# 2006 Research Topics in Membrane Computing

Gheorghe Păun

Institute of Mathematics of the Romanian Academy
PO Box 1-764, 014700 Bucharest, Romania
and
Research Group on Natural Computing
Department of Computer Science and Artificial Intelligence
University of Sevilla
Avda Reina Mercedes s/n, 41012 Sevilla, Spain
george.paun@imar.ro, gpaun@us.es

**Summary.** This is a list of research topics prepared on the occasion of the Fourth Brainstorming Week on Membrane Computing, Sevilla, January 30 – February 3, 2006 (hence the title). The selection is subjective, the presentation is informal (in most cases, the precise formulation of the question to investigate is already part of the research task), the bibliographical information is partial (giving only a starting point for searching the literature; as usual, for a complete bibliography the reader should refer to [36]).

## 1 Introduction

The present notes mainly intend to keep the tradition of the previous editions of the Brainstorming, when lists of open problems and research topics were circulated – see [24, 26, 27]. Also, continuing in an inductive manner the tradition, with the alternation of lists of twenty six problems, like [24] and [27], and of shorter lists, like [26], the present collection of problems/research topics is shorter – although, ambiguously enough, the title would suggest that there are mentioned more than two thousand problems... This does not mean that, if somebody would have enough time and energy to check the literature, (s)he will not be able to produce such a long list: the number of precise open problems mentioned in area papers is really large (with the mentioning that many of these problems are of a rather technical nature; a good example is that of the borderline between universality and non-universality for various classes of P systems, when taking into account descriptional complexity parameters such as the size of rules, the number of membranes, etc., and where many sharp results were obtained, and, if some problems are still open, they concern pretty small variations of the mentioned parameters – a few such precise problems will be mentioned in Section 2).

Anyway, I stress the fact that this list is subjective, it contains mainly topics which I have touched in the last time and which I find of (a personal) interest. In

some sense, the main goal of this discussion is to call attention to several recently considered directions of research, not to formulate precise questions. Many other topics could be more important or more attractive (of course, the importance and the appeal of a problem is also a context-sensitive issue). This is, for example, the case with the problems related to the computational complexity investigations, where rather sound-nice-unexpected results were recently obtained (see, e.g., [12] and [33]) and where many intriguing open problems and conjectures still wait for research efforts (and ideas to attack them). In spite of this, in what follows, only one complexity problem will be formulated, although for sure many related problems deserve a special attention.

Similarly, it will not be separately considered[1] the fundamental issue of applications in biology and medicine, in general, the problem of keeping a close contact with the biological reality (and to try to provide models, tools, techniques to the biologist), although this is (at least in this moment), independently of personal choices, important for the health and visibility of membrane computing (for getting funding sources, too...). After having several applications which have proved that multiset processing in the compartmentalized structure of cell-like or tissue-like P systems is adequate and relevant, reliable and easy to handle, extensible and easy to understand, etc., etc. (see, e.g., [9]), it is both possible and necessary to prove that this approach is also *useful*, in the practical sense, as defined by users from biology and medicine, not by computer scientists working in membrane computing. The short formulation of this issue can be: let us (try to) pass from post-diction to pre-diction (running our models and programs on research data, not on data already published, useful only for validating the new approach; checking hypotheses through computer simulations and providing conclusions to biologists and physicians, who can then either directly trust these conclusions or can transform them in next level hypotheses to be checked in laboratories, in the traditional way).

Another standard caution concerns the fact that the ordering of the research topics below has no significance and that most of the questions are informally/preliminarily formulated. In general, the problems are selected in such a way that the present list is "as disjoint as possible" from the previous lists, although this is not completely true/posssible, because there are issues which were considered also before and which deserve now to be formulated in new, up-dated terms. (In general, a good meta-problem would be to follow the trace of the problems from the previous lists, checking the literature for solutions to them. This is not easy, but it can be interesting – and somewhat rewarding: for instance, most of the problems considered in [24] were solved, or at least addressed and partially settled. This might motivate the compilation of similar lists, including the present one...).

[1] Well, this paragraph is "separated" enough, hence the issue itself is emphasized enough in this rhetoric way...

## 2 Four Jewel–Problems About Borderlines

Finding the borderline between universality and non-universality (reaching or not the Turing computing power), and between efficiency (solving **NP**-complete problems in polynomial time) and non-efficiency is both a mathematical challenge and a question with a larger significance, including a practical one (non-universality could mean decidability, efficiency is much desired – provided that it can be implemented). These borderlines can be defined either in a qualitative manner (ingredients to be used or not, and making a difference in the power/efficiency of P systems) or in a quantitative manner (size of parameters, hence descriptional complexity).

I have selected only four problems of this type, although many others can be found in the literature, and I have called them "jewels" because they deal with basic classes of P systems and refer to very small differences in the used features and/or the size of parameters; implicitly, this also means that there were many efforts spent around these problems, before arriving to the tinny space where the borderline has to be found.

The first problem is already classic: catalytic P systems are universal when using at least two catalysts ([10]), but not when using only one catalyst *and* having all rules catalytic (each rule is of the form $ca \to cu$; see [14]), but the case of systems with one catalyst and using both catalytic and non-catalytic rules ($ca \to cu$ and $a \to u$, where $a$ is an object, $u$ is a multiset, and $c$ is the catalyst) is still open. In both proofs from [10] and [14] one uses only one membrane, with the catalyst(s) ignored when counting the result in the halting configuration.

A similarly restricted space to search the borderline between universality and non-universality is available now for P systems with symport/antiport rules: systems with three membranes are known to be universal when using minimal symport and antiport rules ($sym_1$ and $anti_1$), or only symport rules of weight at most 2 ($sym_2$), but the case of two membranes is open. For two membranes, universality results were obtained only modulo a terminal alphabet, or when ignoring numbers smaller than a given threshold – see details and more precise definitions in [1]. Are two membranes universal in the "pure" case, when no additional symbols are allowed?

Related to symport/antiport systems, but playing now with the number of objects used in the system, is the following problem. Having a restricted number of objects, the rules cannot be of a bounded weight (or can they, at least "partially", for instance, using bounded symport rules and arbitrary antiport rules, or conversely?), hence a trade-off has appeared between the number $n$ of objects and the number $m$ of membranes. The following results are the currently best ones in this respect (see again [1]): systems with $(n, m) \in \{(5, 1), (4, 2), (3, 3), (2, 4)\}$ are universal. The problem is open whether or not systems with one object and any number of membranes are universal. (The conjecture is that the answer is negative.)

After three problems concerning the universality non-universality borderline, let us consider one jewel–problem concerning the efficiency non-efficiency borderline. It was several times asked the question whether or not P systems with active membranes (hence using membrane division) can solve **NP**-complete problems in polynomial time without using membrane polarizations (with the conjecture that the answer is negative – of course, without adding further features, such as label changing, cooperation, etc.) Actually, it turned out that the problem and the conjecture should take into consideration also other "actors", among them, surprisingly, being necessary to include the "innocent looking" operation of membrane dissolution. Indeed, as proved in [12], P systems without polarizations, using membrane division for only elementary membranes or also for non-elementary membranes, with the construction uniform or semi-uniform, solve in polynomial time exactly the problems in the classical class **P**, *provided that the operation of membrane dissolution is not allowed*. Adding membrane dissolution changes drastically the result, at least in the case when division of non-elementary membranes is allowed and the construction is semi-uniform: **NP**-complete problems can be solved in polynomial time in this framework.

Thus, the borderline between efficiency and non-efficiency is defined in terms of the following features: (i) membrane dissolution, (ii) constructing the system in a non-uniform way, and (iii) using membrane division also for non-elementary membranes (of course, in all cases, the membranes do not have polarizations). Somewhere between using none of these features and using all of them lies the borderline we look for (in the proofs from [12], essential was the use/non-use of membrane dissolution).

I stop here with such technical problems (four jewels are anyway a good collection...), and pass now to more general research topics.

## 3 Where Is the Nucleus?

This question was raised by S. Istrail during the recent Unconventional Computation conference (October 2005, Sevilla). It is simple and provocative. We claim to start from the cell, the eukaryotic cell has a membrane/region which plays a central role in its life, the nucleus. How to bring this in our models? Of course, it is not sufficient to design a membrane of a P system as the distinguished one called "nucleus", but to consider a class of systems where this membrane behaves like the nucleus of a cell. A first possible answer to this challenge is to place in this special membrane somewhat like a *genome*, containing the description of the whole cell (actually, organism), a DNA-like support of information which would codify the whole architecture and functioning of the system.

At this general level, this issue is directly related to the question several times formulated of self-reproduction of P systems. Is this a way to address this latter question?

We can, however, be less ambitious and try to put in the nucleus only data about the functioning of the system, not also about its structure. Otherwise stated,

the membranes are given, but they are empty, they contain no objects and no rules, objects are placed only in the nucleus. Some of these objects can codify standard evolution rules, others can codify usual objects, to be evolved by the rules in the compartments of the system.

For instance, we can place in the nucleus tokens of the form $\langle u \to v \rangle$, where $u$ and $v$ are multisets of objects over a given alphabet $O$, as well as tokens of the form $\langle u \rangle$, where $u$ is a multiset over $O$. Some "promoters" should be also placed in the nucleus, in the form of simple objects $a \in O$.

Then, we provide "transcription rules" (more neutrally, we can say "decoding rules") of the forms

$$\langle u \to v \rangle b \to \langle u \to v \rangle \langle u \to v; @h \rangle,$$
$$\langle u \rangle b \to \langle u \rangle \langle u; @h \rangle,$$

where $b \in O$ plays the role of a promoter for "expressing the genes" $\langle u \to v \rangle$ and $\langle u \rangle$. The idea is that in the presence of $b$, the "genes" $\langle u \to v \rangle$ and $\langle u \rangle$ produce a copy of the rule $u \to v$ and of the multiset $u$, which have as destination region $h$ of the system; the operation consumes the promoter $b$, but reproduces the "genes". In turn, the rule $u \to v$ and the multiset $u$ will move towards region $h$, on the shortest path to the destination, crossing one membrane in each time unit. During this journey, the rule and the multiset are not active (they are encapsulated in the "vesicle" $\langle \ldots \rangle$), and they become active only after reaching region $h$.

In the regions of the system, the rules and the objects behave as usual in membrane computing (the rules are applied in a specified way – maximally parallel, minimally parallel, sequentially – to the existing objects, non-deterministically choosing the rules and the objects), with the following important difference: the rules are present in the multiset sense, and, after being used, they are consumed. For instance, if two copies of a rule $aa \to bc$ were sent to membrane $i$ and there are three copies of $a$ available, then one rule disappears, and the resulting multiset is $abc$; if there are five copies of $a$, then both rules are consumed and we get the multiset $abbcc$.

The objects introduced by rules can have usual target indications, hence in this way certain objects can be sent to the nucleus, thus promoting the expression of further "genes".

At the first sight, the "life" of such a system is much different from the evolution of a usual P system – but this intuition remains to be checked, starting with more precise definitions of the system components and the system functioning. Then, the research topics which naturally appear in this context are many – besides the above-mentioned issue of self-reproduction. The previous idea can be formulated for various types of P systems (for various types of rules); which ones are more attractive? Which is the computing power of such systems? Because the universality is plausible (for specific ingredients), it is natural then to ask which is the size (of nucleus) of minimal universal systems? Which is the minimal size of a system with a non-trivial (infinite, cyclic, non-cyclic, complex enough from specific points of view) evolution?

## 4 The Brane-Membrane Bridge

Staying close to biology, let us point out that recently ([5]) a research area somewhat complementary to membrane computing was initiated, with the same starting point, the living cell, but focusing mainly on the operations which directly involve membranes, under the control of proteins embedded in them, and letting aside the biochemistry taking place in the compartments. Two main so-called *brane calculi* were introduced, based on pino-exo-phago and mate-drip-bud operations. The framework and techniques used are those of process algebra, and an important difference from the standard P systems is the fact that the systems introduced in this area are unsynchronized. This is essential in what concerns the computing power of the obtained systems: synchronized brane systems are non-universal, while synchronized systems are universal – see details in [4], [3].

However, the bridge between brane calculi and membrane computing can be established in various ways, and it looks rather fruitful theoretically. The simple starting idea is to consider P systems with brane-like operations with membranes. This was already done in [6] for the mate-drip operations, which were shown to lead to universality; other combinations of operations still wait to be considered – with the mentioning that in [19] there are results of this kind.

A different approach is that considered in [7], where objects are placed both in the regions and on the membranes of a P system, and they evolve both by means of membrane operations and directly, by means of multiset rewriting rules; the objects can move from membranes to regions and conversely. A restricted framework is investigated in [22], where the membranes have "proteins" placed on them, and the objects from the compartments evolve by means of symport/antiport-like rules which are controlled by the proteins. Specifically, rules of several forms are considered (the fact that a protein $p$ is on a membrane (with label) $i$ is written in the form $[_i p|$; "res" indicates that we use here rules of a "restricted" form):

| Type | Rule | Effect |
|---|---|---|
| 1res | $[_i p\|a \to [_i p\|b$ | |
| | $a[_i p\| \to b[_i p\|$ | modify an object, but not move |
| 2res | $[_i p\|a \to a[_i p\|$ | |
| | $a[_i p\| \to [_i p\|a$ | move one object unmodified |
| 3res | $[_i p\|a \to b[_i p\|$ | |
| | $a[_i p\| \to [_i p\|b$ | modify and move one object |
| 4res | $a[_i p\|b \to b[_i p\|a$ | interchange two objects |
| 5res | $a[_i p\|b \to c[_i p\|d$ | interchange and modify two objects |

In these rules, the protein is not changed, but a generalization is to allow rules of the forms below (now, "cp" means "change protein"), where $p, p'$ are two proteins (possibly equal; if $p = p'$, then the rules of type *cp* become rules of type *res*):

| Type | Rule | Effect (besides changing also the protein) |
|---|---|---|
| 1cp | $[_i p\|a \to [_i p'\|b$ | |
| | $a[_i p\| \to b[_i p'\|$ | modify an object, but not move |
| 2cp | $[_i p\|a \to a[_i p'\|$ | |
| | $a[_i p\| \to [_i p'\|a$ | move one object unmodified |
| 3cp | $[_i p\|a \to b[_i p'\|$ | |
| | $a[_i p\| \to [_i p'\|b$ | modify and move one object |
| 4cp | $a[_i p\|b \to b[_i p'\|a$ | interchange two objects |
| 5cp | $a[_i p\|b \to c[_i p\|d$ | interchange and modify two objects |

An intermediate case can be that of changing proteins, but in a restricted manner, by allowing at most two states for each protein, $p, \bar{p}$, and the rules either as in the first table (without changing the protein), or changing from $p$ to $\bar{p}$ and back (like in the case of bistable catalysts). Rules with such flip-flop proteins are denoted by $nff, n = 1, 2, 3, 4, 5$ (note that in this case we allow both rules which do not change the protein and rules which switch from $p$ to $\bar{p}$ and back).

Various combinations of these rules were shown in [22] to be universal, but still several combinations wait to be examined.

Besides these specific problems, the general issue of combining ideas from the two areas, brane calculi and membrane computing, deserves to be emphasized as an interesting research topic – at least because the cell does not separate its life in two distinct biochemistries, one considering what happens in compartments and one considering the membranes themselves...

## 5 Spiking Neural P Systems

The question of incorporating ideas from neuro-biology into membrane computing was formulated several times as a research topic, and there are several contributions to this issue, including a chapter in [25]. Still, this research direction is not at all explored as it deserves to be; the neurons functioning and especially their cooperation in various constructions, the brain included, is a huge source of ideas. Recent contributions were added to this topic by introducing so-called *spiking neural P systems*, which capture the important idea of neural biology concerning the way the neurons communicate by means of "spikes", electrical impulses of identical intensity and shape, but occurring at time moments which are carrying information in the distance between them. We refer to [20], [21] for details and further references about the biological processes related to spiking and about the way they are used in neural computing (one speaks in the last years about a "third generation" neural computing based on spiking neurons).

The way the idea is modeled in terms of P systems is rather simple: one considers only one type of objects, the spike, denoted by $a$, and neurons linked by synapses (elementary membranes placed in the nodes of a directed graph), containing spikes and rules for handling them. These rules are of two forms:

(1) $E/a^r \to a; t$, where $E$ is a regular expression over $\{a\}$, $r \geq 1$, and $t \geq 0$;

(2) $a^s \to \lambda$, for some $s \geq 1$, with the restriction that $a^s \notin L(E)$ for any rule $E/a^r \to a;t$ of type (1) from the same neuron.

The rules of type (1) are *firing* (we also say *spiking*) *rules*, the rules of type (2) are *forgetting rules*.

A neuron gets fired when using a rule $E/a^r \to a;t$, and this is possible only if the neuron contains $n$ spikes such that $a^n \in L(E)$ and $n \geq r$. This means that the regular expression $E$ "covers" exactly the contents of the neuron. The use of a rule $E/a^r \to a;t$ in a step $q$ means firing in step $q$ and spiking in step $q+t$. That is, if $t = 0$, then the spike is produced immediately, in the same step when the rule is used. If $t = 1$, then the spike will leave the neuron in the next step, and so on. In the interval between using the rule and releasing the spike, the neuron is assumed *closed* (in the refractory period), hence it cannot receive further spikes, and, of course, cannot fire again. This means that if $t \geq 1$ and another neuron emits a spike in any moment $q, q+1, \ldots, q+t-1$, then its spike will not pass to the neuron which has used the rule $E/a^r \to a;t$ in step $q$. In the moment when the spike is emitted, the neuron can receive new spikes (it is now free of internal electricity and can receive new electrical impulses). This means that if $t = 0$, then no restriction is imposed, the neuron can receive spikes in the same step when using the rule. Similarly, the neuron can receive spikes in moment $t$, in the case $t \geq 1$.

If a neuron $\sigma_i$ spikes, its spike is replicated in such a way that one spike is sent to *all* neurons $\sigma_j$ such that there is a synapse from $\sigma_i$ to $\sigma_j$ (we write $(i,j) \in syn$), *and* $\sigma_j$ is open at that moment. If a neuron $\sigma_i$ fires and either it has no outgoing synapse, or all neurons $\sigma_j$ such that $(i,j) \in syn$ are closed, then the spike of neuron $\sigma_i$ is lost; the firing is allowed, it takes place, but it produces no spike.

By using a forgetting rule $a^s \to \lambda$, $s$ spikes are simply removed ("forgotten"). Like in the case of spiking rules, the left hand side of a forgetting rule must "cover" the contents of the neuron, that is, $a^s \to \lambda$ is applied only if the neuron contains exactly $s$ spikes.

Note that each neuron uses at most one rule at a time – hence the neurons work in a sequential manner (but the system itself is synchronized: in each time unit, each neuron which can use a rule should do it).

One of the neurons is designated as the output neuron of the system and when it spikes, besides spikes sent to other neurons along synapses, a spike is also sent to the environment. In this way, the system produces a *spike train*, a sequence of time units when we have spikes leaving the system. The number of steps elapsed between two consecutive spikes can be considered as being computed by the system, with many possibilities to define the computed set of numbers: taking all intervals, taking only the interval between the first two spikes, considering alternately the intervals (we take the first interval, we ignore the second one, we take the third interval, and so on), considering all computations or only the halting ones. In this way, the spiking neural P systems behave as number computing devices. It is also possible to consider the spike train itself as the output of a computation, codified as a binary sequence: we write 1 for a time unit when the system sends a spike into

the environment and 0 for a time unit when no spike is sent out. In the non-halting case we get then an infinite binary sequence. If also input neurons are considered, then we can work in the accepting case or even with spiking neural P systems as transducers of binary strings/sequences.

The system from Figure 1 illustrates this discussion, and it also introduces the way of graphically representing a spiking neural P system working in the generating mode: neurons (represented by ovals) placed in the nodes of a graph whose edges represent the synapses, with the output neuron also having an arrow pointing to the environment; in each neuron we specify the existing spikes and the rules. In turn, if a firing rule $E/a^r \to a; d$ has $L(E) = \{a^r\}$, then we write it in the simpler form $a^r \to a; d$.

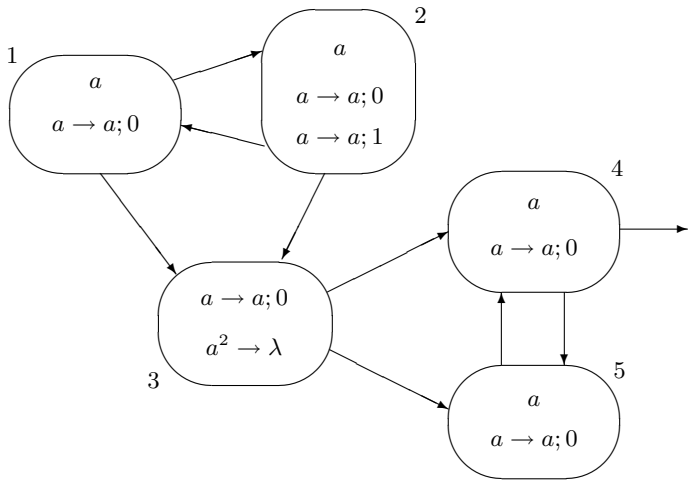


**Fig. 1.** A non-deterministic SN P system with infinite spike trains

The reader can check that this system generates the spike trains described by the binary sequences of the forms $1^k0^\omega$, for all $k \geq 2$.

Many of the above mentioned possibilities of using a spiking neural P systems were considered in [15], [30], [31], and part of them were also investigated in some detail. In particular, two main results were proved in [15] for the case of considering only the distance between the first two spikes, and then extended in [30] to many other cases:

1. universality in the case when no bound is imposed on the number of spikes present in neurons,
2. a characterization of semilinear sets of numbers when the number of spikes present in the neurons is bounded.

The universality result is obtained for systems with a small number of rules in each neuron (at most two), and small numbers of spikes consumed in firing and forgetting rules (at most three), but without any bound on the number of neurons. Improving the proof from this point of view as well as many other open problems were formulated in the mentioned papers.

A lot of problems are also open in what concerns the case of infinite sequences, starting with the necessary comparison of the family of sequences computed in this framework with infinite sequences recognized by finite automata, by Turing machines, or appearing in other areas (see, e.g., [32], [34], [35]).

Not mentioned in the papers cited above is the following natural question: how can a spiking neural P system compute a string? Taking prefixes or subwords of spike trains is one possibility. Working with halting computations and considering the binary description of their spike trains is another possibility. More flexible is the idea of looking for *representations* of languages in terms of languages produced by spiking neural P systems (e.g., via a gsm mapping, which can discard a finite prefix and an infinite suffix of a spike train, maybe also translating the remaining subsequence). This seems to be a non-trivial task, which however might be more feasible if we import an idea from neural-like P systems as discussed in [25] to spiking neural P systems: associating a state to each neuron, and using the states in rules of the following forms

(1) $qE/a^r \to q'a;t$, where $E$ is a regular expression over $\{a\}$, $r \geq 1$, $t \geq 0$;
(2) $qa^s \to q'$, for some $s \geq 1$,

with $q, q'$ being states of the neuron (changed in this way when using the rules).

The study of such systems remains to be carried out (are states useful also from other points of view, for instance, in order to find representations also for Turing computable infinite sequences of bits?). The question still looks non-trivial, at least in the previous formalization, because the states are local, there is no way to communicate among neurons other than by spikes, as in the systems without states. Maybe also relaxing the condition of not having forgetting rules $a^s \to \lambda$ with $a^s \in L(E)$ for firing rules $E/a^r \to a;t$ can also help (a further degree of non-determinism is then allowed).

Not touched in the above mentioned papers are other issues which are natural to be considered, importing them from classic neural computing or from biology: learning, adaptation, self healing, etc., which actually are not explored yet in membrane computing in general.

## 6 Applications in Economics

This is another possible area of applications, which promises to be fruitful both from a theoretical point of view, through the many suggestions coming from economy, and from a practical point of view. Some attempts towards using P systems as a framework for modeling economic processes were made in [29] (with several

previous applications, of a more specific type, being reported by the Polish group of researchers in membrane computing, see, e.g., [17] and the references therein). One starts from the encouraging observation that many processes taking place in economy are of a "chemical type": handling discrete objects, by means of "reactions" which transform certain objects in other objects. The versatility of the multiset rewriting was convincingly proved in [29], by considering an idealized case study, of a producers-retailers interplay, similar to a market functioning (with prices varying according to the agents behavior and commodities abundance), while the software previously used to simulate biological processes was successfully used to simulate this idealized economic process.

Still, many difficulties (differences from the biological framework) were identified: the role played by psycho-social ingredients (the trust between agents, based on the history of their previous collaboration), the long-distance transmembrane communication, across several membranes in one step, the need to have performance criteria for choosing the rules to apply (minimizing the costs and maximizing the profit), and do on. All these make necessary both considering new types of P systems, but also to write new programs for simulating them. The possible reward makes this effort worth carrying out, because, although the use of mathematical models in economics has a long and successful history, still the economists are looking for new tools/techniques especially of a computational type: just because the economy is so much influenced by human factors, as those mentioned above, classic mathematical models are not always useful and heuristic approaches are preferred, based on computer simulations, experiments, adjustments of parameters, and so on – much similar to biological experiments.

This is a work to be done mainly in collaboration with an economist, but also many theoretical aspects appear in this framework. One example is that of numerical P systems, introduced in [28]. They are rather different in style from usual P systems, so that I introduce them in some details, using the example presented in Figure 2.

In the regions of a membrane structure, we have *numerical variables* and *production-repartition programs*. Such a program (program $l$ from region $i$) is of the form

$$pr_{l,i} = (F_{l,i}(x_{1,i}, \ldots, x_{k_i,i}), c_{l,1}|v_1 + c_{l,2}|v_2 + \ldots + c_{l,n_i}|v_{n_i}),$$

where $F_{l,i}(x_{1,i}, \ldots, x_{k_i,i})$ is the "production function" and $c_{l,1}|v_1 + c_{l,2}|v_2 + \ldots + c_{l,n_i}|v_{n_i}$ defines the "repartition protocol". Specifically, we proceed as follows. Denote $C_{l,i} = \sum_{s=1}^{n_i} c_{l,s}$. We start with given initial values for variables (indicated in square brackets in Figure 2). At any time $t \geq 0$, we compute $F_{l,i}(x_{1,i}(t), \ldots, x_{k_i,i}(t))$. The value $q = F_{l,i}(x_{1,i}(t), \ldots, x_{k_i,i}(t))/C_{l,i}$ represents the "unitary portion" to be distributed to variables $v_1, \ldots, v_{n_i}$, according to coefficients $c_{l,1}, \ldots, c_{l,n_i}$ in order to obtain the values of these variables at time $t+1$. Specifically, $v_{l,s}$ will receive $q \cdot c_{l,s}, 1 \leq s \leq n_i$. If a variable receives such "contributions" from several neighboring compartments, then they are added in order to produce the next value of the variable. A variable used in a production function is

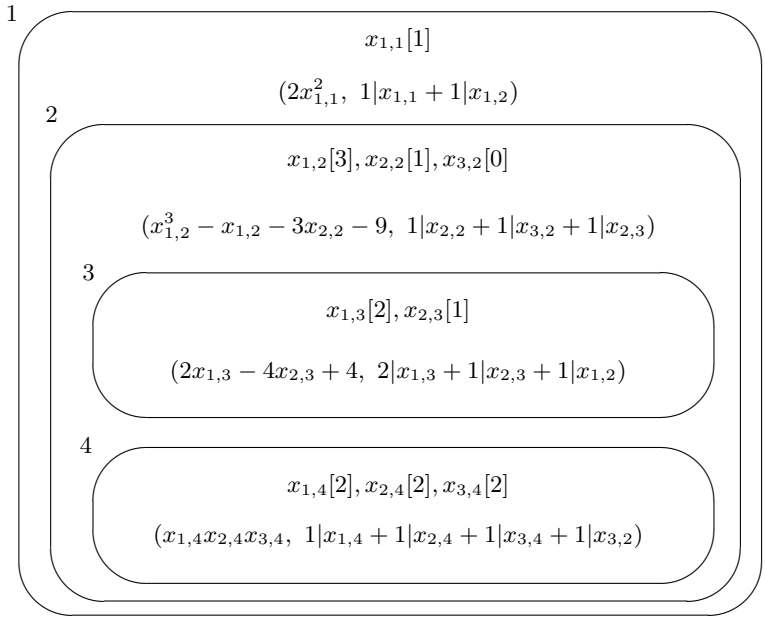


**Fig. 2.** An example of a numerical P system

consumed, reset to 0; if a variable does not appear in a production function, then its value remains unchanged – of course, it can increase by receiving "production portions" from its region or from the neighboring regions.

In this way, we pass from given values of the variables to next values.

The system can be considered as a number computing device by taking as the results of a computation all positive values assumed during the computation by a specified variable from a specified membrane. In such a case, universality is proved in [28] for numerical P systems with a small number of membranes (eight and seven), using polynomials as production functions of rather low degrees and with a small number of variables (degree five in both cases, and five and six variables, respectively).

Of course, the above parameters need to be checked for optimality. Many other research topics are formulated in [28], but we do not recall them here as it is possible that the reader will have his/her own favorite problems about these systems – in general, concerning the applications in economics and related theoretical developments.

## 7 A Quick Mentioning of Other Topics

As said in the Introduction, the primary goal of these notes is not to mention precise problems, but mainly to call reader's attention to some recent ideas which promise to have fruitful continuations.

One such idea is that of *minimal parallelism*, proposed in [8] and then investigated in [16]: instead of the maximally parallel mode of using the rules (choosing a multiset of rules which is maximal, no further rule can be applied), let us ask that from each set of rules (associated with a region or with a membrane), if at least one rule *can* be applied, then at least one *must* be applied; if more rules can be used, then any number of rules may be applied, the only restriction (hence the only information) is to have *at least one*. Clearly, this is much more relaxed than the maximal parallelism (and also than the sequential mode, where *at most one* rule is applied), hence at the first sight this way of using the rules is weaker and more difficult to handle than the maximal mode. This is somewhat confirmed in the papers mentioned above, where universality results were obtained by adding certain ingredients to systems known to be universal in the maximally parallel case without those ingredients. Improving these results and considering the minimal parallelism also for other classes of P systems remain as an attractive research topics.

Another interesting combination of ingredients was considered in [23]: P systems with string-objects, processed by context-free rewriting rules (such systems are known not to be universal) without target indications, and with symport-like rules for transporting strings across membranes. These symport rules are of the form $(E, in)$ or $(E, out)$, where $E$ is a regular expression. The idea is that a string which belongs to $L(E)$ can be moved as indicated by the rule. Interesting enough, very particular forms of the regular expressions used in the transport rules suffice for universality; consider expressions of the next forms:

1. $E = U_1^* W U_2^*$ or $E = U^*$,
2. $E = U^* W$, $E = W U^*$, or $E = U^*$,
3. $E = U^*$,

where $U$, $U_1$, $U_2$ are alphabets and $W$ is a set of symbols.

Rewriting-symport systems with symport rules of types 1 and 2 characterize $RE$, and those with rules of type 3 characterize $ET0L$ – with the important mentioning that the universality is proved for systems with an arbitrary number of membranes, hence the problem arises to find a bound on this number.

This type of P systems is illustrative for a more general idea, of combining ingredients of classes of P systems which were never combined, such as string-objects with operations or rules used traditionally only for symbol-objects.

A sort of magical procedure in molecular computing (and nano-technology) is *self-assembly*, the self-running processes which build higher order structures from simple starting "bricks" – see more accurate definitions as well as references in [18].

Thus, self-assembly means first *self-* and then *-assembly*, which makes interesting to consider as a general case the self-running processes – as any computation in a P system can be considered: because there is no sequencing of instructions, like in a usual programming language, the rules from a region of a P system can be considered as a "program" which works in an opportunistic manner (data driven), thus, somewhat like in self-assembly. To be closer to the self-assembly philosophy, it is of interest to consider cases where the "elementary bricks" are as simple as possible, and this was the strategy recently followed in [11]. A class of population P systems were considered, with cells of a very restricted form (able of only two simple operations, multiset rewriting and objects exchange with neighboring cells, and with only one exit and one entering channel, which establish links between cells), which evolve in a sort of self-running/self-assembly manner and lead to a surprising emergent result (with universality again, for reduced values of parameters, e.g., the number of types of cells).

Related research is worth carrying out, as self-assembly can provide nice computing (and complex systems) ideas.

A sort of constant challenge in natural computing is to check in what extent classic levels of computability are "natural", they correspond to levels of computability defined in bio-inspired terms. The typical example is that of DNA computing by splicing, where regular languages and recursively enumerable languages have direct (and in many cases easy) characterizations, but this is not the case with other intermediate families of languages, such as the linear, context-free, and context-sensitive languages.

The situation is the same in membrane computing, with the additional difficulty (which sometimes makes the problem senseless) that we mainly work with sets of numbers and the length sets of regular, linear, and context-free languages (matrix languages without appearance checking included) are the same. However, when considering the result of a computation defined in the external mode, as the sequence of symbols which exit the system, we deal with strings, hence the problem becomes relevant whether or not languages of other types than the regular and the recursively enumerable ones can be characterized.

Recently, a characterization of context-sensitive languages was obtained in [13], using the following type of symport/antiport P systems: one considers accepting one-membrane systems, with an input alphabet $\Sigma \subseteq O$ containing a distinguished symbol \$ (the end marker), the environment containing all objects from $O - \Sigma$ (and no object from $\Sigma$), and rules of the following four types:

1. $(u, out; v, in)$, where $u, v \in (V - \Sigma)^*$ with $|u| \geq |v|$.
2. $(u, out; va, in)$, where $u, v \in (V - \Sigma)^*$ with $|u| \geq |v|$, and $a \in \Sigma$. A rule of this type is called a *read-rule*.
3. $(u, out; v, in)|_a$, where $u, v \in (V - \Sigma)^*$, and $a \in \Sigma$ ($a$ is a promoter). Note that there is no restriction on the relative lengths of $u$ and $v$. In particular, the length of $v$ can be greater than the length of $u$.

4. For every $a \in \Sigma$, there is at least one rule of the form $(a, out; v, in)$ in the set $R_1$, where $v \in (V - \Sigma)^*$. Moreover, this is the only type of rules for which $a$ can appear on the left part of the rule.

Such a system accepts a language in the following way. An input string $x = a_1 \dots a_n\$$, where $a_i$ is in $\Sigma - \{\$\}$ for $1 \leq i \leq n$, is provided in the environment. The symbols are brought into the system in the order they appear in $x$ by means of read-rules (i.e., rules of the form $(u, out; va, in)$), maybe several applied at the same step, due to the maximal parallelism of using the rules. Note that rules of types 1, 3, and 4 do not consume any input symbol from $x$ and that any symbol $a \in \Sigma$ that is imported from the environment by a rule of type 2 is always transported back to the environment in the following step by a rule of type 4. When a rule of type 4 is applied, the symbol $a$ that is exported to the environment does not get inserted to the input string, that is, it is never brought again in the system. A string is accepted if, after reading it completely, the computation eventually halts.

Systems of this type characterize the context-sensitive languages; if some restrictions are removed or certain changes are made in the definition, then universality results are obtained or characterizations of regular languages.

No characterization of linear or of context-free languages is given in [13]. This remains as a part of a general research topic: find characterizations of language families from Chomsky hierarchy, Lindenmayer hierarchies, Marcus contextual hierarchies, in terms of symport/antiport systems or of other types of P systems which generate/recognize languages.

## References

1. A. Alhazov, R. Freund, Y. Rogozhin: Computational power of symport/antiport: history, advances and open problems. *Membrane Computing. 6th Intern. Workshop, WMC2005, Vienna, Austria, July 2005. Revised Selected and Invited Papers*, LNCS 3850, Springer-Verlag, Berlin, 2006, 1–30.
2. F. Bernardini, M. Gheorghe, N. Krasnogor, J.-L. Giavitto: On self-assembly in population P systems. In *Proc. Unconventional Computing, UC05*, Sevilla, LNCS 3699, Springer, Berlin, 2005, 46–57.
3. N. Busi: On the computational power of the mate/bud/drip brane calculus: interleaving vs, maximal parallelism. *Membrane Computing. 6th Intern. Workshop, WMC2005, Vienna, Austria, July 2005. Revised Selected and Invited Papers*, LNCS 3850, Springer-Verlag, Berlin, 2006, 144–158.
4. N. Busi, R. Gorrieri: On the computational power of brane calculi. *Third Workshop on Computational Methods in Systems Biology*, Edinburgh, 2005.
5. L. Cardelli: Brane calculi. Interactions of biological membranes. In *Computational Methods in Systems Biology. International Conference CMSB 2004, Paris, France, May 2004, Revised Selected Papers* (V. Danos, V. Schachter, eds.), LNCS 3082, Springer-Verlag, Berlin, 2005, 257–280.
6. L. Cardelli, Gh. Păun: An universality result for a (mem)brane calculus based on mate/drip operations. *Cellular Computing. Complexity Aspects* (M.A. Gutierrez-Naranjo, Gh. Păun, M.J. Perez-Jimenez, eds.), Fenix Editora, Sevilla, 2005, 75–94, and *Intern. J. Found. Computer Sci.*, 17, 1 (2006), 49–68.

7. M. Cavaliere, A. Riscos-Nunez, R. Brijder, G. Rozenberg: Membrane systems with marked membranes. Submitted, 2005.
8. G. Ciobanu, L. Pan, Gh. Păun, M.J. Perez-Jimenez: P systems with minimal parallelism. Submitted, 2005.
9. G. Ciobanu, Gh. Păun, M.J. Pérez-Jiménez, eds.: *Applications of Membrane Computing*. Springer-Verlag, Berlin, 2006.
10. R. Freund, L. Kari, M. Oswald, P. Sosik: Computationally universal P systems without priorities: Two catalysts are sufficient. *Theoretical Computer Sci.*, 330, 2 (2005), 251–266.
11. M. Gheorghe, Gh. Păun: Computing by self-assembly: DNA molecules, polyominoes, cells. Submitted, 2005.
12. M.A. Guttierez-Naranjo, M.J. Perez-Jimenez, A. Riscos-Nunez, F.J. Romero-Campero: On the power of dissolution in P systems with active membranes. *Membrane Computing. 6th Intern. Workshop, WMC2005, Vienna, Austria, July 2005. Revised Selected and Invited Papers*, LNCS 3850, Springer-Verlag, Berlin, 2006, 224–240
13. O.H. Ibarra, Gh. Păun: Characterizations of context-sensitive languages and other language classes in terms of symport/antiport P systems. *Theoretical Computer Sci.*, 2006 (to appear).
14. O.H. Ibarra, Z. Dang, O. Egecioglu: Catalytic membrane systems, semilinear sets, and vector addition systems. *Theoretical Computer Sci.*, 312, 2-3 (2004), 378–400.
15. M. Ionescu, Gh. Păun, T. Yokomori: Spiking neural P systems. *Fundamenta Informaticae*, 71, 2-3 (2006), 279–308.
16. T.-O. Ishdorj: Minimal parallelism for polarizationless P systems. Submitted, 2006.
17. W. Korczynski: Păun's systems and accounting. *Proc. WMC6*, Vienna, July 2005, 461–464.
18. N. Krasnogor, S. Gustafson: A family of conceptual problems in the automated design of systems self-assembly. In *Proc. Second Intern. Conf. on the Foundations of Nanoscience: Self-Assembled Architectures and Devices*, 2005, 31–35.
19. S.N. Krishna: Universality results for a brane calculus. *Theoretical Computer Sci.*, 2006 (to appear).
20. W. Maass: Computing with spikes. *Special Issue on Foundations of Information Processing of TELEMATIK*, 8, 1 (2002), 32–36.
21. W. Maass, C. Bishop, eds.: *Pulsed Neural Networks*, MIT Press, Cambridge, 1999.
22. A. Păun, B. Popa: P systems with proteins on membranes. *Fundamenta Informaticae*, to appear.
23. A. Păun, B. Popa: Rewriting P systems with communication by symport rules. Submitted, 2006.
24. Gh. Păun: Computing with membranes (P systems): Twenty six research topics. *Auckland Univ., CDMTCS Report* No 119, 2000 (www.cs.auckland.ac.nz/CDMTCS)
25. Gh. Păun: *Membrane Computing. An Introduction*. Springer-Verlag, Berlin, 2002.
26. Gh. Păun: Further open problems in membrane computing: *Proc. Second Brainstorming Week on Membrane Computing*, Sevilla, 2004, TR 01/04 of Research Group on Natural Computing, Sevilla University, 2004, 354–365.
27. Gh. Păun: Further twenty six open problems in membrane computing. *Proc. Third Brainstorming Week on Membrane Computing*, Sevilla, 2005, TR 01/05 of Research Group on Natural Computing, Sevilla University, 2005, 249–262.
28. Gh. Păun, R. Păun: Membrane computing and economics: Numerical P systems. *Fundamenta Informaticae*, 2006.

29. Gh. Păun, R. Păun: Membrane computing as a framework for modeling economic processes. *Proc. SYNASC 05*, IEEE Press, 2005, 11–18.
30. Gh. Păun, M.J. Pérez-Jiménez, G. Rozenberg: Spike trains in spiking neural P systems. *Intern. J. Found. Computer Sci.*, 2006 (to appear).
31. Gh. Păun, M.J. Pérez-Jiménez, G. Rozenberg: Infinite spike trains in spiking neural P systems. Submitted, 2006.
32. D. Perrin, J.-E. Pin: *Infinite Words*. Elsevier, Amsterdam, 2004.
33. A. Rodriguez-Paton, P Sosik: Membrane computing and complexity theory: Characterization of PSPACE. Submitted, 2006.
34. G. Rozenberg, A. Salomaa: *The Mathematical Theory of L Systems*. Academic Press, New York, 1980.
35. N.J.A. Sloane, S. Plouffe: *The Encyclopedia of Integer Sequences*. Academic Press, New York, 1995.
36. The P Systems Web Page: `http://psystems.disco.unimib.it`.