Membrane Systems: An Introduction

Gheorghe Păun

Institute of Mathematics of the Romanian Academy
PO Box 1-764, 7014700 Bucureşti, 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
E-mail: `george.paun@imar.ro, gpaun@us.es`

Abstract. Membrane Computing (MC) is part of the powerful trend in computer science known under the name of Natural Computing. Its goal is to abstract computing models from the structure and the functioning of the living cell. The present paper is a short and informal introduction to MC, presenting the basic ideas, the central (types of) results, and the main directions of research.

1 Membrane Computing – Starting From Cells

In the last decade, the continuous and mutually beneficial collaboration of informatics with biology became simply spectacular. Two landmark examples are the completion of the genome project, a great success of bio-informatics, of using computer science in biology, and the successful Adleman's experiment (1994) of using DNA molecules as a support for computing. The latter example is illustrative for the direction of research opposite to the traditional one, of using computers in biology: in Adleman's experiment, biological materials and techniques were used in order to solve a computational problem. This was the "official birth certificate" of what is now called DNA Computing, and this gave a decisive impulse to *Natural Computing*.

Membrane Computing is the youngest branch of Natural Computing. It starts from the observation that one of the most marvellous machineries evolved by nature are the cells. The cell is the smallest living unit, a microscopic "enterprise", with a complex structure, an intricate inner activity, and an exquisite relationship with its environment. Both substances, from ions to large macromolecules, and information are processed in a cell, according to involved reactions, organized in a robust and at the same time sensitive manner, having as the goal the processes themselves, the life itself of the cell and of the structures where the cells are included – organs, organisms, populations.

Thus, a double challenge emerged: to check whether or not the often made statements about the "computations" taking place in a cell (see, e.g., [2] and [3]) are mere metaphoras or they correspond to computations in the standard (mathematical) understanding of this term, and, more ambitiously, having in mind the encouraging experience of other branches of Natural Computing, to get inspired from the structure and the functioning of the living cell and define new computing models, possibly of interest for computer science, for computability in general.

Membrane computing emerged as an answer to this double challenge, proposing a series of models (actually, a general framework for devising models) inspired from the cell structure and functioning, as well as from the cell organization in tissue. These models, called P systems, were investigated as mathematical objects, with the main goals being of a (theoretical) computer science type: computation power (in comparison with Turing machines and their restrictions), and usefulness in solving computationally hard problems. The field (founded in 1998; the paper [4] was first circulated on web) simply flourished at this level. Comprehensive information can be found in the web page at `http://psystems.disco.unimib.it`; see also [5].

In this paper we discuss only the cell-like P systems, whose study is much more developed than that of tissue-like P systems or of neural-like P systems, only recently investigated in more details.

In short, such a system consists of a hierarchical arrangement of *membranes* (understood as three-dimensional vesicles), which delimits *compartments* (also called *regions*), where abstract *objects* are placed. These objects correspond to the chemicals from the compartments of a cell, and they can be either unstructured, a case when they can be represented by symbols from a given alphabet, or structured. In the latter case, a possible representation of objects is by strings over a given alphabet. Here we discuss only the case of symbol-objects. Corresponding to the situation from reality, where the number of molecules from a given compartment matters, also in the case of objects from the regions of a P system we have to take into consideration their multiplicity, that is why we consider *multisets* of objects assigned to the regions of P systems. These objects evolve according to *rules*, which are also associated with the regions. The intuition is that these rules correspond to the chemical reactions from cell compartments and the reaction conditions are specific to each compartment, hence the evolution rules are localized. The rules say both how the objects are changed and how they can be moved (we say *communicated*) across membranes. By using these rules, we can change the *configuration* of a system (the multisets from its compartments); we say that we get a *transition* among system configurations. The way the rules are applied imitates again the biochemistry (but goes one further step towards computability): the reactions are done in *parallel*, and the objects to evolve and the rules by which they evolve are chosen in a *non-deterministic* manner, in such a way that the application of rules is maximal. A sequence of transitions forms a *computation*, and with computations which *halt* (reach a configuration where no rule is applicable) we associate a *result*, for instance, in the form of the multiset of objects present in the halting configuration in a specified membrane.

All these basic ingredients of a membrane computing system (a P system) will be discussed further below. This brief description is meant, on the one hand, to show the passage from the “real cell” to the “mathematical cell”, as considered in MC, and, on the other hand, to give a preliminary idea about the computing model we are investigating.

It is important to note at this stage the generality of the approach. We start from the cell, but the abstract model deals with very general notions: membranes interpreted as separators of regions, objects and rules assigned to regions; the basic data structure is the multiset; the rules are used in the non-deterministic maximally parallel manner, and in this way we get sequences of transitions, hence computations. In such terms, MC can be interpreted as a *bio-inspired framework for distributed parallel processing of multisets.*

We close this introductory discussion by stressing the basic similarities and differences between MC and the other areas of Natural Computing. All these areas start from biological facts and abstract computing models. Neural and Evolutionary Computing are already implemented (rather successfuly, especially in the case of Evolutionary Computing) on the usual computer. DNA Computing has a bigger ambition, that of providing a new hardware, leading to bio-chips, to "wet computers". For MC it seems that the most realistic attempt for implementation is *in silico* (this started already to be a trend and some successes are already reported) rather than *in vitro* (no attempt was made yet).

2 The Basic Classes of P Systems

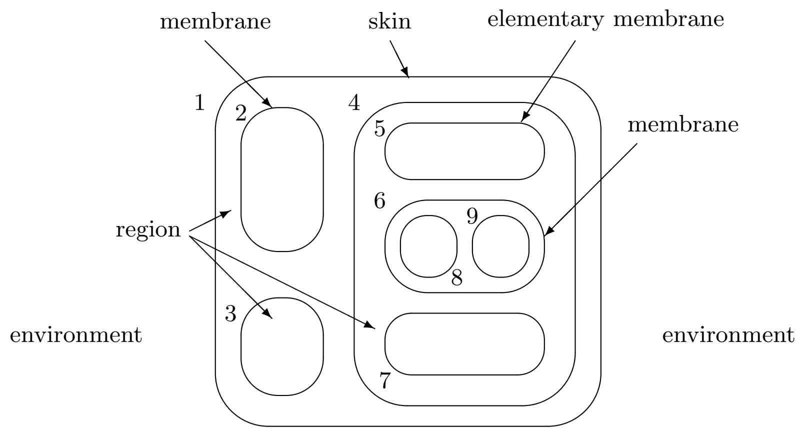
We introduce now the fundamental ideas of MC in a more precise way. What we look for is a computing device, and to this aim we need *data structures, operations* with these data structures, an *architecture* of our "computer", a systematic manner to define *computations* and *results* of computations.

Inspired from the cell structure and functioning, the basic elements of a *membrane system* (*P system*) are (1) the *membrane structure* and the sets of (2) *evolution rules* which process (3) *multisets* of (4) *objects* placed in the compartments of the membrane structure.

A membrane structure is a hierarchically arranged set of membranes. A suggestive representation is as in the figure from the next page. We distinguish the external membrane (corresponding to the plasma membrane and usually called the *skin* membrane) and several internal membranes (corresponding to the membranes present in a cell, around the nucleus, in Golgi apparatus, vesicles, etc); a membrane without any other membrane inside is said to be *elementary.* Each membrane uniquely determines a compartment, also called *region*, the space delimited from above by it and from below by the membranes placed directly inside, if any exists.

In the basic class of P systems, each region contains a multiset of symbol-objects, which correspond to the chemicals swimming in a water solution in a cell compartment; these chemicals are considered here as unstructured, that is why we describe them by symbols from a given alphabet.

The objects evolve by means of evolution rules, which are also localized, associated with the regions of the membrane structure. The rules correspond to the chemical reactions possible in the compartments of a cell. The typical form of such a rule is $aad \to (a, here)(b, out)(b, in)(b, in)$, with the following meaning: two copies of object a and one copy of object d react and the reaction produces one copy of a and three copies of b; the new copy of a remains in the same region (indication *here*), one of the copies of b exits the compartment, going to the surrounding region (indication *out*) and the others two enter one or two of the directly inner membranes (indication *in*). We say that the objects a, b, b, b are *communicated* as indicated by the commands associated with them in the right hand member of the rule. When an object exits a membrane, it will go to the surrounding compartment; in the case of the skin membrane this is the environment, hence the object is "lost", it never comes back into the system. If no inner membrane exists (that is, the rule is associated with an elementary membrane), then the indication *in* cannot be followed, and the rule cannot be applied.



The communication of objects through membranes reminds the fact that the biological membranes contain various (protein) channels through which the molecules can pass (in a passive way, due to concentration difference, or in an active way, with a consumption of energy), in a rather selective manner. The fact that the communication of objects from a compartment to a neighboring compartment is controlled by the "reaction rules" is attractive mathematically, but not quite realistic from a biological point of view, that is why there also were considered variants where the two processes are separated: the evolution is controlled by rules as above, without target indications, and the communication is controlled by specific rules (by symport/antiport rules – see below).

A rule as above, with several objects in its left hand member, is said to be *cooperative*; a particular case is that of *catalytic* rules, of the form $ca \to cx$, where a is an object and c is a catalyst, appearing only in such rules, never changing. A rule of the form $a \to x$, where a is an object, is called *non-cooperative.*

The rules associated with a compartment are applied to the objects from that compartment, in a *maximally parallel way*: all objects which can evolve by means of local rules should do it (we assign objects to rules, until no further assignment is possible). The used objects are “consumed”, the newly produced objects are placed in the compartments of the membrane structure according to the communication commands assigned to them. The rules to be used and the objects to evolve are chosen in a non-deterministic manner. In turn, all compartments of the system evolve at the same time, synchronously (a common clock is assumed for all membranes). Thus, we have two layers of parallelism, one at the level of compartments and one at the level of the whole “cell”.

A membrane structure and the multisets of objects from its compartments identify a *configuration* of a P system. By a non-deterministic maximally parallel use of rules as suggested above we pass to another configuration; such a step is called a *transition.* A sequence of transitions constitutes a *computation.* A computation is successful if it halts, it reaches a configuration where no rule can be applied to the existing objects. With a halting computation we can associate a *result* in various ways. The simplest possibility is to count the objects present in the halting configuration in a specified elementary membrane; this is called *internal output.* We can also count the objects which leave the system during the computation, and this is called *external output.* In both cases the result is a number. If we distinguish among different objects, then we can have as the result a vector of natural numbers. The objects which leave the system can also be arranged in a sequence according to the moments when they exit the skin membrane, and in this case the result is a string. This last possibility is worth emphasizing, because of the qualitative difference between the data structure used inside the system (multisets of objects, hence numbers) and the data structure of the result, which is a string, it contains a positional information, a syntax.

Because of the non-determinism of the application of rules, starting from an initial configuration, we can get several successful computations, hence several results. Thus, a P system *computes* (one also uses to say *generates*) a set of numbers, or a set of vectors of numbers, or a language.

Of course, the previous way of using the rules from the regions of a P system reminds the non-determinism and the (partial) parallelism from cell compartments, with the mentioning that the maximality of parallelism is mathematically oriented (rather useful in proofs); when using P systems as biological models, this feature should be replaced with more realistic features (e.g., reaction rates, probabilities, partial parallelism).

An important way to use a P system is the automata-like one: an *input* is introduced in a given region and this input is *accepted* if and only if the computation halts. This is the way for using P systems, for instance, in solving decidability problems.

We do not give here a formal definition of a P system. The reader interested in mathematical and bibliographical details can consult the mentioned monograph [5], as well as the relevant papers from the web bibliography mentioned above. Of course, when presenting a P system we have to specify: the alphabet of objects, the membrane structure (usually represented by a string of labelled matching parentheses), the multisets of objects present in each region of the system (represented by strings of symbol-objects, with the number of occurrences of a symbol in a string being the multiplicity of the object identified by that symbol in the multiset represented by the considered string), the sets of evolution rules associated with each region, as well as the indication about the way the output is defined.

Many modifications/extensions of the very basic model sketched above are discussed in the literature, but we do not mention them here. Instead, we only briefly discuss the interesting case of *computing by communication.*

In the systems described above, the symbol-objects were processed by multiset rewriting-like rules (some objects are transformed into other objects, which have associated communication targets). Coming closer to the trans-membrane transfer of molecules, we can consider purely communicative systems, based on the three classes of such transfer known in the biology of membranes: *uniport, symport*, and *antiport* (see [1] for details). Symport refers to the transport where two (or more) molecules pass together through a membrane in the same direction, antiport refers to the transport where two (or more) molecules pass through a membrane simultaneously, but in opposite directions, while the case when a molecule does not need a "partner" for a passage is referred to as uniport.

In terms of P systems, we can consider object processing rules of the following forms: a symport rule (associated with a membrane i) is of the form (ab, in) or (ab, out), stating that the objects a and b enter/exit together membrane i, while an antiport rule is of the form $(a, out; b, in)$, stating that, simultaneously, a exits and b enters membrane i. A natural generalization is to move more than two objects simultaneously, for instance, considering antiport rules of the form $(x, out; y, in)$, where x, y are arbitrary multisets of objects.

A P system with symport/antiport rules has the same architecture as a system with multiset rewriting rules: alphabet of objects, membrane structure, initial multisets in the regions of the membrane structure, sets of rules associated with the membranes, possibly an output membrane – with one additional component, the set of objects present in the environment. This is an important detail: because by communication we do not create new objects, we need a supply of objects, in the environment, otherwise we are only able to handle a finite population of objects, those provided in the initial multiset. Also the functioning of a P system with symport/antiport rules is the same as for systems with multiset rewriting rules: the transition from a configuration to another configuration is done by applying the rules in a non-deterministic maximally parallel manner, to the objects available in the regions of the system and in the environment, as requested by the used rules. When a halting configuration is reached, we get a result, in a specified output membrane.

3 Computational Completeness; Universality

As we have already mentioned, many classes of P systems, combining various ingredients as those described above, are able to simulate Turing machines, hence they are *computationally complete.* Always, the proofs of results of this type are constructive, and this have an important consequence from the computability point of view: there are *universal* (hence *programmable*) P systems. In short, starting from a universal Turing machine (or an equivalent universal device), we get an equivalent universal P system. Among others, this implies that in the case of Turing complete classes of P systems, the hierarchy on the number of membranes always collapses (at most at the level of the universal P systems). Actually, the number of membranes sufficient in order to characterize the power of Turing machines by means of P systems is always rather small.

We only mention here two of the most interesting universality results:

1. P systems with symbol-objects with catalytic rules, using only two catalysts and two membranes, are universal.

2. P systems with symport/antiport rules of a rather restricted size (example: three membranes, symport rules with two objects each and no antiport rules, or only minimal symport and antiport rules) are universal.

We can conclude that the compartmental computation in a cell-like membrane structure (using various ways of communicating among compartments) is rather powerful. The "computing cell" is a powerful "computer".

4 Computational Efficiency

The computational power is only one of the important questions to be dealt with when defining a new computing model. The other fundamental question concerns the computing *efficiency*. Because P systems are parallel computing devices, it is expected that they can solve hard problems in an efficient manner – and this expectation is confirmed for systems provided with ways for producing an exponential workspace in a linear way. Three main such possibilities have been considered so far in the literature, and *all of them were proven to lead to polynomial solutions to* **NP***-complete problems*: *membrane division*, *membrane creation*, and *string replication*. Using them, polynomial solutions to `SAT`, the Hamiltonian Path problem, the Node Covering problem, the problem of inverting one-way functions, the Subset-sum, and the Knapsack problems were reported (note that the last two are numerical problems, where the answer is not of the yes/no type, as in decidability problems). Details can be found in [5], [6], as well as in the web page of the domain.

Roughly speaking, the framework for dealing with complexity matters is that of *accepting P systems with input*: a family of P systems of a given type is constructed starting from a given problem, and an instance of the problem is introduced as an input in such systems; working in a deterministic mode (or a *confluent* mode: some non-determinism is allowed, provided that the branching converges after a while to a unique configuration), in a given time one of the answers yes/no is obtained, in the form of specific objects sent to the environment. The family of systems should be constructed in a uniform mode (starting from the size of instances) by a Turing machine, working a polynomial time.

This direction of research is very active at the present moment. More and more problems are considered, the membrane computing complexity classes are refined, characterizations of the **P**$\neq$**NP** conjecture were obtained in this framework, improvements are looked for. An important recent result concerns the fact that **PSPACE** was shown to be included in $\mathbf{PMC}_D$, the family of problems which can be solved in polynomial time by P systems with the possibility of dividing both elementary and non-elementary membranes [7].

5 Concluding Remarks

This paper was intended as a quick and general introduction to Membrane Computing, an invitation to this recent branch of Natural Computing.

The starting motivation of the area was to learn from the cell biology new ideas, models, paradigms useful for informatics – and we have informally presented a series of details of this type. The mathematical development was quite rapid, mainly with two types of results as the purpose: computational universality and computational efficiency. Recently, the domain started to be used as a framework for modelling processes from biology (but also from linguistics, management, computer graphics, etc.), and this is rather important in view of the fact that P systems are (reductionistic, but flexible, easily scallable, algorithmic, intuitive) models of the whole cell; modelling the whole cell was often mentioned as an important challenge for the bio-computing in the near future – see, e.g., [8].

We have recalled only a few classes of P systems and only a few (types of) results. A detailed presentation of the domain is not only beyond the scope of this text, but also beyond the dimensions of a monograph; furthermore, the domain is fastly emerging, so that, the reader interested in any research direction, a more theoretical or a more practical one, is advised to follow the developments, for instance, through the web page mentioned in Section 2.

References

1. B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, P. Walter, *Molecular Biology of the Cell*, 4th ed., Garland Science, New York, 2002.
2. D. Bray, Protein Molecules as Computational Elements in Living Cells. *Nature*, 376 (July 1995), 307–312.
3. S. Ji, The Cell as the Smallest DNA-based Molecular Computer, *BioSystems*, 52 (1999), 123–133.
4. Gh. Păun, Computing with Membranes, *Journal of Computer and System Sciences*, 61, 1 (2000), 108–143 (and Turku Center for Computer Science-TUCS Report 208, November 1998, `www.tucs.fi`).
5. Gh. Păun, *Computing with Membranes: An Introduction*, Springer, Berlin, 2002.
6. M. Pérez-Jiménez, A. Romero-Jiménez, F. Sancho-Caparrini, *Teoría de la Complejidad en Modelos de Computatión Celular con Membranas*, Kronos, Sevilla, 2002.

7. P. Sosik, The Computational Power of Cell Division in P Systems: Beating Down Parallel Computers? *Natural Computing*, 2, 3 (2003), 287–298.
8. M. Tomita, Whole-Cell Simulation: A Grand Challenge of the 21st Century, *Trends in Biotechnology*, 19 (2001), 205–210.