.B. Pavel, I. Dumitrache, Gh. Păun: On the power of enzymatic numerical P systems. Submitted, 2011.

15 P Systems Formal Verification and Testing

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

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

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

Issues related to formal verification (through model checking) and model-based testing of various classes of P systems are mentioned – especially,extensions of existing results are pointed out, either using new tools or tackling classes of P systems useful in various frameworks.

Required Notions: cell P system, active membrane, tissue P system, model checking, model based testing, P-lingua

In the last years, more complex applications of P systems have been built and used to study the behaviour of various systems in biology, economics and linguistics. Models based on P systems have been introduced to investigate problems in distributed computing and process synchronisation. All these applications and models allow to simulate the systems studied and to identify their various properties. It is important that all these applications are correctly implemented and produce the right results. In order to check their correctness, formal verification and testing methods and tools are employed. Formal verification based on model checking and model-based testing will be presented below.

15.1 Formal verification of P systems through model checking

Model checking is an automated technique for verifying if a state-based model of a system meets a given specification. Using a temporal logic formula searches through the entire state space to check whether the property holds or fails are executed. If a property violation is discovered, then a counterexample is returned [3]. 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 [8], Spin [9] or ProB [7]. The decidability of model checking properties for P systems has also been studied in [4]

Most research has focussed on cell P systems with a static structure, but, more recently, P systems with active membranes, in particular with division rules, have also been investigated [10]. 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. However, the state explosion problem normally associated with model checking still hampers such approaches,

in particular in the case of P systems with cell division, for which the number of states can grow exponentially. To address this, more efficient implementations, as well as ways of reducing the number of states through the construction of *approximate* state-based models (possibly using inference techniques such as that presented in [11]) might be investigated.

The following problems regarding the formal verification of P systems are expected to be investigated:

- new methods and techniques for formally verifying variants of P systems with a dynamical structure used to model systems solving problems from computer science or engineering;
- identification of invariants using Daikon, a tool which dynamically detects program properties based on execution traces;
- developing an integrated environment for specifying and formally verifying P systems using P-lingua, Daikon, and one or more model checking tools;
- extending the existing approaches to other classes of P systems (e.g., tissue P systems).

15.2 Model-based testing of P systems

Testing is the main means of software validation and an essential part of system development; all software applications, irrespective of their use and purpose, are tested before being released. In testing, programs are run on a set of sample data in order to expose faults in the code. This means that an essential (and in many cases the most time consuming) part of testing is selecting such sample data. This process is called *test generation*. As in the last years there have been significant developments in using the P systems paradigm to model, simulate and formally verify various systems (in biology, economics, linguistics, graphics, computer science, etc.), test generation methods for systems modelled as P systems must also exist.

In the last years, a number of approaches to testing P systems have been developed. One approach involves the definition of a number of coverage criteria (such as simple rule coverage, in which each rule of the P system must be covered at least once, and more complex variants) and the selection of test data to meet these criteria [5]. An extension of this strategy involves mutation analysis: here, a test selection criterion is defined by producing a slightly modified version of the system (called a mutant) and the selected test data must distinguish between the original model and the mutant [8]. Another approach to P system testing is based on finite state machine conformance techniques [6]. This involves the construction of a state-based approximation of the P system (called a deterministic finite cover automaton) and the application of conformance testing techniques for such a finite state model [12].

Essentially, all the aforementioned techniques have been developed in the context of cell P systems with a fixed structure. The challenge for the future is to

extend these to P systems with active membranes, as well as to 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 behaviour of interactive systems, is expected, and may have an important practical impact. Ultimately, suitable tools will have to be developed and integrated within the aforementioned modeling, verification and testing.

Acknowledgement. This work was partially supported by project MuVet, Romanian National Authority for Scientific Research (CNCS UEFISCDI) grant number PN-II-ID-PCE-2011-3-0688.

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. Workshop on Membrane Computing*, WMC 2007, LNCS 4860, Springer, 2007, 138–159.
3. E.M. Clarke, O. Grumberg, D.A. Peled: *Model checking*. MIT Press, Cambridge, MA, USA, 1999.
4. Z. Dang, O.H. Ibarra, C. Li, G. Xie: On the decidability of model-checking for P systems. *Journal of Automata, Languages and Combinatorics* 11 (2006), 279–298.
5. M. Gheorghe, F. Ipate, C. Dragomir: Formal verification and testing based on P Systems. *Int. Workshop on Membrane Computing*, WMC 2009, LNCS 5957, Springer, 2010, 54–65.
6. F. Ipate, M. Gheorghe: Finite state based testing of P systems. *Natural Computing*, 8 (2009), 833–846.
7. F. Ipate, A. Ţurcanu: Modelling, verification and testing of P systems using Rodin and ProB. *Ninth Brainstorming Week on Membrane Computing*, 2011, 209–220.
8. F. Ipate, M. Gheorghe, R. Lefticaru: Test generation from P systems using model checking. *Journal of Logic and Algebraic Programming*, 79 (2010), 350–362.
9. F. Ipate, R. Lefticaru, C. Tudose: Formal verification of P systems using Spin. *International Journal of Foundations of Computer Science*, 22 (2011), 133–142.
10. 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, Fontainebleau, 2011, 241–252.
11. F. Ipate: Learning finite cover automata from queries. *Journal of Computer and System Sciences*, 78 (2012), 221–244.
12. F. Ipate: Bounded sequence testing from deterministic finite state machines. *Theoret. Comput. Sci.*, 411 (2010), 1770–1784.

16 Causality, Semantics, Behavior

Oana Agrigoroaiei, Bogdan Aman, Gabriel Ciobanu

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

The connection of MC with process algebra is not very much explored, although this is a very promising research direction. Some issue related to causality, behavior equivalence, type systems, relationships with the Chemical Abstract Machine are mentioned here.

Required Notions: cell P system, maximal parallelism, synchronization, semantics, event, (bi)simulation

16.1 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 works 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 solve the problems proposed above.

16.2 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 premises and are purely local.

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

16.3 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.

P systems consider cells as mechanisms working in a maximal parallel and nondeterministic 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 $\mathbf{N_2}$).

16.4 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 it 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*, 79 (2010), 278–288.
2. O. Agrigoroaiei, G. Ciobanu: Rule-based and object-based event structures for membrane systems. *Journal of Logic and Algebraic Programing*, 79 (2010), 295–303.
3. O. Agrigoroaiei, G. Ciobanu: Quantitative causality in membrane systems. *Proc. Twelfth International Conference on Membrane Computing*, Fontainebleau, 2011, 53–63.
4. B. Aman, G. Ciobanu: Typed membrane systems. *Int. Workshop on Membrane Computing*, WMC 2009, LNCS 5957, Springer, 2010, 169–181.
5. G. Berry, G. Boudol: The chemical abstract machine. *Theoretical Computer Science*, 96 (1992), 217–248.
6. I. Eidhammer, I. Jonassen, W. Taylor: Structure comparison and structure patterns. *Journal of Computational Biology*, 7 (2000), 685–716.
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*, 18 (2011), 5699–5704.

8. Gh. Păun: Some open problems collected during 7th BWMC. *Proc. Seventh Brainstorming Week on Membrane Computing*, 2009, vol. 2, 197–206.
9. B. Russell: *The Principles of Mathematics*, vol. I, Cambridge University Press, 1903.
10. J. Wells: The essence of principal typings. LNCS 2380, Springer, 2002, 913–925.

17 Kernel P Systems

Marian Gheorghe

Department of Computer Science, University of Piteşti, Romania

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

The issue of a common generalization of several classes of P systems is proposed, and some basic ideas towards such a goal are presented.

Required Notions: tissue P system, P system with dynamic structure, regular expression

Different variants of P systems have been used for specifying simple algorithms [5, 2], classes of **NP**-complete problems [7] and various applications [6]. More specific classes of P systems have been recently considered for modeling some distributed algorithms and problems [8]. In many cases the evolution of the system investigated requires some specific behavior or the use of certain rules, maybe with some constraints, which are not always the same as the ones exhibited by the model in its initial definition. It helps in many cases to have some flexibility with the modeling approach, especially in the specification stage, as it shortens the model and makes it clearer. Although there is a powerful specification language, called P-lingua, with implementations for various variants of P systems [9], there is no unified framework that allows us to simulate, verify and test the behavior of the specified systems. In this respect, it is suggested here a *kernel P system* (*kP system*, for short) that, in the first stage, will be a low level specification language including the most used concepts from P systems.

The generic structure of a kP system might be a graph-like structure as in tissue P systems. Such a model uses a set of symbols, labels of membranes, rules of various types and a certain strategy to run them against the multiset of objects available in each region. The rules in each compartment will be of two types: (i) *object processing rules* which transform and transport objects between compartments or exchange objects between compartments and environment, and (ii) *system structure rules* responsible for changing the system's topology. Each rule has a guard, defined using activators and inhibitors in a general way. The execution strategy is defined such that maximally parallel or sequential manners are captured

and each compartment has its own strategy. Rewriting and communication rules based on promoters and inhibitors are considered together with a special set of symport/antiport rules. Additional features like membrane division, creation, dissolution, bond creation and destruction are used to deal with the system structure.

The key concept of a compartment is first introduced and then the definition of a kP system.

Definition 1. *Given an alphabet A, of elements named* objects*, and an alphabet L of* labels*, a* compartment *is a tuple $C = (l, w_0, R^\sigma)$, where $l \in L$ is the label of the compartment, w_0 is the initial multiset over A, and R^σ denotes "the DNA code", i.e., the set of rules, denoted R, applied in this compartment and a regular expression, σ, over $Lab(R)$, the labels of the rules of R.*

Definition 2. *A* kernel P system *is a tuple $k\Pi = (A, L, IO, \mu, C_1, \ldots, C_n)$, where A and L are, as in Definition 1, the* alphabet *of objects and the set of* labels*, respectively; IO is a multiset of objects from A, called* environment*; μ defines the membrane structure, which is a graph, (V, E), where V are vertices, $V \subseteq L$ (the nodes are labels of these compartments), and E edges; $C_1, \ldots, C_n$ are the n compartments of the system – the inner part of each compartment is called the* region *which is delimited by a* membrane*; the labels of the compartments are from L and initial multisets are over A.*

We first discuss various types of rules. It is assumed that the rules below belong to the same compartment, C_i, labeled l_i. Each rule might have a regular expression associated with. When a rule involves more than a compartment, then each compartment might have its own regular expression attached to it. $RE(A \cup \bar{A})$ denotes the set of regular expressions over $A \cup \bar{A}$; each such regular expression defines conditions involving promoters, elements from A, and/or inhibitors, elements from $\bar{A}$. The interpretation of a regular expression $g \in RE(A \cup \bar{A})$, associated with a rule, is that all the promoters appearing in g must be present in the current multiset and none of the inhibitors must appear there. We call this regular expression, g, *guard*. A rule with such a guard is applicable when this is evaluated to true.

A rule can have one of the following types:

- **rewriting and communication** rule: $x \to y\ \{g\}$, where $x \in A^+$, $y \in A^*$, $g \in RE(A \cup \bar{A})$; the right hand side, y, has the form $y = (a_1, t_1) \ldots (a_h, t_h)$, where $a_j \in A$ and $t_j \in L$, $1 \leq j \leq h$, is an object and a target (i.e., the label of a compartment), respectively; the target t_j must be either the label of the current compartment, l_i (more often ignored) or of an existing neighbor of it ($(l_i, t_j) \in E$) or an unspecified one, $*$; otherwise, the rule is not applicable; if a target t_j refers to a label that appears more than once, then one of the involved compartments will be nondeterministically chosen; if t_j is $*$, then the object a_j is sent to a compartment arbitrarily chosen;
- **input-output** rule, is a form of symport/antiport rule: $(x/y)\ \{g\}$, where $x, y \in A^*$, $g \in RE(A \cup \bar{A})$; x from the current region, l_i, is sent to the environment and y from the environment is brought into the current region;

- **system structure rules**; the following types are considered:
  - **membrane division** rule: $[]_{l_i} \to []_{l_{i_1}} \dots []_{l_{i_h}}\ \{g\}$, where $g \in RE(A \cup \bar{A})$; the compartment l_i will be replaced by h compartments obtained from l_i, i.e., the content of them will coincide with that of l_i; their labels are $l_{i_1}, \dots, l_{i_h}$, respectively; all the links of l_i are inherited by each of the newly created compartments;
  - **membrane dissolution** rule: $[]_{l_i} \to \lambda\ \{g\}$; the compartment l_i will be destroyed together with its links;
  - **link creation** rule: $[]_{l_i}; []_{l_j} \to []_{l_i} - []_{l_j}\ \{cg\}$; the current compartment l_i is linked to l_j and, if more than one l_j exists, then one of them will be nondeterministically picked up; cg, called compound guard, describes an expression $l_i.g_1\ op\ l_j.g_2$, where g_1, g_2 are regular expressions referring to compartments l_i and l_j, respectively; op is either *and* or *or*, standing for either both guards are true or at least one is true. If one of the guards is empty then op is no longer used; a compound guard defines a Boolean condition across the two compartments;
  - **link destruction** rule: $[]_{l_i} - []_{l_j} \to []_{l_i}; []_{l_j}\ \{cg\}$; this is the opposite of link creation and means that compartments l_i, l_j are disconnected; as usual, when more than a link, $(l_i, l_j) \in E$, exists, then only one is considered by this rule; cg is a compound guard.

  The usual behavior of P systems requiring that rewriting and communication, and symport/antiport (input-output) rules are applied in a maximal parallel way (or sequentially in some cases), whereas membrane division, creation, dissolution rules and creation and destruction of links are executed one per membrane, will be considered in this context as well, but in a rather more general way.

  The main challenges of this approach are

1. a rigorous definition of the syntax and semantics of kP systems;
2. comparisons between (fragments) of kP systems and well-known variants of P systems;
3. establishing general algorithms to translate different classes of P systems into kP systems (similar to [1, 4]);
4. defining operational semantics for kP systems and providing implementations in model checkers, like Spin, Maude, similar to [3].

Further steps in developing this unified framework might consist of adding other useful modeling features like the possibility of defining rules and compartments using indexes, a certain concept of a module, various other semantics. It is intended to keep the kernel system as generic as possible such that some of the above mentioned extensions will be introduced in a rather syntactical manner allowing to map them into the basic variant, without additional semantics.

Acknowledgement. This work was partially supported by project MuVet, Romanian National Authority for Scientific Research (CNCS, UEFISCDI) grant number PN-II-ID-PCE-2011-3-0688.

References

1. A. Alhazov, L. Pan, Gh. Păun: Trading polarizations for labels in P systems with active membranes. *Acta Informatica*, 41 (2004), 111–144.
2. A. Alhazov, D. Sburlan: Static sorting P systems. In [6], 2006, 215–252.
3. O. Andrei, G. Ciobanu, D. Lucanu, A rewriting logic framework for operational semantics for membrane systems. *Theoretical Computer Sci.*, 373 (2007), 163–181.
4. R. Barbuti, A. Maggiolo-Schettini, P. Milazzo, S. Tini: Membrane systems working in generating and accepting modes: Expressiveness and encodings. *Membrane Computing, 11th International Conference, CMC2010, Jena, Germany, August 2010* (M. Gheorghe et al., eds), LNCS 6501, Springer, 2011, 103–118.
5. R. Ceterchi, C. Martín-Vide: P systems with communication for static sorting. *Pre-Proc. Brainstorming Week on Membrane Computing, Tarragona, February 2003* (M. Cavaliere et al., eds.), Technical Report no 26, Rovira i Virgili Univ., Tarragona, TR 26/03, URV, 2003, 101–117.
6. G. Ciobanu, Gh. Păun, M.J. Pérez-Jiménez, eds.: *Applications of Membrane Computing*, Springer, 2006.
7. D. Díaz-Pernil, M.A. Gutiérrez-Naranjo, M.J. Pérez-Jiménez: A uniform family of tissue P systems with cell division solving 3-COL in a linear time. *Theoretical Computer Science*, 404 (2008), 76–87.
8. R. Nicolescu, M.J. Dinneen, Y.-B. Kim: Structured modelling with hyperdag P systems. Part A. *Membrane Computing, Seventh Brainstorming Week, BWMC 2009, Sevilla, Spain, February 2009* (A.R. Gutiérrez-Escudero at al., eds.), Universidad de Sevilla, 2009, 85–107.
9. The P-lingua Website: `http://www.p-lingua/wiki/index.php/Main_Page`.

18 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`

Some possibilities of bridging MC (P systems) and reaction systems are discussed, the basic idea being of importing ideas from a research area to another one.

Required Notions: cell P system, multiset, reaction system, halting, fypercomputing

Reaction systems (we call them *R systems*) form a recently introduced research area aiming to model the evolution of (bio)chemicals by means of (bio)reactions,

in a framework based on the following two fundamental assumptions (we recall them 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.

With these postulates in mind, let us consider first some possibilities of passing from R systems to P systems.

Moving from multisets, which are basic in P systems, to sets (actually, to multisets with an infinite multiplicity of their elements, called ω multisets in Section 13) 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.

However, as we have mentioned in Section 13, considering P systems with ω multisets leads in an easy way to fypercomputations.

The second assumption of the reaction systems theory (no permanence of objects) looks easier to handle in terms of MC. The immediate idea is to simply remove (by a "deletion rule") 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 MC. 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* (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 possibilities were considered in various papers in MC.

Part of these possibilities are checked in [7] from where we recall the following result:

Theorem 2. *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.

The case of defining the result of a computation of symport/antiport P systems by means of signals – objects or events – also 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 2 [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?

The R systems area has a series of notions of the dynamical systems type which were not too much investigated for P systems (time, events, modules, structure, causality, and so on), and this is also a promising direction of research.

Let us now briefly explore the other direction, from P to R.

The R systems are not meant to define computations, their behavior is deterministic, from a set of symbols we precisely pass to a unique set of symbols. However, starting from an R system, a "generative device" can be defined, based on passing from a configuration to another one (without input from the environment), provided that some nondeterminism is introduced in the R system functioning. Three possibilities of this kind were proposed in [7]: (i) working with tabled R systems, as in Lindenmayer systems (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 k 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, threshold k), possible hierarchies.

Of course, a general research topic is to find other ways of building a (string or graph) computing device in terms of R systems. A possible question is also the possibility to introduce membranes in the R systems area or other MC ingredients – thus getting a sort of PR systems. (An attempt of this kind is reported in [5], where so-called reaction automata are introduced, but these devices violates both postulates of R systems and use so many ingredients of P systems – multisets, parallelism, nondeterminism, halting – that they are just P automata with a new name.)

References

1. A. Ehrenfeucht, G. Rozenberg: Basic notions of reaction systems, *Proc. DLT 2004* (C.S. Calude et al., 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. F. Okubo, S. Kobayashi, T. Yokomori: On the properties of languages classes defined by bounded reaction automata. *Theoretical Computer Sci.*, in press.
6. Gh. Păun: Towards fypercomputations (in membrane computing). LNCS, Springer, to appear.
7. Gh. Păun, M.J. Pérez-Jiménez: Towards bridging two cell-inspired models: P systems and R systems. *Theoretical Computer Sci.*, to appear.

19 P Systems and Evolutionary Computing Interactions

Gexiang Zhang

School of Electrical Engineering
Southwest Jiaotong University, Chengdu, P.R. China
zhdxdylan@126.com

Problems related to the so-called membrane algorithms (actually, distributed evolutionary computing, with the distribution controlled by means of membranes, as well as with other MC ingredients used) are mentioned, both in the direction of improving the optimization techniques and in looking for more complex/practical applications.

Required Notions: evolutionary computing, cell P system, active membranes, membrane algorithm

As a relatively young branch of natural computing, MC has gone through thirteen years of intensive research involving areas of theoretical computer science as well as applications in various fields, including systems biology, graphics, linguistics, parallel and distributed computing. However, these applications, in terms of varieties and types, are relatively small compared to a very broad range of applications of evolutionary computing. A natural question would be, whether some combinations of these two models might benefit from the large scope of applications evolutionary computing has already shown so far, and the rigorous and sound theoretical development membrane systems have proved for all its variants.

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

Membrane-inspired evolutionary algorithms (MIEAs): Since membrane computing was initiated in 1998, a large number of theoretical results, such as various variants of membrane systems and their computational power and efficiency came forth [1]. On the one hand, the way MC is extended into real-world applications is not easy to be addressed and represents an ongoing issue. On the other hand, the hybridization of different computing techniques is an attractive research topic in the area of evolutionary computing, due to a better performance than their counterpart approaches. What can the young paradigm of MC bring to evolutionary computation? Fortunately, MIEAs, formerly called membrane algorithms [2, 3], create a bridge between MC and various real-world applications. 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 [3, 4]. The comparative analysis of dynamic behaviors of an instance of MIEAs shows the appropriate combination of MC and evolutionary computation can produce a better capability to balance exploration and exploitation [5], which are two contradictory factors directly related to the performance of an optimization algorithm. Until now, MIEAs have been studied in conjunction with cell P systems with a fixed membrane structure and by considering an evolutionary computing approach as a subalgorithm put inside a membrane [1, 6, 7]. Further research topics are listed below.

1. Consider further combinations of features that make full use of the characteristics of both MC models and evolutionary computing, such as the consideration of cell P systems with active membranes, tissue P systems and population P systems.
2. Usually, in an MIEA an evolutionary algorithm is used as a subalgorithm placed inside a membrane. This idea can be extended. A membrane structure can be used as a framework of the organization of several different types of evolutionary operators, as shown in [8], or several distinct kinds of evolutionary mechanisms, such as a genetic algorithm, evolutionary programming, evolution strategy, differential evolution and particle swarm optimization. Furthermore, the flexible communication rules can be used at the level of genes, instead of at the level of individuals shown in [6, 4].
3. The single-objective problems are usually involved in the investigations reported in the literature. The framework of P systems can offer better population diversity in MIEAs, hence further work can turn to solve problems in a complex environment, such as multi-objective, dynamic, peaked optimization problems, and with/without constraints, to check whether P systems can bring a better performance to evolutionary algorithms.
4. More real-world application problems, such as power system optimization, software/hardware co-design and vehicle route plan, can be solved by using MIEAs.

5. A deep performance analysis and evaluation of MIEAs is necessary to reveal the roles of P systems played in the hybrid optimization algorithms, on the basis of the previous work [5].

Automated design of membrane computing models (ADMCMs): The automated synthesis of some types of MC 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 [9]. This is quite a complex problem as it involves a great number of parameters (rules, objects, combination of rules) and many semantics associated with P systems.

Membrane evolutionary algorithms (MEAs): MEAs will focus on implementing evolutionary algorithms within a P system environment in order to take advantage of the parallelism and distribution of MC, given that recent investigations are studying the implementation of P 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.

Acknowledgement. This work was supported by the National Natural Science Foundation of China (61170016), the Program for New Century Excellent Talents in University and the Project-sponsored by SRF for ROCS, SEM.

References

1. Gh. Păun, G. Rozenberg, A. Salomaa, eds.: *The Oxford Handbook of Membrane Computing*. Oxford University Press, 2010.
2. T.Y. Nishida: Membrane algorithm with brownian subalgorithm and genetic subalgorithm. *International Journal of Foundations of Computer Science*, 18 (2007), 1353–1360.
3. G.X. Zhang, C.X. Liu, H.N. Rong: Analyzing radar emitter signals with membrane algorithms. *Mathematical and Computer Modelling*, 52 (2010), 1997–2010.
4. G.X. Zhang, J.X. Cheng, M. Gheorghe: A membrane-inspired approximate algorithm for traveling salesman problems. *Romanian Journal of Information Science and Technology*, 14 (2011), 3–19.
5. G.X. Zhang, C.X. Liu, M. Gheorghe: Diversity and convergence analysis of membrane algorithms. *Proc. of the Fifth International Conference on Bio-Inspired Computing: Theories and Application*, 2010, 596–603.
6. G.X. Zhang, M. Gheorghe, C.Z. Wu: A quantum-inspired evolutionary algorithm based on P systems for Knapsack Problem. *Fundamenta Informaticae*, 87 (2008), 93–116.
7. J.X. Cheng, G.X. Zhang, X.X. Zeng: A novel membrane algorithm based on differential evolution for numerical optimization. *International Journal of Unconventional Computing*, 7 (2011), 159–183.
8. G.X. Zhang, M. Gheorghe, Y.Q. Li: A membrane algorithm with quantum-inspired subalgorithms and its application to image processing. *Natural Computing*, 2012, DOI: 10.1007/s11047-012-9320-2. (published online)

9. X.L. Huang, G.X Zhang, H.N. Rong, F. Ipate: Evolutionary design of a simple membrane system. *Proc. CMC 2011, Fontainebleau, France, August 2011* (M. Gheorghe et al., eds.), LNCS 7184, Springer, 2012, 203–214.

20 Metabolic P Systems

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

A dynamical inverse problem for MP systems is formulated, then a possible topological extension of P systems is suggested.

Required Notions: Multiset, multiset rewriting systems, P systems, discrete dynamical systems, metabolism, function approximation.

20.1 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 multisets of objects and rules are internally placed. Rules transform and move objects among membranes. Metabolic P systems, shortly 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 MNC (Models of Natural Computing and Bioinformatics) group. A brief introduction on MP systems and references can be found in the Scholarpedia page "Metabolic P Systems".

One of the most recent results about MP systems was the discovery of a methodology for solving dynamical inverse problems, in the sense we are going to explain [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).

An MP grammar G is a "generator" of time series, determined by the 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 with 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*, the matter quantities consumed/produced by the rules in that state.

An MP graph is a natural graphical representation of G. An MP grammar becomes an MP system when values for the *time interval*, the *population unit*, and the *metabolite masses* are added. An MP grammar G generates the (infinite) time series $(X[i] \mid i \in \mathbb{N})$, from the initial values I, 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:

$$X[i+1] - X[i] = \sum_{j=1}^{m} (\beta_j(X) - \alpha_j(X))\varphi_j(s[i]).$$

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 an 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 in terms of matrix tensor product), 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.

20.2 A (vague) topological problem

Membrane computing is based on the intuition of a membrane as a spatial entity closing a subspace (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 MC 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 happens at the level of morphogenesis in biological systems.

References

1. V. Manca, R. Lombardo: Computing with multi-membranes. *Proc. CMC 2011, Fontainebleau, France, August 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 stoichiometric stepwise regression for MP systems. *Int. J. Foundations of Computer Science*, 22 (2011), 97–106.
4. V. Manca, L. Marchetti: Solving dynamical inverse problems by means of metabolic P systems. *BioSystems*, 2012, doi:10.1016/j.biosystems.2011.12.006.
5. V. Manca: Metabolic P systems. *Scholarpedia*, 2011.

21 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
thomas.hinze@uni-jena.de

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

Issues arising from the use of P systems in modeling biological processes are discussed – especially concerning various circular evolutions of biological processes.

Required Notions: circadian rhythm, cellular dynamics, backtracking

21.1 Motivation

Endogenous oscillations have been identified to be essential for the function of numerous systems found in biology as well as engineering [1]. 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 [4].

Exploration of chronobiological systems emerges as a growing research field within bioinformatics focusing on various applications in medicine, agriculture, and material sciences [8]. Particularly, 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 [6]. 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.

There are numerous types of controllable core oscillators found in circadian clocks. The majority reveals the Goodwin type, a cyclic gene regulatory network composed of mutual activating and inhibiting gene expressions [8]. The most effective way to influence its frequency is modification of protein degradation rates. Furthermore, core oscillators can be of post-translational type [7], exploiting a cyclic scheme of protein phosphorylation, complex formation, or decomposition. Here, the involved catalysts affect the frequency. The third and most complex type of core oscillators includes *compartmental dynamics* [4] aimed to be advantageously modeled using P systems combining a representation of dynamical structures with tracing their spatiotemporal behavior.

21.2 Resulting Challenges

Within the domain of strictly continuous signals quantified by real numbers, modeling and analysis of oscillating behavior has been well-studied [1]. 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. 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. *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.

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* [5]. 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 [3]. 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 their effects to sets of molecular objects and local reactions and transportation rules (configuration update schemes).

21.3 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 [2, 5] 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. 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 dynamical structures.

References

1. J. Aschoff: A survey on biological rhythms. *Biological Rhythms*, 4 (1981), 3–10.
2. F. Fontana, V. Manca: Discrete solutions to differential equations by metabolic P systems. *Theoretical Computer Science*, 372 (2007), 165–182.
3. G. Gruenert, B. Ibrahim, T. Lenser, M. Lohel, T. Hinze, P. Dittrich: Rule-based spatial modeling with diffusing, geometrically constrained molecules. *BMC Bioinformatics*, 11 (2010), 307.
4. 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 (2009), 411–426.
5. T. Hinze, R. Fassler, T. Lenser, N. Matsumaru, P. Dittrich: Event-driven metamorphoses of P systems. *Proc. WMC 2009* (D. Corne et al., eds.), LNCS 5391, Springer, 2009, 231–245.
6. T. Hinze, C. Bodenstein, B. Schau, I. Heiland, S. Schuster: Chemical analog computers for clock frequency control based on P modules. *Proc. CMC 2011, Fontainebleau, France, August 2011* (M. Gheorghe et al., eds.), LNCS 7184, Springer, 2012, 182–202.
7. 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 (2007), 841–853.
8. V.K. Sharma, A. Joshi: Clocks, genes, and evolution. The evolution of circadian organization. *Biological Rhythms* (V. Kumar, ed.), Springer, 2002, 5–23.

22 Simulating Cells Using P Systems

Andrei Păun

Department of Computer Science, Louisiana Tech University, Ruston, USA

Bioinformatics Department, National Institute of Research and Development for Biological Sciences, Bucharest, Romania
apaun@latech.edu, apaun@fmi.unibuc.ro

Possibilities and difficulties encountered in (Ruston group) research on modeling biological processes by means of P systems are discussed, with emphasis on complexity, noise, implementations.

Required Notions: P system models, Gillespie's algorithm, mass action law

To achieve a greater understanding of the biological processes the technology will need to improve and evolve from the current state of "big populations" to discrete events. By big populations we mean the following fact: to be able to perform a specific experiment, the researcher needs a large number of "elements" (say cells) in the same state that is investigated and only then the experiment can reach a

conclusion. Once the number of "elements" in the experiment is decreasing, most of our methods to investigate properties of those "elements" become hard/impossible to describe/investigate. Obviously this statement is rather broad and there are some techniques such as FRET analysis that look at discrete events/elements, but we claim that the majority of the current bio-molecular techniques do require large multiplicities of the "element" investigated.

The aforementioned fact has to be understood by the researcher looking to model/simulate cells. It describes the state of the research tools in that area. The modeler can help that particular area by offering better insight into the sub-cellular processes through simulation and prediction. One could immediately point out that since we have a "technological" problem (as stated before) which is precluding us to gain insight into the "discrete" processes, then how can one hope to simulate the sub-cellular mechanisms. The answer is two-fold: (1) cells prove to respond mostly in the same fashion to similar stimuli, meaning that the inherent stochasticity of these systems does not "break" the response pathways (making the simulation from this perspective "easy" as we need to simulate the "important" events, not all the noise associated with the gene regulation mechanisms and their stochasticity); (2) even if we do not know a mechanism, once a model is built based on our best knowledge and we see it diverging from reality in a specific point, we know where to start investigating for other processes/reactions.

There is also a philosophical motivation to using P systems for a cell simulator: P systems were defined to capture the compartmentalized structure of the eukaryotic cells, and indeed this compartmentalization could prove one of the best features of a cell simulator. Furthermore: due to the current biomolecular techniques involving large multiplicities for a species the simulation techniques in the area focussed on ordinary differential equations (ODE) as continuous mathematics both has powerful tools and are easily implemented. But we claim that a continuous mathematics approach in this area of sub-cellular simulation may not be the best approach as some processes have been seen to behave discretely, and in several pathways we can see the multiplicity of some multiprotein complexes appear in very small numbers (below 10). In such cases a discrete simulation technique such as Gillespie's algorithm would be preferable to the simulators based on ODE [4].

Incidentally we have also defined a discrete simulation technique in [2] which was repeatedly improved (see references in [5]) and was lately named NWA with memory. The motivation behind the NWA algorithm was simple: we wanted a discrete mathematics based simulation technique that would be faster than Gillespie's algorithm.

22.1 Brief Description for Current Cellular Models and Simulators

In order to plausibly model the biochemistry of life, individual biochemical interactions need to occur asynchronously over different lengths of time. The model relies on the *law of mass action*. The law states that reaction rate is directly proportional to the number of reactants available in the system. In other words, the time

required to execute a rule in the cell is dependent on the number of its reacting species. We note that the rule application is not considered to be instantaneous; the kinetics that are giving the reaction speed model the time required by the molecules involved in the rule to couple together (if the reaction is of second order or higher) as well as the time required for the actual reaction to take place.

The law of mass action gives us the power to temporally describe the evolving configurations of our system. To understand the asynchrony of rule execution, we need to discuss the *kinetic rates* pertaining to the law of mass action. The kinetics of a chemically reactive system are often described as concentration-based values. This is common for the types of experiments used to derive the rates, typically involving enormous populations (millions) of cells. The cells are often lysed as a large population, molecules are measured in terms of light intensity and data are given as concentrations of species across cell population. These values can be averaged across the cell population, yielding concentrations per cell. We rely on these values to fit our models, but the values are derived from entire cell populations instead of individual cells. Hence, the interesting phenotypic, biochemical and physiological characteristics of individual cells can be sometimes overgeneralized (or lost) in lieu of the behavior of the majority of the cells in the population.

Some labs employ techniques to measure single-cell dynamics. For example, interesting results/models on p53 have been reported in [7], where it is shown that individual cells undergo not dampened oscillations, as reported in [1], but each individual cell instead exhibits a different numbers of oscillations. The average behavior for the cell population appeared to be dampened, but individual cells did not behave this way.

We are collaborating with Mark DeCoster's biomedical laboratory from Louisiana Tech University in order to study single cell data via a high-speed imaging system. It is our hope that future collaborations will help unlock some of the secrets behind Fas-induced apoptosis. Regardless of whether data comes from large cell populations or single cell dynamics, we, as modelers, must remain vigilant and build the best models with the data available to us.

Using the law of mass action and discrete kinetic constants we can define the *Waiting Time* (WT) of a reaction in the P system. The WT is a value assigned to each reaction, signifying the next timepoint for a single execution of the reaction. As molecular multiplicities will change throughout a simulation, from one configuration to the next, so will the WTs of reactions utilizing those molecules.

We used a min-heap for sorting reactions, where the top of the heap is the reaction with the smallest WT – i.e., the next reaction to be executed. However, we need to use nonstandard methods for maintaining the heap, due to the asynchrony of the rules and the sharing of reactants. These nonstandard methods are similar to those proposed by Gibson and Bruck [3] in their modification to the Gillespie algorithm.

To clarify, when a rule is applied, multiple nodes can have changes to their WT, since the multiplicities of particular species of the system have changed. These species can be shared over multiple reactions. Hence, multiple WT potentially

can fail the min-heap property throughout the tree simultaneously at each new configuration. In order to handle this, we use heap maintenance methods similar to those proposed by Gibson and Bruck [3] in their modification of the Gillespie algorithm.

22.2 Improving the Simulators

The following "open problems" are mostly for the simulator developed by our group but should be relevant for other simulators as well.

1. increase the stochasticity at the level of the heap by applying a modified Monte Carlo simulation technique for the first 3 levels of the heap (the fastest 7 reactions),

2. faster implementation such as using C rather than C++ or Java,

3. GPU implementation of the simulator for parallel simulations and identifying "decision points" in the pathway; also running the same model several times could identify the minority from the majority (this information could be lost in an ODE framework for simulation),

4. bigger and better models for sub-cellular mechanisms,

5. using Manca's Log-gain theory to gather stoichiometric data to be used in simulations [6],

6. implementing the simulation framework as a plug-in in CoPasi for broader dissemination and usage.

Acknowledgements. The author acknowledges support from UEFISCDI – PNII-TE 92/2010.

References

1. R.L. Bar-Or, R. Maya, L.A. Segel, U. Alon, A.J. Levine, M. Oren: Generation of oscillations by the p53-Mdm2 feedback loop: A theoretical and experimental study. *Proc. Natl. Acad. Sci., USA*, 97 (2000), 11250–11255.
2. S. Cheruku, A. Păun, F.J. Romero-Campero, M.J. Pérez-Jiménez, O.H. Ibarra: Simulating FAS-induced apoptosis by using P systems. *Progress in Natural Science*, 17 (2007), 424–431.
3. M.A. Gibson, J. Bruck: Efficient exact stochastic simulation of chemical systems with many species and many channels. *Journal of Physical Chemistry A*, 104 (2000), 1876–1889.
4. D.T. Gillespie: Exact stochastic simulation of coupled chemical Reactions," *The Journal of Physical Chemistry*, vol. 81, no. 25, 1977, pp. 2340–2361.
5. J. Jack, A. Păun: Discrete modeling of biochemical signaling with memory enhancement. *LNBI Transactions on Computational Systems Biology*, 5750 (2009), 200–215.
6. V. Manca: The metabolic algorithm for P systems: Principles and applications. *Theoretical Computer Science*, 404 (2008), 142–155.
7. G. Lahav, N. Rosenfeld, A. Sigal, N. Geva-Zatorsky, A.J. Levine, M.B. Elowitz, U. Alon: Dynamics of the p53-Mdm2 feedback loop in individual cells. *Nature Genetics*, 36 (2004), 147–150.

23 P Systems for Computational Systems and Synthetic Biology

Marian Gheorghe[1,2], **Vincenzo Manca**[3], **Francisco-José Romero-Campero**[4]

[1]Department of Computer Science, University of Piteşti, Romania

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

[3]Department of Computer Science, University of Verona, Italy
vincenzo.manca@univr.it

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

Deterministic and stochastic P system models are discussed in the context of specifying fairly complex biological systems; their usage for systems and synthetic biology is also presented.

Required Notions: metabolic P systems, dynamical inverse problem, stochastic P systems, Gillespie algorithm, systems biology, synthetic biology

The approaches based on P systems aiming to provide coherent descriptions of fairly complex biological systems are either deterministic or stochastic [5]. Two such variants are discussed below, but some more variants of the above mentioned types of P systems are available in the current literature, see [15] and Sections 21 and 22 of this paper.

Metabolic P systems (MP systems for short) were introduced in 2004 as a particular kind of P systems devised for modeling metabolic processes [7]. Their main goal consists in solving dynamical inverse problems (DIPs) by means of discrete systems. A general algorithm, called Log-Gain Stoichiometric Stepwise Regression (LGSS), providing MP solutions to DIPs was obtained, in a systematic way, by integrating finite difference recurrent equations, least square method, stepwise regression, and related Fisher tests, within a suitable linear algebra framework where solutions can be expressed as ordinary and tensor products among matrices [11]. A MATLAB implementation of LGSS was developed by Luca Marchetti [12].

Many successful applications of MP theory to biological dynamics were developed, starting from classical examples (Lotka-Volterra, Brusselator, Mitotic Oscillator) [10]. Presently, the two main applications under investigation concern the insulin-glucose dynamics in diabetes pathologies and genetic expression in a kind of breast cancer (in cooperation with endocrinologists and clinicians in Italy, Verona and in USA, Detroit). A synthetic description and references is given by Vincenzo Manca in [8, 9].

Stochastic P systems, SP system for short, are rule-based discrete and stochastic multicompartmental systems used as abstract structures to model stochastic cellular systems [14]. The key difference between the original P systems and SP systems consists in a stochastic constant that is specifically associated with each rule. This constant is used to determine in a specific state or configuration of the system the probability of applying the corresponding rule and the time elapsed between rule applications according to Gillespie's stochastic simulation algorithm [6].

SP systems allow the incremental and parsimonious design of models by providing modelers with the feature of modularity explicitly [4]. A P system module consists of a finite set of rewriting rules that may contain some free variables in their objects, labels and stochastic constants. Modules can be arranged in libraries so they can be reused to define the rewriting rules of different models. In this respect, modules act like macros that get expanded once the corresponding module variables are instantiated with specific molecular species names, numerical values for the stochastic constants and compartment names.

A variant of SP systems, *lattice population P systems* [18], allow modelers to represent multi-cellular systems with specific geometries by distributing copies of given individual stochastic P systems over the points of a finite geometrical lattice.

SP systems have been implemented in the software tool for the specification, simulation, analysis and optimization of systems and synthetic biology models, Infobiotics workbench [3].

These systems have been used to model signal transduction pathways [13, 1], bacterial gene regulation [17], bacterial populations [16], metapopulations [2] and synthetic biology problems [19].

Membrane computing has made very significant contributions in certain areas of computer science and has produced some impact with respect to a number of applications. It remains a challenge to show how it copes with complex applications, especially in systems and synthetic biology. Some of these challenges are listed below:

- identify more complex systems to be specified by one of the variants of P systems described above or presented in [15];
- extend the current variants with additional features in order to cope with more complex applications;
- create a repository of illustrative biological case studies;
- develop additional complementary approaches that help analyzing biological systems – data sensitivity analysis, property data extraction and verification, hierarchies of languages allowing to map P system specifications into biochemical reactions;
- implement adequate tools exploiting the latest technologies and create benchmark problems to assess them.

Acknowledgement. M.G.'s work was partially supported by project MuVet, Romanian National Authority for Scientific Research (CNCS, UEFISCDI) grant number PN-II-ID-PCE-2011-3-0688.

References

1. D. Besozzi, P Cazzaniga, S. Cocolo, G. Mauri, D. Pescini: Modelling diffusion in a signal transduction pathway: the use of virtual volumes in P systems. *Int. J. Found. Comput. Sci.*, 22 (2011), 89–96.
2. D. Besozzi, P Cazzaniga, D. Pescini, G. Mauri: Modelling metapopulations with stochastic membrane systems. *BioSystems*, 91 (2008), 499–514.
3. J. Blakes, J. Twycross, F.J. Romero-Campero, N. Krasnogor: The Infobiotics Workbench: an integrated in silico modelling platform for Systems and Synthetic Biology. *Bioinformatics*, 27 (2011), 3323–3324.
4. H. Cao, F.J. Romero-Campero, S. Heeb, M. Cámara, N. Krasnogor: Evolving cell models for systems and synthetic biology. *Syst. Synth. Biol.*, 4 (2010), 55–84
5. M. Gheorghe, V. Manca, F.J. Romero-Campero: Deterministic and stochastic P systems for modelling cellular processes. *Natural Computing*, 9 (2010), 457–473.
6. D.T. Gillespie: Stochastic simulation of chemical kinetics. *Annual Review of Physical Chemistry*, 58 (2007), 35–55.
7. V. Manca: The metabolic algorithm for P systems: Principles and applications.*Theoretical Computer Science*, 404 (2008), 142–155.
8. V. Manca: Metabolic P systems.*Scholarpedia*, 6 (2010), 9273.
9. V. Manca: Fundamentals of metabolic P systems. Chapter 6 in [15], 475–498.
10. V. Manca L. Bianco, F. Fontana: Evolutions and oscillations of P systems: Theoretical considerations and application to biological phenomena. *Proc. WMC 2004, Milan, Italy, June 2004*, LNCS 3365, Springer, 2005, 63–84.
11. V. Manca, L. Marchetti: Solving dynamical inverse problems by means of metabolic P systems. *BioSystems*, to appear. DOI:10.1016/j.biosystems.2011.12.006.
12. L. Marchetti, V. Manca: A methodology based on MP theory for gene expression analysis. *Proc. CMC 2011, Fontainebleau, France, August 2011*, LNCS 7184, Springer, 2012, 300–313.
13. A. Păun, M.J. Pérez-Jiménez, F.J. Romero-Campero: Modeling signal transduction using P systems. *Proc. WMC 2006, Leiden, The Netherlands, July 2006*, LNCS 4361, Springer, 2006, 100–122.
14. Gh. Păun, F.J. Romero-Campero: Membrane computing as a modeling framework: cellular systems case studies. *Proc. Formal Methods for Computational Biology, 8th Intern. School*, Bertinoro, Italy, June 2008 (M. Bernardo et al, eds.), LNCS 5012, Springer, 2008, 168–214.
15. Gh. Păun, G. Rozenberg, A. Salomaa, eds.: *The Oxford Handbook of Membrane Computing*. Oxford Univ. Press, 2010.
16. F.J. Romero-Campero, M.J. Pérez-Jiménez: A model of the quorum sensing system in Vibrio fischeri using P systems. *Artificial Life*, 14 (2008), 95–109.
17. F.J. Romero-Campero, M.J. Pérez-Jiménez: Modelling gene expression control using P systems: The Lac Operon, a case study. *BioSystems*, 91 (2008), 438–457.
18. F.J. Romero-Campero, J. Twycross, M. Cámara, M. Bennett, M. Gheorghe, N. Krasnogor: Modular assembly of cell systems biology models using P systems. *Int. J. Found. Comput. Sci.*, 20 (2009), 427–442.

19. J. Smaldon, F.J. Romero-Campero, F. Fernández Trillo, M. Gheorghe, C. Alexander, N. Krasnogor: A computational study of liposome logic: towards cellular computing from the bottom up. *Syst. Synth. Biol.*, 4 (2010), 157–179.

24 Biologically Plausible Applications of SN P Systems for an Explanation of Brain Cognitive Functions

Adam Obtulowicz

Institute of Mathematics, Polish Academy of Sciences, Warsaw, Poland
A.Obtulowicz@impan.pl

Some conjectures about the possibility of using SN P systems and extension of them for modeling features of the brain (such as learning, modularity) are formulated.

Required Notions: spiking neuron, SN P system, learning

The (hierarchical) clustering (scene segmentation in particular) and binding (feature integration) problem solution in cortical neural networks 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 networks have been proposed, where spiking neural networks are the most biologically plausible models, see [16], [17], [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 SN P systems (cf. [10]) has been proposed in [8]. Thus, one formulates the following question:

> Do SN 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 SN 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 SN 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 an SN P system solving XOR problem.*

If we compare precise timing of spikes approach for spiking neural networks to the number of spikes approach for SN 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 SN 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 SN P systems equipped with Hebbian learning procedures, respectively. The SN P systems constituting that network could correspond to brain network modules realizing simultaneously various cognitive functions, respectively.

On the other hand, since SN 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* (*modeled*) *by the following hybrid constructs*:

1. *a two-level construct of a* spiking super-neural P system *which is an SN 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 regard to the possibility of their molecular implementation which is suggested by recent findings outlined in [1].

References

1. A. Bandyopadhyay, D. Fujita, R. Pati: Architecture of a massively parallel processing nano-brain operating 100 billion molecular neurons simultaneously. *International Journal of Nanotechnology and Molecular Computation*, 1 (2009), 50–80.

2. S.M. Bohte: *Spiking Neural Networks*. Professorschrift, Leiden University, 2003.
3. O. Booij: *Temporal Pattern Classification using Spiking Neural Networks*. M.Sc. Thesis, Amsterdam University 2004.
4. O. Booij, Hieu tat Nguyen: A gradient descent rule for spiking neurons emitting multiple spikes. *Applications of Spiking Neural Networks* (S.M. Bohte, J.N. Kok, eds.), *Information Processing Letters*, Amsterdam, 2005.
5. R. Ceterchi, I.A. Tomescu: Spiking neural P systems–a natural model for sorting networks. *Proc. Sixth Brainstorming Week on Membrane Computing* (D. Diaz-Pernil et al., eds.), Sevilla, February 4–8, 2008, RGNC Report 01/2008, Fenix Editora, Sevilla, 2008, 93–105.
6. D. Geary: *The Origin of Mind: Evolution of Brain, Cognition, and General Intelligence*. American Psychological Association 2005.
7. W. Gerstner: Population dynamics of spiking neurons: Fast transients, asynchronous states, and locking. *Neural Computation*, 12 (2000), 43–89.
8. M.A. Gutiérez-Naranjo, M.J. Pérez-Jiménez: A spiking neural P systems based model for Hebbian learning. *Proc. 9th Workshop on Membrane Computing* (P. Frisco et al., eds.), Edinburgh, July 28–31, 2008, 189–207.
9. M. Ionescu, D. Sburlan: Some applications of spiking neural P systems. *Proc. 8th Workshop on Membrane Computing* (Eleftherakis et al., eds.), Thessaloniki, June 25–28, 2007, 383–394.
10. M. Ionescu, Gh. Păun, T. Yokomori: Spiking neural P systems. *Fund. Inform.*, 71 (2006), 279–308.
11. A. Kasiński, F. Ponulak: Comparison of supervised learning methods for spike time coding in spiking neural networks. *Int. J. Appl. Math. Comput. Sci.*, 16 (2006), 101–113.
12. A. Knoblauch, G. Palm: 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), 168–184.
13. K. MacDonald, D. Chiappe: Review of [6] in *Human Ethology Bulletin*, 21 (2006), 14–18.
14. B. Meftah, A. Benyettou, O. Lezoray, W. Qingxiang: Image clustering with spiking neuron network. *World Congress on Computational Intelligence, International Joint Conference on Neural Networks*, Hong-Kong, 2008.
15. S.C. Moore: *Back-propagation in Spiking Neural Networks*. M.Sc. Thesis, University of Bath 2002, `http://www.simonchristianmoore.co.uk/Thesis4.html`.
16. T. Natschläger, B. Ruf: Spatial and temporal pattern analysis via spiking neurons. *Network: Comp. Neural Systems*, 9 (1998), 319–332.
17. B. Ruf: *Computing and Learning with Spiking Neurons–Theory and Simulation*. Doctoral Thesis, Technische Universität Graz, 1998.

25 Computer Vision

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

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

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

Some possibilities to employ MC techniques in computer vision (especially in thresholding, smoothing, homology theory) are discussed.

Required Notions: array grammar, array-rewriting P system, cell and tissue P systems

Computer vision is probably one of the challenges for computer scientists in the next years. 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.

In MC, there is a large tradition in handling information structured as two dimensional objects (see, e.g., [2, 3, 9, 16]). The main motivation for these studies is to bring together P systems 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 [13].

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, $\mathcal{B}$ is the new one, and $tar \in \{here, in, out\}$ indicates the place of the picture after the substitution has been made.

Recently, a new research line has been open by applying well-known MC techniques for solving problems from digital imagery. For example, 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. Segmentation has shown its utility, for example, in bordering tumors and other pathologies or computer-guided surgery. In [5, 8, 10, 11] we can find several approaches to this problem with MC techniques. Other problems, as *thresholding* [4] or *smoothing* [18] have also been considered in the framework of MC. Special attention deserves [14], where the *symmetric dynamic programming stereo* (SDPS) algorithm [15] 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 provides techniques and algorithms for handling digital images from a topological point of view. Recently, the

links between algebraic topology and MC have started to be explored via *homology theory* [6, 7, 12]. *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. 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 a similar way with other applications of P systems, the theoretical advantages of the MC techniques for computer vision need a powerful software and hardware for an effective implementation. The use of these new technologies for the parallel implementation of P systems techniques applied to computer vision have started to be explored with promising experimental results [1, 17, 18].

An appropriate combination of MC 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 MC.

References

1. J. Carnero, D. Díaz-Pernil, M.A. Gutiérrez-Naranjo: Designing tissue-like P systems for image segmentation on parallel architectures. *Ninth Brainstorming Week on Membrane Computing* (M.A. Martínez del Amor et al., eds.), Fénix Editora, Sevilla, 2011, 43–62
2. R. Ceterchi, R. Gramatovici, N. Jonoska, K.G. Subramanian: Tissue-like P systems with active membranes for picture generation. *Fundamenta Informaticae*, 56 (2003), 311–328.
3. R. Ceterchi, M. Mutyam, Gh. Păun, K.G. Subramanian: Array-rewriting P systems. *Natural Computing*, 2 (2003), 229–249.
4. 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*, 13 (2010), 131–140.
5. H.A. Christinal, D. Díaz-Pernil, P. Real: Segmentation in 2D and 3D image using tissue-like P system. LNCS 5856, Springer, 2009, 169–176.
6. H.A. Christinal, D. Díaz-Pernil, P. Real: Using membrane computing for obtaining homology groups of binary 2D digital images. LNCS 5852, Springer, Berlin, 2009, 383–396.
7. H.A. Christinal, D. Díaz-Pernil, P. Real: P systems and computational algebraic topology. *Mathematical and Computer Modelling*, 52 (2010), 1982–1996.
8. 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 (2011), 2206–2212.
9. K.S. Dersanambika, K. Krithivasan: Contextual array P systems. *Intern. Journal of Computer Mathematics*, 81 (2004), 955–969.

10. D. Díaz-Pernil, M.A. Gutiérrez-Naranjo, H. Molina-Abril, P. Real: A bio-inspired software for segmenting digital images. *Proc. Fifth International Conference on Bio-Inspired Computing. Theories and Applications* BIC-TA (A.K. Nagar et al., eds.), vol. 2 (2010), 1377–1381.
11. 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*, 2011 http://dx.doi.org/10.1007/s11047-011-9287-4.
12. 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*, 13 (2010), 141–152.
13. C. Ferretti, G. Mauri, C. Zandron: P systems with string objects. Chapter 7 of *The Oxford Handbook of Membrane Computing* (Gh. Păun, G. Rozenberg, A. Salomaa, eds.), Oxford University Press, 2010, 168–197.
14. G. Gimel'farb, R. Nicolescu, S. Ragavan: P systems in stereo matching. LNCS 6855, Springer, Berlin, 2011, 285–292.
15. G. Gimel'farb: Probabilistic regularisation and symmetry in binocular dynamic programming stereo. *Pattern Recognition Letters*, 23 (2002), 431–442.
16. S.N. Krishna, R. Rama, K. Krithivasan: P systems with picture objects. *Acta Cybernetica*, 15 (2001), 53–74.
17. 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. LNCS 6855, Springer, 2011, 277–284.
18. 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. *Intern. Journal of Natural Computing Research*, 2 (2011), 25–34.

26 Open Problems on Simulation of Membrane Computing Models

Manuel García-Quismondo, Luis F. Macías-Ramos, Miguel A. Martínez-del-Amor, Ignacio Pérez-Hurtado, Luis Valencia-Cabrera

Research Group on Natural Computing
Dpt. of Computer Science and Artificial Intelligence, University of Sevilla, Spain
{mgarciaquismondo,lfmaciasr,mdelamor,perezh,lvalencia}@us.es

Research ideas related to the extension of the P-lingua dedicated languages and on the implementation of P systems on reconfigurable or parallel hardware (e.g., NVIDIA architectures) are mentioned.

Required Notions: P system models, GPU computing, P-Lingua, MeCoSim

The development of P system simulators, and of other related software tools, becomes a critical point in the processes of model validation and virtual experimentation. For this purpose, a software framework for specifying and simulating

P systems, called P-Lingua, was developed [7]. Moreover, a generic software to generate graphical applications based on P-Lingua, called MeCoSim, was also developed. Finally, in order to accelerate the simulation by implementing P systems parallelism on high performance platforms, some simulators were developed by using GPU computing [5]. Research in all these directions are under development.

(A) Simulation Framework: P-Lingua and PLinguaCore

P–Lingua has been successfully applied to ecosystem modeling problems [6], formal model checking and to solve computationally hard problems [7]. It supports several P system models, such as *active membrane* models [7], *Tissue* P system models and Spiking Neural P Systems (*SN P*) [9] systems.

Nevertheless, there is still plenty to do in order to extend the capabilities of P-Lingua. Both expressivity and functionality issues should be improved to renew the P-Lingua menu and attract new users. First, inclusion of parsing directives should be implemented in order to, say, modify the behavior of existing models. A fast learner example would be the 'asynchronous' behavior, that is, cracking the universal clock that reigns the computation process in most of standard models (this is already included for the case of SN P Systems). Then, integration of new models, such as numerical P systems and some specific types of SN P Systems, incorporating *weights* and *astrocytes with their different flavors*, remains unexplored. Finally, re-factoring of the work done to bring some exotic elements of the *reaction systems*.

(B) Generic end-user graphical applications: MeCoSim

In the last few years, there have been some interesting, user-friendly, successful software applications for modeling and simulating P systems, mainly focused on biological systems: *MetaPlab* [4] for internal mechanisms of biological systems by means of MP System; *BioSimWare* [1] and *Infobiotics* [2] for P system based multicompartmental stochastic simulations of complex biological systems, the last one including Synthetic Biology; and *EcoSim* [6], a family of probabilistic simulators for different ecosystems.

However, a general application for the study, analysis, modeling, visual simulation, model checking, optimization of as many as possible variants of P systems has not been provided. A first approach has been developed with *MeCoSim* [10]. Some plugins have been developed to provide some analysis and model checking capabilities. It has been successfully applied as an assistant tool for the iterative design of ecosystem models, and to solve computationally hard problems by using tissue and SN P system models.

Nevertheless, there are many challenges to solve. The core of the visual application should include more analysis and modeling tools to ease the work of the P systems designer. Also, some *interfaces* to communicate with different simulation engines should be developed to run simulation against simulators implemented in different local or remote platforms and architectures. It should integrate with different applications for formal model checking, enabling the user to extract and/or validate properties of the studied models. Eventually, new P systems *models* should

be added to MeCoSim, providing the general functionalities and new possible plugins to many potential P systems designers.

(C) Simulation on High Performance Platforms: GPU computing

So far, there have been many efforts on the development of GPGPU based simulators for P systems. In fact, the following P systems models have been successfully simulated by means of GPUs: P systems with active membranes and division rules [5], a family of P systems with active membranes solving `SAT` in linear time [5], SN P systems (SNP) with and without delays [3], and ENPSs [8].

On the other hand, there exist many other models which, to the best of our knowledge, are yet to be simulated by means of GPGPU. These models include tissue P systems, population dynamics P systems, stochastic P systems, hyperdag P systems, numerical P systems and string P systems, to name just a few.

Another challenge is the *integration* of GPU simulators on end-user MC software frameworks. Although some steps have been taken in this direction with the P-Lingua automatic generation [7] of P system files to be parsed by GPU simulators [5], there is still a long way to walk for an efficient interaction between these two kinds of technological tools.

Last but not least, a thorough *performance comparison* between GPU simulators and other HPC approaches is yet to be developed. These approaches include reconfigurable hardware (FGPA, DSP), computer clusters with OpenMP and MPI, etc. The need for some works on this direction has been previously noticed [5, 8]. Finally, it is also interesting to port current developments of GPU simulators to the last GPU platforms, based on both NVIDIA and AMD ATI architectures, and on GPU based clusters.

Acknowledgements. The authors are supported by the project TIN 2009-13192 from "Ministerio de Economía y Competitividad" of Spain, and by "Proyecto de Excelencia con Investigador de Reconocida Valía" P08-TIC-04200 from Junta de Andalucía, both co-financed by FEDER funds. Manuel García-Quismondo is also supported by the National FPU Grant Programme from the Spanish Ministry of Education.

References

1. D. Besozzi, P. Cazzaniga, G. Mauri, D. Pescini: BioSimWare: A software for the modeling, simulation and analysis of biological systems. LNCS 6501, Springer, 2011, 119–143.
2. J. Blakes, J. Twycross, F.J. Romero–Campero, N. Krasnogor: The infobiotics workbench: an integrated in silico modelling platform for Systems and Synthetic Biology. *Bioinformatics*, 27 (2011), 3323–3324.
3. F.C. Cabarle, H. Adorna, M.A. Martínez-del-Amor: A spiking neural P system simulator based on CUDA. LNCS 7184, Springer, Berlin, 2012, 87–103.
4. A. Castellini, V. Manca: MetaPlab: A computational framework for metabolic P systems. *Pre-proceedings of WMC'08*, 2008, Edinburgh, Scotland.

5. J.M. Cecilia, J.M. García, G.D. Guerrero, M.A. Martínez-del-Amor, I. Pérez-Hurtado, M.J. Pérez-Jiménez: Simulation of P systems with active membranes on CUDA. *Briefings in Bioinformatics*, 11 (2010), 313–322.
6. M.A. Colomer, A. Margalida, D. Sanuy, M.J. Pérez–Jiménez: A bio–inspired computing model as a new tool for modeling ecosystems: The avian scavengers as a case study. *Ecological modelling*, 222 (2011), 33–47.
7. M. García-Quismondo, R. Gutiérrez-Escudero, M.A. Martínez-del-Amor, E. Orejuela-Pinedo, I. Pérez-Hurtado: P-Lingua 2.0: A software framework for cell-like P systems. *International Journal of Computers, Communications and Control*, 4 (2009), 234–243.
8. M. García-Quismondo, M.J. Pérez–Jiménez, L.F. Macías–Ramos: Implementing ENPS by means of GPUs for AI applications. *Beyond AI. Interdisciplinary Aspects of Artificial Intelligence (BAI 2011)*, 08/12/2011–09/12/2011, Pilsen, Czech Republic.
9. L.F. Macías-Ramos, I. Pérez-Hurtado, M. García-Quismondo, L. Valencia-Cabrera, M.J. Pérez-Jiménez, A. Riscos-Núñez: A P-Lingua based simulator for spiking neural P systems. LNCS 7184, Springer, 2012, 257–281.
10. I. Pérez-Hurtado, L. Valencia-Cabrera, M.J. Pérez-Jiménez, M.A. Colomer, A. Riscos-Núñez: MeCoSim: A general purpose software tool for simulating biological phenomena by means of P Systems. *IEEE Proceedings of BIC-TA 2010*, I, 637–643.

Closing Remarks

As also said in the Introduction, this collection of open problems and research topics in MC was initially meant to be a working material, for 10th BWMC, and it was updated and completed several times. However, no such list can be complete, neither uniform, in what concerns the type of problems, their technicality, difficulty, range of interest. As expected, some problems are local, others are very general, while the sections are not at all uniform in style (we have preserved in a great extent the contributors writing). Moreover, many further research ideas wait to be addressed in MC, for instance, in the P and dP automata area, the SN P systems area, complexity, dynamical systems approach – not to speak about applications (from biology and bio-medicine, to ecology, robot control, approximate optimization). 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, and, on the other hand, because it points out active research areas of MC, indicating its "frontiers". Actually, this "mega-paper" proved already to be useful during the 10th BWMC, where several of the proposed research topics were addressed – the paper incorporates some changes due to these recent progresses.

Acknowledgements

The work of Gh. Păun and M.J. Pérez-Jiménez was supported by Proyecto de Excelencia con Investigador de Reconocida Valía, de la Junta de Andalucía, grant P08 – TIC 04200. The work of M. Gheorghe was partially supported by project

MuVet, Romanian National Authority for Scientific Research (CNCS, UEFISCDI) grant number PN-II-ID-PCE-2011-3-0688.

The editors are much indebted to all MC researchers who have contributed to this "mega-paper", to Grzegorz Rozenberg for many suggestions during the preparation of the paper, as well to two anonymous referees for carefully reading a previous version of the text.