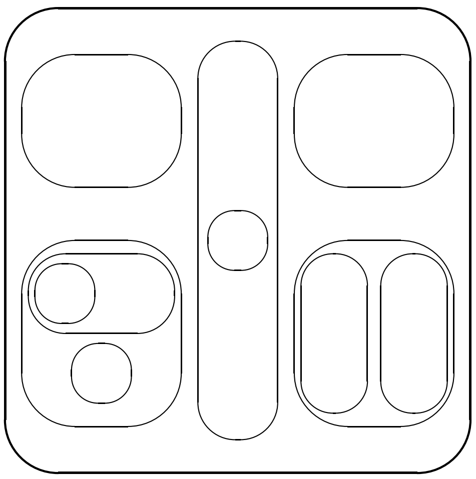
Brainstorming Workshop on Uncertainty in Membrane Computing

Proceedings

Palma de Mallorca, November 8-10, 2004



© del text: els autors
© de la edició: Departament de Matemàtiques i Informàtica de la Universitat de les Illes Balears
Edició: Departament de Matemàtiques i Informàtica. Universitat de les Illes Balears. Campus Universitari. 07122 Palma de Mallorca

ISBN: 84–7632–897–4
DL: PM1650–2004

Brainstorming Workshop on Uncertainty in Membrane Computing

Departament de Matemàtiques i Informàtica
Escola Universitària Politècnica
Universitat de les Illes Balears
Palma de Mallorca, November 8 – 10, 2004

Program Committee
Antonio di Nola
Georghe Păun
Adam Obtułowicz
Mario Pérez-Jiménez
Francesc Rosselló

Organizing Committee
Ricardo Alberich
Jaume Casasnovas
Mercè Llabrés
José Miró-Juliá
Jairo Rocha
Francesc Rosselló

Preface

Since its very inception as a computational model mimicking the behavior of cells, it was apparent the interest of including uncertainty and approximation into membrane computing models. Recall Gheorge Păun's words in his first list of problems, dated October 2000:

> *Membrane Computing comes from biology and in biology and biochemistry the processes are nondeterministic, the result is only approximately/probabilistically true. Up to now, only 'crisp' mathematics was used in P systems area (the same is in a great extent true also for DNA Computing). What about 'approximate' mathematical approaches, using probabilities, fuzzy sets, or rough set theory? [...] What about 'approximate' computing, whatever this can mean?*

Several approaches to membrane computing under uncertainty have been proposed lately, based indeed on probabilities, rough sets, and fuzzy sets, and the topic seemed mature enough to motivate a meeting that would bring together researchers from different groups interested in the specific problem of including uncertainty into membrane computing.

The University of the Balearic Islands hosted the *Brainstorming Workshop on Uncertainty in Membrane Computing* from November 8 to 10 2004 with this aim. About 30 people coming from universities in Austria, England, Italy, Japan, Poland, Romania, and Spain gathered for three days in an informal atmosphere, similar to that of the annual Brainstorming Week on Membrane Computing.

The first day was devoted to tutorials, and during the other two days some participants presented their work in progress. This volume collects the written versions of some of the tutorials and the presentations. It also includes a paper by Solomon Marcus on uncertainty written specially for this volume, which we warmly thank.

This workshop was sponsored by the Ministry of Science and Education of Spain, the Department of Mathematics and Computer Science, and the Polytechnic School of the UIB. These proceedings have been published by the Department of Mathematics and Computer Science of the UIB.

Palma, November 2004

Contents

(Imprecise Topics about) Handling Imprecision in P Systems

Antonio Di Nola[1], Gheorghe Păun[2,3], Mario J. Pérez-Jiménez[2], Francesc Rosselló[4]

[1]Department of Mathematics and Computer Science
University of Salerno, 84081 Baronissi, Salerno, Italy
E-mail: adinola@unisa.it

[2]Institute of Mathematics of the Romanian Academy
PO Box 1-764, 014700 Bucureşti, Romania
E-mail: george.paun@imar.ro

[3]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: {gpaun, marper}@us.es

[4]Dept. Mathematics and Computer Science
Research Insitute of health Science (IUNICS)
University of the Balearic Islands, 07122, Palma de Mallorca, Spain
E-mail: cesc.rossello@uib.es

A mathematician from an office placed in the Sevilla central building of the university proudly says to one placed in his office from the southern Sevilla building of t he university:

– Look, I am closer than you to Palma de Mallorca!

He is right, but a biologist placed in Palma de Mallorca smiles, because (s)he s ees no difference...

1 (A Sort of) Introduction

The standard P systems are beautiful mathematical toys (maybe of interest for computer scientists, linguists, etc) which can easily get a smile from Palma-de-Mallorca-of-biologists. Working with multisets, hence precisely counting the objects from the compartments of a system (a cell), assuming a universal clock, which ticks uniformly for all compartments, and using the rules (reactions) in the maximal parallel manner, or, the other extreme case, in the sequential manner, are three features which make the life of mathematicians easier and nicer, but which are science fiction for today biologists. The specification *today* suggests that *tomorrow* can change the things. After all, more than one biologist is convinced that cellular processes are exact and deterministic, and that it is our knowledge that is inexact and incomplete, and maybe someday this knowledge will improve. However, waiting for tomorrow is not always a good strategy (for sure, not for our employers. . .), so that a sort of obsession wanders the science in general and membrane computing in particular: to become more and more realistic! Many papers are motivated in these terms, most of them succeeding to move (at most? at least?) from the southern building of Sevilla university to the central one (we do not mention them in the bibliography which closes this note, but only those which succeed – or at least promise – bigger steps; for the general bibliography of the area, the reader can consult the known web address `http://psystems.disco.unimib.it`), and only part of them already greeting the biologist from the Valencia beaches. . .

The problem (difficulty) is not that numbers are no longer sure things, thus contradicting Galileo, but that *reality is imprecision* (and complexity, but this is a related but different topic). Even if God does not play dice (how could Einstein know it?! and, what did he mean with that sentence, actually?), for us, the humans, dices are indispensable. And not only dices (probabilities), but also many other sources/forms of uncertainty, in most cases originating in the partial knowledge we have about processes, phenomena, systems we want to understand. Probability, partial information, fuzzyness, randomness, stochasticity, ambiguity, noise (not to mention incompleteness, undecidability, intractability) are only a few of the main terms related to this issue.

Coping with uncertainty is not only a challenge coming from "reality", from practitioners – in our case, mainly from biology. Biology is invoked here (in general, in membrane computing) mainly because this is the field from where the membrane computing is inspired and where it promises to return in the near future tools and applications relevant for biologists. However, membrane computing started as a branch of (theoretical) computer science, with the aim of learning something useful, or at least intellectually interesting, for computer science from the study of the (structure and functioning of the) living cell. The initial goal had nothing related to any promise to biologists. And yet at this theoretical level the challenge to deal with various forms of uncertainty appears. Because mathematics has developed several tools (theories) for handling uncertainty —probability theory, with many branches, fuzzy set theory, rough set theory, approximate reasoning and approximate algorithmics, etc.— it is a natural task for the mathematician to bring such tools in membrane computing, with or without having in mind (and motivating the papers by) how realistic the models are, from a biological or non-biological point of view.

We want to add here a word of caution for those intending to be realistic. There are indeed many mathematical approaches to uncertainty, and it is mathematically correct to develop an "uncertain" formal computational model by mixing any previous "crisp" computational model with any one one of these approaches, but some of these models may not be sound from the point of view of real world uncertainty modelling.

For instance, a fuzzy set $F : X \to [0, 1]$ (or with range any frame) assigns to each element of the (crisp) set X "the value with which this element belongs to the set F," but it can also be understood, dually, as assigning to each element of X "its value of having property F" or even "its value of approximating the element F." In this way, fuzzy set theory appears as suitable to work with mixtures, non-crisp properties, and degrees of approximation, but not to work with other types of uncertainties. Thus, for instance, our lack of knowledge of the place where a molecule is at a given moment shouldn't be modelled by means of a raw fuzzy set, but using the probability values of all possible places. Other approaches, like possibility theory of belief theory, can be used here, but then they model more than simply our "lack of knowledge." And although a probability distribution is a fuzzy set, it is not sound to use general fuzzy set theory methods, like aggregation techniques or distances between fuzzy sets, to handle probabilities. On its turn, rough set theory approximates crisp sets "from below" and "from above," and it can be used in a natural way to model approximations set-theoretically, but not numerically. And so on...

This long (and, admittedly, imprecise) discussion is intended to stress the fact that the topics/suggestions mentioned below are not necessarily meant to "bring P systems closer to biology" (sometimes, one even writes "back to biology"), although such a goal is implicit and it would be a nice "by-product" of the possible results obtained in the study of these topics/suggestions. The questions formulated below are just natural from a mathematical point of view (although in their formulation we will use biological motivation/metaphora).

Then, an important point we want to make is the fact that this note is explicitly meant to foster discussions, researches, collaborations during the Brainstorming Workshop on Uncertainty in P Systems, Palma de Mallorca, November 2004. This is not a research paper, is only a positional paper, a provocation to the participants in the meeting. The choice of issues is subjective, their ordering has no significance (of importance), the list is not meant to be exhaustive, the classification below is approximate. And, of course, many formulations are imprecise enough; already formulating in a rigorous manner such topics would be a matter of investigation.

Finally, a warning/precaution related to the bibliography: we have mentioned many titles at the end, all we know in this moment in membrane computing area related to the topic of this discussion, but we will cite very few of them in the text, although many of them are directly related to the issues we discuss. Also, we do not mention any book or paper related to the mathematical approaches to uncertainty/imprecision, for instance, about fuzzy or rough set theories; there are huge bibliographies (on the web) about these topics and the reader can easily find such an information.

2 The Identification of Objects

Let us start "from inside", from the objects swimming and evolving in the compartments of a membrane structure.

Are we (always) entitled to say that the object a is in region i? Actually, what means "the object" a? If we "see" a molecule x, how much are we sure that it is of type a and not of type b? What about considering estimations for x to be of one of the types from a given list, for instance, expressed in the form of probabilities? We can then discuss about objects with probability .8 to be a and .2 to be b. This can be also a matter of possibilities, leading to the use of possibility values and theory instead of probability ones, of beliefs, closer to fuzzy set theory, or of similarities, which lead to classification in terms of rough sets theory. We can go further (making the life of the mathematician still harder), working with objects x identified by statements of the form "x is an object from the set $\{a, b, c\}$ with probability .66 and from the set $\{d, e\}$ with probability .34." The two sets are here disjoint, but in a general case they can be not. What can we do with additional statements of the form "with probability 1, x is not from the set $\{a, f\}$?"

We have here also another sensitive issue, related to the fact that in the compartments of a P system we deal with *copies* of objects. What is a copy? (S. Marcus posed somewhere a much more dramatic question: do there exist copies?) If we cannot precisely identify an object, then we cannot precisely say when one object is a copy of the other (hence that they are indistinguishable, for instance, for the evolution rules). Can we work with *similar* objects, instead of *identical* objects? Equality is an equivalence relation, similarity is only a tolerance relation (it is not transitive), leading to tolerance classes which are not necessarily disjoint. This raises problems when handling the objects, as they are not "crispy" classified. Should we return to numerical estimations of the similarity, which could be used for instance to model inexact, mutated or simply modified copies, or should we remain in a qualitative framework (e.g., working with a similarity relation)?

And finally, do always these questions matter? For instance, in molecular biology, methylated DNA can act in some reactions as usual, non-methylated DNA, but it may lead to errors (with a certain probability!) in other reactions, yielding inexact results, and it cannot be involved at all in other reactions that need definitely non-methylated DNA molecules. Should we impose consequently that some distinctions, or similarities, matter as far as some rewriting rules are concerned, but not for the remaining ones?

3 The Place of Objects

After identifying the objects, we have to place them in the compartments of a membrane structure, maybe also taking into account the environment, and this is again a rich source of imprecision. Even if we try in a laboratory to introduce ourselves a molecule in a compartment of a cell, we cannot be always sure that the result is the one we expect; still more difficult is to fish for a molecule in a given compartment. Furthermore, many chemicals from the cell are macromolecules, maybe long chains of atoms, which can be placed with an end in a compartment and the other end in another compartment (like the many proteins embedded in the membranes). We have also to mention that the molecules move continuously across membranes.

In short, it makes sense to assign probabilities (estimations) to the fact that a given object is placed in a given compartment, and this is directly related to the next point.

4 Describing the Multisets

Which objects are at a given time in a given compartment, and how many copies of each? From the probabilities assigned to the presence of objects in a given compartment we can have a probabilistic estimation of the contents of that compartment. Maybe more natural is to have fuzzy set estimations or rough set approximations of multisets from compartments (at least, because the general study of fuzzy and rough multisets is well developed and can provide tools to use in our area).

Rough set theory looks particularly attractive, because the basic issue of this theory is to approximate a set by an upper and a lower approximation, the latter one *surely* included in the set, the former one *surely* including the set, in between having a border, as larger as higher the imprecision is. By enlarging the available information, the two approximations can converge to the real set, thus making smaller and smaller the border. How this attractive idea can be used in a P system? For instance, which rules should be used for evolving the objects from the border? Should they participate in the same (cooperative) rules with objects from the lower approximation and/or from outside the upper approximation?

On its turn, fuzzy set theory can be used to represent statements like "there are around seven copies of reactive a in that membrane," by defining, for instance, the content of a membrane as a mapping sending each reactive a to a fuzzy natural number, i.e., a mapping $n_a : \mathbb{N} \to [0,1]$ that has some suitable properties.

Other, not so well-know approaches already introduced to handle approximately defined quantities, or specially tailored *a posteriori*, can be used to define generalized multisets that model other kinds of uncertain contents of membranes. For instance, techniques imported from interval calculus can be used to manipulate multisets sending a reactive to, say, "somewhere between 5 and 9."

Of course, we can try to learn something (or to get challenges) from the biologist or from the bio-chemist. On the one hand, the former one will push us towards linguistic logic, because (s)he currently works with statements using such terms as "many molecules", "sufficiently large population", "high enough pressure," and so on. Fuzzy set theory can be used in this connection, since statements like “there are many copies of reactive a in that membrane” correspond to define the multisets that describe the membranes contents as mappings that send each reactive to a linguistic variable, and these multisets can be manipulated using techniques imported from fuzzy control theory.

On the other hand, bio-chemists work with probabilities, reaction rates, concentrations, gradients, stoichiometric constants, etc.. All these mean real (well, rational) numbers, associated both with the contents of membranes and with the reactions taking place in/on the membranes. Multisets with real multiplicities associated with objects have been already considered, e.g., in [9, 15, 16]. In particular, the first cited paper started a systematic study of P systems with non-discrete multisets, but the topic is far from being exhausted.

It should be mentioned that in most biological applications of P systems reported so far, the rules have associated probabilities/reaction rates, sometimes dynamically computed, in accordance with the current population of objects, using standard techniques from biochemistry (e.g., stoichiometric constants), which, interestingly enough, brings continuous mathematics aspects in the functioning of P systems, which, in the basic version, are essentially discrete machineries.

5 The Level of Rules

All the previous sources of uncertainty and ideas about ways to capture/handle them in P systems at the level of objects and multisets have a direct connection with the way the evolution rules are defined and applied. How precise is a rule defined? Are the multisets from its left and right hand members precise or not? If not, what this means? What about the relation between the left and the right hand multisets, whatever their definition is? Furthermore, in the case of multiset-rewriting rules, where targets are associated with the objects newly introduced by a rule, we can question the precision of these targets, and associate with each object all targets, with probabilities assigned to them: for instance, something like $(a : here.5, out.2, in.3)$. In this way, probabilities are assigned to the presence of an object in a given compartment, hence even if we start from a precisely known system, after a while the place of objects will be only probabilistically known.

Besides, are the reactions the same at different moments, or their result also depends on parameters other than the contents of the compartment where they are used?

When a rule can/should be used? This has to do with the probability for a reaction to take place, which, in turn, depends on the concentration of reactants, but also on reaction conditions (temperature, pH, available energy, etc). We do not repeat the previous discussion about reaction rates, stoichiometry, etc., but these terms are highly relevant here. An important issue concerns the relationship between non-determinism and various forms of uncertainty; intuitively, by assigning probabilities (rates of reaction) to rules, we diminish the non-determinism in using the rules, as the probability induces some priority relation among rules. This topic deserves a more detailed examination, for instance, in relation with the resolution of computationally hard problems by means of P systems.

Finally: if the multisets of objects from compartments are fuzzy or rough sets, what about applying to them sets (or multisets) of rules which are also fuzzy or rough?

6 The Clock and the Parallelism

These two issues are related. Without an external clock it is difficult (but probably not impossible) to define the functioning and the result of a P system, but assuming that all compartments have the same clock and that all rules/reactions last the same amount of time is far from... Palma de Mallorca. Getting rid of the internal clock is a great topic and the first results started to appear, see, e.g., [7, 8, 22]. "No clock" does not necessarily mean "no internal time" (for instance, no known duration of rules), but the internal time can be different from a compartment to another one, with rules of different durations, maybe expressed in non-integer numbers, maybe simply unknown. The collaboration between (cooperative) rules can be achieved through the objects they produce, the rules can also be synchronized by signals or promoters/inhibitors, with a great flexibility in what concerns the moment where the necessary objects become available.

The question of time is directly connected to that of parallelism. Maximal parallelism is powerful (e.g., because it can provide information about the whole multiset from a compartment), the sequential use of rules is easy to handle, but the truth is somewhere *in media res*. How to deal with "partial parallelism," what this means and how can it be estimated?

7 Other (Related) Issues

Of course, there are many other things to discuss in this framework. We have said nothing explicit about the environment, which also can be described in imprecise terms. Then, we mentioned above only multiset-rewriting rules, but the same issues can be formulated for symport and antiport rules, which move multisets (of which type?) of objects (how precisely known?) from a region to another one (how precisely defined?) and that can even degenerate (with some probability?) with each movement.

In the previous sections we have talked about imprecision in general, without mentioning possible *degrees of imprecision*. Given a system, can we evaluate its degree of imprecision? Given two systems, can we say that one of them is better identified than the other one? (Of course, such a comparison asks for an external observer —a crisp one, maybe.) In what terms, using which measures? Maybe entropy, maybe other criteria. When a system contains "too much" imprecision, so that it makes no sense to further work with it, because the information we get is irrelevant (non trustful)? How the degree of imprecision can be decreased, what is the information we can get and which we have to look for in order to improve the knowledge about a system? Working with degrees of approximation and the asymptotic convergence of approximations to the set one looks for, are standard issues in rough set theory; it remains to implement them also in P systems.

The previous topic is related to using approximation in an operational manner, for instance, in terms of probabilistic or randomized algorithms. If we cannot solve a problem with reasonable resources (polynomial space and time), then let us try to have an approximate solution, with a well estimated degree of accuracy, or an optimal solution which is however found only with some precisely estimated chance, but using reduced resources (larger the resources, bigger the probabilities to get a good solution or to get a solution at all). In membrane computing the challenge is clear: "solving" hard problems in polynomial time, but not using an exponential workspace (even created in the natural way provided by membrane division or string replication), but a polynomial workspace, by paying in the accuracy of the certainty of the solution. The question waits to be systematically approached.

Finally, there are many other sources of imprecision. Non-determinism and confluence (in the strong sense, all configurations lead to a unique configuration, or in the weak sense, all configurations lead to configurations from a given class, having a specified property) are standard properties of P systems. How are they related to imprecision? What means in our context ambiguity or synonymy, to mention only two sources of imprecision from linguistics?

We hope that the reader will take the turn and address these questions (at least by reformulating them in mathematical terms) or related ones, thus moving the area closer to reality/biology (and the reader her-himself to Palma...).

References

[1] I.I. Ardelean, M. Cavaliere, Modelling biological processes by using a probabilistic P system software, *Natural Computing*, 2, 2 (2003), 173–197.

[2] I. Ardelean, M. Cavaliere, Playing with a probabilistic P system simulator: Mathematical and biological problems, *Brainstorming Week on Membrane Computing*, Tarragona, February 2003, TR 26/03, URV, 2003, 37–45.

[3] D. Besozzi, C. Zandron, Dynamical probabilistic P systems, DNA10 (poster?).

[4] M. Buzzi, *Calcolo con membrane. P sistemi probabilistici*, Master Thesis, Univ. of Como, 2003.

[5] J. Casasnovas, F. Rosselló, Scalar and fuzzy cardinalities of crisp and fuzzy multisets, submitted, 2003.

[6] J. Casasnovas, J. Miró, M. Moyà, F. Rosselló, An approach to membrane computing under inexactitude, *Intern. J. Foundations of Computer Sci.*, to appear.

[7] M. Cavaliere, Towards asynchronous P systems, *Pre-proceedings of Fifth Workshop in Membrane Computing, WMC5*, Milano, Italy, 2004, 161–173.

[8] M. Cavaliere, D. Sburlan, Time-independent P systems, *Membrane Computing. International Workshop WMC5, Milano, Italy, 2004*, LNCS ??, Springer, 2005.

[9] A. Cordón-Franco, F. Sancho-Caparrini, Non-discrete P systems, *Pre-proceedings of Fifth Workshop in Membrane Computing, WMC5*, Milano, Italy, 2004, 205–207.

[10] R. Freund, Asynchronous P systems, *Pre-proceedings of Fifth Workshop in Membrane Computing, WMC5*, Milano, Italy, 2004, 12–28.

[11] V. Manca, On the dynamics of P systems, *Pre-proceedings of Fifth Workshop in Membrane Computing, WMC5*, Milano, Italy, 2004, 29–43.

[12] S. Marcus, Tolerance multisets, *Multiset Processing. Mathematical, Computer Science and Molecular Computing Points of View*, LNCS 2235, Springer, 2001, 217–223.

[13] S. Miyamoto, Fuzzy multisets and their generalizations, *Multiset Processing. Mathematical, Computer Science and Molecular Computing Points of View*, LNCS 2235, Springer, 2001, 225–236.

[14] M. Mutyam, Probabilistic rewriting P systems, *Int. J. Found. Computer Sci.*, 14, 1 (2003), 157–166.

[15] T.Y. Nishida, Multiset and K-subset transforming systems, *Pre-proc. Workshop on Multiset Processing*, Curtea de Argeş, Romania, TR 140, CDMTCS, Univ. Auckland, 2000, 193–202, and *Multiset Processing. Mathematical, Computer Science and Molecular Computing Points of View*, LNCS 2235, Springer, 2001, 255–266.

[16] T.Y. Nishida, Simulation of photosynthesis by a K-subset transforming system with membranes, *Pre-proc. Workshop on Membrane Computing*, Curtea de Argeş, 2001, *Technical Report* 17/01 of RGML, URV, Tarragona, 223–228, and *Fundamenta Informaticae*, 49, 1-3 (2002), 249–259.

[17] A. Obtułowicz, Probabilistic P systems, *Pre-proceedings of Workshop on Membrane Computing*, Curtea de Argeş, Romania, August 2002, MolCoNet Publication No 1, 2002, 331–332, and LNCS 2597, Springer, 2003, 377–387.

[18] A. Obtułowicz, Mathematical models of uncertainty with a regard to membrane systems, *Brainstorming Week on Membrane Computing*, Tarragona, February 2003, TR 26/03, URV, 2003, 241–246, and *Natural Computing*, 2, 3 (2003), 251–263.

[19] A. Obtułowicz, General multi-fuzzy sets and fuzzy membrane systems, *Pre-proceedings of Fifth Workshop in Membrane Computing, WMC5*, Milano, Italy, 2004, 316–326.

[20] A. Obtułowicz, Gh. Păun, (In search of) Probabilistic P systems, *BioSystems*, 70, 2 (2003), 107–121.

[21] F. Sancho-Caparrini, A note on complexity measures for probabilistic P systems, *Proceedings of the Second Brainstorming Week on Membrane Computing, Sevilla, February 2004*, Technical Report 01/04 of Research Group on Natural Computing, Sevilla University, Spain, 2004 443–448, and *JUCS*, 10, 5 (2004), 559–539.

[22] D. Sburlan, Clock-free P systems, *Pre-proceedings of Fifth Workshop in Membrane Computing, WMC5*, Milano, Italy, 2004, 372–383.

[23] A. Syropoulos, Fuzzifying P systems, submitted, 2004.

Tutorials

Plasma membrane, compartmentation, transport, and imprecisions

Óscar Moya Mesa

Department of Biology,
University of the Balearic Islands,
07122 Palma de Mallorca (Spain)
E-mail: `omoya@uib.es`

A cell is delimited by its plasma membrane. It is a compartmentation that allows qualitative and quantitative differences between its inner contents and the environment.

A cell, or even a group of cells, can be considered different compartments where membrane properties determine the communication between them due to their properties:

- they act as highly selective filters and molecular transport mechanisms for;
- they control the entry of nutrients and exit of residual products;
- they generate differences between contents and environment;
- they have systems to detect external signals allowing the cellular compartment to react to environmental (external compartment) changes.

There is a basic common structure in all eucariotic cell membranes. Phospholipids are disposed in a bilayer, as those assimetric molecules create spontaneously this structure in an aqueous environment. A phospholipid has a head group attached via a phosphat group to a 3-carbon glycerol backbone, and two fatty acid tails attached to the remaining two carbons of the glycerol. The head is polar, property that confers this part of the molecule affinity to water, in contrast to the tails that are hydrophobic. The simpler phospholipid membrane structure would be a liposome, where the polar heads are exposed to water because of their hydrophilic properties and the tails are in the middle of the bilayer. But a plasma membrane is much more than a liposome: proteins, glucosacarids and cholesterol are other components. Proteins are specially important as they play a key role in transport, signal detection, and regulation.

Without proteins, lipid membranes would be relatively impermeable to ions and many other small molecules, but permeable to water. In fact, they are permeable to almost all molecules: apolar and polar molecules without electrochemical charge can freely go from one side to the other; even polar charged ones can, as it is only a matter of time that they cross from one side to the other. Molecular size and polarity will determine the speed of molecular interchange between both sides, but in biological terms of funtionality we can say that proteins are necessary to mediate transport to allow cellular activity.

Living organisms, and in fact cells, need to communicate with their environment, they need the entry and exit of ions, simple, and complex molecules. Most of the small molecules and ions need help to cross the membrane in form of transmembrane channels or active transport. This transport can be passive, depending on concentration and electrical potential gradients, or active (against gradients and with energy use). The first case correspond to the channels, the second to active transport proteins.

These cellular structures, their characteristics of compartmentation and communication have been used in mathematic modelling, in membrane computing. At a first stage very simple models have arisen, where some standard rules can explain how the molecules are placed in the different compartments and how they "move" from one to the other. But biological structures are much more complex. Even more, is not only a matter of complexity what characterizes biological systems, randomness has a lot to say in these processes. Now, the tendence is trying to incorporate these properties in the new models, to get a better approach to reality or simply to try other kinds of computation to test their possibilities. There are models that handle imprecisions, that search how to incorporate the stochasticity associated to biological processes in their calculations.

To do that it is necessary to have a more realistic view about plasma membranes. They are active structures in different ways, one is its fluidity. Both monolayers are not static, their components move along its surface in a more or less aleatory form. This movement is chaotic and depends both on chance and environmental factors. Although the cell has some control, concentrating some specific molecules in concrete zones, depending on needs and functionality, this control is relative and the molecules still move freely and chaotically in theses regions. Chemical reactions have certain degree of randomness as well, molecules do not interact always in the same way, even with a very similar final result.

Biological systems are complex, even our scarce knowledge lets us notice how complicate they can be, how difficult it is to know and control all the factors that are implied. The most simple mechanisms or reactions are regulated and affected by environmental and cellular conditions. There are many sources of imprecision related to:

- transport systems: their specificity, regulatory mechanisms...
- the molecules that will be transported: concentration, difussion rates, gradients, electrochemical charge, solubility, similarity...
- membrane properties: fluidity, surface, dinamics, membrane electrochem-

ical potential, quantity and distribution of their components...

- inner and outer cell physicochemical conditions: pH, temperature, ionic concentration...

In summary there is imprecision due to the lack of knowledge about which factors interfer, how they do it, and to the stochasticity of biological and chemical dinamics. Randomness and complexity in biological systems are the two main causes of imprecision that should be taken into account for membrane computing in a more realistic scenario.

From a biologist point of view, at a first stage it would be preferable to work with well known systems, as modelling them is much easier and the complexity factor can be better controlled. Randomness is still present and could be incorporated via probabilistic calculations. A good first choice could be to model the sodium-potassium pump, a transport system with a key role in osmotic equilibrium and stabilization of cellular volume. Its biological relevance has been the focus of many research efforts in learning how they work, their regulation systems and how external conditions affect their activity.

This work has been partially supported by the Spanish DGES, project REN2003-00024.

Introduction to Membrane Computing

Gheorghe Păun

Institute of Mathematics of the Romanian Academy
PO Box 1-764, 014700 Bucureşti, Romania
E-mail: `george.paun@imar.ro`

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: `gpaun@us.es`

Abstract. This is a comprehensive (and – supposed – friendly) introduction to membrane computing, meant to offer both to computer scientists and to non-computer scientists an up-dated overview of the domain. That is why the panoply of notions which are introduced here is rather large, but the presentation is informal, without any proof and with rigorous definitions given only for the basic types of P systems – symbol-object P systems with multiset rewriting rules, systems with symport/antiport rules, systems with string-objects, tissue-like P systems, and neural-like P systems. Besides a list of (biologically inspired or mathematically motivated) ingredients/features which can be used in systems of these types, we also mention a series of results – as well as a series of research trends and topics. Then, both some applications are briefly mentioned and a discussion is made about the attractiveness of this framework for (possible) applications, especially in biology.

1 (The Impossibility of) A Definition of Membrane Computing

Membrane computing is an area of computer science aiming to abstract computing ideas and models from the structure and the functioning of living cells, as well as from the way the cells are organized in tissues or higher order structures.

In short, it deals with distributed and parallel computing models, processing multisets of symbol-objects in a localized manner (evolution rules and evolving objects are encapsulated into compartments delimited by membranes), with an essential role played by the communication among compartments (with the environment as well). Of course, this is just a rough description of a membrane system – hereafter called P system – of the very basic type, as many different classes of such devices exist.

The essential ingredient of a P system is its *membrane structure*, which can be a hierarchical arrangement of membranes, like in a cell (hence described by a tree), or a net of membranes (placed in the nodes of a graph), like in a tissue, or in a neural net. The intuition behind the notion of a membrane is that from biology, of a three–dimensional vesicle, but the concept itself is generalized/idealized to interpreting a membrane as a *separator* of two regions (of the Euclidean space), a finite "inside" and an infinite "outside", also providing the possibility of a *selective communication* among the two regions.

The variety of suggestions from biology and the range of possibilities to define the architecture and the functioning of a membrane-based-multiset-processing device are practically endless – and already the literature of membrane computing contains a very large number of models. Thus, membrane computing is not a theory related to a specific model, it is a *framework* for devising compartmentalized models. Both because the domain is rather young (the trigger paper is [77], circulated first on web, but related ideas were considered before, in various contexts), but also as a genuine feature, based both on the biological background and the mathematical formalism used, not only there are already proposed many types of P systems, but the flexibility and the versatility of P systems seem to be, in principle, unlimited.

This last observation, as well as the rapid development and enlargement of the research in this area, make impossible a short and faithful presentation of membrane computing, with any good level of completeness.

However, there are a series of notions, notation, models which are already "standard", which have stabilized and can be considered as basic elements of membrane computing. This paper is devoted to presenting mainly such notions and models, together with the associated notation.

The presentation will be both historically and didactically organized, introducing first either notions which were investigated from the beginning in this area, or simpler notions, able to quickly offer an image of membrane computing to the reader who is not familiar with the domain.

The reader has surely noticed that all the previous discussion refers mainly to computer science (goals), and much less to biology. Membrane computing was not initiated as an area aiming to provide models to biology, in particular, models of the cell. Still in this moment, after a considerable development at the theoretical level, the domain is not fully prepared to offer such models to biology – but this is a strong tendency of the recent research and considerable advances towards such achievements were reported (the topic will be discussed later in more details).

2 Membrane Computing as Part of Natural Computing

Before entering into more specific elements of membrane computing, let us spend some time with the relationship of this area with, let us say, using the "local" terminology, the "outside". We have said above that membrane computing is part of computer science. However, the *genus proximus* is natural computing, the general attempt to learn computer science useful ideas, models, paradigms from the way nature – life especially – "computes", in various circumstances where substance and information processing can be interpreted as computations. Classic bio-inspired branches of natural computing are genetic algorithms (more generally, evolutionary computing, with well individualized sub-branches such as evolutionary programming) and neural computing. Both of them have a long history, which can be traced until unpublished works of Turing (see, e.g., [97]), many applications, and a huge bibliography. Both of them are a proof that "it is worth learning from biology", supporting the optimistic observation that many billions of year nature/life has adjusted certain tools and processes which, correctly (luckily?) abstracted and implemented in computer science terms, can prove to be surprisingly useful in many applications.

A more recent branch of natural computing, with an enthusiastic beginning and unconfirmed yet computational applicability (we do not discuss here the by-products, such as the nanotechnology related developments), is DNA computing, whose birth certificate is related to Adleman experiment [1] of solving a (small) instance of the Hamiltonian path problem by handling DNA molecules in a laboratory. According to Hartmanis [50], [51], it was a *demo* that we can compute with bio-molecules, a big event for computability. However, after one decade of research, the domain is still preparing its tools for a possible future practical application and looking for a new breakthrough idea, similar to Adleman's one from 1994. However, at the theoretical level, DNA computing is beautifully developed (see, e.g., [85] and the proceedings of the yearly DNA Based Computers series of conferences). This is both due to the fact that DNA structure and processing suggest a series of new data structures (e.g., the double stranded sequence, with the pairs of corresponding symbols from the two strands being related through a complementarity relation) and operations (e.g., recombination, denaturation, hybridization), but also to the fact that the massive parallelism made possible by the efficiency of DNA as a support of information promises to be useful in solving computationally hard problems in a feasible time. Actually, at the theoretical level one can say that DNA computing started already in 1987, when T. Head [49] has proposed a language theoretic model of what he called the splicing operation (the recombination of DNA molecules, cut by restriction enzymes in fragments pasted together when the sticky ends match).

Both evolutionary computing and DNA computing are inspired from and related to handling DNA molecules. Neural computing considers the neurons are simple finite automata linked in networks of specific types. Thus, these "neurons" are not interpreted as cells, with an internal structure and life, but as "dots on a grid", with a simple input–output function. (The same observation holds true for cellular automata, where again the "cells" are "dots on a grid", only interacting among them, in a rigid structure.) None of these domains considers the cell itself as its main object of research, in particular, none of these domains pays any attention to membranes and compartmentalization – and this is the point where membrane computing enters the stage. Thus, membrane computing can be seen as an extension of DNA (more generally, molecular) computing, from the "one-processor" level to a distributed computing model.

3 Laudation to the Cell (and Its Membranes)

Life (as we know it on Earth, in the traditional meaning of the term, that investigated by biology) is directly related to cells, everything alive consists of cells or has to do in a direct way with cells. The cell is the smallest "thing" unanimously considered as *alive*. It is very small and very intricate in its structure and functioning, has an elaborate internal activity and an exquisite interaction with the neighboring cells, and with the environment in general. It is fragile and robust at the same time, with a way to organize (control) the bio-chemical (and informational) processes which was polished during billions of years of evolution.

Any cell means membranes. The cell itself is defined – separated from its environment – by a membrane, the external one. Inside the cell, several membranes enclose "protected reactors", compartments where specific biochemical processes take place. In particular, a membrane encloses the nucleus (of eukaryotic cells), where the genetic material is placed. Through vesicles enclosed by membranes one can transport packages of molecules from a part of the cell (e.g., from the Golgi apparatus) to other parts of the cell – in such a way that the transported molecules are not "aggressed" during their journey by neighboring chemicals.

Then, the membranes allow a selective passage of substances among the compartments delimited by them. This can be a simple selection by size, in the case of small molecules, or a much more intricate selection, through protein channels, which not only select, but can also move molecules from a low concentration to a higher concentration, perhaps coupling molecules, through so-called symport and antiport processes.

Much more: the membranes of a cell do not only delimit compartments where specific reactions take place in solution, hence *inside* the compartments, but many reactions in a cell develop *on the membranes*, catalyzed by the many proteins bound on them. It is said that when a compartment is too large for the local biochemistry to be efficient, life creates membranes, both in order to create smaller "reactors" (small enough that, through the Brownian motion, any two of the enclosed molecules can collide frequently enough), and in order to create further "reaction surfaces". Anyway, biology contains many fascinating facts from a computer science point of view, and the reader is encouraged to check the validity of this assertion browsing, e.g., through [2], [64], [8].

Life means surfaces inside surfaces, as can be learned already from the title of [52], while S. Marcus puts it in an equational form [69]: *Life = DNA software + membrane hardware.*

Then, there are cells living alone (unicellular organisms, such as ciliates, bacteria, etc.), but in general the cells are organized in tissues, organs, organisms, communities of organisms. All these suppose a specific organization, starting with the direct communication/cooperation among neighboring cells, and ending with the interaction with the environment, at various levels. Together with the internal structure and organization of the cell, all these suggest a lot of ideas, exciting from a mathematical point of view, and potentially useful from a computability point of view. Part of them were already explored in membrane computing, much more still wait for research efforts.

4 Some General Features of Membrane Computing Models

Still remaining at this general level, it is worth mentioning from the beginning some of the basic features of models investigated in this area, besides the essential use of membranes/compartmentalization.

We have mentioned above the notion of a multiset. The compartments of a cell contains substances (ions, small molecules, macromolecules) swimming in an aqueous solution; there is no ordering there, everything is close to everything, the concentration matters, the population, *the number of copies of each molecule* (of course, we are abstracting/idealizing here, departing from the biological reality). Thus, the suggestion is immediate: to work with sets of objects whose multiplicities matter, hence with *multisets.* This is a data structure with peculiar characteristics, not new but not systematically investigated in computer science.

A multiset can be represented in many ways, but the most compact one is in the form of a string. For instance, if the objects a, b, c are present in, respectively, 5, 2, 6 copies each, we can represent this multiset by the string $a^5b^2c^6$; of course, all permutations of this string represent the same multiset.

Both from the string representation of multisets and because of the biochemical background, where standard chemical reactions are common, the suggestion comes to process the multisets from the compartments of our computing device by means of rewriting-like rules. This means rules of the form $u \rightarrow v$, where u and v are multisets of objects (represented by strings). Continuing the previous example, we can consider a rule $aab \rightarrow abcc$. It indicates the fact that two copies of object a together with a copy of object b react, and, as a result of this reaction, we get back a copy of a as well as the copy of b (hence b behaves here as a catalyst), and we produce two new copies of c. If this rule is applied to the multiset $a^5b^2c^6$, then, because aab are "consumed" and then $abcc$ are "produced", we pass to the multiset $a^4b^2c^8$. Similarly, by using the rule $bb \rightarrow aac$, we will get the multiset a^7c^7, which contains no occurrence of object b.

Two important problems arise here.

The first one is related to the *non-determinism.* Which rules should be applied and to which objects? The copies of an object are considered identical, so we do not distinguish among them; whether to use the first rule or the second one is a significant issue, especially because they cannot be used both at the same time (for the mentioned multiset), as they compete for the "reactant" b. The standard solution to this problem in membrane computing is that *the rules and the objects are chosen in a non-deterministic manner* (at random, with no preference; more rigorously, we can say that any possible evolution is allowed).

This is also related to the idea of *parallelism.* Biochemistry is not only (in a certain degree) non-deterministic, but it is also (in a certain degree) parallel. If two chemicals can react, then the reaction does not take place for only two molecules of the two chemicals, but, in principle, for all molecules. This is the suggestion supporting the maximal parallelism used in many classes of P systems: in each step, all rules which can be applied have to be applied to all possible objects. We will come back to this important notion later, but now we only illustrate it with the previous multiset and pair of rules. Using these rules in the maximally parallel manner means to either use the first rule twice (thus involving four copies of a and both copies of b) or once the second rule (it consumes both copies of b, hence the first rule cannot be used at the same time). In the first case, one copy of a remains unused (and the same with all copies of c), and the resulting multiset is $a^3b^2c^{10}$; in the second case, all copies of a and c remain unused, and the resulting multiset is a^7c^7. It deserves to be noted that in the latter case the maximally parallel application of rules corresponds to the *sequential* (one in a time) application of the second rule.

There also are other types of rules used in membrane computing (e.g., symport and antiport rules), but we will discuss them later. Here we close with the observation that membrane computing deals with models which are intrinsically *discrete* (basically, working with multisets of objects, with the multiplicities being natural numbers) and evolve through *rewriting-like* (we can also say reaction-like) rules.

5 Computer Science Related Areas

Rewriting rules are standard rules for handling strings in formal language theory (although other types of rules are also used, both in formal language theory and in P systems, such as insertion, deletion, context-adjoining, etc.). Also working with strings modulo the ordering of symbols is "old stuff": commutative languages (investigated, e.g., in [35]) are nothing else than the permutation closure of languages. In turn, the multiplicity of symbol occurrences in a string corresponds to the Parikh image of the string, which directly leads to vector addition systems, Petri nets, register machines, formal power series.

Then, also the parallelism is considered in many areas of formal languages, and it is the main feature of Lindenmayer systems. These systems deserve a special discussion here, as they are a well developed branch of formal language theory inspired from biology, specifically, from the development of multi-cellular organisms (which can be described by strings of symbols). However, for L systems the cells are considered as symbols, not their structure is investigated but their organization in (mainly linear) patterns. P systems can be seen as dual to L systems, as they zoom in the cell, distinguishing the internal structure and the objects evolving inside it – maybe also distinguishing (when "zooming enough") the structure of the objects themselves, which leads to the category of P systems with string-objects.

However, a difference exists between the kind of parallelism in L systems and that in P systems: there the parallelism is *total* – all symbols of a string should be processed at the same time, here we work with a maximal parallelism – we process as many objects as possible, but not necessarily all of them.

Still closer to membrane computing are the multiset processing languages, the most known of them being Gamma [9, 10]. The standard rules of Gamma are of the form $u \rightarrow v(\pi)$, where u and v are multisets and π is a predicate which should be satisfied by the multiset to which the rule $u \rightarrow v$ is applied. The generality of the form of rules ensures a great expressivity and, in a direct manner, computational universality. What Gamma does not have (at least in the initial versions) is distributivity. Then, membrane computing restricts the form of rules, on the one hand, as imposed by the biological roots, on the other hand, in search of mathematically simple and elegant models.

Also membranes appear, even in Gamma related models, and this is the case with CHAM, the Chemical Abstract Machine of Berry and Boudol, [16], the direct ancestor of membrane systems – with the mentioning that the membranes of CHAM ... are not membranes as in the cell biology, but they correspond to the contents of membranes, multisets and lower level membranes together, while the goals and the approach are completely different, directed to the algebraic treatment of the processes these membranes can undergo. From this point of view, of goals and tools, CHAM has a recent counterpart in the so-called *brane calculus* (of course, "brane" comes from "membrane") from [22] (see also [89] for a related approach), where process algebra is used for investigating the processes taking place *on* membranes and *with* membranes of a cell.

The idea of devising a computing device based on compartmentalization through membranes was also suggested in [66].

Many related areas and many roots, with many common ideas and many differences. In some extent, membrane computing is a synthesis of part of these ideas, integrated in a framework directly inspired from the cell biology, paying the deserved attention to membranes (hence to distribution, hierarchization, communication, localization and to other related concepts), aiming – in the basic types of devices – to find computing models, as elegant (minimalistic) as possible, as powerful as possible (in comparison with Turing machines and their subclasses), and as efficient as possible (able to solve computationally hard problems in a feasible time).

6 The Cell-Like Membrane Structure

We move now towards presenting in a more precise manner the computing models investigated in our area, and we start by introducing one the fundamental ingredients of a P system, namely, the *membrane structure*.

The meaning of this notion is illustrated in Figure 1, and this is what we can see when looking (through mathematical glasses, hence abstracting as much as necessary in order to obtain a formal model) to a standard cell.

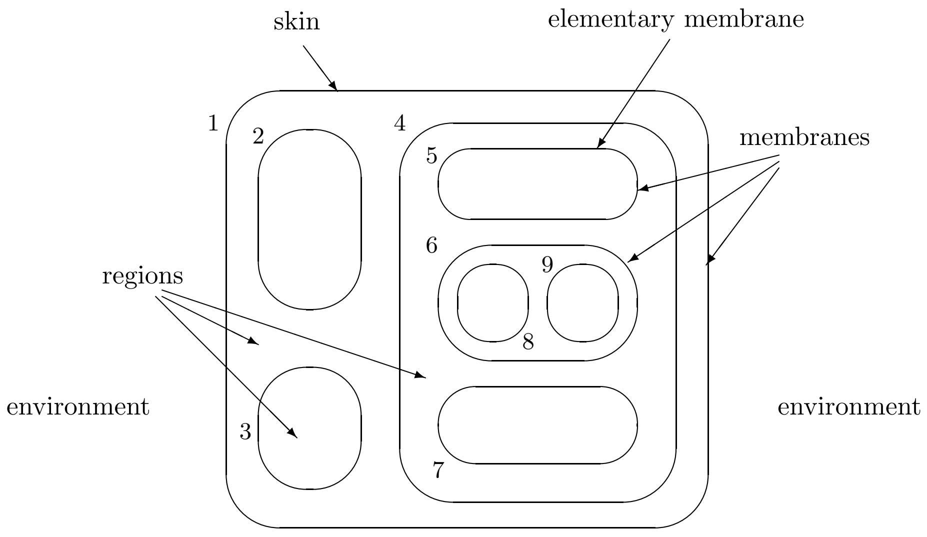


Figure 1: A membrane structure

Thus, as suggested by Figure 1, a membrane structure is a hierarchically arranged set of membranes, contained in a distinguished external membrane (corresponding to the plasma membrane and usually called the *skin* membrane). Several membranes can be placed inside the skin membrane (they correspond to the membranes present in a cell, around the nucleus, in Golgi apparatus, vesicles, mitochondria, etc.); a membrane without any other membrane inside it is said to be *elementary.* Each membrane 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. Clearly, the correspondence membrane–region is one-to-one, that is why we sometimes use interchangeably these terms.

Usually, the membranes are identified by *labels* from a given set of labels. In Figure 1, we use numbers, starting with number 1 assigned to the skin membrane (this is the standard labelling, but the labels can be more informative "names" associated with the membranes). Also, in the figure the labels are assigned in a one-to-one manner to membranes, but this is possible only in the case of membrane structures which cannot grow (indefinitely), otherwise several membranes should have the same label (we will see later such cases). Due to the membrane–region correspondence, we identify by the same label a membrane and its associated region.

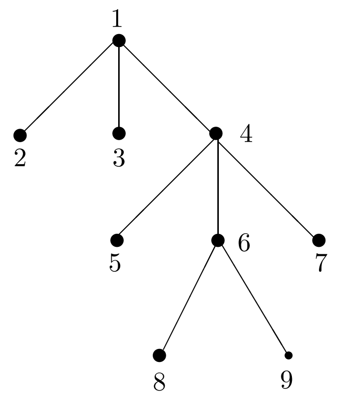


Figure 2: The tree describing the membrane structure from Figure 1

Clearly, a hierarchical structure of membranes can be represented by a rooted tree; Figure 2 gives the tree which describes the membrane structure from Figure 1. The root of the tree is associated with the skin membrane and the leaves with the elementary membranes. In this way, graph–theoretic notions are brought into the stage, such as the distance in the tree, the level of a membrane, the height/depth of the membrane structure, as well as terminology, such as parent/child membrane, ancestor, etc.

Directly suggested by the tree representation is the symbolic representation of a membrane structure, by strings of labelled matching parentheses. For instance, a string corresponding to the structure from Figure 1 is the following one:

$$[_1\ [_2\ ]_2\ [_3\ ]_3\ [_4\ [_5\ ]_5\ [_6\ [_8\ ]_8\ [_9\ ]_9\ ]_6\ [_7\ ]_7\ ]_4\ ]_1.$$

An important aspect should now be noted: the membranes from the same level can float around, that is, the tree representing the membrane structure is not oriented; in terms of parentheses expressions, two sub-expressions placed at the same level represent the same membrane structure. For instance, in the previous case, the expression

$$[_1\ [_3\ ]_3\ [_4\ [_6\ [_8\ ]_8\ [_9\ ]_9\ ]_6\ [_7\ ]_7\ [_5\ ]_5\ ]_4\ [_2\ ]_2\ ]_1$$

is an equivalent representation of the same membrane structure.

7 Evolution Rules and the Way of Using Them

In the basic variant of P systems, each region contains a multiset of symbol-objects, which correspond to the chemicals swimming in a 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. Actually, there are three main types of rules: (1) multiset-rewriting rules (one uses to call them, simply, evolution rules), (2) communication rules, and (3) rules for handling membranes.

In this section we present the first type of rules. They correspond to the chemical reactions possible in the compartments of a cell, hence they are of the form $u \to v$, where u and v are multisets of objects. However, in order to make the compartments cooperate, we have to move objects across membranes, and to this aim we add *target indications* to the objects produced by a rule as above (to the objects from multiset v). These indications are: *here, in, out*, with the meaning that an object having associated the indication *here* remains in the same region, one having associated the indication *in* goes immediately into a directly lower membrane, non-deterministically chosen, and *out* indicates that the object has to exit the membrane, thus becoming an element of the region surrounding it. An example of evolution rule is $aab \to (a, here)(b, out)(c, here)(c, in)$ (this is the first of the rules considered in Section 4, with target indications associated with the objects produced by rule application). After using this rule in a given region of a membrane structure, two copies of a and one b are consumed (removed from the multiset of that region), and one copy of a, one of b, and two of c are produced; the resulting copy of a remains in the same region, and the same happens with one copy of c (indications *here*), while the new copy of b exits the membrane, going to the surrounding region (indication *out*), and one of the new copies of c enters one of the child membranes, non-deterministically chosen. If no such child membrane exists, that is, the membrane with which the rule is associated is elementary, then the indication *in* cannot be followed, and the rule cannot be applied. In turn, if the rule is applied in the skin region, then b will exit into the environment of the system (and it is “lost” there, as it can never come back). In general, the indication *here* is not specified (an object without an explicit target indication is supposed to remain in the same region where the rule is applied).

It is important to note that in this initial type of systems we are going to describe we do not provide similar rules for the environment, as we do not care about the objects present there; later we will consider types of P systems where also the environment takes part in the system evolution.

A rule as above, with at least two objects in its left hand side, is said to be *cooperative*; a particular case is that of *catalytic* rules, of the form $ca \to cv$, where c is an object (called catalyst) which assists the object a to evolve into the multiset v; rules of the form $a \to v$, where a is an object, are called *non-cooperative*.

The rules can also have the form $u \to v\delta$, where δ denotes the action of *membrane dissolving*: if the rule is applied, then the corresponding membrane disappears and its contents, object and membranes alike, are left free in the surrounding membrane; the rules of the dissolved membrane disappear at the same time with the membrane. The skin membrane is never dissolved.

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. However, the fact that the communication of objects from a compartment to a neighboring compartment is controlled by the "reaction rules" is mathematically attractive, but not quite realistic from a biological point of view, that is why there were also 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).

It is also worth noting that evolution rules are stated in terms of *names of objects*, while their application/execution is done using *copies of objects* – remember the example from Section 4, where the multiset $a^5b^2c^6$ was processed by a rule of the form $aab \to a(b, out)c(c, in)$, which, in the maximally parallel manner, is used twice, for the two possible sub-multisets aab.

We have arrived in this way at the important feature of P systems, concerning *the way of using the rules.* The key phrase in this respect is: *in the maximally parallel manner, non-deterministically choosing the rules and the objects.*

More specifically, this means that we assign objects to rules, non-deterministically choosing the objects and the rules, until no further assignment is possible. More mathematically stated, we look to the *set* of rules, and try to find a *multiset* of rules, by assigning multiplicities to rules, with two properties: (i) the multiset of rules is *applicable* to the multiset of objects available in the respective region, that is, there are enough objects in order to apply the rules a number of times as indicated by their multiplicities, and (ii) the multiset is *maximal*, no further rule can be added to it (because of the lack of available objects).

Thus, an evolution step in a given region consists in finding a maximal applicable multiset of rules, removing from the region all objects specified in the left hand of the chosen rules (with the multiplicities as indicated by the rules and by the number of times each rule is used), producing the objects from the right hand sides of rules, and then distributing these objects as indicated by the targets associated with them. If at least one of the rules introduces the dissolving action δ, then the membrane is dissolved, and its contents become part of the immediately upper membrane – provided that this membrane was not dissolved at the same time, a case where we stop in the first upper membrane which was not dissolved (at least the skin remains intact).

8 A Formal Definition of a Transition P System

Systems based on multiset-rewriting rules as above are usually called *transition P systems*, and we preserve here this terminology (although "transitions" are present in all types of systems).

Of course, when presenting a P system we have to specify: the alphabet of objects (a usual finite non-empty alphabet of abstract symbols identifying the objects), the membrane structure (it can be represented in many ways, but the most used one is by a string of labelled matching parentheses), the multisets of objects present in each region of the system (represented in the most compact way by strings of symbol-objects), the sets of evolution rules associated with each region, as well as the indication about the way the output is defined – see below.

Formally, a *transition P system* (of degree m) is a construct of the form

$$\Pi = (O, C, \mu, w_1, w_2, \ldots, w_m, R_1, R_2, \ldots, R_m, i_o),$$

where:

1. O is the (finite and non-empty) alphabet of *objects*,
2. $C \subset O$ is the set of *catalysts*,
3. μ is a membrane structure, consisting of m membranes, labelled with $1, 2, \ldots, m$; one says that the membrane structure, and hence the system, is *of degree m*,
4. $w_1, w_2, \ldots, w_m$ are strings over O representing the *multisets of objects* present in the regions $1, 2, \ldots, m$ of the membrane structure,
5. $R_1, R_2, \ldots, R_m$ are finite *sets of evolution rules* associated with the regions $1, 2, \ldots, m$ of the membrane structure,
6. i_o is either one of the labels $1, 2, \ldots, m$, and then the respective region is the *output region* of the system, or it is 0, and then the result of a computation is collected in the environment of the system.

The rules are of the form $u \to v$ or $u \to v\delta$, with $u \in O^+$ and $v \in (O \times Tar)^*$, where[1] $Tar = \{here, in, out\}$. The rules can be cooperative (with u arbitrary), non-cooperative (with $u \in O - C$), or catalytic (of the form $ca \to cv$ or $ca \to cv\delta$, with $a \in O - C, c \in C, v \in ((O - C) \times Tar)^*$); note that the catalysts never evolve and never change the region, they only help the other objects to evolve.

A possible restriction about the region i_o in the case when it is an internal one is to consider only regions enclosed by elementary membranes for output (that is, i_o should be the label of an elementary membrane of μ).

In general, the membrane structure and the multisets of objects from its compartments identify a *configuration* of a P system. The *initial configuration* is given by specifying the membrane structure and the multisets of objects available in its compartments at the beginning of a computation, hence $(\mu, w_1, \ldots, w_m)$. During the evolution of the system, by means of applying the rules, both the multisets of objects and the membrane structure can change. We will see how this is done in the next section; here we conclude with an **example**

[1]By V^* we denote the set of all strings over an alphabet V, the empty string λ included, and by V^+ we denote the set $V^* - \{\lambda\}$, of all non-empty strings over V.

of a P system, represented in a pictorial way in Figure 3. It is important to note that adding to the initial configuration the set of rules, placed in the corresponding regions, we have a complete and concise presentation of the system (the indication of the output region can also be added in a suitable manner, for instance, writing "output" inside it).

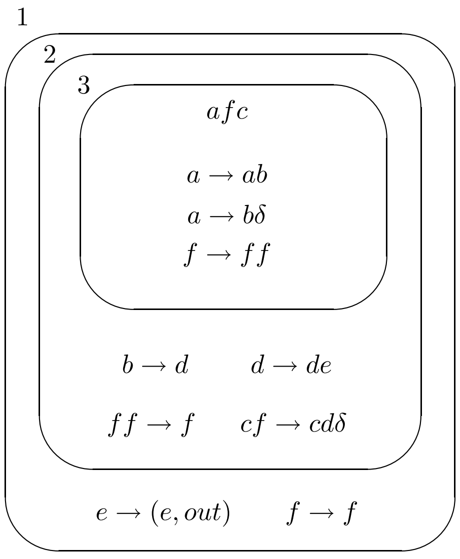


Figure 3: The initial configuration of a P system, rules included

9 Defining Computations and Results of Computations

In their basic variant, membrane systems are synchronous devices, in the sense that a global clock is assumed, which marks the time for all regions of the system. In each time unit a transformation of a configuration of the system – we call it *transition* – takes place by applying the rules in each region, in a *non-deterministic* and *maximally parallel manner*. As explained in the previous sections, this means that the objects to evolve and the rules governing this evolution are chosen in a non-deterministic way, and this choice is "exhaustive" in the sense that, after the choice was made, no rule can be applied in the same evolution step to the remaining objects.

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, and the output region i_o still exists in the halting configuration (in the case when i_o is the label of a membrane, it can be dissolved during the computation). With a successful computation we can associate a *result* in various ways. If we have an output region specified, and this is an internal region, then we have an *internal output*: we count the objects present in the output region in the halting configuration and this number is the result of the computation. When we have $i_o = 0$, we 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 (if several objects exit at the same time, then all their permutations are accepted as a substring of the result). Note that non-halting computations provide no output (we cannot know when a number is "completely computed" before halting), and if the output membrane is dissolved during the computation, then the computation aborts, no result is obtained (of course, this makes sense only in the case of the internal output).

A possible extension of the definition is to consider a *terminal* set of objects, $T \subseteq O$, and to count only the copies of objects from T, discarding the objects from $O - T$ present in the output region. This allows some additional leeway in constructing and "programming" a P system, because we can ignore some auxiliary objects (e.g., the catalysts).

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, depending on the way the output is defined). The case when we get a language is important in view of the qualitative difference between the "loose" data structure we use inside the system (vectors of numbers) and the data structure of the result, strings, where we also have a "syntax", a positional information.

For a given system Π we denote by $N(\Pi)$ the set of numbers computed by Π in the above way. When we consider the vector of multiplicities of objects from the output region, we write $Ps(\Pi)$. In turn, in the case when we take as (external) output the strings of objects leaving the system, then we denote the language of these strings by $L(\Pi)$.

Let us illustrate the previous definitions by examining the computations of the system from Figure 3 – with the output region being the environment.

We have objects only in the central membrane, that with label 3, hence only here we can apply rules. Specifically, we can repeatedly apply the rule $a \rightarrow ab$ in parallel with $f \rightarrow ff$, and in this way the number of copies of b grows each step by one, while the number of copies of f is doubled in each step. If we do not apply the rule $a \rightarrow b\delta$ (again in parallel with $f \rightarrow ff$), which dissolves the membrane, then we can continue in this way forever. Thus, in order to ever halt, we have to dissolve membrane 3. Assume that this happens after $n \geq 0$ steps of using the rules $a \rightarrow ab$ and $f \rightarrow ff$. When membrane 3 is dissolved, its contents ($n+1$ copies of b, 2^{n+1} copies of f, and one copy of the catalyst c) are left free in membrane 2, which now can start using its rules. In the next step, all objects b become d. Let us examine the rules $ff \rightarrow f$ and $cf \rightarrow cd\delta$. The second rule dissolves membrane 2, hence passes the contents of this membrane to membrane 1. If among the objects which arrive in membrane 1 there is at least one copy of f, then the rule $f \rightarrow f$ from region 1 can be used forever and the computation never stops; moreover, if the rule $ff \rightarrow f$ is used at least once in parallel with the rule $cf \rightarrow cd\delta$, then at least one copy of f is present. Therefore, the rule $cf \rightarrow cd\delta$ should be used only if region 2 contains only one copy of f (note that, because of the catalyst, the rule $cf \rightarrow cd\delta$ can be used only for one copy of f). This means that the rule $ff \rightarrow f$ was used always for all pairs of f available, that is, in each step the number of copies of f is divided by 2. This is already done once in the step when all copies of b become d, and will be done from now on as long as at least two copies of f are present. Simultaneously, in each step each d produces one copy of e. This process can continue until we get a configuration with only one copy of f present; at that step we have to use the rule $cf \rightarrow cd\delta$ (the rule $ff \rightarrow f$ is no longer applicable), hence also membrane 2 is dissolved. Because we have applied the rule $d \rightarrow de$, in parallel for all copies of d (there are $n+1$ such copies), during $n+1$ steps, we have $(n+1)(n+1)$ copies of e, $n+2$ copies of d (one of them was produced by the rule $cf \rightarrow cd\delta$), and one copy of c present in the skin membrane of the system (the unique membrane still present). The objects e are sent out, and the computation halts. Therefore, we compute in this way the number $(n+1)^2$, for some $n \geq 0$, that is, $N(\Pi) = \{n^2 \mid n \geq 1\}$.

10 Using Symport and Antiport Rules

The multiset rewriting rules correspond to reactions taking place in the cell, *inside* the compartments. However, an important part of the cell activity is related to the passage of substances through membranes, and one of the most interesting ways to handle this trans-membrane communication is by coupling molecules. The process by which two molecules pass together across a membrane (through a specific protein channel) is called *symport*; when the two molecules pass simultaneously through a protein channel, but in opposite directions, the process is called *antiport*.

We can formalize these operations in an obvious way: (ab, in) or (ab, out) are symport rules, stating that a and b pass together through a membrane, entering in the former case and exiting in the latter; similarly, $(a, out; b, in)$ is an antiport rule, stating that a exits and at the same time b enters the membrane. Separately, neither a nor b can cross a membrane – unless we have a rule of the form (a, in) or (a, out), called, for uniformity, *uniport* rule.

Of course, we can generalize these types of rules, by considering symport rules of the form (x, in), (x, out), and antiport rules of the form $(z, out; w, in)$, where x, z, w are multisets of arbitrary size; one uses to say that $|x|$ is the *weight* of a symport rule as above, and $\max(|z|, |w|)$ is the *weight* of the antiport rule[2].

Now, such rules can be used in a P system instead of the target indications *here, in, out*: we consider multiset rewriting rules of the form $u \rightarrow v$ (or $u \rightarrow$

[2]By $|u|$ we denote the length of the string $u \in V^*$, for any alphabet V.

$v\delta$) without target indications associated with the objects from v, as well as symport/antiport rules for communicating the objects among compartments. Such systems, called *evolution–communication* P systems, were considered in [23] (for various restricted types of rules of the two forms).

Here we do not go into this direction, but we stay closer both to the chronological evolution of the domain, and to the mathematical minimalism, and we check whether we can compute using only communication, that is, only symport and antiport rules. This leads to considering one of the most interesting classes of P systems, which we formally introduce here.

A *P system with symport/antiport rules* is a construct of the form

$$\Pi = (O, \mu, w_1, \ldots, w_m, E, R_1, \ldots, R_m, i_o),$$

where:

1. O is the alphabet of objects,
2. μ is the membrane structure (of degree $m \geq 1$, with the membranes labelled in a one-to-one manner with $1, 2, \ldots, m$),
3. $w_1, \ldots, w_m$ are strings over O representing the multisets of objects present in the m compartments of μ in the initial configuration of the system,
4. $E \subseteq O$ is the set of objects supposed to appear in the environment in arbitrarily many copies,
5. $R_1, \ldots, R_m$ are the (finite) sets of rules associated with the m membranes of μ,
6. $i_o \in H$ is the label of a membrane of μ, which indicates the *output* region of the system.

The rules from R can be of two types, symport rules and antiport rules, of the forms as specified above.

The rules are used in the non-deterministic maximally parallel manner. In the usual way, we define transitions, computations, and halting computations. The number (or the vector of multiplicities) of objects present in region i_o in the halting configuration is said to be computed by the system along that computation; the set of all numbers (resp., vectors or numbers) computed in this way by Π is denoted by $N(\Pi)$ (resp., by $Ps(\Pi)$).

We remark here a new component of the system, the set E of objects which are present in the environment in arbitrarily many copies; because we only move objects across membranes and because we start with finite multisets of objects present in the system, we cannot increase the number of objects necessary for the computation if we do not provide a supply of objects, and this can be done by considering the set E. Because the environment is supposed inexhaustible, the objects from E are supposed inexhaustible, irrespective how many of them are brought into the system, arbitrarily many remain outside.

Another new feature is that this time the rules are associated with the membranes, not with the regions, and this is related to the fact that each rule governs the communication through a specific membrane.

The P systems with symport/antiport rules have a series of attractive characteristics: they are fully based on biological types of multiset processing rules; the environment takes a direct part into the evolution of the system; the computation is done only by communication, no object is changed, the objects only move across membranes; no object is created or destroyed, hence the conservation low is observed (as given in the previous sections, this is not valid for multiset rewriting rules, because, for instance, rules of the form $a \to aa$ or $ff \to f$ are allowed, but by using some dummy objects d available in arbitrarily many copies, we can take care of the conservation low by writing, e.g., $da \to aa$ and $ff \to fd$).

11 An Example (Like a Proof. . .)

Because P systems with symport/antiport rules constitute an important class of P systems, it is worth considering an example; however, instead of a simple example (actually, it is not very easy to find simple symport/antiport systems computing non-trivial sets of numbers), we give directly a general construction, for simulating a *register machine*. In this way, we also introduce one of the widely used proof techniques for the universality results in this area. (Of course, the biologist reader can safely skip this section.)

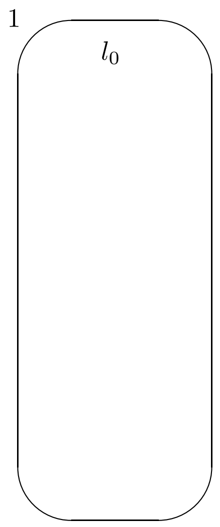
Informally speaking, a register machine consists of a specified number of counters (also called registers) which can hold any natural number, and which are handled according to a program consisting of labelled instructions; the counters can be increased or decreased by 1 – the decreasing being possible only if a counter holds a number greater than or equal to 1 (we say that it is non-empty) –, and checked whether they are non-empty.

Formally, a (non-deterministic) *register machine* is a device $M = (m, B, l_0, l_h, R)$, where $m \geq 1$ is the number of counters, B is the (finite) set of instruction labels, l_0 is the initial label, l_h is the halting label, and R is the finite set of instructions labelled (hence uniquely identified) by elements from B (R is also called the *program* of the machine). The labelled instructions are of the following forms:

- $l_1 : (\mathtt{add}(r), l_2, l_3)$, $1 \leq r \leq m$ (add 1 to counter r and go non-deterministically to one of the instructions with labels l_2, l_3),
- $l_1 : (\mathtt{sub}(r), l_2, l_3)$, $1 \leq r \leq m$ (if counter r is not empty, then subtract 1 from it and go to the instruction with label l_2, otherwise go to the instruction with label l_3),
- $l_h : \mathtt{halt}$ (the halt instruction, which can only have the label l_h).

A counter machine generates a k-dimensional vector of natural numbers in the following manner: we distinguish k counters as output counters (without loss of generality, they can be the first k counters), and we start computing with all m counters being empty, with the instruction labelled by l_0; if the computation reaches the instruction l_h : `halt` (we say that it halts), then the values of counters $1, 2, \ldots, k$ is the vector generated by the computation. The set of all vectors from $\mathbf{N}^k$ generated in this way by M is denoted by $Ps(M)$. If we want to generate only numbers (1-dimensional vectors), then we have the result of a computation in counter 1, and the set of numbers computed by M in this way is denoted by $N(M)$. It is known (see [73], [40]) that non-deterministic counter machines with $k+2$ counters can compute any set of Turing computable k-dimensional vectors of natural numbers (hence machines with three counters generate exactly the family of Turing computable sets of numbers).

Now, a register machine can be easily simulated by a P system with symport/antiport rules. The idea is illustrated in Figure 4, where we have represented the initial configuration of the system, the rules associated with the unique membrane, as well as the set E, of objects present in the environment.



$$E = \{a_r \mid 1 \leq r \leq m\} \cup \{l, l', l'', l''', l^{iv} \mid l \in B\}$$

$$\left.\begin{array}{l}(l_1, out; a_r l_2, in)\\(l_1, out; a_r l_3, in)\end{array}\right\} \text{ for } l_1 : (\texttt{add}(r), l_2, l_3)$$

$$\left.\begin{array}{l}(l_1, out; l_1' l_1'', in)\\(l_1' a_r, out; l_1''', in)\\(l_1'', out; l_1^{iv}, in)\\(l^{iv} l_1''', out; l_2, in)\\(l^{iv} l_1', out; l_3, in)\end{array}\right\} \text{ for } l_1 : (\texttt{sub}(r), l_2, l_3)$$

$$(l_h, out) \qquad \text{for } (l_h : \texttt{halt})$$

Figure 4: An example of symport/antiport P system

The value of each register r is represented by the multiplicity of object $a_r, 1 \leq r \leq m$, in the unique membrane of the system. The labels from B, as well as primed version of them, are also objects of our system. We start with the unique object l_0 present in the system. In the presence of a label–object l_1 we can simulate the corresponding instruction l_1 : $(\texttt{add}(r), l_2, l_3)$ or $l_1 : (\texttt{sub}(r), l_2, l_3)$.

The simulation of an `add` instruction is clear, so that we discuss only a `sub` instruction. The object l_1 exits the system in exchange of the two objects $l_1' l_1''$ (rule $(l_1, out; l_1' l_1'', in)$). In the next step, if any copy of a_r is present in the system, then l'_1 has to exit (rule $(l'_1 a_r, out; l'''_1, in)$), thus diminishing the number of copies of a_r by one, and bringing inside the object l'''_1; if no copy of a_r is present, which corresponds to the case when the register r is empty, then the object l'_1 remains inside. Simultaneously, rule $(l''_1, out; l^{iv}_1, in)$ is used, bringing inside the "checker" l^{iv}_1. Depending on what this object finds in the system, either l'''_1 or l'_1, it introduces the label l_2 or l_3, respectively, which corresponds to the correct continuation of the computation of the register machine.

When the halt instruction is reached, that is, the object l_h is introduced, this object is just expelled into the environment and the computation stops.

Clearly, the (halting) computations in Π directly correspond to (halting) computations in M, hence $N(M) = N(\Pi)$.

12 A Large Panoply of Possible Extensions

We have mentioned the flexibility and the versatility of the formalism of membrane computing, and we have already mentioned several types of systems, making use of several types of rules, with the output of a computation defined in various ways. We continue here in this direction, by presenting a series of possibilities of changing the form of rules and/or the way of using them. The motivation for such extensions comes both from biology, from the natural desire to capture more and more biological facts, and from mathematics and computer science, from the desire to have more powerful or more elegant models.

First, let us return to the basic target indications, *here, in, out*, associated with the objects produced by rules of the form $u \to v$; *here* and *out* precisely indicate the region where the object is to be placed, but *in* introduces a degree of non-determinism in the case when there are several inner membranes. This non-determinism can be avoided by indicating also the label of the target membrane, that is, using target indications of the form in_j, where j is a label. An intermediate possibility, more specific than *in* but not completely unambiguous like in_j, is to assign both to objects and to membranes *electrical polarizations*, $+, -$, and 0. The polarizations of membranes are given from the beginning (or can be changed during the computation), the polarization of objects is introduced by rules, using rules of the form $ab \to c^+c^-(d^0, tar)$. The charged objects have to go to any lower level membrane of opposite polarization, while objects with neutral polarization either remain in the same region or get out, depending on the target indication $tar \in \{here, out\}$ (this is the case with d in the previous rule).

A spectacular generalization, considered recently in [33], is to use indications in_j, for j being any membrane from the system, hence the object is "teleported" immediately at any distance in the membrane structure; also, commands of the form in^* and out^* were used, with the meaning that the object should be sent to (one of) the elementary membranes from the current membrane or to the skin region, respectively, no matter how far the target is.

Then, we have considered the membrane dissolution action, represented by the symbol δ; we may imagine that such an action decreases the thickness of the membrane from the normal thickness, 1, to 0. A dual action can be also used, of increasing the thickness of a membrane, from 1 to 2. We indicate this action by τ. Assume that δ also decreases the thickness from 2 to 1, that the thickness cannot have other values than 0 (membrane dissolved), 1 (normal thickness), and 2 (membrane impermeable), and that when both δ and τ are introduced simultaneously in the same region, by different rules, then their actions cancel, the thickness of the membrane does not change. In this way, we get a nice possibility to control the work of the system: if a rule introduces a target indication *in* or *out* and the membrane which has to be crossed by the respective object has thickness 2, hence it is non-permeable, then the rule cannot be applied.

Let us look now to the catalysts. In the basic definition they never change their state or their place, like ordinary objects do. A "democratic" decision is to let also the catalysts to evolve – in certain limits. Thus, *mobile catalysts* were proposed, moving across membranes like any object (but still not changing themselves). Then, the catalysts were allowed to change their state, for instance, oscillating between c and $\bar{c}$. Such a catalyst is called *bi-stable*, and the natural generalization is to consider k-stable catalysts, allowed to change along k given forms. Note that in all cases the number of catalysts is not changed, we do not produce or remove catalysts (unless if they leave the system), and this is important in view of the fact that the catalysts are in general used for inhibiting the parallelism (a rule $ca \to cv$ can be used simultaneously at most as many times as many copies of c are present).

There are several possibilities for controlling the use of rules, in general, leading to a decrease of the degree of non-determinism of a system. For instance, a mathematically and biologically motivated possibility, is to consider a *priority* relation on the set of rules from a given region, in the form of a partial order relation on the set of rules from that region. This corresponds to the fact that certain reactions/reactants are more active than others, and can be interpreted in two ways: as a competition for reactants/objects, or in a strong sense. In the latter sense, if a rule r_1 has priority over a rule r_2 and r_1 can be applied, then r_2 cannot be applied, irrespective whether rule r_1 leaves objects which it cannot use. For instance, if $r_1 : ff \to f$ and $r_2 : cf \to cd\delta$, like in the example from Section 8, and the current multiset is $fffc$, because rule r_1 can be used, consuming two copies of f, we do not also use the second rule for the remaining fc. In the weak interpretation of the priority, this is allowed: the rule with the maximal priority takes as many objects as possible, and, if still there are remaining objects, then the next rule in the decreasing order of priority is used for as many objects as possible, and, if still remain unused objects, we continue in this way until no further rule can be added to the multiset of applicable rules.

Also directly coming from bio-chemistry are the rules with *promoters* and *inhibitors*, written in the form $u \to v|_z$ and $u \to v|_{\neg z}$, respectively, where u, v, z are multisets of objects; in the case of promoters, the rule $u \to v$ can be used in a given region only if all objects from z are present in the same region, and they are different from the (copies of) objects from u; in the inhibitors case, no object from z should be present in the region, and different from the objects from u. The promoting objects can evolve at the same time by other rules, or by the same rule $u \to v$, but by another instance of it (e.g., $a \to b|_a$ can be used twice in a region containing two copies of a, with each instance of $a \to b|_a$ acting on one copy of a and promoted by the other copy, but cannot be used at all in a region where a appears only once).

An interesting combination of rewriting–communication rules are those considered in [92], where rules of the following three forms are proposed: $a \to (a, tar)$, $ab \to (a, tar_1)(b, tar_2)$, $ab \to (a, tar_1)(b, tar_2)(c, come)$, where a, b, c are objects, and tar, tar_1, tar_2 are target indications of the forms *here, in, out* or even in_j, with j the label of a membrane. Such a rule just moves objects from a region to another one, with the mentioning that rules of the third type can be used only in the skin region and the indication $(c, come)$ means that a copy of c is brought into the system from the environment. Clearly, these rules are different from the symport/antiport rules; for instance, the two objects ab from a rule $ab \to (a, tar_1)(b, tar_2)$ start from the same region, and can go into different directions, one up and one down in the membrane structure.

We have left in the end one of the most general type of rules, introduced in [15] under the name of *boundary rules*, directly capturing the idea that many reactions take place on the inner membranes of a cell, maybe depending on the contents of both the inner and the outer region adjacent to that membrane. These rules are of the form $xu[_i vy \to xu'[_i v'y$, where x, u, u', v, v', y are multisets of objects and i is the label of a membrane. The meaning is that in the presence of the objects from x outside and of objects from y inside the membrane i, the multiset u from outside changes to multiset u' and, simultaneously, the multiset v from inside is changed into v'. The generality of this kind of rules is apparent − and it can be decreased by imposing various restrictions on the involved multisets.

There also are other variants considered in the literature, especially in what concerns the way of controlling the use the rules, but we do not continue here in this direction.

13 P Systems with Active Membranes

We pass now to presenting a class of P systems, which, together with the basic transition systems and the symport/antiport systems, is one of the three central types of cell-like P systems considered in membrane computing. Like in the above case of boundary rules, they start from the observation that the membranes play an important role in the reactions which take place in a cell, and, moreover, they can evolve themselves, either changing their characteristics or even getting divided.

Especially this last idea has motivated the class of *P systems with active membranes*, which are constructs of the form

$$\Pi = (O, H, \mu, w_1, \ldots, w_m, R),$$

where:

1. $m \geq 1$ (the initial degree of the system);
2. O is the alphabet of *objects*;
3. H is a finite set of *labels* for membranes;
4. μ is a *membrane structure*, consisting of m membranes having initially neutral polarizations, labelled (not necessarily in a one-to-one manner) with elements of H;
5. $w_1, \ldots, w_m$ are strings over O, describing the *multisets of objects* placed in the m regions of μ;
6. R is a finite set of *developmental rules*, of the following forms:

   (a) $[_h a \rightarrow v]_h^e$,
   for $h \in H, e \in \{+, -, 0\}, a \in O, v \in O^*$
   (object evolution rules, associated with membranes and depending on the label and the charge of the membranes, but not directly involving the membranes, in the sense that the membranes are neither taking part in the application of these rules nor are they modified by them);

   (b) $a[_h\ ]_h^{e_1} \rightarrow [_h b]_h^{e_2}$,
   for $h \in H, e_1, e_2 \in \{+, -, 0\}, a, b \in O$
   (*in* communication rules; an object is introduced in the membrane, possibly modified during this process; also the polarization of the membrane can be modified, but not its label);

   (c) $[_h a\ ]_h^{e_1} \rightarrow [_h]_h^{e_2} b$,
   for $h \in H, e_1, e_2 \in \{+, -, 0\}, a, b \in O$
   (*out* communication rules; an object is sent out of the membrane, possibly modified during this process; also the polarization of the membrane can be modified, but not its label);

   (d) $[_h a\ ]_h^e \rightarrow b$,
   for $h \in H, e \in \{+, -, 0\}, a, b \in O$
   (dissolving rules; in reaction with an object, a membrane can be dissolved, while the object specified in the rule can be modified);

   (e) $[_h a\ ]_h^{e_1} \rightarrow [_h b\ ]_h^{e_2} [_h c\ ]_h^{e_3}$,
   for $h \in H, e_1, e_2, e_3 \in \{+, -, 0\}, a, b, c \in O$
   (division rules for elementary membranes; in reaction with an object, the membrane is divided into two membranes with the same label, possibly of different polarizations; the object specified in the rule is replaced in the two new membranes by possibly new objects; the remaining objects are duplicated and may evolve in the same step by rules of type (a)).

The objects evolve in the maximally parallel manner, used by rules of type (a) or by rules of the other types, and the same is true at the level of membranes, which evolve by rules of types $(b) - (e)$. Inside each membrane, the rules of type

(a) are applied in the parallel way, with each copy of an object being used by only one rule of any type from (a) to (e). Each membrane can be involved in only one rule of types $(b), (c), (d), (e)$ (the rules of type (a) are not considered to involve the membrane where they are applied). Thus, in total, the rules are used in the usual non-deterministic maximally parallel manner, in a bottom-up way (first we use the rules of type (a), and then the rules of other types; in this way, in the case of dividing membranes, in the newly obtained membranes we duplicate the result of using first the rules of type (a)). Also as usual, only halting computations give a result, in the form of the number (or the vector) of objects expelled into the environment during the computation.

The set H of labels has been specified because it is also possible to allow the change of membrane labels. For instance, a division rule can be of the more general form

(e') $[_{h_1} a\,]_{h_1}^{e_1} \to [_{h_2} b\,]_{h_2}^{e_2} [_{h_3} c\,]_{h_3}^{e_3}$,
for $h_1, h_2, h_3 \in H, e_1, e_2, e_3 \in \{+, -, 0\}, a, b, c \in O$.

The change of labels can also be considered for rules of types (b) and (c). Also, we can consider the possibility of dividing membranes in more than two copies, or even of dividing non-elementary membranes (in such a case, all inner membranes are duplicated in the new copies of the membrane).

It is important to note that in the case of P systems with active membranes, the membrane structure evolves during the computation, not only by decreasing the number of membranes, due to dissolution operations (rules of type (d)), but also increasing the number of membranes, by division. This increase can be exponential in a linear number of steps: using successively a division rule, due to the maximal parallelism, in n steps we get 2^n copies of the same membrane. This is one of the most investigated ways of obtaining an exponential working space in order to trade time for space and solve computationally hard problems (typically **NP**-complete problems) in a feasible time (typically polynomial or even linear).

Some details can be found in Section 20, but we illustrate here the way of using membrane division in such a framework by an example dealing with the generation of all 2^n truth–assignments possible for n propositional variables.

Assume that we have the variables $x_1, x_2, \dots, x_n$; we construct the following system, of degree 2:

$$\begin{array}{rcl}
\Pi & = & (O, H, \mu, w_1, w_2, R), \\
O & = & \{a_i, c_i, t_i, f_i \mid 1 \leq i \leq n\} \cup \{\texttt{check}\}, \\
H & = & \{1, 2\}, \\
\mu & = & [_1[_2\ ]_2]_1, \\
w_1 & = & \lambda, \\
w_2 & = & a_1 a_2 \ldots a_n c_1, \\
R & = & \{[_2 a_i]_2^0 \to [_2 t_i]_2^0 [_2 f_i]_2^0 \mid 1 \leq i \leq n\} \\
 & \cup & \{[_2 c_i \to c_{i+1}]_2^0 \mid 1 \leq i \leq n-1\} \\
 & \cup & \{[_2 c_n \to \texttt{check}]_2^0,\ [_2\texttt{check}]_2^0 \to \texttt{check}[_2\ ]_2^+\}.
\end{array}$$

We start with the objects $a_1, \ldots, a_n$ in the inner membrane and we divide this membrane, repeatedly, by means of rules $[_2 a_i]_2^0 \to [_2 t_i]_2^0 [_2 f_i]_2^0$; note that the object a_i used in each step is non-deterministically chosen, but each division replaces that object by t_i (for *true*) in one membrane and with f_i (for *false*) in the other membrane, hence after n steps the obtained configuration is the same irrespective which was the order of expanding the objects. Specifically, we get 2^n membranes with label 2, each one containing a truth–assignment for the n variables. Actually, simultaneously with the division, we have to use the rules of type (a) which make evolve the “counter” c, hence at each step we increase by one the subscript of c. Therefore, when all variables are expanded, we get the object `check` in all membranes (the rule of type (a) is used first, and after that the result is duplicated in the newly obtained membranes). In step $n + 1$, this object exits each copy of membrane 2, changing its polarization to positive – this object is meant to signal the fact that the generation of all truth–assignments is completed, and we can start checking the truth values of the (clauses of a) propositional formula.

The previous example was chosen also for showing that the polarizations of membranes are not used during generating the truth–assignments, but it might be useful after that – and, up to now, this is the case in all polynomial time solutions to **NP**-complete problems obtained in this framework, in particular, for solving `SAT` (satisfiability of propositional formulas in the conjunctive normal form). This is an important *open problem* in this area: whether or not the polarizations can be avoided. This can be done if other ingredients are considered, such as label changing or division of non-elementary membranes, but without adding such features the best result obtained so far is that from [3] where it is proved that the number of polarizations can be reduced to two.

14 A Panoply of Possibilities for Having a Dynamical Membrane Structure

Membrane dissolving and dividing are only two of the many possibilities of handling the membrane structures. One of the early investigated additional possibility is membrane *creation*, based on rules of the form $a \to [_h v]_h$, where a is an object, v is a multiset of objects, and h is a label from a given set of labels. Using such a rule in a membrane j, we create a new membrane, with label h, having inside the objects specified by v. Because we know the label of the new membrane, we know the rules which can be used in its region (a "dictionary" of possible membranes is given, specifying the rules to be used in any membrane with labels in a given set). Because rules for handling membranes are of a more general interest (e.g., for applications), we illustrate them in Figure 5, where the reversibility of certain pairs of operations is also made visible.

For instance, converse to membrane division can be considered the operation of *merging* the contents of two membranes; formally, we can write such a rule in the form $[_{h_1} a]_{h_1} [_{h_2} b]_{h_2} \to [_{h_3} c]_{h_3}$, where a, b, c are objects and h_1, h_2, h_3 are labels (we have considered the general case, where the labels can be changed).

Actually, the merging operation can be considered also as the reverse of the *separation* operation, formalized as follows: let $K \subseteq O$ be a set of objects; a separation with respect to K is done by a rule of the form $[_{h_1}\ ]_{h_1} \to [_{h_2} K]_{h_2} [_{h_3} \neg K]_{h_3}$, with the meaning that the contents of membrane h_1 is split into two membranes, with labels h_2 and h_3, the first one containing all objects from K and the second one containing all objects which are not in K.

Simple to formalize are also the operations of *endocytosys* and *exocytosys* (we use these general names, although in biology there are distinctions depending on the size of the objects and the number of objects moved – phagocytosys, picocytosys, etc.).

For instance, $[_{h_1} a]_{h_1} [_{h_2}\ ]_{h_2} \to [_{h_2} [_{h_1} b]_{h_1}]_{h_2}$, for $h_1, h_2 \in H, a, b \in V$, is an endocytosys rule, stating that an elementary membrane labelled h_1 enters the adjacent membrane labelled h_2, under the control of object a; the labels h_1 and h_2 remain unchanged during this process, however, the object a may be modified to b during the operation. Similarly, the rule $[_{h_2} [_{h_1} a]_{h_1}]_{h_2} \to [_{h_1} b]_{h_1} [_{h_2}\ ]_{h_2}$, for $h_1, h_2 \in H, a, b \in V$, indicates an exocytosys operation: an elementary membrane labelled h_1 is sent out of a membrane labelled h_2, under the control of object a; the labels of the two membranes remain unchanged, but the object a from membrane h_1 may be modified during this operation. In the case of endocytosys, membrane h_2 can be a non-elementary one.

Finally, let us mention the operation of *gemmation*, by which a membrane is created inside a membrane h_1 and sent to a membrane with label h_2; the moving membrane is dissolved inside the target membrane h_2 thus releasing its contents there. In this way, multisets of objects can be transported from a membrane to another one in a protected way: the enclosed objects cannot be processed by the rules of the regions through which the travelling membrane passes. The travelling membrane is created with a label of the form $@_{h_2}$, which indicates that it is a temporary membrane, having to get dissolved inside the membrane with label h_2. Corresponding to the situation from biology, in [17], [18] one considers only the case where the membranes h_1, h_2 are adjacent, and directly placed in the skin membrane, but the operation can be generalized.

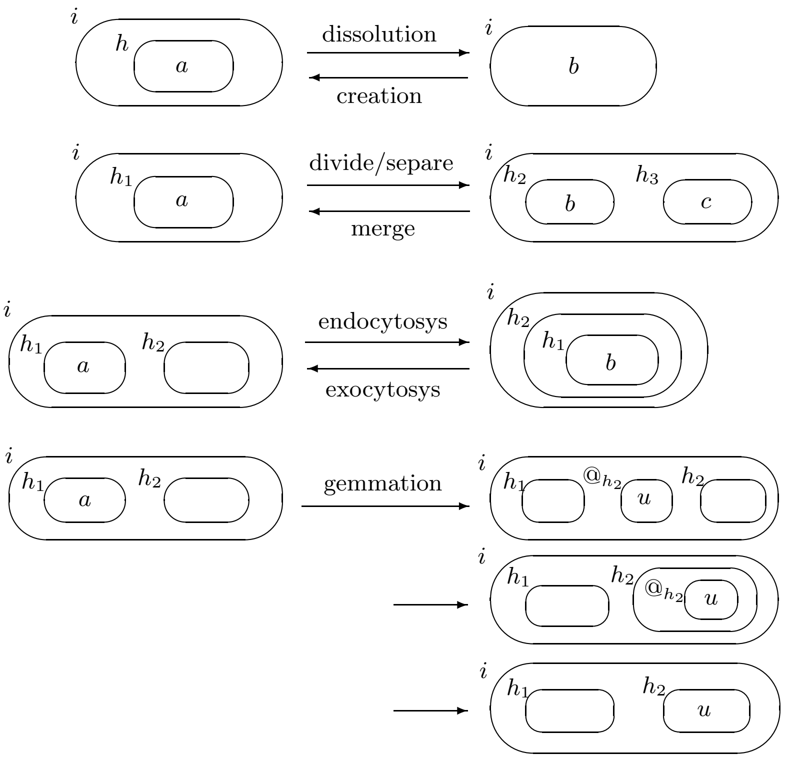


Figure 5: Membrane handling operations

Anyway, a gemmation rule is of the form $a \to [_{@_{h_2}} u]_{@_{h_2}}$, where a is an object and u a multiset of objects (but it can be generalized by creating several travelling membranes at the same time, with different destinations); the result of applying such a rule is as illustrated in the bottom of Figure 5, with the important mentioning that the crossing of one membrane takes one time unit (it is supposed that the travelling membrane finds the shortest path from the region where it is created to the target region).

Several other operations with membranes were considered, e.g., in the context of applications to linguistics, [13], as well as in [57], and in other papers, but we do not enter into further details here.

15 Structuring the Objects

In the previous classes of P systems, the objects were considered atomic, identified only by their name, but in a cell many chemicals are complex molecules (e.g., proteins, DNA molecules, other large macro-molecules), whose structure can be described by strings or more complex data, such as trees, arrays, etc. Also from a mathematical point of view is natural to consider P systems with string–objects.

Such a system has the form

$$\Pi = (V, T, \mu, M_1, \dots, M_m, R_1, \dots, R_m),$$

where V is the alphabet of the system, $T \subseteq V$ is the terminal alphabet, μ is the membrane structure (of degree $m \geq 1$), $M_1, \dots, M_m$ are finite sets of strings present in the m regions of the membrane structure, and $R_1, \dots, R_m$ are finite sets of string–processing rules associated with the m regions of μ.

We have given here the system in the general form, with a specified terminal alphabet (we say that the system is *extended*; if $V = T$, then the system is said to be *non-extended*), and without specifying the type of rules. These rules can be of various forms, but here we consider only two cases: rewriting and splicing.

In a *rewriting P system*, the string-objects are processed by rules of the form $a \to u(tar)$, where $a \to u$ is a context-free rule over the alphabet V and *tar* is one of the target indications *here, in, out.* When such a rule is applied to a string x_1ax_2 in a region i, we obtain the string x_1ux_2, which is placed in region i, in any inner region, or in the surrounding region, depending on whether *tar* is *here, in*, or *out*, respectively. The strings which leave the system do not come back; if they are composed only of symbols from T, then they are considered as generated by the system, and included in the language $L(\Pi)$.

There are several differences from the previous classes of P systems: we work with *sets* of string-objects, not with multisets; in order to introduce a string in the language $L(\Pi)$ we do not need to have a halting computation, because the strings do not change after leaving the system; each string is processed by only one rule (the rewriting is sequential at the level of strings), but in each step all strings from all regions which can be rewritten by local rules are rewritten by one rule.

In a *splicing P system*, we splicing rules as those from DNA computing (see [49], [85]), that is, of the form $u_1\#u_2\$u_3\#u_4$, where u_1, u_2, u_3, u_4 are strings over V. For four strings $x, y, z, w \in V^*$ and a rule $r : u_1\#u_2\$u_3\#u_4$, we write

$$\begin{aligned} (x, y) \vdash_r (z, w) \quad \text{if and only if} \quad & x = x_1u_1u_2x_2,\ y = y_1u_3u_4y_2, \\ & z = x_1u_1u_4y_2,\ w = y_1u_3u_2x_2, \\ & \text{for some } x_1, x_2, y_1, y_2 \in V^*. \end{aligned}$$

We say that we splice x, y at the sites u_1u_2, u_3u_4, respectively, and the result of the splicing (obtained by recombining the fragments obtained by cutting the strings as indicated by the sites) are the strings z, w.

In our case we add target indications to the two resulting strings, that is, we consider rules of the form $r : u_1\#u_2\$u_3\#u_4(tar_1, tar_2)$, with tar_1, tar_2 one of *here, in, out.* The meaning is as standard: after splicing the strings x, y from a given region, the resulting strings z, w are moved to the regions indicated by tar_1, tar_2, respectively. The language generated by such a system consists again of all strings over T sent into the environment during the computation, without considering only halting computations.

We do not give here an example of a rewriting or a splicing P system, but we pass to introducing an important extension of rewriting rules, namely, *rewriting with replication*, [60]. In such systems, the rules are of the form $a \to (u_1, tar_1)||(u_2, tar_2)|| \dots ||(u_n, tar_n)$, with the meaning that by rewriting a string x_1ax_1 we get n strings, $x_1u_1x_2, x_1u_2x_2, \dots, x_1u_nx_2$, which have to be moved in the regions indicated by targets $tar_1, tar_2, \dots, tar_n$, respectively. In this case we work again with halting computations, and the motivation is that if we do not impose the halting condition, then the strings $x_1u_ix_2$ evolve completely independently, hence we can replace the rule $a \to (u_1, tar_1)||(u_2, tar_2)|| \dots ||(u_n, tar_n)$ with n rules $a \to (u_i, tar_i), 1 \le i \le n$, without changing the language; that is, replication makes a difference only in the halting case.

The replicated rewriting is important for the possibility to replicate strings, thus enlarging the workspace, and indeed, this is one of the frequently used ways to generate an exponential workspace in linear time, used then for solving computationally hard problems in polynomial time.

Besides these types of rules for string processing, also other kinds of rules were used, such as insertion and deletion, context adjoining in the sense of Marcus contextual grammars [76], splitting, conditional concatenation, and so on, sometimes with motivations from biology, where several similar operations can be found, e.g., at the genome level.

16 Tissue–Like P Systems

We pass now to consider a very important generalization of the membrane structure, passing from the cell-like structure, described by a tree, to a tissue–like structure, with the membranes placed in the nodes of an arbitrary graph (which corresponds to the complex communication networks established among adjacent cells, by making their protein channels to cooperate, moving molecules directly from one cell to another cell, [64]). Actually, in the basic variant of tissue–like P systems, this graph is a virtually total one, what matters is the communication graph, dynamically defined during the computation. In short, several (elementary) membranes – also called cells – are freely placed in a common environment; they can communicate either with each other or with the environment by symport/antiport rules. Specifically, we consider antiport rules is of the form $(i, x/y, j)$, where i, j are labels of cells or, at most one, is zero, identifying the environment, and x, y are multisets of objects. The meaning is that the multiset x is moved from i to j, at the same time with moving the multiset y from j to i. If one of the multisets x, y is empty, then we have, in fact, a symport rule. Therefore, the communication among cells is done either directly, in one step, or indirectly, through the environment: one cell throws some objects out and other cells can take these objects, in the next step or later. As in symport/antiport P systems, the environment contains a specified set of objects in arbitrarily many copies. A computation develops as standard, starting from the initial configuration and using the rules in the non-deterministic maximally parallel manner. When halting, we count the objects from a specified cell, and this is the result of the computation.

The graph plays a more important role in so-called *tissue-like P systems with channel-states*, [41], which are constructs of the form

$$\Pi = (O, T, K, w_1, \ldots, w_m, E, syn, (s_{(i,j)})_{(i,j)\in syn}, (R_{(i,j)})_{(i,j)\in syn}, i_o),$$

where O is the alphabet of *objects*, $T \subseteq O$ is the alphabet of *terminal* objects, K is the alphabet of *states* (not necessarily disjoint of O), $w_1, \ldots, w_m$ are strings over O representing the initial multisets of objects present in the cells of the system (it is assumed that we have m cells, labelled with $1, 2, \ldots, m$), $E \subseteq O$ is the set of objects present in arbitrarily many copies in the environment, $syn \subseteq \{(i,j) \mid i,j \in \{0,1,2,\ldots,m\}, i \neq j\}$ is the set of links among cells (we call them *synapses*; 0 indicates the environment) such that for $i,j \in \{0,1,\ldots,m\}$ at most one of $(i,j),(j,i)$ is present in syn, $s_{(i,j)}$ is the *initial state* of the synapse $(i,j) \in syn$, $R_{(i,j)}$ is a finite set of rules of the form $(s, x/y, s')$, for some $s, s' \in K$ and $x, y \in O^*$, associated with the synapse $(i,j) \in syn$, and, finally, $i_o \in \{1,2,\ldots,m\}$ is the *output* cell.

We note the restriction that there is at most one synapse among two given cells, and the synapse is given as an ordered pair (i,j), with which a state from K is associated. The fact that the pair is ordered does not restrict the communication among the two cells (or between a cell and the environment), because we work here in the general case of antiport rules, specifying simultaneous movements of objects in the two directions of a synapse.

A rule of the form $(s, x/y, s') \in R_{(i,j)}$ is interpreted as an antiport rule $(i, x/y, j)$ as above, acting only if the synapse (i,j) has the state s; the application of the rule means (1) moving the objects specified by x from cell i (from the environment, if $i = 0$) to cell j, at the same time with the move of the objects specified by y in the opposite direction, as well as (2) changing the state of the synapse from s to s'.

The computation starts with the multisets specified by $w_1, \ldots, w_m$ in the m cells; in each time unit, a rule is used on each synapse for which a rule can be used (if no rule is applicable for a synapse, then no object passes over it and its state remains unchanged). Therefore, the use of rules is sequential at the level of each synapse, but it is parallel at the level of the system: all synapses which can use a rule must do it (the system is synchronously evolving). The computation is successful if and only if it halts and the result of a halting computation is the number of objects from T present in cell i_o in the halting configuration (the objects from $O - T$ are ignored when considering the result). The set of all numbers computed in this way by the system Π is denoted by $N(\Pi)$. Of course, also vectors can be computed, by considering the multiplicity of objects from T present in cell i_o in the halting configuration.

A still more elaborated class of systems, called *population P systems*, were investigated in the last time in a series of papers by F. Bernardini and M. Gheorghe – see, e.g., [14] – with motivations related to the dynamics of cells in skin-like tissues, populations of bacteria, colonies of ants. These systems are highly dynamical; not only the links between cells, corresponding to the channels from the previous model, with states assigned to these channels, can change during the evolution of the system, but also the cells can change their name, can disappear (get dissolved) and can divide, thus producing new cells; these new cells inherit, in a well specified sense, the links with the neighboring cells of the parent cell. The generality of this model makes it rather attractive for applications in areas as those mentioned above, related to tissues, populations of bacteria, etc.

17 Neural–Like P Systems

The next step in enlarging the model of tissue-like P systems is to consider more complex cells, for instance, moving the states from the channels between cells to the cells themselves – still preserving the network of synapses. This last term directly suggests the neural motivation of these attempts, aiming to capture something from the intricate structure of neural networks, of the way the neurons are linked and cooperate in the most efficient computer ever invented, the human brain.

We do not recall here the formal definition of a neural-like P system, but we refer to [82] for details, and here we only present the general idea behind these systems.

We again use a population of cells (each one identified by its label) linked by a specified set of synapses. This time, each cell has at every moment a state from a given finite set of states, a contents, in the form of a multiset of objects from a given alphabet of objects, and a set of rules for processing these objects.

The rules are of the form $sw \to s'(x, here)(y, go)(z, out)$, where s, s' are states and w, x, y, z are multisets of objects; in state s, the cell consumes the multiset w and produces the multisets x, y, z; the objects from multiset x remain in the cell, those of multiset y have to be communicated to the cells towards which there are synapses starting in the current cell; a multiset z, with the indication *out*, is allowed to appear only in a special cell, which is designated as the output cell, and for this cell, the use of the previous rule entails sending the objects of z to the environment.

The computation starts with all cells in specified initial states, with initially given contents, and proceeds by processing the multisets from all cells, simultaneously, according to the local rules, redistributing the obtained objects along synapses, and sending a result into the environment through the output cell; a result is accepted only when the computation halts.

Because of the use of states, there are several possibilities of processing the multisets of objects from each cell. In the *minimal* mode, a rule is chosen and applied once to the current pair state–multiset. In the *parallel* mode, a rule is chosen, e.g., $sw \to s'w'$, and used in the maximally parallel manner: the multiset w is identified in the cell contents, in the maximal manner, and the rule is used for processing all these instances of w. Finally, in the *maximal* mode, we apply in the maximally parallel manner all rules of the form $sw \to s'w'$, that is, with the same states s and s' (note the difference with the parallel mode, where in each step we choose a rule and we use only this rule as many times as possible).

Then, there also are three possibilities to move the objects between cells (of course, we only move objects produced by rules in multisets with the indication *go*). Assume that we have applied a rule $sw \to s'(x, here)(y, go)$ in a given cell i. In the *spread* mode, the objects from y are non-deterministically distributed to all cells j such that (i, j) is a synapse of the system. In the *one* mode, all the objects from y are sent to one cell j, again provided that the synapse (i, j) exists. Finally, we can also replicate the objects of y and each object from y is sent to all cells j such that (i, j) is an available synapse – this is the *replicative* mode.

Note that the states ensure a powerful way to control the work of the system, that the parallel and maximal modes are efficient ways to process the multisets, and that the replicative mode of distributing the objects provides the possibility of increasing exponentially the number of objects, in linear time. All together, these features make the neural-like P system both very powerful and very efficient computing devices. However, this class of P systems still waits for a systematic investigation – maybe starting with questioning their very definition, and changing this definition in such a way to capture more realistic brain–like features.

18 Other Ways of Using a P System; P Automata

In all previous sections we have considered the various types of P systems as *generative devices*: starting from an initial configuration, because of the non-determinism of using the rules we can proceed along various computations, at the end of which we get a result; in total, all successful computations provide a set of numbers, of vectors or numbers, or a language (set of strings), depending on the way the result of a computation is defined. This approach, grammar oriented, is only one possibility, mathematically attractive and important theoretically, but not useful from a practical point of view, when dealing with specific problems to solve and specific functions to compute. However, a P system can be used also for computing functions and for solving problems (in a standard algorithmic manner).

Actually, besides the generative approach, there are two other general (related) ways of using a P system: in the *accepting* mode, and in the *transducer* mode. In both cases, an input is provided to the system, in a way depending on the type of systems at hand. For instance, in a symbol-object P system, besides the initial multisets present in the regions of the membrane structure, we can introduce a multiset w_0 in a specified region, just adding the objects of w_0 to the objects present in that region. In the string case, a string can be added, possibly inserted in one of the existing strings. The computation proceeds, and if it halts, then we say that the input is accepted (or recognized). In the transducer mode, we do not only have to halt, but we also collect an output, from a specified output region, internal to the system or the environment.

Now, an important distinction appears, between systems which behave deterministically (in each moment, at most one transition is possible, hence either the computation stops, or it continues in a unique mode), and those which work in a non-deterministic way. Such a distinction does not makes much sense in the generative mode, especially if only halting computations provide a result, at their end: such a system can generate only a single result. In the case of computing functions or solving problems (e.g., decidability problems), the determinism is obligatory.

Again a distinction is in order: actually, we are not interested in the way the system behaves, deterministically or non-deterministically, but in the uniqueness and the reliability of the result. If, for instance, we ask whether or not a propositional formula in conjunctive normal form is satisfiable or not, we do not care how the result is obtained, but we want to make sure that it is the right one: yes or no. Whether or not the truth–assignments were created as in the example from Section 13, expanding the variables in a random order, is not relevant, important is that after n steps we get the same configuration. This brings into the stage the important notion of *confluence*. A system is *strongly confluent* if, starting from the initial configuration and behaving we-do-not-care-how, after a while it reaches a configuration from where the computation continues in a deterministic way. Because we are only interested in the result of computations (e.g., in the answer, yes or no, to a decidability problem), we can relax the previous condition, to a *weak confluence* property: irrespective how the system works, it always halts and all halting computations provide the same result. These notions will be essentially invoked when discussing the efficiency of P systems, as in Section 20.

Here let us consider in some details the accepting mode of using a P system. Given, for instance, a transition P system Π, let us denote by $N_a(\Pi)$ the set of all numbers accepted by Π, in the following sense: we introduce a^n, for a specified object a, into a specified region of Π, and we say that n is accepted if and only if there is a computation of Π, starting from this augmented initial configuration, which halts. In the case of systems taking objects from the environment, such as the symport/antiport or the communicative ones [92], we can consider that the system accepts/recognizes the sequence of objects taken from the environment during a halting computation (if several objects are brought into the system at the same time, then all their permutations are accepted as substrings of the accepted string). Similar strategies can be followed for all types of systems, tissue-like and neural-like included (but P automata were first introduced in the symport/antiport case, in [37] – see also [39]).

The above set $N_a(\Pi)$ was defined in general, for non-deterministic systems, but, clearly, in the accepting mode the determinism can be imposed (the non-determinism is moved to the environment, to the "user", which provides an input, unique, but non-deterministically chosen, from which the computation starts). Note that the example of a P system with symport/antiport rules from Section 11 works in the same manner for an accepting register machine (a number is introduced in the first register and it is accepted if and only if the computation halts; in such a case, the `add` instructions can be deterministic, that is, with labels l_2, l_3 identical (one simply writes $l_1 : (\mathtt{add}(r), l_2)$, with the continuation unique), and in this case the P system itself is deterministic.

19 Universality

The initial goal of membrane computing was to define computability models inspired from the cell biology, and indeed a large part of the investigations in this area was devoted to producing computing devices and examining their computing power, in comparison with the standard models in computability theory, Turing machines and their restricted variants. As it turns out, most of the considered classes of P systems are equal in power with Turing machines. In a rigorous manner, we have to say that they are Turing complete (or computationally complete), but because the proofs are always constructive, starting the constructions from these proofs from universal Turing machines or from equivalent devices, we obtain universal P systems (able to simulate any other P system of the given type, after introducing a "code" of the particular system as an input in the universal one – the precise definition should be given for every particular type of systems). That is why we speak about *universality* results, and not about computational completeness.

All classes of systems considered above, whether cell-like, tissue-like, or neural-like, with symbol-objects or string-objects, working in the generative or the accepting modes, of course, with certain combinations of features, are known to be universal. The cell turns out to be a very powerful "computer", both when standing alone and in tissues.

In general, for P systems working with symbol-objects, these universality results are proved by simulating computing devices which are known to be universal, and which either work with numbers or do not essentially use the positional information from strings. This is true/possible for register machines, matrix grammars (in the binary normal form), programmed grammars, regularly controlled grammars, graph-controlled grammars (but not for arbitrary Chomsky grammars and for Turing machines, which can be used only in the case of string-objects). The example from Section 11 illustrates a universality proof for the case of P systems with symport/antiport rules (with rules of a sufficiently large weight – see below stronger results from this point of view).

We do not enter here in other details, than specifying some notations which are already standard in membrane computing and, after that, mentioning some universality results of a particular interest.

As for notations, the family of sets $N(\Pi)$ of numbers (we keep from here the symbol N) generated by P systems of a specified type (we keep P), working with symbol-objects (O), having at most m membranes, and using features/ingredients from a given *list* is denoted by $NOP_m($*list-of-features*$)$. If we compute sets of vectors, then we write $PsOP_m(\dots)$, with Ps coming from "Parikh set". When the systems work in the accepting mode, one writes $N_aOP_m(\dots)$, and when string-objects are used, one replaces N with L (from "languages") and O with S (from "strings"), thus obtaining families $LSP_m(\dots)$. The case of tissue-like systems is indicated by adding the letter t before P, thus obtaining $NOtP_m(\dots)$, while for neural-like systems one uses instead the letter n. When the number of membranes is not bounded, the subscript m is replaced by $*$, and this is a general convention, used also for other parameters.

Now, in what concerns the list of features, they can be taken from an endless pool: using cooperative rules is indicated by *coo*, catalytic rules are indicated by *cat*, with the mentioning that the number of catalysts matters, hence we use cat_r in order to indicate that we use systems with at most r catalysts; bi-stable catalysts are indicated by $2cat$ ($2cat_r$, if at most r catalysts are used); similarly, mobile catalysts are indicated by $Mcat$. When using a priority relation, we write *pri*, for the actions δ, τ we write simply δ, τ. Membrane creation is represented by *mcre*, endocytosys and exocytosys operations are indicated by *endo, exo*, respectively. In the case of P systems with active membranes, one directly lists the types of rules used, from (a) to (e), as defined and denoted in Section 13.

For systems with string-objects, one write *rew*, $repl_d$, *spl* for indicating that one uses rewriting rules, replicated rewriting rules (with at most d copies of each string produced by replication), splicing rules, respectively.

In the case of (cell-like or tissue-like) systems using symport/antiport rules, we have to specify the maximal weight of the used rules, and this is done by writing $sym_p, anti_q$, meaning that symport rules of weight at most p and antiport rules of weight at most q are allowed.

There are many other features, with notations of the same type (as mnemonic as possible), which we do not recall here. Sometimes, when it is important to show in the name of the discussed family that a specific feature fe is *not* allowed, one uses to write nFe – for instance, $nPri$ for not using priorities (note the capitalization of the initial name of the feature), $n\delta$, etc.

Specific examples of families of numbers (we do not consider here also sets of vectors or languages, although, as we have said above, a lot of universality results are known for all cases) appear in the few universality results which we recall below. In these results, NRE denotes the family of Turing computable sets of numbers (the notation comes from the fact that these numbers are the length sets of recursively enumerable languages, those generated by Chomsky type-0 grammars, or many types of regulated rewriting grammars, and recognized by Turing machines). The family NRE is also the family of sets of numbers generated/recognized by register machines. When dealing with vectors of numbers, hence with the Parikh images of languages (or with the sets of vectors generated/recognized by register machines), we write $PsRE$.

Here are some universality results (for the proofs, see the mentioned papers):

1. $NRE = NOP_1(cat_2)$, [38].
2. $NRE = NOP_3(sym_1, anti_1) = NOP_3(sym_2, anti_0)$, [4].
3. $NRE = NOP_3((a), (b), (c))$, [65].
4. $NRE = NOP_9(endo, exo)$, [59].
5. $NRE = NSP_3(repl_2)$, [61].

All results above hold true also for vectors of numbers.

In all these results, the number of membranes sufficient for obtaining the universality is pretty small (the equality $NRE = NOP_9(endo, exo)$ was only recently proven and it is very probable that it will be improved in the number of membranes). Actually, in all cases when the universality holds (and the code of a particular system is introduced in a universal system in such a way that the membrane structure is not modified), the hierarchy on the number of membranes collapses, because a number of membranes as large as the degree of the universal system suffices.

Still, "the number of membranes matters", as we read already in the title of [54]: there are (sub-universal) classes of P systems for which the number of membranes induces an infinite hierarchy of families of sets of numbers (see also [55]).

20 Solving Computationally Hard Problems in Polynomial Time

The computational power (the "competence") 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*, the resources used for solving problems. In general, the research in natural computing is especially concerned with this issue. 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 time.

We have discussed above three basic ways to construct such an exponential space in cell-like P systems, namely, membrane division (the same effect has the separation operation, as well as other operations which replicate partially or totally the contents of a membrane), membrane creation (combined with the creation of exponentially many objects), and string replication. Similar possibilities are offered by cell division in tissue-like systems and by objects replication in neural-like systems. Also the possibility to use a pre-computed exponential workspace, unstructured and non-active (e.g., with the regions containing no object) was considered.

In all these cases polynomial or pseudo–polynomial solutions to **NP**-complete problems were obtained. The first problem addressed in this context was `SAT` [81], [79] (the solution was improved in several respects in other subsequent papers), but similar solutions are reported in the literature for the Hamiltonian Path problem, the Node Covering problem, the problem of inverting one-way functions, the Subset-sum, and the Knapsack problems (note that the last two are numerical problems, where the answer is not of the yes/no type, as in decidability problems), and for several other problems. Details can be found in [82], [87], as well as in the web page of the domain, [102].

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, or, in the case of the weak confluence, all computations stop in a determined time and give the same result), 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 problem instances) by a Turing machine, working a polynomial time. A more relaxed framework is that where a *semi-uniform* construction is allowed: carried out in polynomial time by a Turing machine, but starting from the instance itself to be solved (the condition to have a polynomial time construction ensures the "honesty" of the construction: the solution to the problem cannot be found during the construction phase).

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. The **PSPACE**-complete problem used in this proof was `QSAT` (see [92], [5] for details).

There also are many *open problems* in this area. We have mentioned already the intriguing question whether polynomial solutions to **NP**-complete problems can be obtained through P systems with active membranes without polarizations (and without label changing possibilities of other additional features). In general, the borderline between *efficiency* (the possibility to solve **NP**-complete problems in polynomial time) and *non-efficiency* is a challenging topic. Anyway, we know that membrane division cannot be avoided ("Milano theorem": a P system without membrane division can be simulated by a Turing machine with a polynomial slowdown, see [100], [101]).

21 Focusing on the Evolution

The computational power is of interest for theoretical computer science, computational efficiency is of interest for practical computer science, but none of these is of a direct interest for biology. Actually, this last statement is not at all correct: if a biologist is interested in simulating a cell – and this seems to become a major concern of to-day biology, see [58], [53] and other sources – then the generality of the model (its comparison with Turing machines and its restrictions) is directly linked to the possibility of solving algorithmically questions about the model. Just an example: is a given configuration reachable from the initial configuration? Imagine that the initial configuration represents a healthy cell and we are interested whether a sickness state is ever reached. Then, if both healthy and non-healthy configurations can be reached, the question appears whether we can find the "bifurcation configurations", and this is again a reachability issue. The relevance of such a "purely theoretical" problem is clear, and its answer directly depends on the generality (hence the power) of the model. Then, of course, the time needed for answering the question is a matter of computational complexity. So, both the power and the efficiency are, indirectly, of interest also for biologists, so we (the biologists, too) should be more careful when asserting that a given type of "theoretical" investigation is not of interest for biology.

Still, the immediate concern of biological research is the evolution of biological systems, their life, whatever this means, not the result of a specific evolution. Otherwise stated, halting computations are of interest for computer science, of direct interest for biology is the computation/evolution itself. Although membrane computing was not intended initially to deal with such issues, a series of recent investigations indicate a strong tendency towards considering P systems as dynamical systems. This does not concern only the fact that, besides the rules for object evolution, a more complete panoply of possibilities were imagined for making also the membrane structure evolve, with specific developments in the case of tissue-like and population P systems, where also the links between cells are evolving, but this concerns especially the formulation of questions which are typical for dynamical systems study. Trajectories, periodicity and pseudo-periodicity, stability, attractors, basins, oscillations and many other concepts were brought in the framework of membrane computing – and the enterprise is not trivial, as these concepts were initially introduced in areas handled with continuous mathematics tools (mainly differential equations). A real program of defining discrete dynamical systems, with direct application to the dynamics of P systems, was started by V. Manca and his collaborators; we refer to [19], [68], [67], [15], etc. for details.

22 Recent Developments

Of course, the specification "recent" is risky, as it can soon become obsolete, but still we want to mention here some directions of research and some results which were not presented before – after just repeating the fact that topics such as complexity classes and polynomial solutions to hard problems, dynamical systems approaches, population P systems (in general, systems dealing with populations of cells, as in tissue-like or neural-like systems) are of a strong current interest which will probably lead to significant theoretical and practical results. To these trends we can add another general, and not very structured yet, topic: using non-crisp mathematics, handling uncertainty by means of probabilistic, fuzzy sets, rough sets theories.

However, we want here to also point out a few more precise topics.

One of them concerns the role of time in P systems. The synchronization and the existence of a global clock are too strong assumptions (from a biological point of view). What about P systems where there exists no internal clock, and all rules have different times to get applied? This can mean both that the duration needed by a rule to get applied can differ from the duration of another rule, and, the extreme possibility, that the duration is not known at all. In the first case, we can have a timing function, assigning durations to rules, in the second case even such an information is missing (e.g., a rule $a \to v$ should wait for a rule $c \to ba$ to be completed in order to have an available a to process). How the power of a system depends on the timing function? Are there time-free systems, which generate the same set of numbers irrespective which is the time function which associates durations with its rules? Such questions are addressed in a series of papers by M. Cavaliere and D. Sburlan; see e.g., [27], [28].

Another powerful idea explored by M. Cavaliere and his collaborators is that of coupling a simple bio-inspired *system*, Sys, such as a P system without a large computing power, with an *observer* Obs, a finite state machine which analyzes the configurations of the system Sys along the evolutions of the system; from each configuration either a symbol is produced or nothing (that is, the "result" of that configuration is the empty string λ); in a stronger variant, the observer can also reject the configuration and hence the system evolution, trashing it. The couple (Sys, Obs), for various simple systems and multiset processing finite automata, proved to be a very powerful computing device, universal even for very weak systems Sys. Details can be found in [24], [25].

An idea recently explored is that of trying to bound the number of objects used in a P system, and still computing all Turing computable numbers. The question can be seen as "orthogonal" on the usual questions concerning the number of membranes and the size of rules, as, intuitively, one of these parameters should be left free in order to codify and handle an arbitrary amount of information by using a limited number of objects. The first results of this type were given in [84] and they are surprising: in formal terms, we have $NRE = NOP_4(obj_3, sym_*, anti_*)$ (P systems with four membranes and symport and antiport rules of arbitrary weight are universal even when using only three objects). In turn, two objects (but without a bound on the number of membranes) are sufficient in order to generate all sets of vectors computed by so-called (see [46]) partially blind counter machines (for sets of numbers the result is not so interesting, because partially blind counter machines accept only semilinear sets of numbers, while the sets of vectors they accept can be non-semilinear).

Other interesting topics recently investigated which we only list here concern the reversibility of computations in P systems [63], energy accounting (associating quanta of energy to objects or to rules, handled during the computation) [43], [42], [62], relations with grammar systems and with colonies [83], descriptional complexity, non-discrete multisets [74], [34].

We close this section by mentioning the notion of the *Sevilla carpet* introduced in [31], which proposes a way to describe the time-and-space complexity of a computation in a P system by considering the two–dimensional table of all rules used in each time unit of a computation. This corresponds to the Szilard language from language theory, with the complication now that we use several rules in the same step, and each rule is used several times. Considering all the information concerning the rules we can get a global evaluation of the complexity of a computation – as nicely illustrated, for instance, in [90] and [47].

23 Applications; The Attractiveness of Membrane Computing as a Modelling Framework

Finally, let us shortly discuss some applications of membrane computing – starting however with a general discussion about the features of this area of research which make it attractive for applications in several disciplines, especially for biology.

First, there are several keywords which are genuinely proper to membrane computing and which are of interest for many applications: *distribution* (with the important system-part interaction, emergent behavior, non-linearly resulting from the composition of local behaviors), *discrete mathematics* (continuous mathematics, especially systems of differential equations, has a glorious history of applications in many disciplines, such as astronomy, physics, meteorology, but it failed to prove adequate for linguistics, and cannot cover more than local processes in biology because of the complexity of processes and, in many cases, their imprecise character; then, a basic question is whether the biological reality is of a continuous nature or of a discrete nature – as languages proved to be, which the second possibility ruling out the usefulness of many tools from continuous mathematics), *algorithmicity* (by definition, P systems are computability models, of the same type as Turing machines or other classic representations of algorithms, hence easy to be simulated on a computer), *scalability/extensibility* (this is one of the main difficulties of using differential equations in biology), *transparency* (multiset rewriting rules are nothing else than reaction equations as customarily used in chemistry and bio-chemistry, without any "mysterious" notation and, behind the notation, "mysterious" behavior), *parallelism* (a dream of computer science, a common sense in biology), *non-determinism* (let us compare the "program" of a P system, which is a set of instructions/rules, with the only structure being that imposed by localization to regions, but without any ordering/structure inside each region, with the rigid sequences of instructions of programs written in usual programming languages), *communication* (with the marvellous and still not completely understood way the life is coordinating the many processes taking place in a cell, and in tissues, organs, organisms, in contrast with the costly way of coordinating/synchronizing computations in parallel electronic computing architectures, where the communication time become prohibitive with the increase of the number of processors), and so on and so forth.

Then, for biology, besides the easy understanding of the formalism and the transparency of the (graphical and symbolic) representations, encouraging should be also the simple observation that membrane computing emerged as a bio-inspired research area, explicitly looking to the cell for finding computability models (though, not looking initially for models of relevance for the biological research), hence it is just natural to try to improve these models and use them in the study of the very originating ground. This should be put in contrast with the attempt to "force" models and tools developed in other scientific areas, e.g., in physics, to cover biological facts, presumably of a genuinely different nature as that of the area for which these models and tools were created and proven to be adequate/useful.

Coming back to the important distinction between continuous and discrete tools, it should be emphasized that significant results can be obtained by computer simulations of discrete data, in particular, of multisets. This has been convincingly proven in many cases – see [94], [93], [95], [74], [68], [67] – and reminds of the assertion made in several places that cellular automata can be an alternative/substitute for differential equations; comparing the rigid structure of cellular automata with the flexibility of membrane systems we can safely infer that if this assertion is valid for cellular automata, then it is still "more valid" for P systems.

Now, in what concerns the applications themselves reported up to now, they are developed at various levels. In many cases, what is actually used is the *language* of membrane computing, having in mind three dimensions of this aspect: (i) the long list of concepts either newly introduced, or related in a new manner in this area, (ii) the mathematical formalism of membrane computing, and (iii) the graphical language, the way to represent membranes, cell-like structures, tissue-like structures. It is easy to illustrate each of these three points, e.g., producing a list of concepts already mentioned in the present paper, or recalling some of the stabilized notations (for instance, representing a membrane by square brackets, with labels and possibly electrical charges, with the multiset rewriting rules, the symport and antiport rules), but we only say some words about the graphical language. Many ingredients are, in a great extent, known: Euler–Wenn diagrams (here, without intersection and with a unique superset, corresponding to the skin membrane), with labels assigned to membranes, with multisets (not sets!) of objects placed inside, with arrows describing the communication channels in the case of tissue-like and neural-like systems; what is essentially new is that also the rules for the evolution of the system are written in compartments in the case of multiset or string rewriting rules, and near membranes in the case of symport/antiport and boundary rules (which are associated with membranes). Thus, not only the state of the system is displayed, but also the "evolution engine", the rules. Also, the localization is apparent, both for objects and rule.

However, this level of application/usefulness is only a preliminary, superficial one. The next level is to use tools, techniques, results of membrane computing, and here there appears an important question: to which aim? Solving problems already stated, e.g., by biologists, in other terms and another framework, could be an impressive achievement, and this is the most natural way to proceed – but not necessarily the most efficient one, at least at the beginning. New tools can suggest new problems, which either cannot be formulated in a previous framework (in plain language, as it is the case in biology, whatever specialized the specific jargon is, or using other tools, such as differential equations) or have no chance to be solved in the previous framework. Problems of the first type (already examined by biologists, mainly experimentally) concern, for instance, correlations of processes, of the presence/absence of certain chemicals, their multiplicity (concentration, population) in a given compartment, their interaction, while of the second type are topics related to the trajectories of bio-systems when modelled as dynamical systems (e.g., a sequence of configurations can be finite or infinite, while in the latter case it can be periodic, ultimately periodic, almost periodic, quasi-periodic, etc., notions which are not yet present in the index of notions of biological books).

Applications of all these types were reported in the literature of membrane computing. As expected and as natural, most applications were carried out in biology, but also applications in computer graphics (where the compartmentalization seems to add a significant efficiency to well-known techniques based on L systems, [44]), linguistics (both as a representation language for various concepts related to language evolution, dialogue, semantics [13], and making use of the parallelism, in solving parsing problems in an efficient way [45]), management (again, mainly at the level of the formalism and the graphical language, see, e.g., [11], [12]), in devising sorting and ranking algorithms [7], handling 2D structures [29], etc.

In turn, the applications in biology follow in most cases a scenario of the following type: one examines a piece of reality, in general from the biochemistry of the cell, one writes a P system modelling the respective process, one writes a program simulating that system (or one uses one of the existing programs), and one performs a large number of experiments with the program (this is much cheaper than conducting laboratory experiments), tuning certain parameters, and looking for the evolution of the system (usually, for the population of certain objects). We do not recall any detail here, but we refer to the chapter of [82] devoted to biological applications, as well as to the papers available in the web page [102], and, especially, to the forthcoming volume [30]. Anyway, the investigations are somewhat preliminary, but the progresses are obvious and the hope is to have in the near future applications of an increased interest for biologists.

This hope is supported also by the fact that more and more powerful simulations/implementations of various classes of P systems are available, with better and better interfaces, which allow for the friendly interaction with the program. We avoid to plainly say that we have "implementations" of P systems, because of the inherent non-determinism and the massive parallelism of the basic model, features which cannot be implemented, at least in principle, on the usual electronic computer – but which can be implemented on a dedicated, reconfigurable, hardware, as done in [88], or on a local network, as reported in [32] and [96]. This does not mean that simulations of P systems on usual computers are not useful; actually, such programs were used in all biological applications mentioned above, and can also have important didactic and research applications. An overview of membrane computing software reported in literature (some programs are available in the web page [102]) can be found in [48].

24 Closing Remarks

The present paper should be seen only as a general overview of membrane computing, with the choice of topics intended to be as pertinent as possible, but, of course, not completely free of a subjective bias. The reader interested in further technical details, formal definitions, proofs, research topics and open problems, or in details concerning the applications (and the software behind them) is advised to consult the comprehensive web page from `http://psystems.disco.unimib.it`. A complete bibliography of membrane computing can be found there, with many papers available for downloading (in particular, one can find there the proceedings volumes of the yearly Workshops on Membrane Computing, as well as of the yearly Brainstorming Weeks on Membrane Computing).

References

[1] L.M. Adleman: Molecular Computation of Solutions to Combinatorial Problems. *Science*, 226 (November 1994), 1021–1024.

[2] B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, P. Walter: *Molecular Biology of the Cell*, 4th ed. Garland Science, New York, 2002.

[3] A. Alhazov, R. Freund: On the Efficiency of P Systems with Active Membranes and Two Polarizations. In [72], 147–161.

[4] A. Alhazov, M. Margenstern, V. Rogozhin, Y. Rogozhin, S. Verlan: Communicative P Systems with Minimal Cooperation. In [72], 162–178.

[5] A. Alhazov, C. Martín-Vide, L. Pan: Solving a PSPACE-Complete Problem by P Systems with Restricted Active Membranes. *Fundamenta Informaticae*, 58, 2 (2003), 67–77.

[6] A. Alhazov, C. Martín-Vide, Gh. Păun, eds.: *Pre-Proceedings of Workshop on Membrane Computing*, WMC 2003, Tarragona, Spain, July 2003. Technical Report 28/03, Rovira i Virgili University, Tarragona, 2003.

[7] A. Alhazov, D. Sburlan: Static Sorting Algorithms for P Systems. In [70], 17–40.

[8] I.I. Ardelean: The Relevance of Biomembranes for P Systems. *Fundamenta Informaticae*, 49, 1–3 (2002), 35–43.

[9] J.-P. Banâtre, A. Coutant, D. Le Métayer: A Parallel Machine for Multiset Transformation and Its Programming Style. *Future Generation Computer Systems*, 4 (1988), 133–144.

[10] J.-P. Banâtre, P. Fradet, D. Le Métayer: Gamma and the Chemical Reaction Model: Fifteen Years After. In [21], 17–44.

[11] J. Bartosik: Paun's Systems in Modeling of Human Resource Management. *Proc. Second Conf. Tools and Methods of Data Transformation*, WSU Kielce, 2004.

[12] J. Bartosik, W. Korczynski: Systemy membranowe jako modele hierarchicznych struktur zarzadzania. *Mat. Pokonferencyjne Ekonomia, Informatyka, Zarzadzanie. Teoria i Praktyka*, Wydzial Zarzadzania AGH, Tom II, AGH 2002.

[13] G. Bel Enguix, M.D. Jiménez-Lopez: Linguistic Membrane Systems and Applications. In [30].

[14] F. Bernardini, M. Gheorghe: Population P Systems. *Journal of Universal Computer Science*, 10, 5 (2004), 509–539.

[15] F. Bernardini, V. Manca: Dynamical Aspects of P Systems. *BioSystems*, 70, 2 (2003), 85–93.

[16] G. Berry, G. Boudol: The Chemical Abstract Machine. *Theoretical Computer Science*, 96 (1992), 217–248.

[17] D. Besozzi: *Computational and Modelling Power of P Systems.* PhD Thesis, Univ. degli Studi di Milano, 2004.

[18] D. Besozzi, C. Zandron, G. Mauri, N. Sabadini: P Systems with Gemmation of Mobile Membranes. *Proc. ICTCS 2001*, Torino, *LNCS* 2202 (A. Restivo, S.R. Della Rocca, L. Roversi, eds.), Springer-Verlag, Berlin, 2001, 136–153.

[19] C. Bonanno, V. Manca: Discrete Dynamics in Biological Models. *Romanian Journal of Information Science and Technology*, 5, 1-2 (2002), 45–67.

[20] C. Calude, Gh. Păun: Bio-Steps Beyond Turing. *BioSystems*, 2004.

[21] C.S. Calude, Gh. Păun, G. Rozenberg, A. Salomaa, eds.: *Multiset Processing. Mathematical, Computer Science, and Molecular Computing Points of View. Lecture Notes in Computer Science*, 2235, Springer, Berlin, 2001.

[22] L. Cardelli: Brane Calculus. *Proc. Computational Methods in Systems Biology '04*, Springer, to appear.

[23] M. Cavaliere: Evolution-Communication P Systems. In [86], 134–145.

[24] M. Cavaliere, P. Leupold: Evolution and Observation – A New Way to Look at Membrane Systems. In [70], 70–87.

[25] M. Cavaliere, P. Leupold: Evolution and Observation. A Non-Standard Way to Generate Formal Languages. *Theoretical Computer Science*, 321, 2-3 (2004), 233–248.

[26] M. Cavaliere, C. Martin-Vide, Gh. Păun, eds.: *Proceedings of the Brainstorming Week on Membrane Computing, Tarragona, February 2003.* Technical Report 26/03, Rovira i Virgili University, Tarragona, 2003.

[27] M. Cavaliere, D. Sburlan: Time-Independent P Systems. In [72], 239–258.

[28] M. Cavaliere, D. Sburlan: Time and Synchronization in Membrane Systems. *Fundamenta Informaticae*, 64 (2005), 65–77.

[29] R. Ceterchi, R. Gramatovici, N. Jonoska, K.G. Subramanian: Generating Picture Languages with P Systems. In [26], 85–100.

[30] G. Ciobanu, Gh. Păun, M.J. Pérez–Jiménez, eds.: *Applications of Membrane Computing.* Springer, Berlin, 2005.

[31] G. Ciobanu, Gh. Păun, Gh. Ştefănescu: Sevilla Carpets Associated with P Systems. In [26], 135–140.

[32] G. Ciobanu, G. Wenyuan. A P System Running on a Cluster of Computers. In [70], 123–139.

[33] L. Colson, N. Jonoska, M. Margenstern: λP Systems and Typed λ-Calculus. In [72], 1–18.

[34] A. Cordón-Franco, F. Sancho-Caparrini: Approximating Non-Discrete P Systems. In [72], 288–296.

[35] S. Crespi–Reghizzi, D. Mandrioli: Commutative Grammars. *Calcolo*, 13, 2 (1976), 173–189.

[36] E. Csuhaj-Varjú, J. Kelemen, A. Kelemenová, Gh. Păun, G. Vaszil: Cells in Environment: P Colonies. Submitted, 2004.

[37] E. Csuhaj-Varju, G. Vaszil: P Automata or Purely Communicating Accepting P Systems. In [86], 219–233.

[38] R. Freund, L. Kari, M. Oswald, P. Sosik: Computationally Universal P Systems Without Priorities: Two Catalysts Suffice. *Theoretical Computer Science*, 2004.

[39] R. Freund, M. Oswald: A Short Note on Analysing P Systems. *Bulletin of the EATCS*, 78 (2003), 231–236.

[40] R. Freund, A. Păun: Membrane Systems with Symport/Antiport Rules: Universality Results. In [86], 270–287.

[41] R. Freund, Gh. Păun, M.J. Pérez-Jiménez: Tissue-Like P Systems with Channel-States. *Brainstorming Week on Membrane Computing*, Sevilla, February 2004, TR 01/04 of Research Group on Natural Computing, Sevilla University, 2004, 206–223, and *Theoretical Computer Science*, 2004, in press.

[42] P. Frisco: *Theory of Molecular Computing. Splicing and Membrane Systems.* PhD Thesis, Leiden University, The Netherlands, 2004.

[43] P. Frisco, S. Ji: Towards a Hierarchy of Info-Energy P Systems. In [86], 302–318.

[44] A. Georgiou, M. Gheorghe, F. Bernardini: Generative Devices Used in Graphics. In [30].

[45] R. Gramatovici, G. Bel Enguix: Parsing with P Automata. In [30].

[46] S.A. Greibach: Remarks on Blind and Partially Blind One-Way Multicounter Machines. *Theoretical Computer Science*, 7 (1978), 311–324.

[47] M.A. Gutiérrez-Naranjo, M.J. Pérez-Jiménez, A. Riscos-Núñez: On Descriptive Complexity of P Systems. In [72], 321–331.

[48] M.A. Gutiérrez-Naranjo, M.J. Pérez-Jiménez, A. Riscos-Núñez. Available Membrane Computing Software. In [30].

[49] T. Head: Formal Language Theory and DNA: An Analysis of the Generative Capacity of Specific Recombinant Behaviors. *Bulletin of Mathematical Biology*, 49 (1987), 737–759.

[50] J. Hartmanis: About the Nature of Computer Science. *Bulletin of the EATCS*, 53 (June 1994), 170–190.

[51] J. Hartmanis: On the Weight of Computation. *Bulletin of the EATCS*, 55 (Febr. 1995), 136–138.

[52] J. Hoffmeyer: Surfaces Inside Surfaces. On the Origin of Agency and Life. *Cybernetics and Human Knowing*, 5, 1 (1998), 33–42.

[53] M. Holcombe: Computational Models of Cells and Tissues: Machines, Agents and Fungal Infection. *Briefings in Bioinformatics*, 2, 3 (2001), 271–278.

[54] O.H. Ibarra: The Number of Membranes Matters. In [70], 218–231.

[55] O.H. Ibarra: On Membrane Hierarchy in P Systems. *Theoretical Computer Science*, 2004.

[56] O.H. Ibarra: On Determinism Versus Nondeterminism in P Systems. Submitted, 2004.

[57] M. Ionescu, T.-O. Ishdorj: Replicative–Distribution Rules in P Systems with Active Membranes. *Proc. of ICTAC2004, First Intern. Colloq. on Theoretical Aspects of Computing*, Guiyang, China, 2004.

[58] H. Kitano: Computational Systems Biology. *Nature*, 420, 14 (2002), 206–210.

[59] S.N. Krishna, Gh. Păun, P Systems with Mobile Membranes. *Theoretical Computer Science*, 2005.

[60] S.N. Krishna, R. Rama: P Systems with Replicated Rewriting. *Journal of Automata, Languages and Combinatorics*, 6, 3 (2001), 345–350.

[61] S.N. Krishna, R. Rama, H. Ramesh: Further Results on Contextual and Rewriting P Systems. *Fundamenta Informaticae*, 64 (2005), 235–246.

[62] A. Leporati, C. Zandron, G. Mauri. Simulating the Fredkin Gate with Energy-Based P systems. *Journal of Universal Computer Science*, 10, 5 (2004), 600–619.

[63] A. Leporati, C. Zandron, G. Mauri: Universal Families of Reversible P Systems. *Proc. Conf. Universal Machines and Computations 2004*, Sankt Petersburg, 2004.

[64] W.R. Loewenstein: *The Touchstone of Life. Molecular Information, Cell Communication, and the Foundations of Life.* Oxford University Press, New York, Oxford, 1999.

[65] M. Madhu, K. Krithivasan: Improved Results About the Universality of P Systems. *Bulletin of the EATCS*, 76 (Febr. 2002), 162–168.

[66] V. Manca: String Rewriting and Metabolism. A Logical Perspective. In Gh. Păun, ed.: *Computing with Bio-Molecules. Theory and Experiments*, Springer, Singapore, 1998, 36–60.

[67] V. Manca, L. Bianco, F. Fontana: Evolution and Oscillation in P Systems: Applications to Biological Phenomena. In [72], 63–84.

[68] V. Manca, G. Franco, G. Scollo: State Transition Dynamics. Basic Concepts and Molecular Computing Perspectives. In M. Gheorghe. ed.: *Molecular Computational Models. Unconventional Approaches*, Idea Group, London, 2004.

[69] S. Marcus: Bridging P Systems and Genomics: A Preliminary Approach. In [86], 371–376.

[70] C. Martín-Vide, G. Mauri, Gh. Păun, G. Rozenberg, A. Salomaa, eds.: *Membrane Computing. International Workshop, WMC2003, Tarragona, Spain, Revised Papers. Lecture Notes in Computer Science*, 2933, Springer, Berlin, 2004.

[71] C. Martín-Vide, Gh. Păun, J. Pazos, A. Rodríguez-Patón: Tissue P Systems. *Theoretical Computer Science*, 296, 2 (2003), 295–326.

[72] G. Mauri, Gh. Păun, M.J. Pérez-Jiménez, G. Rozenberg, A. Salomaa, eds.: *Membrane Computing. International Workshop WMC5, Milan, Italy, 2004. Revised Papers, Lecture Notes in Computer Science*, 3365, Springer, Berlin, 2005.

[73] M. Minsky: *Computation – Finite and Infinite Machines.* Prentice Hall, Englewood Cliffs, NJ, 1967.

[74] T.Y. Nishida: Simulations of Photosynthesis by a K-subset Transforming System with Membranes. *Fundamenta Informaticae*, 49, 1-3 (2002), 249–259.

[75] A. Păun, Gh. Păun: The Power of Communication: P Systems with Symport/Antiport. *New Generation Computing*, 20, 3 (2002), 295–306.

[76] Gh. Păun: *Marcus Contextual Grammars.* Kluwer, Dordrecht, 1997.

[77] 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`).

[78] Gh. Păun: Computing with Membranes – A Variant. *International Journal of Foundations of Computer Science*, 11, 1 (2000), 167–182.

[79] Gh. Păun: Computing with Membranes: Attacking NP-Complete Problems. In I. Antoniou, C.S. Calude, M.J. Dinneen, eds.: *Unconventional Models of Computation*, Springer, London, 2000, 94–115.

[80] Gh. Păun: From Cells to Computers: Computing with Membranes (P Systems). *BioSystems*, 59, 3 (2001), 139–158.

[81] Gh. Păun: P Systems with Active Membranes: Attacking NP-Complete Problems. *Journal of Automata, Languages and Combinatorics*, 6, 1 (2001), 75–90.

[82] Gh. Păun: *Computing with Membranes: An Introduction.* Springer, Berlin, 2002.

[83] Gh. Păun: Grammar Systems vs. Membrane Computing: A Preliminary Approach. *Workshop on Grammar Systems*, MTA SZTAKI, Budapest, 2004, 225–245.

[84] Gh. Păun, J. Pazos, M.J. Pérez-Jiménez, A. Rodríguez-Patón: Symport/Antiport P Systems with Three Objects Are Universal. *Fundamenta Informaticae*, 64 (2005), 345–358.

[85] Gh. Păun, G. Rozenberg, A. Salomaa: *DNA Computing. New Computing Paradigms*. Springer, Berlin, 1998.

[86] Gh. Păun, G. Rozenberg, A. Salomaa, C. Zandron, eds.: *Membrane Computing. International Workshop, WMC-CdeA 2002, Curtea de Argeş, Romania, Revised Papers. Lecture Notes in Computer Science*, 2597, Springer, Berlin, 2003.

[87] M. Pérez-Jiménez, A. Romero-Jiménez, F. Sancho-Caparrini: *Teoría de la Complejidad en Modelos de Computatión Celular con Membranas*. Editorial Kronos, Sevilla, 2002.

[88] B. Petreska, C. Teuscher: A Hardware Membrane System. In [70], 269–285.

[89] A. Regev, E.M. Panina, W. Silverman, L. Cardelli, E. Shapiro: BioAmbients – An Abstraction for Biological Compartments. *Theoretical Computer Science*, 325 (2004), 141–167.

[90] A. Riscos–Núñez: *Programacion celular. Resolucion eficiente de problemas numericos NP-complete*. PhD Thesis, Univ. Sevilla, 2004.

[91] P. Sosik: The Computational Power of Cell Division in P Systems: Beating Down Parallel Computers? *Natural Computing*, 2, 3 (2003), 287–298.

[92] P. Sosik, J. Matysek: Membrane Computing: When Communication Is Enough. In C.S. Calude, M.J. Dinneen, F. Peper, eds., *Unconventional Models of Computation 2002, Lecture Notes in Computer Science*, 2509, Springer, Berlin, 2002, 264–275.

[93] Y. Suzuki, Y. Fujiwara, H. Tanaka, J. Takabayashi: Artificial Life Applications of a Class of P Systems: Abstract Rewriting Systems on Multisets. In [21], 299–346.

[94] Y. Suzuki, H. Tanaka: Chemical Oscillation in Symbolic Chemical Systems and Its Behavioral Pattern. In Y. Bar-Ylam, ed.: *Proc. Intern. Conference on Complex Systems*, New England Complex Systems Institute, 1997, 1–7.

[95] Y. Suzuki, H. Tanaka: Abstract Rewriting Systems on Multisets, and Its Application for Modelling Complex Behaviours. In [26], 313–331.

[96] A. Syropoulos, P.C. Allilomes, E.G. Mamatas, K.T. Sotiriades: A Distributed Simulation of P Systems. In [70], 355–366.

[97] C. Teuscher: *Alan Turing. Life and Legacy of a Great Thinker.* Springer, Berlin, 2003.

[98] M. Tomita: Whole-Cell Simulation: A Grand Challenge of the 21st Century. *Trends in Biotechnology*, 19 (2001), 205–210.

[99] G. Vaszil: On the Size of P Systems with Minimal Symport/Antiport. *Pre-Proceedings of Workshop on Membrane Computing, WMC5, Milano, Italy*, June 2004, 422–431.

[100] C. Zandron: *A Model for Molecular Computing: Membrane Systems.* PhD Thesis, Univ. degli Studi di Milano, 2001.

[101] C. Zandron, C. Ferretti, G. Mauri: Solving NP-Complete Problems Using P Systems with Active Membranes. In I. Antoniou, C.S. Calude, M.J. Dinneen, eds.: *Unconventional Models of Computation*, Springer, London, 2000, 289–301.

[102] The Web Page of Membrane Computing:
`http://psystems.disco.unimib.it`

An Approach to Computational Complexity in Membrane Computing

Mario J. Pérez-Jiménez

Research Group on Natural Computing
Department of Computer Science and Artificial Intelligence
University of Sevilla
Avda. Reina Mercedes s/n, 41012 Sevilla, Spain
Mario.Perez@cs.us.es

1 Introduction

The necessity to define in a satisfactory way what means a *definite method* for solving mathematical problems was studied by A. Turing who investigated how such methods should be applied *mechanically*, and, moreover, he formalized the task of performing such methods in terms of the operations of a *machine* able to read and write symbols on a tape divided into parts called cells (simulating how a person can solve a problem with paper and pencil manipulating symbols).

The *theory of computation* deals with the *mechanical solvability* of problems, that is, searching solutions that can be described by a finite sequence of elementary processes or instructions. The first goal in the theory of computation is general problem solving; that is, to develop principles and special methods that are able to solve any problem from a certain class of questions.

A *computational model* tries to capture those aspects of mechanical solutions of problems that are relevant to these solutions, including their inherent limitations. In some sense, we can think that computational models design machines according to certain necessity.

From a practical point of view, the goal of computation theory is to take real–life problems and try to solve them through a method capable of being simulated by a machine when we use a suitable language to communicate that problem to the machine (a language is a system of signs used to communicate information between different parties).

Abstract machines are formal computing devices that we use to investigate properties of real computing devices. Computable languages are a special type of formal languages that can be processed by abstract machines that represent computers.

If we have a mechanically solvable problem and we have a specific algorithm solving it that can be implemented in a real machine, then it is very important to know how much computational resources (time or memory) are required for a given instance, in order to recognize the limitations of the real device.

One of the main goals of the *theory of computational complexity* is the study of the efficiency of algorithms and their data structures through the analysis of the resources required for solving problems (that is, according to their intrinsic computational difficulty). This theory provides a classification of the *abstract problems* that allows us to detect their inherent complexity from the computational solutions point of view.

Of course, such a classification demands a precise and formal definition of the concept of *abstract problem* and the model to be considered.

The following parameters are used to specify a *complexity class* within a general computational framework:

- First: the *model* of computation, D (in our case, recognizer P systems).
- Second: the *mode* of computation, M (in our case, non-deterministic and parallel).
- Third: the resource, r, that we wish to bound (usually time and space).
- Finally, we must specify an upper *bound* of the resources, f (a total recursive function from natural numbers to natural numbers).

Then, a complexity class is defined as the set of all languages decided by the device D operating in mode M and such that for any input string, u, D expends at most $f(|u|)$ units of the resource r, to accept or reject u.

Many interesting problems of the real world are presumably intractable and hence it is not possible to execute algorithmic solutions in an electronic computer when we deal with instances of those problems whose size is large. The theoretical limitations of the Turing machines in terms of computational power are also practical limitations to the digital computers.

Natural Computing is a new computing area inspired by nature, using concepts, principles and mechanisms underlying natural systems. *Evolutionary Algorithms* use different concepts from biology. *Neural Networks* are inspired in the structures of the brain and nervous system. *DNA Computing* is based on the computational properties of DNA molecules and on the capacity to handle them. *Membrane Computing* is inspired by the structure and functioning of living cells.

These two last models of computation provide unconventional devices with an attractive property (*computational efficiency*), they are able to create an exponential workspace in polynomial time (and, in some sense, trading space for time).

Can some unconventional devices be used to solve presumably intractable problems in a feasible time? The answer is affirmative at least from a theoretical point of view.

In this paper we provide a systematic and formal framework for the design of polynomial solutions to hard problems, and to classify them according to their polynomial solvability by cell–like membrane systems. Complexity classes in the framework of membrane computing and their relationships with the problems they contain, are the main subjects of this paper.

The paper is structured as follows. The next section is devoted to describe in an informal way the deterministic and non-deterministic mode of computation in a computing model. In Sections 3, 4, and 5 combinatorial optimization problems and decision problems are introduced, and a relationship between them from a complexity point of view is showed. The **P** versus **NP** problem is presented in Section 6, and in Section 12 a characterization of that problem is obtained. In Section 7 weakly and strongly **NP**–complete problems are studied. Sections 8 and 9 are devoted to present the general framework (recognizer membrane systems) within a theory of computational complexity developed here. Deterministic and non-deterministic polynomial complexity classes in membrane systems are introduced in Sections 10 and 11. Finally, we study the P systems with the capability to construct an exponential workspace in polynomial time, and the polynomial complexity classes associated with them.

2 Determinism versus Non-Determinism

A model of computation is properly given when we formally define the concept of mechanical procedure (*algorithm*). For that, it is necessary to syntactically define it, and determine precisely how such procedures can be executed (the semantic of the model).

The devices (systems or machines) modelling mechanical procedures can be represented through *configurations* (containing a complete description of the current state of the device). These configurations can evolve according to the semantic of the model. Formally, the semantic defines the concept of *transitions* from a configuration of the system to a next configuration; that is, the semantic of the model specifies what means *next configuration* of a given configuration. A configuration which has no next configuration is called a *halting configuration.*

A *computation* or execution of a device of a model is a sequence (finite or infinite) of configurations such that each configuration (except the first) is obtained from the previous one by a (step of) transition of the system. That is, a computation starts with an *initial configuration* of the system, and then it proceeds step by step, and halts when reaches a halting configuration (and then the result is encoded in this configuration).

When we use the devices of a model of computation to solve certain kind of problems on strings (in particular to *recognize* a language), it is necessary to define what means to *accept* or *reject* a string. In this case it is possible to consider two *modes of computation* in a computing model.

- The *deterministic mode* is characterized by the following fact: each configuration has *at most* one next configuration. In a deterministic device, given a current configuration, the next configuration of the system is uniquely determined, if any.

- The *non-deterministic mode* verifies the following property: each non halting configuration hast *at least* a next configuration. In a non-deterministic device several next configurations can be reached from a current configuration.

The computation of deterministic devices can be viewed as a tree with only one branch, whereas the computation of a non-deterministic device can be viewed as a tree having many possible branches. The root of the tree corresponds to the beginning of the computation, and every node in the tree corresponds to a point of the computation at which the machine has eventually multiple choices. Each branch of this tree determines one computation of the system.

Next we define what means to accept or reject a string by a deterministic or non-deterministic device (whose answers are only *yes* or *no*).

- A deterministic device M *accepts* (respectively, *rejects*) a string a if the answer of M on input a is *yes* (respectively, *no*).

- A non-deterministic device M *accepts* a string a if there exists a computation of M with input a such that the answer is *yes*.

Let us note the difference between the definition of acceptance by deterministic and non-deterministic devices. An input string a is accepted by a deterministic machine M, if *the* computation of M on input a halts and answers *yes*. A non-deterministic machine M accepts a string a if there exists *some* computation of M on input a answering *yes*; that is, there exists a sequence of non-deterministic choices that answers *yes*. In this case, it is possible that we accept a string but that there exists another computation with the same input that either halts and answers *no*, or does not halt.

Thus, a deterministic device can (mechanically) reject a string, but this is not the case in non-deterministic machines. How can we decide (in a mechanical way) whether there exists a non halting computation?

Non-deterministic Turing machines are like existential quantifiers: they accept an input string if there exists an accepting path in the corresponding computation tree. In some sense, we can affirm that non-deterministic devices do not properly capture the intuitive idea underlying the concept of algorithm, because the result of such a machine on an input (that is, the output of a computation) is not reliable, since the answer of the device is not always the same.

Non-determinism can be considered as a generalization of determinism (the computation may branch at each configuration), and it can be viewed as a kind of parallel computation where several "processes" can be run simultaneously.

3 Combinatorial Optimization Problems

Roughly speaking, when we deal with *combinatorial optimization problems* we wish to find the *best* solution (according to a given criterion) among a class of possible (candidate or feasible) solutions. That is, in this kind of problems there can be many possible solutions, each one has associated a value (a positive rational number), and we aim to find a solution with the optimal (minimum or maximum) value.

For example, a *vertex cover* of an undirected graph is a set of vertices such that any edge of the graph has, at least, an endpoint in that set. Then, we may want to find one of the smallest vertex covers among all possible vertex covers in the input graph. This is the combinatorial optimization problem called *Minimum Vertex Cover Problem.* The main ingredients in this problem are the following: (a) the collection of all undirected graphs, (b) the finite set of all vertex covers associated with a given undirected graph, and (c) the cardinality of each vertex cover of a given undirected graph.

We can formalize these ideas in the following definition.

Definition 1 *A combinatorial optimization problem, X, is a tuple (I_X, s_X, f_X) where:*

- *I_X is a language over a finite alphabet.*
- *s_X is a function whose domain is I_X and, for each $a \in I_X$, the set $s_X(a)$ is finite.*
- *f_X is a function (the objective function) that assigns to each instance $a \in I_X$ and each $c_a \in s_X(a)$ a positive rational number $f_X(a, c_a)$.*

The elements of I_X are called *instances* of the problem X. For each instance $a \in I_X$, the elements of the finite set $s_X(a)$ are called *candidate* (or *feasible*) *solutions* associated with the instance a of the problem. For each instance $a \in I_X$ and each $c_a \in s_X(a)$, the positive rational number $f_X(a, c_a)$ is called *solution value* for c_a. The function f_X provides the criterion to determine the *best* solution.

For example, the *Minimum Vertex Cover* problem is a combinatorial optimization problem such that I_X is the set of all undirected graphs, and for each undirected graph G, $s_X(G)$ is the set of all vertex covers of G; that is, each vertex cover of the graph is a candidate solution for the problem. The objective function f_X is defined as follows: for each undirected graph G and each vertex cover C of G, $f_X(G, C)$ is the cardinality of C.

Definition 2 *Let $X = (I_X, s_X, f_X)$ be a combinatorial optimization problem. An optimal solution for an instance $a \in I_X$ is a candidate solution $c \in s_X(a)$ associated with this instance such that,*

- *either for all $c' \in s_X(a)$ we have $f_X(a, c) \leqslant f_X(a, c')$ (and then we say that c is a minimal solution for a),*
- *either for all $c' \in s_X(a)$ we have $f_X(a, c) \geqslant f_X(a, c')$ (and then we say that c is a maximal solution for a).*

A *minimization* (respectively, *maximization*) *problem* is a combinatorial optimization problem such that each optimal solution is a minimal (respectively, maximal) solution.

That is, an optimization problem seeks the best of all possible candidate solutions, according to a simple cost criterion given by the objective function. For example, the *Minimum Vertex Cover* problem is a minimization problem because a minimal solution associated with an undirected graph G, provides one of the smallest vertex covers of G.

An *approximation computational device*, $\mathcal{D}$, for a combinatorial optimization problem, X, provides a candidate solution $c \in s_X(a)$ for each instance $a \in I_X$. If the provided solution is always optimal, then $\mathcal{D}$ is called an *optimization computational device* for X.

For instance, an approximation machine for the *Minimum Vertex Cover* problem needs only find some vertex cover associated with each undirected graph, whereas an optimization machine must always find a vertex cover with the least cardinality associated with each undirected graph.

Having in mind that until now polynomial time optimization algorithms have not be found for many presumably intractable problems (it is believed that this kind of solutions can never be found), it is convenient to find an approximation algorithm running in polynomial time and such that, for all problem instances the candidate solution given by the algorithm is *close*, in a sense, to an optimal solution.

4 Decision Problems

An important class of combinatorial optimization problems is the class of decision problems, that is, problems that require either an *yes* or a *no* answer.

Definition 3 *A decision problem, X, is a pair (I_X, θ_X) such that I_X is a language over a finite alphabet (whose elements are called instances) and θ_X is a total boolean function (that is, a predicate) over I_X.*

Therefore, a decision problem $X = (I_X, \theta_X)$ can be viewed as a combinatorial optimization problem $X = (I_X, s_X, f_X)$ where for each instance $a \in I_X$ we have the following:

- $s_X(a) = \{\theta_X(a)\}$ (the only possible candidate solution associated with instance a is 0 or 1, depending on the answer of the problem to a).
- $f_X(a, \theta_X(a)) = 1$.

Thus, each decision problem can be considered as a minimization (or maximization) problem.

There exists a natural correspondence between languages and decision problems in the following way. Each language L, over an alphabet Σ, has a decision problem, X_L, associated with it as follows: $I_{X_L} = \Sigma^*$, and $\theta_{X_L} = \{(x,1) \mid x \in L\} \cup \{(x,0) \mid x \notin L\}$; reciprocally, given a decision problem $X = (I_X, \theta_X)$, the language L_X over the alphabet of I_X corresponding to it is defined as follows: $L_X = \{a \in I_X \mid \theta_X(a) = 1\}$.

Usually, NP-completeness has been studied in the framework of decision problems. Many abstract problems are not decision problems, but combinatorial optimization problems, in which some value must be optimized (minimized or maximized). In order to apply the theory of NP-completeness to combinatorial optimization problems, we must consider them as decision problems.

We can transform any combinatorial optimization problem into a roughly equivalent decision problem by supplying a target/threshold value for the quantity to be optimized, and asking the question whether this value can be attained. Next we give two examples.

1. The *Minimum Vertex Cover Problem.*

   Optimization version: Given an undirected graph G, find the cardinality of a *minimal* set of a vertex cover of G.

   Decision version: Given an undirected graph G, and *a positive integer* k, determine whether or not G has a vertex cover of size *at most* k.

2. The *Common Algorithmic Problem* [9].

   Optimization version: given a finite set S and a family F of subsets of S, find the cardinality of a *maximal* subset of S which does not include any set belonging to F.

   Decision version: given a finite set S, a family F of subsets of S, and a positive integer k, we are asked whether there is a subset A of S such that the cardinality of A is *at least* k, and which does not include any set belonging to F.

If a combinatorial optimization problem can be *quickly* solved, then its decision version can be quickly solved as well (because we only need to compare the solution value with a threshold value). Similarly, if we can make clear that a decision problem is hard, we also make clear that its associated combinatorial optimization problem is hard.

For example, let A be a polynomial time algorithm for the optimization version of the Minimum Vertex Cover problem. Then we consider the following polynomial time algorithm for the decision version of the Minimum Vertex Cover problem: given an undirected graph G, and a positive integer k, if $k < A(G)$ (here $A(G)$ is the cardinality of a smallest vertex cover of G), then answer *no*; otherwise, the answer is *yes*.

Reciprocally, let B be a polynomial time algorithm for the decision version of the Minimum Vertex Cover problem. Then we consider the following polynomial time algorithm for the optimization version of the Minimum Vertex Cover problem: given an undirected graph G, repeatedly while $k \leqslant$ number of vertices of G (starting from $k = 0$, and in the next step considering $k + 1$) we execute the algorithm A on input (G, k), until we reach a first *yes* answer, and then the result is k.

5 Solving Decision Problems

Recall that, in a natural way, each decision problems has associated a language over a finite alphabet. Next, we define the solvability of decision problems through the recognition of languages associated with them.

In order to specify the concept of solvability we work with an universal computing model: Turing machines.

Let M be a Turing machine such that the result of any halting computation is *yes* or *no*. Let L be a language over an alphabet Σ.

If M is a *deterministic* device (with Σ as working alphabet), then we say that M *recognizes* or *decides* L whenever, for any string a over Σ, if $a \in L$, then the answer of M on input a is *yes* (that is, M accepts a), and the answer is *no* otherwise (that is, M reject a).

If M is a *non-deterministic* Turing machine, then we say that M *recognizes* or *decides* L if the following is true: for any string a over Σ, $a \in L$ if and only if there exists a computation of M with input a such that the answer is *yes*. That is, an input string a is accepted by M if there is *an* accepting computation of M on input a. But now we do not have a mechanical criterion to reject an input string.

Recall that any deterministic Turing machine with multiple tapes can be simulated by a deterministic Turing machine with one tape with a polynomial loss of efficiency, whereas the simulation of non-determinism through determinism involves an exponential loss of efficiency.

In the context of computation theory, we consider a problem X to be solved when we have a *general* (definite) *method* (described in a model of computation) that works for any instance of the problem. From a practical point of view, such methods only run over a finite set of instances whose sizes depend on the available resources.

We say that a Turing machine M solves a decision problem X if M *recognizes* the language associated with X; that is, for any instance a of the problem: (1) in the deterministic case, the machine (with input a) output *yes* if the answer of the problem is *yes*, and the output is *no* otherwise; (2) in the non-deterministic case, some computation of the machine (with input a) output *yes* if the answer of the problem is *yes*.

Due to the fact that we represent the instances of abstract problems as strings we can consider their size in a natural manner: the size of an instance is the length of the string. Then, how do the resources required to execute a method increase according to the size of the instance? This is a relevant question in computational complexity theory.

6 The P versus NP Problem

In order to solve an abstract problem by a computational device, problem instances must be represented (encoded) in an adequate way that the device understands.

Given a problem it is possible to use different *reasonable* encoding schemes to represent the instances (we do not attempt to define *reasonable*, however informally we say that *reasonable* means [7] to codify instances in a concise manner, without irrelevant information, and the numbers occurring in them should be represented in binary, or any fixed base other than 1). It is easy to prove that the input sizes that different reasonable encoding schemes determine differ, at most, polynomially from one another.

Recall that complexity classes provide a way to group decision problems of similar computational complexity.

P is the class of all decision problems solvable (or languages recognized)

by some deterministic Turing machine in a time bounded by a polynomial on the size of the input. Having in mind that all *reasonable* deterministic computational models are polynomially equivalent (that is, any one of them can simulate another with only a polynomial loss of efficiency), this class is the same for all models of computation that are polynomially equivalent to the deterministic Turing machine with one tape. Moreover, informally speaking, **P** corresponds to the class of problems having a *feasible* algorithm that gives an answer in a *reasonable* time; that is, problems that are realistically solvable on a machine (even for large instances of the problem).

NP is the class of all decision problems solvable in a polynomial time by non-deterministic Turing machines; that is, for every accepted input there exists at least one accepting computation taking an amount of steps bounded by a polynomial on the length of the input. This class is invariant for all reasonable non-deterministic computational models because all of them are polynomially equivalent.

Every deterministic Turing machine can be considered as a non-deterministic one, so we have $\mathbf{P} \subseteq \mathbf{NP}$. In terms of the previously defined classes, the **P** *versus* **NP** problem can be expressed as follows: is it verified the relation $\mathbf{NP} \subseteq \mathbf{P}$? That is, the **P** versus **NP** problem is the problem of determining whether every language recognized by some non-deterministic Turing machine in polynomial time is also can be recognized by some deterministic Turing machine in polynomial time.

The $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$ question is one of the outstanding open problems in theoretical computer science. The relevance of this question is not only the inherent pleasure of solving a mathematical problem, but in this case an answer to it would provide information of high economical interest. On the one hand, a negative answer to this question would confirm that the majority of current cryptographic systems are secure from a practical point of view. On the other hand, a positive answer would not only show the uncertainty about the secureness of these systems, but also this kind of answer is expected to come together with a general procedure provides a deterministic algorithm solving most of **NP**-complete problem in polynomial time (furthermore, mathematics would be *transformed* because real computers will be able to find a formal proof of any theorem which has a proof of reasonable length).

In the last years several computing models using powerful tools from nature have been developed (because of this, they are known as *bio-inspired* models) and several solutions in polynomial time to problems from the class **NP** have been presented, making use of non-determinism and/or of an exponential amount of space. This is the reason why a practical implementation of such models (in biological, electronic, or other mediums) could provide a significant advance in the resolution of **NP**-complete problems.

7 Strongly NP–Complete Problems

The *Subset Sum* problem is the following: given a finite set A, a weight function, $w : A \to \mathbf{N}$, and a constant $k \in \mathbf{N}$, determine whether or not there exists a subset $B \subseteq A$ such that $w(B) = k$.

It is well known that *Subset Sum* can be solved in time $O(n \cdot k)$, using a dynamic programming algorithm. Hence, that algorithm is polynomial in the number of input items n and the magnitude of the items k. But such a algorithm is not a polynomial algorithm because its time bound is not a polynomial function on the size of the input (that is, of the order $\Omega(n \cdot \log k)$). Then we say that such a algorithm is *pseudo-polynomial*, and that the problem can be solved in *pseudo-polynomial time*. Nevertheless if we represent the input in *unary* form then that algorithm becomes a polynomial algorithm.

Definition 4 *An algorithm that solves a problem X will be called a pseudopolynomial time algorithm for X if its running time would be polynomial if all input numbers associated with each instance were expressed in unary notation.*

The *Knapsack* and *Partition* problems are also **NP**–complete problems that can be solved by a pseudo-polynomial time algorithm.

Often, problems which can be solved in pseudo-polynomial time are also called *weakly* **NP**–*complete* problems. The existence of a pseudo-polynomial time algorithm for a given **NP**–complete problem illustrate that the problem is not so *intractable* after all.

Thus it is important to determine whether a problem is weakly **NP**–complete, or whether it has the following stronger property.

Definition 5 *A problem is said to be* **NP**–*complete in the strong sense if the variant of it in which any instance of size n is restricted to contain integers of size at most $p(n)$, where p is a polynomial, remains* **NP**–*complete.*

That is, the strongly **NP**–complete problems remains **NP**–complete even if all numbers in the input are written in unary notation.

For example, the decision version of the *Minimum Vertex Cover* problem is a strongly **NP**–complete problem since the numbers in the input (an undirected graph) are bounded by a polynomial in the number of vertices (input size).

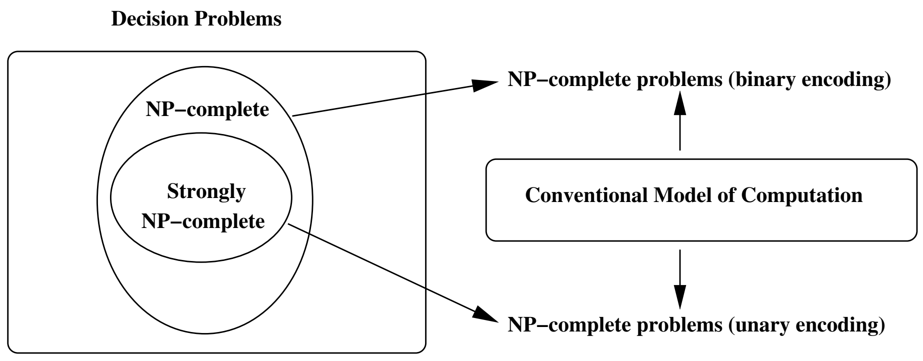

Decision Problems
NP–complete
Strongly NP–complete
NP–complete problems (binary encoding)
Conventional Model of Computation
NP–complete problems (unary encoding)

Figure 1: NP–Completeness and codification of instances

Other strongly **NP**–complete problems are the following: *3–Partition*, *Sat*, *Clique*, *HPP* (Hamiltonian Path Problem), *TSP* (Travelling Salesman Problem), and *Bin Packing*.

What happens if a strongly **NP**–complete problem can be solved by a pseudo-polynomial time algorithm? Let X be such a problem. Then the variant Y of it in which all input numbers of X are written in unary notation is also **NP**–complete. Moreover, if A is a pseudo-polynomial time algorithm solving X, then it is also a polynomial time algorithm that solves Y. Hence, **P**=**NP**.

Theorem 1 *The following propositions are equivalent:*

1. $\mathbf{P} = \mathbf{NP}$.
2. *Every strongly* **NP**–*complete problem can be solved by a pseudo-polynomial time algorithm.*
3. *There exists a strongly* **NP**–*complete problem that can be solved by a pseudo-polynomial time algorithm.*

Thus, to prove **P**=**NP** suffices to find a strongly **NP**–complete problem solvable in pseudo-polynomial time. Recall that the concept of solvability above mentioned is formally associated with deterministic Turing machines.

However, P systems take multisets as input and handle them through computations. Hence the inputs in P systems are provided in *unary*, so it is necessary to analyze with more details when it is said that a problem is polynomial-time solvable in the framework of membrane computing (particularly, concerning the size of the problem instances).

In this context we can say that polynomial solutions to **NP**–complete problems in the framework of membrane computing, can be considered, in a sense, as *pseudo-polynomial* solutions in the classical sense.

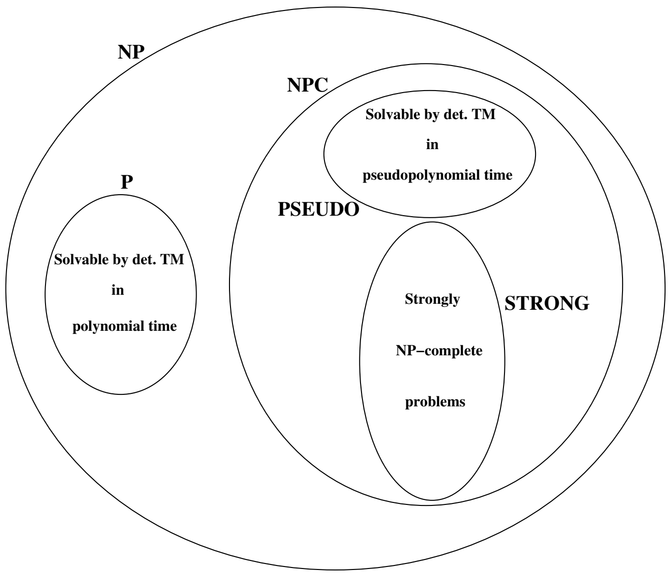

NP
NPC
Solvable by det. TM in pseudopolynomial time
PSEUDO
P
Solvable by det. TM in polynomial time
Strongly NP–complete problems
STRONG

Figure 2: Kinds of NP–complete problems

8 Recognizer Membrane Systems

Membrane computing is a recent branch of natural computing initiated in [22]. It has been developed basically from a theoretical point of view.

Membrane systems – usually called P systems – are distributed parallel computi ng models inspired by the structure and functioning of living cells.

Membrane systems have several *syntactic* ingredients: a *membrane structure* consisting of a hierarchical arrangements of membranes embedded in a *skin* membrane, and delimiting *regions* or compartments where multisets of *objects* and sets (eventually empty) of (evolution) *rules* are placed.

Also, P systems have two main *semantic* ingredients: their inherent *parallelism* and *non-determinism.* The objects inside the membranes can evolve according to given rules in a synchronous (in the sense that a global clock is assumed), parallel, and non-deterministic manner.

Can this parallelism and non-determinism be used to solve hard problems in a feasible time? The answer is affirmative, but we must point out two considerations. On the one hand, we have to deal with the non-determinism in such a way that the solutions obtained from these devices are algorithmic solutions in the classic sense; that is, the answers of the computations of the system must be reliable. On the other hand, the drastic decrease of the execution time from an exponential to a polynomial one is not achieved for free, but by the use of an exponential workspace (in the form of membranes or string–objects), although this space is created in polynomial (often linear) time.

In this paper we use membrane computing as a framework to attack the resolution of decision problems. In order to solve this kind of problems and having in mind the relationship between the solvability of a problem and the recognition of the language associated with it, we consider P systems as *recognizer languages* devices.

Moreover, for technical reasons we only work with devices such that all computations halt, and such that the result (*yes* or *no* answer, because we deal with recognition of strings) is collected in the environment (and in the last step of the computation).

All these restrictions make more difficult the process of designing families of recognizer P systems to solve decision problems.

Definition 6 *A* recognizer P system *is a P system with external output such that:*

1. *The working alphabet contains two distinguished elements* yes *and* no.
2. *All computations halt.*
3. *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 in the last step of the computation.*

In recognizer P systems, we say that a computation $\mathcal{C}$ is an *accepting computation* (respectively, *rejecting computation*) if the object *yes* (respectively, *no*) appears in the environment associated with the corresponding halting configuration of $\mathcal{C}$. Hence, these devices send to the environment an accepting or rejecting answer, in the end of their computations.

If we want these kind of systems to properly solve decision problems and capture the true algorithmic concept, it is necessary to require a condition of *confluence*; that is, the system must always give the same answer. In order to accept or reject a string it should be enough to read the answer of *any* computation of the system. In this manner, an observer outside the system can identify the exact moment when the system halts, and know the answer.

Since P systems work in a non-deterministic manner, we need to adapt the usual definition of acceptance in non-deterministic Turing machines.

9 Soundness and Completeness

In order to assure that a family of recognizer P systems solves a decision problem, two main properties must to be proved: for each instance of th e problem,

(a) if *there exists an* accepting computation of the membrane system processing it, answering *yes*, then the problem also answer *yes* for that instance (*soundness*);

(b) if the problem answers *yes*, then *any* computation of the system processing that instance, answer *yes* (*completeness*).

If we demand that the family of membrane systems is sound and complete, then it satisfies a condition of *confluence*: every computation of a system from the family has the same output.

Next, we formalize these ideas in the following definition.

Definition 7 *Let $X = (I_X, \theta_X)$ be a decision problem. Let $\mathbf{\Pi} = (\Pi(w))_{w \in I_X}$ be a family of recognizer membrane systems* without input.

- *We say that the family $\mathbf{\Pi}$ is sound with regard to X if the following is true: for each instance of the problem $w \in I_X$, if there exists an accepting computation of $\Pi(w)$, then $\theta_X(w) = 1$.*
- *We say that the family $\mathbf{\Pi}$ is complete with regard to X if the following is true: for each instance of the problem $w \in I_X$, if $\theta_X(w) = 1$, then every computation of $\Pi(w)$ is an accepting computation.*

The soundness property means that if we obtain an *acceptance response* of the system (associated with an instance) through some computation, then the answer of the problem (for that instance) is *yes.* The completeness property means that if we obtain an *affirmative* response to the problem, then any computation of the system must be an accepting one.

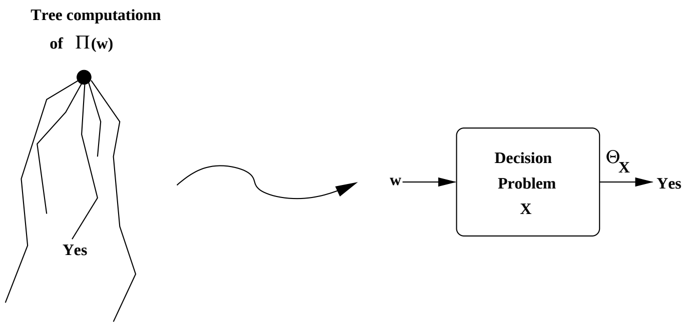


Figure 3: Soundness of a family of P systems without input

These concepts can be extended to families of recognizer P systems *with input membrane* in a natural way, but in this case a P system belonging to the family can process several instances of the problem provided that an appropriate input, depending on the instance, is supplied to the system.

Definition 8 *Let $X = (I_X, \theta_X)$ be a decision problem. Let $\mathbf{\Pi} = (\Pi(n))_{n \in \mathbf{N}}$ be a family of recognizer membrane systems* with input. *A* polynomial encoding *of X in $\mathbf{\Pi}$ is a pair (cod, s) of polynomial time computable functions over I_X such that for each instance $w \in I_X$, $s(w)$ is a natural number and $cod(w)$ is an input multiset of the system $\Pi(s(w))$.*

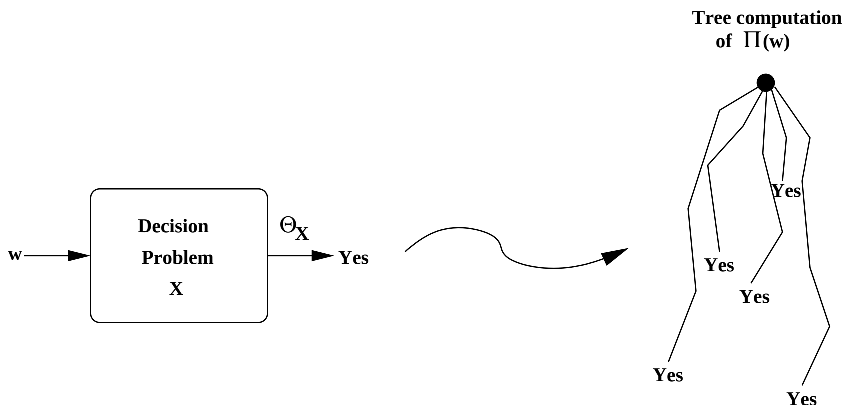


Figure 4: Completeness of a family of P systems without input

Definition 9 *Let $X = (I_X, \theta_X)$ be a decision problem. Let $\mathbf{\Pi} = (\Pi(n))_{n\in\mathbf{N}}$ be a family of recognizer membrane systems* with input. *Let (cod, s) be a polynomial encoding of X in $\mathbf{\Pi}$.*

- *We say that the family $\mathbf{\Pi}$ is sound with regard to (X, cod, s) if the following is true: for each instance of the problem $w \in I_X$, if there exists an accepting computation of $\Pi(s(w))$ with input $cod(w)$, then $\theta_X(w) = 1$.*
- *We say that the family $\mathbf{\Pi}$ is complete with regard to (X, cod, s) if the following is true: for each instance of the problem $u \in I_X$, if $\theta_X(u) = 1$ then every computation of $\Pi(s(u))$ with input $cod(u)$ is an accepting computation.*

The soundness property means that if given an instance we obtain an *acceptance response* of the system associated with it (and individualized by the appropriate input multiset) through some computation, then the answer of the problem (for that instance) is *yes.* The completeness property means that if we obtain an *affirmative* response to the problem, then any computation of the system associated with it (and individualized by the appropriate input multiset) must be an accepting one.

10 Polynomial Complexity Classes in Membrane Systems

Next, we consider different complexity classes in the framework of recognizer membrane systems.

10.1 Recognizer membrane systems without input

The first results about *solvability* of **NP**–complete problems in polynomial time (even linear) by membrane systems were given by Gh. Păun [24], C. Zandron et al. [42], S.N. Krishna et al. [11], and A. Obtułowicz [15] in the framework of P systems that lack an input membrane. Thus, the constructive proofs of such results design *one* system for *each* instance of the problem.

In this context, next we define polynomial complexity classes in recognizer membrane systems without input. In order to solve a decision problem we need, then, to associate with each instance of the problem a system which decides the instance. We impose these systems to be *confluent* in the following sense: an instance of the problem will have a positive answer if and only if *every* (or, equivalently, there exists a) computation of the corresponding system is an accepting computation.

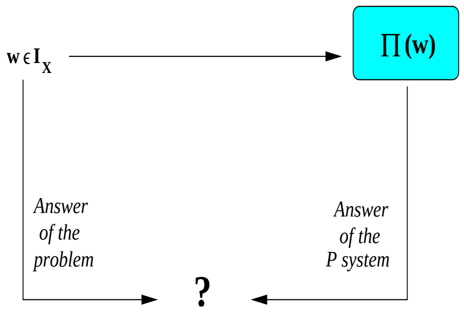


Figure 5: Complexity class for membrane systems without input

We also demand that *every* computation is bounded, in execution time, by a polynomial function. This is because we do not only want to obtain the same answer, independently of the chosen computation, but that all the computations consume, at most, the same amount of resources (in time).

Definition 10 *Let $\mathcal{R}$ be a class of recognizer P systems without input membrane. A decision problem $X = (I_X, \theta_X)$ is solvable in polynomial time by a family $\mathbf{\Pi} = (\Pi(w))_{w \in I_X}$, of P systems of type $\mathcal{R}$, and we denote it by $X \in \mathbf{PMC}^*_{\mathcal{R}}$, if the following is true:*

- *The family $\mathbf{\Pi}$ is polynomially uniform by Turing machines; that is, there exists a deterministic Turing machine working in polynomial time which constructs the system $\Pi(w)$ from the instance $w \in I_X$.*
- *The family $\mathbf{\Pi}$ is polynomially bounded; that is, there exists a polynomial function $p(n)$ such that for each $w \in I_X$, all computations of $\Pi(w)$ halt in at most $p(|w|)$ steps.*
- *The family $\mathbf{\Pi}$ is sound and complete with regard to X.*

Note that in this complexity class we consider two different tasks: the first one is the construction of the family, which we require to be done in polynomial time (sequential time by deterministic Turing machines). The second one is the execution of the systems of the family, in which we imposed that the total number of steps performed by their computations are bounded by the function g (parallel time by non-deterministic membrane systems).

As a direct consequence of working with recognizer membrane systems is the fac t that these complexity classes are closed under complement.

Moreover, the complexity classes are closed under polynomial time reduction, in the classical sense. Recall that if $X = (I_X, \theta_X)$ and $Y = (I_Y, \theta_Y)$ are decision problems, then we say that X is reducible to Y in polynomial time if there exists a polynomial time function f from I_X to I_Y verifying the following condition: for each $w \in I_X$ we have $\theta_X(w) = 1$ if and only if $\theta_Y(f(w) = 1$.

Proposition 2 *Let $\mathcal{R}$ be a class of recognizer P systems without input membrane. Let X and Y be two decision problems such that X is reducible to Y in polynomial time. If $Y \in \mathbf{PMC}^*_{\mathcal{R}}$, then $X \in \mathbf{PMC}^*_{\mathcal{R}}$.*

The *Hamiltonian Path Problem* can be solved in *quadratic time* by a family $\mathcal{R}$ of recognizer P systems without input in an uniform way (see [25]). Then $\mathbf{NP} \subseteq \mathbf{PMC}^*_{\mathcal{R}}$.

10.2 Recognizer membrane systems with input

A computation of a Turing machine starts when the machine is in the initial state and we “write” a string in the input tape of the machine. Then, the machine starts to compute according to the transition function. In the definitions of basic P systems that have been initially considered, there is no membrane in which we can “introduce” input objects before allowing the system to begin to work. However, it is easy to consider input membranes in this kind of devices.

In this section we deal with recognizer membrane systems *with an input membrane* and we propose to solve hard problems in an *uniform* way in the following sense: all instances of a decision problem that have the same *size* (according to a prefixed polynomial time computable criterion) are processed by the same system, to which an appropriate input, that depends on the specific instance, is supplied.

Now, we formalize these ideas in the following definition.

Definition 11 *Let $X = (I_X, \theta_X)$ be a decision problem. We say that X is solvable in polynomial time by a family of recognizer membrane systems* with input $\mathbf{\Pi} = (\Pi(n))_{n \in \mathbf{N}}$, *and we denote it by $X \in \mathbf{PMC}_{\mathcal{R}}$, if the following is true:*

- *The family $\mathbf{\Pi}$ is polynomially uniform by Turing machines; that is, there exists a deterministic Turing machine that constructs in polynomial time the system $\Pi(n)$ from $n \in \mathbf{N}$.*
- *There exists a polynomial encoding (cod, s) of X in $\mathbf{\Pi}$ such that:*

- *The family* $\mathbf{\Pi}$ *is polynomially bounded with regard* (X, cod, s)*; that is, there exists a polynomial function* $p(n)$ *such that for each* $w \in I_X$ *every computation of the system* $\Pi(s(w))$ *with input* $cod(w)$ *is halting and, moreover, it performs at most* $p(|w|)$ *steps.*
- *The family* $\mathbf{\Pi}$ *is sound and complete with regard to* (X, cod, s)*.*

Note that in the above definition and in order to decide about an instance, w, of a decision problem, first of all we need to compute the natural number $s(w)$, obtain the input multiset $cod(w)$, and construct the system $\Pi(s(w))$. This is properly a *pre-computation stage*, running in polynomial time expressed by a number of *sequential steps* in the framework of the Turing machines. After that, we execute the system $\Pi(s(w))$ with input $cod(w)$. This is properly the *computation stage*, also running in polynomial time, but now it is described by a number of *parallel steps*, in the framework of membrane computing.

As mentioned above, these complexity classes are closed under complement.

Moreover, these complexity classes are closed under polynomial time reduction, in the classical sense.

Proposition 3 *Let* $\mathcal{R}$ *be a class of recognizer P systems with input membrane. Let* X *and* Y *be two decision problems such that* X *is reducible to* Y *in polynomial time. If* $Y \in \mathbf{PMC}_{\mathcal{R}}$*, then* $X \in \mathbf{PMC}_{\mathcal{R}}$*.*

The *Satisfiability Problem* can be solved in linear time by a family $\mathcal{R}$ of recognizer P systems with input (see [35]). Then $\mathbf{NP} \subseteq \mathbf{PMC}_{\mathcal{R}}$.

11 (Non-Deterministic) Polynomial Complexity Classes in Membrane Systems

According to the usual manner of considering acceptance by non-deterministic Turing machines, we can consider non-deterministic complexity classes in P systems without requiring them to be confluent, that is, *characterizing* the acceptance of an input string by the existence of an accepting computation.

Definition 12 *Let* $\mathcal{R}$ *be a class of recognizer P systems without input membrane. A decision problem* $X = (I_X, \theta_X)$ *is non-deterministically solvable in polynomial time by a family* $\mathbf{\Pi} = (\Pi(w))_{w \in I_X}$*, of P systems of type* $\mathcal{R}$*, and we denote it by* $X \in \mathbf{NPMC}^*_{\mathcal{R}}$*, if the following is true:*

- *The family* $\mathbf{\Pi}$ *is polynomially uniform by Turing machines.*
- *The family* $\mathbf{\Pi}$ *is polynomially bounded.*
- *The family* $\mathbf{\Pi}$ *is sound and complete with regard to* X*, in the following sense: for each instance* $w \in I_X$ *of the problem,* $\theta_X(w) = 1$ *if and only if there exists an accepting computation of* $\Pi(w)$*.*

Note that in this definition, in contrast to the corresponding definition for deterministic complexity classes, we only demand that for each instance w with affirmative answer there exists at least *one* accepting computation of the system $\Pi(w)$, instead of demanding *every* computation of the system to be an accepting one.

Again, this class is closed under polynomial time reduction, but notice that it does not have to be closed under complement.

Let us denote by $\mathcal{T}$ the class of recognizer *transition* P systems (see [22]). In [35] we construct a family of recognizer transition P systems solving *HPP* (in the directed version with two distinguished nodes) in *linear time*, in a non-deterministic manner. That is, we have the following:

Proposition 4 *HPP* $\in \mathbf{NPMC}^*_{\mathcal{T}}$, *and* $\mathbf{NP} \subseteq \mathbf{NPMC}^*_{\mathcal{T}}$.

In a similar way we can define non-deterministic complexity classes for recognizer membrane systems with input.

Definition 13 *Let* $X = (I_X, \theta_X)$ *be a decision problem. We say that* X *is non-deterministically solvable in polynomial time by a family of recognizer membrane systems* with input $\mathbf{\Pi} = (\Pi(n))_{n\in\mathbf{N}}$, *and we denote it by* $X \in \mathbf{NPMC}_{\mathcal{R}}$, *if the following is true:*

- *The family* $\mathbf{\Pi}$ *is polynomially uniform by Turing machines.*
- *There exists a polynomial encoding* (cod, s) *of* X *in* $\mathbf{\Pi}$ *such that:*
  - *The family* $\mathbf{\Pi}$ *is polynomially bounded with regard to* (X, cod, s).
  - *The family* $\mathbf{\Pi}$ *is sound and complete with regard to* (X, cod, s), *but now in the following sense: for each instance* $w \in I_X$ *of the problem,* $\theta_X(w) = 1$ *if and only if there exists an accepting computation of* $\Pi(s(w))$ *with input* $cod(w)$.

This class is closed under polynomial time reduction, but it does not have to be closed under complement.

In [35] we construct a family of recognizer transition P systems solving *SAT* in *constant time*, in a non-deterministic manner. That is, we have the following:

Proposition 5 *SAT* $\in \mathbf{NPMC}_{\mathcal{T}}$, *and* $\mathbf{NP} \subseteq \mathbf{NPMC}_{\mathcal{T}}$.

12 Characterizing the P $\neq$ NP Relation through P Systems

In this section we show how it is possible to attack the **P** versus **NP** problem within the framework of membrane computing.

We consider deterministic Turing machines as language decision devices. That is, the machines halt over any string on the input alphabet, with the halting state being equal to the accepting state, in the case that the string belongs to the decided language, and with that state equal to the rejecting state, in the case that the string does not belong to that language.

It is possible to associate with a Turing machine a decision problem, which will permit us to say when such a machine is simulated by a family of P systems.

Definition 14 *Let TM be a Turing machine with input alphabet Σ_{TM}. The* decision problem associated with TM *is the problem $X_{TM} = (I, \theta)$, where $I = \Sigma_{TM}^*$, and for every $w \in \Sigma_{TM}^*$, $\theta(w) = 1$ if and only if TM accepts w.*

Obviously, the decision problem X_{TM} is solvable by the Turing machine TM.

Definition 15 *We say that a Turing machine TM is simulated in polynomial time by a family of recognizer P systems if $X_{TM} \in \mathbf{PMC}_{\mathcal{R}}$.*

In P systems, evolution rules, communication rules and rules involving dissolution are called *basic rules.* That is, by applying this kind of rules the size of the structure of membranes does not increase. Hence, it is not possible to construct an exponential working space in polynomial time using only basic rules in a P system.

In Chapter 9 of [39], and following the ideas from [40], we state that every deterministic Turing machine can be simulated in polynomial time by a family of systems of the class $\mathcal{R}$.

Proposition 6 *Let TM be a deterministic Turing machine working in polynomial time. Then TM can be simulated in polynomial time by a family of recognizer P systems using only basic rules.*

In [37], we proved the following result that can be considered as a reciprocal of the above proposition.

Proposition 7 *If a decision problem is solvable in polynomial time by a family of recognizer P systems (using only basic rules), then there exists a Turing machine solving it in polynomial time.*

From the above propositions, we establish characterizations of the $\mathbf{P} \neq \mathbf{NP}$ relation by means of the polynomial time unsolvability of **NP**–complete problems by families of recognizer P systems.

Theorem 8 *The following conditions are equivalent:*

(1) $\mathbf{P} \neq \mathbf{NP}$.

(2) There exists an **NP**–*complete decision problem unsolvable in polynomial time by a family of of recognizer P systems using only basic rules.*

(3) Each **NP**–*complete decision problem is unsolvable in polynomial time by a family of of recognizer P systems using only basic rules.*

From the constructive proof given in [37], we can deduce the following nice result characterizing the class **P**.

Proposition 9 *Let $\mathcal{T}$ the class of recognizer transition P systems. Then $\mathbf{P} = \mathbf{PMC}_{\mathcal{T}}$.*

13 P Systems with Active Membranes

P systems with membrane division were introduced in [24], and in this variant the number of membranes can increase exponentially in polynomial time.

Next, we define P systems with active membranes using 2-division for elementary membranes, with polarizations, but without cooperation and without priorities (and without permitting the change of membrane labels by means of any rule).

Definition 16 *A P system with active membranes using 2-division for elementary membranes is a tuple* $\Pi = (\Sigma, H, \mu, \mathcal{M}_1, \dots, \mathcal{M}_m, R)$, *where:*

1. $m \geqslant 1$, *is the initial degree of the system;*
2. Σ *is an alphabet of symbol-objects;*
3. H *is a finite set of labels for membranes;*
4. μ *is a membrane structure, of* m *membranes, labelled (not necessarily in a one-to-one manner) with elements of* H*;*
5. $\mathcal{M}_1, \dots, \mathcal{M}_m$ *are strings over* Σ*, describing the initial multisets of objects placed in the* m *regions of* μ*;*
6. R *is a finite set of evolution rules, of the following forms:*
   (a) $[\, a \rightarrow \omega \,]_h^{\alpha}$ *for* $h \in H$,$\alpha \in \{+, -, 0\}$, $a \in \Sigma$, $\omega \in \Sigma^*$*: This is an object evolution rule, associated with a membrane labelled with h and depending on the polarity of that membrane, but not directly involving the membrane.*
   (b) $a\,[\ \ ]_h^{\alpha_1} \rightarrow [\, b \,]_h^{\alpha_2}$ *for* $h \in H$, $\alpha_1, \alpha_2 \in \{+, -, 0\}$, $a, b \in \Sigma$*: An object from the region immediately outside a membrane labelled with h is introduced in this membrane, possibly transformed into another object, and, simultaneously, the polarity of the membrane can be changed.*
   (c) $[\, a \,]_h^{\alpha_1} \rightarrow b\,[\ \ ]_h^{\alpha_2}$ *for* $h \in H$, $\alpha_1, \alpha_2 \in \{+, -, 0\}$, $a, b \in \Sigma$*: An object is sent out from membrane labelled with h to the region immediately outside, possibly transformed into another object, and, simultaneously, the polarity of the membrane can be changed.*
   (d) $[\, a \,]_h^{\alpha} \rightarrow b$ *for* $h \in H$, $\alpha \in \{+, -, 0\}$, $a, b \in \Sigma$*: A membrane labelled with h is dissolved in reaction with an object. The skin is never dissolved.*
   (e) $[\, a \,]_h^{\alpha_1} \rightarrow [\, b \,]_h^{\alpha_2}\ [\, c \,]_h^{\alpha_3}$ *for* $h \in H$, $\alpha_1, \alpha_2, \alpha_3 \in \{+, -, 0\}$, $a, b, c \in \Sigma$*: An elementary membrane can be divided into two membranes with the same label, possibly transforming some objects and their polarities.*

These rules are applied according to the following principles:

- All the rules are applied in parallel and in a maximal manner. In one step, one object of a membrane can be used by only one rule (chosen in a non-deterministic way), but any object which can evolve by one rule of any form, must evolve.
- If a membrane is dissolved, its content (multiset and internal membranes) is left free in the surrounding region.
- If at the same time a membrane labelled by h is divided by a rule of typ e (e) and there are objects in this membrane which evolve by means of rules of type (a), then we suppose that first the evolution rules of type (a) are used, and then the division is produced. Of course, this process takes only one step.
- The rules associated with membranes labelled by h are used for all copies of this membrane. At one step, a membrane can be the subject of *only one* rule of types (b)-(e).

Note that these P systems have some important properties:

- They use three electrical charges.
- The polarization of a membrane can be modified by the application of a rule.
- The label of a membrane cannot be modified by the application of a rule.
- They do not use cooperation neither priorities.

Let us denote by $\mathcal{AM}$ the class of recognizer P systems with active membranes using 2-division for elementary membranes.
In this class of recognizer membrane systems:

- Some weakly **NP**–complete problems are solvable in polynomial time: for example, *Knapsack* ([30]), *Subset Sum* ([29]), *Partition* ([8]) $\in \mathbf{PMC}_{\mathcal{AM}}$.
- Some strongly **NP**–complete problems are solvable in polynomial time: for example, the following problems *SAT* ([35]), *Clique* ([3]), *Bin Packing* ([32]), *CAP* ([33]) belong to the complexity classes $\mathbf{PMC}_{\mathcal{AM}}$.

Recall that polynomial time solutions to strongly **NP**–complete problems by recognizer membrane systems, can be considered as pseudo-polynomial solutions in the classical sense.

Having in mind that the complexity class $\mathbf{PMC}_{\mathcal{AM}}$ is closed under complement and polynomial time reductions we have the following result.

Proposition 10 $\mathbf{NP} \subseteq \mathbf{PMC}_{\mathcal{AM}}$, *and* $\mathbf{co\text{-}NP} \subseteq \mathbf{PMC}_{\mathcal{AM}}$.

P. Sosik in [41] provides a semi–uniform efficient solution to *QBF* (satisfiability of quantified propositional formulas), a well known **PSPACE**–complete problem, in the framework of P systems with active membranes but using 2–division for non–elementary membranes. Hence we have the following result.

Proposition 11 *Let $\mathcal{AM}^*$ be the class of recognizer P systems with active membranes using 2-division for non–elementary membranes. Then,* $\mathbf{PSPACE} \subseteq \mathbf{PMC}^*_{\mathcal{AM}^*}$.

This result shows that the complexity classes $\mathbf{PMC}_{\mathcal{AM}}$ and $\mathbf{PMC}^*_{\mathcal{AM}^*}$ are not precise enough to describe classical complexity classes below **NP**. Therefore, it is challenging to investigate weaker variants of P systems with active membranes ab le to characterize classical complexity classes (especially, the classes **NP** and **PSPACE**).

In [1], universality has been achieved removing the polarization of membranes from P systems with active membranes but allowing the change of membrane labels by means of communication rules and membrane division rules. Moreover, in this framework it is possible to solve **NP**–complete problems (e.g., the *SAT* problem) in linear time.

Several efficient solutions to **NP**–complete problems have been obtained within the following variants of membrane systems with active membranes:

- P systems using 2–division for elementary membranes, without cooperation, without priorities, without label changing, but using only two electrical charges ([1], [38]).
- P systems using 2–division for elementary membranes, without cooperation, without priorities, without changing of membrane labels, without polarizations, but using bi–stable catalysts ([31]).
- P systems without polarizations, without cooperation, without priorities, without label changing, without division, but using three types of membrane rules: separation, merging, and release ([18]).
- P systems with separation rules instead of division rules, in two different cases: in the first, using polarizations and separation rules; and in the second one, without polarizations, without change of membrane labels but using separation rules with change of membrane labels ([19]).

It is easy to obtain solutions to **NP**–complete problems through P systems with active membranes using 2-division for elementary membranes, without polarizations, without priorities, without label changing possibilities, but using cooperation (or trading cooperation by priority).

But, what happens if we consider only recognizer P systems with active membranes using 2-division for elementary membranes, without polarizations, without cooperation, without priority, and without changing of membrane labels? Let $\mathcal{AM}^0$ be the class of recognizer P systems of this kind.

What is exactly the class of decision problems solvable in polynomial time by families of systems belonging to $\mathcal{AM}^0$? Is the relation $\mathbf{P} = \mathbf{PMC}_{\mathcal{AM}^0}$ true?

Another interesting questions about the relationship between classical and cellular complexity classes are the following ones:
Question 1: Is there a classical complexity class $\mathcal{C}$, such that $\mathcal{C} = \mathbf{PMC}_{\mathcal{AM}}$?
Question 2: Given a classical complexity class $\mathcal{C}$, determine a (minimal in a descriptive sense) class of recognizer P systems $\mathcal{F}$ such that $\mathcal{C} = \mathbf{PMC}_{\mathcal{F}}$?

14 Conclusions

In this paper, some polynomial complexity classes in recognizer membrane systems, without or with input, and capturing the "classical" deterministic and non-deterministic modes of computation, have been introduced.

The complexity classes corresponding to membrane systems without input (respectively, with input) provide the general framework to design solutions to decision problems in a *semi–uniform* (respectively, *uniform*) way.

In this context we have proven that membrane computing offers a new way to attack the **P** versus **NP** problem.

The convenience of characterizing classical complexity classes through these new classes is an interesting topic in order to study the minimal ingredients required, from membrane systems point of view, to obtain certain *computational efficiency.*

Acknowledgement

The author wish to acknowledge the support of the project TIC2002-04220-C03-01 of the Ministerio de Ciencia y Tecnología of Spain, cofinanced by FEDER funds.

References

[1] A. Alhazov, R. Freund, Gh. Păun, P systems with active memb ranes and two polarizations. *Proceedings of the Second Brainstorming Week on Membrane Computing* (Gh. Păun, A. Riscos, A. Romero, F. Sancho, eds.), Report RGNC 01/04, 2004, 20–35.

[2] A. Alhazov, T.-O. Ishdorj, Membrane operations in P systems with active membranes. *Proceedings of the Second Brainstorming Week on Membrane Computing* (Gh. Păun, A. Riscos, A. Romero, F. Sancho, eds.), Report RGNC 01/04, 2004, 37–52.

[3] A. Alhazov, C. Martín–Vide, L. Pan, Solving graph problems by P systems with restricted elementary active membranes. *Aspects of Molecular Computing* (N. Jonoska, Gh. Păun, G. Rozenberg, eds.), Lecture Notes in Computer Science, 2950 (2004), 1–22.

[4] J. Castellanos, Gh. Păun, A. Rodrguez–Patn, P systems with worm–objects. *IEEE 7th International Conference on String Processing and Information Retrieval, SPIRE 2000*, La Coruña, Spain, 64–74.

[5] A. Cordón–Franco, M.A. Gutiérrez–Naranjo, M.J. Pérez–Jiménez, F. Sancho–Caparrini, Implementing in Prolog an effective cellular solution for the Knapsack problem. *Membrane Computing* (C. Martín-Vide, Gh. Păun, G. Rozenberg, A. Salomaa, eds.), Lecture Notes in Computer Science, 2933 (2004), 140-152.

[6] E. Czeiler, Self–activating P systems. *Membrane Computing* (Gh. Păun, G. Rozenberg, A, Salomaa, C. Zandron, eds.), Lecture Notes in Computer Science, 2597 (2003), 234–246.

[7] M.R. Garey, D.S. Johnson, *Computers and Intractability. A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York, 1979.

[8] M.A. Gutiérrez–Naranjo, M.J. Pérez–Jiménez, A. Riscos–Núñez, A fast P system for finding a balanced 2-partition. *Soft Computing*, in press.

[9] T. Head, M. Yamamura, S. Gal, Aqueous computing: writing on molecules. *Proceedings of the Congress on Evolutionary Computation 1999*, IEEE Service Center, Piscataway, NJ, 1999, 1006–1010.

[10] M. Ito, C. Martín–Vide, Gh. Păun, Characterization of Parikh sets of ET0L languages in terms of P systems. In *Words, Semigroups, and Transducers* (M. Ito, Gh. Păun, S. Yu, eds.), World Scientific, Singapore, 2001, 239–254.

[11] S.N. Krishna, R. Rama, A variant of P systems with active membranes: Solving NP–complete problems. *Romanian Journal of Information Science and Technology*, 2, 4 (1999), 357–367.

[12] S.N. Krishna, R. Rama, P systems with replicated rewriting. *Journal of Automata, Languages and Combinatorics*, 6, 1 (2001), 345–350.

[13] S.N. Krishna, R. Rama, Breaking DES using P systems. *Theoretical Computer Science*, 299, 1-3 (2003), 495–508.

[14] M. Madhu, K. Kristhivasan, P systems with membrane creation: Universality and efficiency. *Proceedings Third International Conference on Universal, Machines and Computations*, Chisinau, Moldova, 2001 (M. Margenstern, Y. Rogozhin, eds.), Lecture Notes in Computer Science, 2055 (2001), 276–287.

[15] A. Obtułowicz, Deterministic P systems for solving SAT problem. *Romanian Journal of Information Science and Technology*, 4, 1–2 (2001), 551–558.

[16] A. Obtułowicz, On P systems with active membranes: Solving the Integer Factorization problem in a polynomial time. In *Multiset Processing. Mathematical, Computer Science, and Molecular Computing Points of View* (C.S. Calude, Gh. Păun, G. Rozenberg, A. Salomaa, eds.), Lecture Notes in Computer Science, 2235 (2001), 267–285.

[17] A. Obtułowicz, Note on some recursively family of P systems with active membranes. Submitted, 2004.

[18] L. Pan, A. Alhazov, T.-O. Ishdorj, Further remarks on P systems with active membranes, separation, merging, and release rules. *Proceedings of the Second Brainstorming Week on Membrane Computing* (Gh. Păun, A. Riscos, A. Romero, F. Sancho, eds.), Report RGNC 01/04, 2004, 316–324.

[19] L. Pan, T.-O. Ishdorj, P systems with active membranes and separation rules. *Journal of Universal Computer Science*, 10, 5 (2004), 630–649.

[20] L. Pan, C. Martín–Vide, C. Solving multiset 0–1 knapsack problem by P systems with input and active membranes. *Proceedings of the Second Brainstorming Week on Membrane Computing* (Gh. Păun, A. Riscos, A. Romero, F. Sancho, eds.), Report RGNC 01/04, 2004, 342–353.

[21] A. Păun, On P systems with membrane division. In *Unconventional Models of Computation* (I. Antoniou, C.S. Calude, M.J. Dinneen, eds.), Springer, London, 2000, 187–201.

[22] Gh. Păun, Computing with membranes, *Journal of Computer and System Sciences*, 61, 1 (2000), 108–143, and *Turku Center for Computer Science-TUCS Report* Nr. 208, 1998.

[23] Gh. Păun, Computing with membranes: Attacking **NP**–complete problems. In *Unconventional Models of Computation* (I. Antoniou, C.S. Calude, M.J. Dinneen, eds.), 2000, 94–115.

[24] Gh. Păun, P systems with active membranes: Attacking **NP**–complete problems. *Journal of Automata, Languages and Combinatorics*, 6, 1 (2001), 75–90.

[25] Gh. Păun, *Membrane Computing. An Introduction*, Springer-Verlag, Berlin, 2002.

[26] Gh. Păun, M.J. Pérez–Jiménez, A. Riscos–Núñez, P systems with tables of rules. *Theory is Forever. Essays Dedicated to Arto Salomaa on the Ocassion of His 70th Birthday* (J. Karhumaki, H. Maurer, Gh. Păun, G. Rozenberg, eds.), Lecture Notes in Computer Science, 3113 (2004), 235-249.

[27] Gh. Păun, G. Rozenberg, A guide to membrane computing. *Theoretical Computer Science*, 287 (2002), 73–100.

[28] Gh. Păun, Y. Suzuki, H. Tanaka, T. Yokomori, On the power of membrane division in P systems. *Theoretical Computer Science*, 324, 1 (2004), 61–85.

[29] M.J. Pérez–Jiménez, A. Riscos–Núñez, Solving the Subset-Sum problem by P systems with active membranes. *New Generation Computing*, in press.

[30] M.J. Pérez–Jiménez, A. Riscos–Núñez, A linear time solution to the Knapsack problem using active membranes. *Membrane Computing* (C. Martn-Vide, Gh. Păun, G. Rozenberg, A. Salomaa, eds.). Lecture Notes in Computer Science, 2933 (2004), 250–268.

[31] M.J. Pérez–Jiménez, F.J. Romero-Campero, Trading polarizations for bi-stable catalysts in P systems with active membranes. In this volume.

[32] M.J. Pérez–Jiménez, F.J. Romero-Campero, An efficient family of P systems for packing items into bins. *Journal of Universal Computer Science*, 10, 5 (2004), 650–670.

[33] M.J. Pérez–Jiménez, F.J. Romero-Campero, Attacking the Common Algorithmic problem by recognizer P systems. *Pre-proceedings of the Machines, Computations and Universality, MCU'2004 (abstracts)*, September 21-26, 2004, Sankt Petesburg, p. 27.

[34] M.J. Pérez–Jiménez, A. Romero-Jiménez, F. Sancho-Caparrini, *Teoría de la Complejidad en Modelos de Computación con Membranas*, Ed. Kronos, Sevilla, 2002.

[35] M.J. Pérez–Jiménez, A. Romero–Jiménez, F. Sancho–Caparrini, Complexity classes in models of cellular computing with membranes. *Natural Computing*, 2, 3 (2003), 265–285.

[36] M.J. Pérez–Jiménez, A. Romero–Jiménez, F. Sancho–Caparrini, Solving VALIDITY problem by active membranes with input. *Proceedings of the Brainstorming Week on Membrane Computing* (M. Cavaliere, C. Martín-Vide, Gh. Păun, eds.), Report GRLMC 26/03, 2003, 279–290.

[37] M.J. Pérez–Jiménez, A. Romero–Jiménez, F. Sancho–Caparrini, The P versus NP problem through cellular computing with membranes. *Aspects of Molecular Computing. Essays Dedicated to Tom Head on the Ocassion of His 70th Birthday* (N. Jonoska, Gh. Păun, G. Rozenberg, eds.), Lecture Notes in Computer Science, 2950 (2004), 338–352.

[38] A. Riscos-Núñez, *Programacin celular: Resolucin eficiente de problemas numricos* **NP**–*completos.* PhD. Thesis, University of Seville, Spain, 2004.

[39] A. Romero-Jiménez, *Complexity and Universality in Cellular Computing Models*, PhD. Thesis, University of Seville, Spain, 2003.

[40] A. Romero-Jiménez, M.J. Pérez–Jiménez, Simulating Turing machines by P systems with external output. *Fundamenta Informaticae*, 49, 1-3 (2002), 273–287.

[41] P. Sosik, The computational power of cell division. *Natural Computing*, 2, 3 (2003), 287–298.

[42] C. Zandron, C. Ferreti, G. Mauri, Solving NP-complete problems using P systems with active membranes. In *Unconventional Models of Computation, UMC'2K* (I. Antoniou, C. Calude, M.J. Dinneen, eds.), Springer-Verlag, Berlin, 2000, 289–301.

[43] C. Zandron, G. Mauri, C. Ferreti, Universality and normal forms on membrane systems. *Proceedings International Workshop on Grammar Systems, 2000* (R. Freund, A. Kelemenova, eds.), Bad Ischl, Austria, July 2000, 61–74.

Contributions and work in progress reports

P systems with vague boundaries: the t-norm approach

Stefano Aguzzoli[1], **Daniela Besozzi**[2], **Brunella Gerla**[3], **Corrado Manara**[3]

[1]Università degli Studi di Milano
Dipartimento di Scienze dell'Informazione
Via Comelico 39, 20135 Milano, Italy
E-mail: `aguzzoli@dsi.unimi.it`

[2]Università degli Studi di Milano
Dipartimento di Informatica e Comunicazione
Via Comelico 39, 20135 Milano, Italy
E-mail: `besozzi@dico.unimi.it`

[3]Università degli Studi di Salerno
Dipartimento di Matematica e Informatica
Via Ponte don Melillo, 84084 Fisciano (SA), Italy
E-mail: `{bgerla,cmanara}@unisa.it`

1 Introduction

In the everyday life we solve a lot of problems not caring at all about *precision* and perfection of the solution. The increase of precision leads to an increase of the amount of information whose *significance* then decreases until a point is reached, after which precision and significance are mutually excluding characteristics. Then, imprecision (vagueness) cannot be avoided and often is necessary to convey relevant information [4].

This vagueness, called *fuzziness*, can raise during the process of grouping objects having some property P. In general, P cannot characterize unambiguously the group of objects because there can exist some *borderline* elements, which make unsharp the boundaries of the set

$$X = \{x \mid x \text{ has the property } P\}.$$

This led to the development of *fuzzy set theory* and *fuzzy logic* (see [8, 9, 7] and references therein).

Since the introduction of the notion of fuzzy set , the term “fuzzy logic” has been largely used but it is important to make some distinctions. In its *wide sense*, fuzzy logic is a synonymous of fuzzy set theory. In its *narrow sense*, it can be considered as a logical system which aims at a formalization of approximate reasoning. Fuzzy logic originates with the attempt to handle concepts which admit many (more than two) degrees of truth and it is based on a *comparative notion of truth*: one statement may be more true than another one. From this point of view fuzzy logic is worth studying [2].

The path from initial considerations about fuzziness to a formal logical system is not straightforward. Nowadays, the various approaches to *many-valued logics* found in the literature are competing as natural candidates to offer to the engineering discipline of fuzzy logic the theoretical foundations that have been lacking for several years.

Fuzzy logic is naturally described as the logic of degrees of truth, thus differentiating itself from logics of belief, and from probabilistic logic and modal logic, which are not truth-functional. Connectives of a logic behave truth-functionally when the value of the connection of some propositions is a function of the value of the same propositions only.

In order to set a formal framework to deal with fuzzy or uncertain reasoning, if we require the set of truth-values to be linearly ordered, and the connectives of the logic to be truth-functional, then a major tool used in fuzzy logic for modelling uncertain information is the definition of suitable *triangular norms*, *t-norms* for short [3].

In this work we propose a t-norm based approach for handling imprecision in *P systems*. P systems, initially proposed in [5], are a class of distributed and parallel computing devices inspired by the architecture of living cells and the way biological substances are both modified and moved among internal organelles. In a P system, each compartment (an organelle inside the cell) can be seen as a computing unit, having its own data and its local program (molecular substances and biochemical reactions), and all compartments considered as a whole (the cell) can be seen as an “unconventional” computing device. In particular, each compartment is delimited and separated from the rest by a membrane; the whole computing unit is formally characterized by a membrane structure, where membranes can be hierarchically placed inside a unique external membrane delimiting the entire cell. All membranes are semi-permeable barriers, which either allow some substances to move inwards or outwards, and consequently change their location in the membrane structure, or block the movement of some other substances. The biological substances and reactions are represented by means of objects and evolution rules. Objects are usually symbols or strings over a given alphabet, evolution rules are given as rewriting rules with target indications, thus describing both the transformation and the communication of objects.

A computation in P systems is obtained by starting from an initial configuration, identified by the membrane structure, the objects and the rules initially present inside it, and then letting the system evolve. The application of rules is performed in a nondeterministic and maximal parallel manner: all the applicable rules have to be used to modify all objects which can be the subject of a rule, and this is done in parallel for all membranes (a universal clock is assumed to exist). Whenever no rule can be further applied, the computation halts and the output is defined in terms of the objects sent out the external membrane or, alternatively, collected inside a specified membrane. No output is obtained if the computation never halts (that is, whenever a rule can be continuously applied).

Further notions on many variants of P systems, as well as an updated bibliography, can be found in [6] and at `http://psystems.disco.unimib.it`.

2 Triangular norms

Definition 1 (t-norm) A t-norm is any operator $\overset{\wedge}{*}\colon [0,1]^2 \to [0,1]$ satisfying:

1. *Associativity*: $x \overset{\wedge}{*} (y \overset{\wedge}{*} z) = (x \overset{\wedge}{*} y) \overset{\wedge}{*} z$.
2. *Commutativity*: $x \overset{\wedge}{*} y = y \overset{\wedge}{*} x$.
3. *Monotonicity* in each argument: If $y_1 \leqslant y_2$ then $x \overset{\wedge}{*} y_1 \leqslant x \overset{\wedge}{*} y_2$. If $x_1 \leqslant x_2$ then $x_1 \overset{\wedge}{*} y \leqslant x_2 \overset{\wedge}{*} y$.
4. *Absorption*: $x \overset{\wedge}{*} 0 = 0$ and *Unity*: $x \overset{\wedge}{*} 1 = x$.

A t-norm $\overset{\wedge}{*}$ is *continuous* if it is continuous as a real-valued function with respect to each variable.

A *t-conorm* is the dual operator of a t-norm.

Definition 2 (t-conorm) A t-conorm is any operator $\overset{*}{\vee}\colon [0,1]^2 \to [0,1]$ satisfying:

1. *Associativity*: $x \overset{*}{\vee} (y \overset{*}{\vee} z) = (x \overset{*}{\vee} y) \overset{*}{\vee} z$.
2. *Commutativity*: $x \overset{*}{\vee} y = y \overset{*}{\vee} x$.
3. *Monotonicity* in each argument: If $y_1 \leqslant y_2$ then $x \overset{*}{\vee} y_1 \leqslant x \overset{*}{\vee} y_2$. If $x_1 \leqslant x_2$ then $x_1 \overset{*}{\vee} y \leqslant x_2 \overset{*}{\vee} y$.
4. *Absorption*: $x \overset{*}{\vee} 0 = x$ and *Unity*: $x \overset{*}{\vee} 1 = 1$.

As we can note, t-norms and t-conorms differ only in the boundary condition imposed.

Triangular norms can be used to model *graded-truth conjunction*. Some natural requirements such a conjunction should satisfy are met by the definition of t-norm. Indeed, the truth degree of the conjunction of propositions A and B should not depend on the order in which A and B are connected. The same is true for the truth degree of conjunctions of several propositions $A_1, A_2, \ldots, A_u$. These two properties are witnessed by commutativity and associativity. It is also natural to assume that the truth degree of the conjunction of a proposition with a complete falsity should be completely false, thus justifying the absorption requirement. Analogously, the conjunction of a proposition with a complete truth should not have smaller truth degree than the proposition has. Finally, we should expect that high truth degrees of propositions A and B would correspond to a high truth degree of their conjunction, and this is assured by the fact that t-norms are increasing functions in each argument. Note also that the absorption and unity properties state that each t-norm coincides with the conjunctive connective of classical logic when properly restricted to the domain $\{0,1\}^2$.

There exist uncountably many t-norms. If we restrict our attention to continuous t-norms only, we shall see that there exist three main t-norms, all the others arising as suitable combinations of them:

- Łukasiewicz t-norm: $x \odot y = \max(0, x + y - 1)$.
- Gödel t-norm: $x \wedge y = \min(x, y)$.
- Product t-norm: $x \cdot y = xy$, product of real numbers.

An analogous approach to *graded-truth implication* requires that the truth degree of A *implies* B should be high when the truth degree of A is not significantly higher than the truth degree of B: then any binary operator $\Rightarrow$, chosen as semantics of an implication connective, should be non-increasing in its first argument and non-decreasing in the second one. To model a sound and powerful rule of *graded-truth modus ponens*, we require that from lower bounds a, c of the truth degrees of propositions A and $A \Rightarrow B$ respectively, we can infer a lower bound b for the truth-degree of B. If we combine a and c by some fixed t-norm $\overset{\wedge}{*}$, then we may require $c = a \Rightarrow b$ to be the maximum value such that $a \overset{\wedge}{*} c \leqslant b$ is satisfied. Actually, the following lemma holds:

Lemma 1 *Given any continuous t-norm $\overset{\wedge}{*}$, there is a unique operator $\Rightarrow_{\overset{\wedge}{*}}$: $[0,1]^2 \to [0,1]$, such that, for all $x, y, z \in [0,1]$:*

$$x \overset{\wedge}{*} z \leqslant y \qquad \textit{if and only if} \qquad z \leqslant x \Rightarrow_{\overset{\wedge}{*}} y.$$

The operator $\Rightarrow_{\overset{\wedge}{*}}$ is called the *residuum* of $\overset{\wedge}{*}$ and is defined by:

$$x \Rightarrow_{\overset{\wedge}{*}} y = \max(z | x \overset{\wedge}{*} z \leqslant y).$$

For any continuous t-norm, the residuum operation coincides with the truth-table of classical implication, when its domain is restricted to $\{0,1\}^2$.

The residuum operators induced by the three main continuous t-norms are:

- Łukasiewicz implication: $x \Rightarrow_{\odot} y = \min(1, 1 - x + y)$.

- Gödel implication: $x \Rightarrow_{\wedge} y = \begin{cases} 1 & \text{if } x \leqslant y \\ y & \text{otherwise.} \end{cases}$

- Product implication: $x \Rightarrow_{.} y = \begin{cases} 1 & \text{if } x \leqslant y \\ y/x & \text{otherwise.} \end{cases}$

The choice of a (continuous) t-norm $\overset{\wedge}{*}$ determines an entire propositional many-valued logic, with its connectives of conjunction, implication, negation, and modus ponens.

Hájek's Basic Logic BL [2], which is presented as a traditional Hilbert system with a finite set of axiom schemata, is the logic of all continuous t-norms and their residua. That is, BL proves a formula φ iff the standard interpretation of φ evaluates identically to 1, in each t-norm algebra

$$([0,1], \overset{\wedge}{*}, \Rightarrow_{\overset{\wedge}{*}}, 0).$$

The class of all algebraic models of BL forms the algebraic variety $\mathcal{BL}$. The study of subvarieties of $\mathcal{BL}$ is the main tool to derive results in all the most important many-valued logics, as Łukasiewicz, Gödel, and Product logics. These results concern both logical and complexity aspects: for the latter, the study of free algebras is of the foremost importance.

Łukasiewicz logic is unique among many-valued propositional logics because all its connectives (primitive and derived) have continuous functions as their semantics.

3 P systems with vague boundaries

We assume the reader is familiar with the basic notions and notations of P systems.

We briefly recall that a *membrane structure* consists of a set of membranes hierarchically embedded in a unique membrane, called the *skin membrane.* The membrane structure is identified with a string of correctly matching square parentheses, placed in a unique pair of matching parentheses; each pair of matching parentheses corresponds to a membrane. Each membrane identifies a region, delimited by it and the membranes (if any) immediately inside it. Usually, a unique label is univocally associated to each membrane. An *object* can be a symbol or a string over a specified finite alphabet V; *multisets* of objects are usually considered in order to describe the presence of multiple copies of any given object. In the following, we will only consider structured objects, that is strings. Objects are modified by means of *evolution rules* which are, usually, context-free rewriting rules with an associated target indication (*tar*, in short) of the form $here, out, in$. The target indication determines the region where the object is communicated after the application of the rule: if $tar = here$, then the object remains in the same region; if $tar = out$, then the object exits from the region where it was placed; if $tar = in$, then the object nondeterministically enters one of the membranes immediately inside the region where the rule is applied, if any inner region exists (otherwise the rule cannot be applied).

In this section we introduce the notion of a P system with *vague boundaries*, which satisfies some peculiar aspects not common with the classical definition of P system:

- each object can be simultaneously present inside many regions, this is formally expressed by assigning a membership value to it, denoting "how much it belongs" to every region;
- each rule can be simultaneously active in many regions, this is formally expressed by assigning a value to it, denoting "how much it is active" inside every region;
- there is no crisp separation of regions, instead each membrane represents a vague boundary with respect to the adjacent regions.

As a consequence, we believe that the communication of objects can be described with a t-norm approach (by evaluating the composition of the truth degree of the objects with the truth degree of the rules) and it is no more necessary to associate target indications to rules.

Formally, a P system with vague boundaries in the t-norm approach is defined as

$$\Pi = (V, T, \mu, M, R, \Phi, (\overset{\wedge}{*}, \overset{*}{\vee}), i_o)$$

where:

- V is the alphabet of the system;
- $T \subseteq V$ is the terminal (or output) alphabet;
- μ is a membrane structure consisting of n membranes, which are injectively labelled by numbers in the set $\{1, \ldots, n\}$;
- $M = \{\sigma_1, \ldots, \sigma_p\}$ is a (multi)set of strings over V, representing the objects initially present in all regions of the system;
- $R = \{r_1, \ldots, r_q\}$ is a finite set of context-free rewriting rules of the form $a \to x$, with $a \in V, x \in V^*$, associated with the regions of μ;
- $\Phi = (\mu_1, \ldots, \mu_n)$ is the membership function initially associated with the regions of μ, where $\mu_i : M \cup R \to [0, 1]$ for all $i = 1, \ldots, n$;
- $(\overset{\wedge}{*}, \overset{*}{\vee})$ is the chosen pair of t-norm and t-conorm;
- i_o is a number in the set $\{1, \ldots, n\} \cup \{\infty\}$, indicating the *output* region.

We denote by m_i the membrane (and its corresponding region) labelled with number i, $i = 1, \ldots, n$, present in the membrane structure μ. Note that, since we do not consider any dissolving or dividing action for membranes, the membrane structure will never be modified during any computation.

As in classical rewriting P systems, for each string that can be the subject of many evolution rules at the same time (possibly applicable on more than one symbol in the string), we consider only one possibility to rewrite it: we apply only one evolution rule (nondeterministically chosen among all applicable rules) and we apply it over only one symbol in the string (nondeterministically chosen among all rewritable symbols). Hence, no parallel rewriting methods will be used here.

We consider the proposition "The string σ_j is in the region delimited by m_i" for every m_i in μ, $\sigma_j \in M$, $j = 1, \ldots, p$, and we denote it by $\mu_i(\sigma_j)$. In the same way, we denote by $\mu_i(r_k)$ the proposition "The rule r_k is active in the region delimited by m_i" for every m_i in μ, $r_k \in R$, $k = 1, \ldots, q$. Hence, we have:

$$\mu_i(\sigma_j) \in [0,1], \quad \forall i \in \{1, \ldots, n\} \; \forall \sigma_j \in M,$$

$$\mu_i(r_k) \in [0,1], \quad \forall i \in \{1, \ldots, n\} \; \forall r_k \in R.$$

Consider two configurations $C^{(t)} = (\mu, M^{(t)})$ of Π at time t and $C^{(t+1)} = (\mu, M^{(t+1)})$ of Π at time $t+1$. For every $\sigma \in M^{(t+1)}$ let

$$H_\sigma = \{(j,k) \;\mid\; \sigma_j \Rightarrow \sigma \text{ by using rule } r_k\}$$

the multiset of couple of indexes (j,k) such that the string σ is obtained from some string σ_j by application of some rule r_k. For every $\sigma \in M^{(t+1)}$ the truth value of the proposition "The string σ is in the region delimited by m_i" is the result of the following combination:

$$\mu_i(\sigma) = \overset{*}{\bigvee}_{(j,k) \in H_\sigma} \left(\mu_i(\sigma_j) \overset{\wedge}{*} \mu_i(r_k) \right).$$

The value $\mu_i(\sigma)$ is evaluated for each string σ and for all membranes m_i, in any configuration of the system. Hence, by considering the dynamical update of "how much" each string "belongs" to every membrane, we can determine the "communication" of strings, that is their movement across the vague boundaries of Π.

Let us display the multiset H_σ as $\{(j_1, k_1), (j_2, k_2), \ldots, (j_u, k_u)\}$. The formula defining $\mu_i(\sigma)$ can be read as follows:

EITHER σ is produced from σ_{j_1} by r_{k_1}
OR σ is produced from σ_{j_2} by r_{k_2}
$\vdots$
OR σ is produced from σ_{j_u} by r_{k_u}

In the theory of fuzzy control the "connective" OR is naturally interpreted by a t-conorm, which generalizes the disjunctive character of classical Boolean disjunction in definition by cases. The t-norm used to combine the membership of σ_j with the membership of r_k generalizes the crisp concept of σ_j AND r_k belonging to the same membrane. Then it is worth examining the possible connections between our description of membrane systems with vague boundaries with the theory of t-norm based fuzzy control.

4 Discussion and future work

Here we collect some ideas for further discussion and future developments of our preliminary proposal:

1. Here we have considered P systems with vague boundaries where only string-objects are present inside the membrane structure. The use of structured objects takes inspiration from the biology of the cell, where long molecules (for instance, proteins) can live across the phospholipidic bilayer of membranes, thus having a part inside and another part outside the organelle delimited by that membrane. What about the natural extension of P systems with vague boundaries to the case of multisets of symbol-objects?

2. In a cell, many transmembrane proteins act as channels or gates for the (selective) passage of biochemical substances. In [1] the functioning of sodium-potassium exchange pump is modelled within the framework of P systems, and the notion of bilayer is defined in order to have a realistic description of the cellular process. Hence, it would be interesting to introduce the same notion of bilayer also in P systems with vague boundaries, and to define the membership values of objects and rules not only for all membranes but also for their corresponding bilayer.

3. It could be interesting to adapt our definition of membranes with vague boundaries to describe hierarchical systems where the notion of sphere of influence plays a key role. In this setting, the concepts of distributions of objects and their topological or metrical relationships could be modeled by adding structure, in the form of logical or analytical constraints, to our basic description.

References

[1] D. Besozzi and G. Ciobanu. A P system description of the sodium-potassium pump. *Submitted work*, 2004.

[2] P. Hájek. *Metamathematics of fuzzy logic.* Kluwer Academic Publishers, Dordrecht, 1998.

[3] E. P. Klement, R. Mesiar, and E. Pap. *Triangular Norms.* Kluwer Academic Publishers, Dordrecht, 2000.

[4] V. Novák, I. Perfilieva, and J. Mo¯cko¯r. *Mathematical principles of fuzzy logic*. Kluwer, Dordrecht, 1999.

[5] Gh. Păun. Computing with membranes. *Journal of Computer and System Sciences*, 61(1):108–143, 2000. (See also *Turku Center for Computer Science-TUCS Report* No 208, 1998, www.tucs.fi).

[6] Gh. Păun. *Membrane Computing. An Introduction*. Springer-Verlag, Berlin, 2002.

[7] J. Yen and R. Langari. *Fuzzy Logic. Intelligence, Control, and Information*. Prentice-Hall, Upper Saddle River, 1999.

[8] L. Zadeh. Fuzzy sets. *Information and Control*, 8(3):338–353, 1965.

[9] L. Zadeh. Soft computing and fuzzy logic. *IEEE Software*, 11(6):48–56, 1994.

P systems under uncertainty: the case of transmembrane proteins

Stefano Aguzzoli[a], **Ioan I. Ardelean**[b], **Daniela Besozzi**[c],
Brunella Gerla[d], **Corrado Manara**[d]

[a]Università degli Studi di Milano
Dipartimento di Scienze dell'Informazione
Via Comelico 39, 20135 Milano, Italy
E-mail: aguzzoli@dsi.unimi.it
[b]Institute of Biology of the Romanian Academy
Centre of Microbiology
Splaiul Independenţei 296
PO Box 56–53, Bucharest 060031, Romania
E-mail: ioan.ardelean@ibiol.ro
[c]Università degli Studi di Milano
Dipartimento di Informatica e Comunicazione
Via Comelico 39, 20135 Milano, Italy
E-mail: besozzi@dico.unimi.it
[d]Università degli Studi di Salerno
Dipartimento di Matematica e Informatica
Via Ponte don Melillo, 84084 Fisciano (SA), Italy
E-mail: {bgerla,cmanara}@unisa.it

1 Introduction

P systems, initially proposed in [77], are a class of distributed and parallel computing devices inspired by the architecture and functioning of living cells. It has to be stressed that these systems were not intended to be a model of the cell, instead their purpose was to investigate some computational features which can be abstracted from the cellular biology. Anyway, in several recent works the framework of P systems has been used to define models of specific cellular processes or structures (see an updated bibliography at `http://psystems.disco.unimib.it`).

In this work we are mainly interested in investigating the modelling power of P systems, and in the extension of the framework, in order to deal with *imprecise* biological information. Indeed, many aspects of the cell functioning are still unknown to biologists and source of *uncertainty*. In these cases uncertainty does not emerge not only because of the lack of knowledge about the occurrence of some event (a mathematical model for this is provided by the probability theory), but it is due to the *vagueness*, that is the capability to use imprecise information to describe the cell functioning. *Fuzzy set theory* and *fuzzy logic* could be useful in this framework. Hence, we propose to approach the problem of integrating fuzzy techniques in P systems to deal with uncertainty in biological systems.

In particular, we will briefly analyze the case of transmembrane proteins. First we consider the P model proposed in [2] for simulating the activity of mechanosensitive channels, where the uncertainty is related to some relevant parameters of the model, and we give some suggestions and open problems concerning how to use fuzzy tools within that model.

Then we propose the investigation of the global behavior of populations of (equal or different) transport proteins, such as ATP-powered pumps, ion channels and transporters, and the study of flux dynamics of the corresponding transported molecules. In this case, the source of uncertainty is concerned with the local position in the cellular membranes of transport proteins and with the local distribution of substances, since the rate and extent of transport is also influenced by the concentrations of substances and by the electric potential that exists across the membrane.

The motivation of this research is to use P systems for modelling the functioning of specific cellular structures and phenomena, having as final goal the production of useful and relevant tools for biologists, and hence motivating further cooperations between scientists working in the areas of P systems and Microbiology. Indeed, the design of appropriate software simulators, based on the corresponding P models, would provide an easier way to check both the effectiveness and the correctness of the models, and hopefully become a tool for testing known data, predicting unknown scenarios and returning meaningful information to biologists.

2 The case of mechanosensitive channels

In this section we first give a biological description of mechanosensitive channels with large conductance (MscL, in short), then we report a sketch and a brief overview of the P system presented in [2] for modelling the functioning of MscL analyzed during patch clamping experiments. The reader is referred to [2] fur further notions and details, as well as for the references therein. Finally, we propose and motivate a fuzzy extension of that model.

2.1 Biological description of mechanosensitive channels

Mechanosensitive channels are homopentameric transmembrane proteins gated by mechanical forces. Their physiological function consists in the protection against severe osmotic downshifts, since they allow the rapid exit of different chemicals and the sudden decrease of the osmotic pressure inside the cell. This event is fundamental for bacterial cell because, when the turgor pressure is too large, the integrity of the cell can be damaged by disruption of cell wall and plasma membrane, followed by cell death. The increase in the pressure exerted against the cellular membrane may be due to natural environmental conditions (e.g., rain falling) or to a suction applied during artificial patch clamping experiments. In correspondence to these distinct situations, in [2] two models for the description of the activity of MscL in *E. coli* and in other prokaryotes are presented. In Section 2.2 we will only consider the *in vitro* model, corresponding to patch clamping experiments.

When the cellular membrane is submitted to a mechanical stretch, it experiences an increase in the *membrane tension*, which causes the progression of the channel from the steady-state closed conformation to an expanded – but still closed – conformation and, through some subconducting open states (which correspond to the breaking away of the sections in the homopentameric structure), to the fully open state (see Figure 1). According to a biological model proposed in [11], we consider the following conformations and their relative notations:

- the closed conformation, denoted by **C**;
- the expanded closed conformation, denoted by **CE**;
- the first subconducting open conformation, denoted by **SO1**, where only one subunit (out of five) is open;
- the second subconducting open conformation, denoted by **SO2**, where two (out of five) subunits are open;
- the third subconducting open conformation, denoted by **SO3**, where three (out of five) subunits are open;
- the fourth subconducting open conformation, denoted by **SO4**, where four (out of five) subunits are open;
- the fully open conformation, denoted by **O** (where all five subunits are open).

Data collected from patch clamping experiments on *E. coli* [12] correspond to the following real values, or interval of values, for the membrane tension (measured in dyne/cm):

(i) $t_C \in [0, 10)$, when no suction is applied to the patch membrane;

(ii) $t_{CE} = 10$, when a suction is applied to the patch membrane, the membrane tension increases and MscL is in the closed expanded substate;

(iii) $t_{SO1}, t_{SO2}, t_{SO3}, t_{SO4} \in (10, 13)$, when the channel is partly open (solutes and water pass from the internal region to the external medium) and shows a flickering through subconducting states;

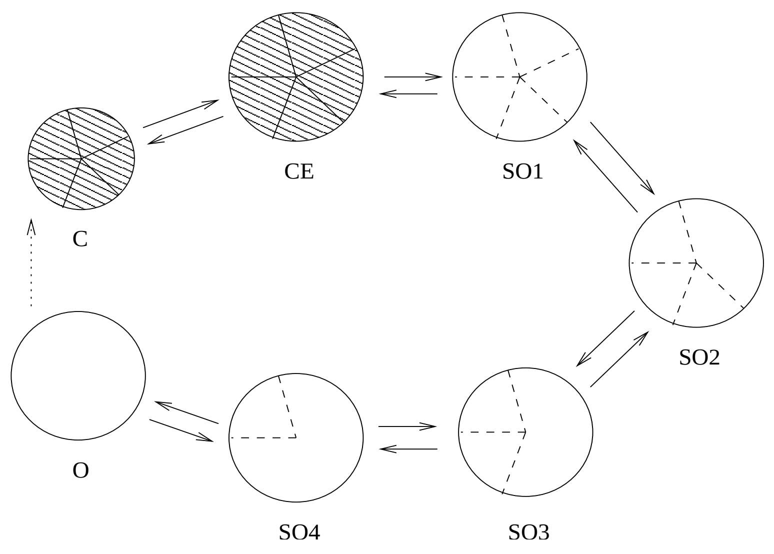


Figure 1: Transitions in MscL: from the closed to the fully open conformation, via subconducting states.

(*iv*) $t_O = 13$, when MscL is fully open, chemicals and water continue to pass from the internal region to the external medium;

(*v*) $t_L \geqslant 14$, when the applied suction is so high to cause the membrane lysis.

Similarly, we can consider the following conductivity values of the subconducting and open states:

- the conductivity of the subconducting state **SO1** is $0.25 \cdot 3.5$nS, that is 0.875nS;
- the conductivity of the subconducting state **SO2** is $0.56 \cdot 3.5$nS, that is 1.96nS;
- the conductivity of the subconducting state **SO3** is $0.74 \cdot 3.5$nS, that is 2.59nS;
- the conductivity of the subconducting state **SO4** is $0.89 \cdot 3.5$nS, that is 3.115nS;
- the conductivity of the subconducting state **O** is 3.5nS.

2.2 A P system model for mechanosensitive channels

As said before, MscL act as transmembrane mechanoelectrical switches, opening in response to lipid bilayer stretch and deformations and converting a mechanical stress of the membrane into gating transitions. The channel open probability, as well as the dynamic of close–to–open transitions, are functions of the membrane tension, an essential parameter described by means of a variable label attached to the membrane [2]. Hence, the evolution rules not only intervene in the transformation and communication of objects, but also in the modification of the label, which is to be interpreted as a key descriptor of the channel status. This is a new interpretation of the membrane label, which becomes a fundamental component of the system used to describe a biological significant counterpart (the status of the channel, in this case), and not just an identifier of a membrane in the membrane structure.

The definition of the variable membrane parameter (the *tension*) in the *in vitro* model is based on the real data reported in Section 2.1. The tension label assumes real positive values in the finite set of labels $Tension = \{t_C, t_{CE}, t_{SO1}, t_{SO2}, t_{SO3}, t_{SO4}, t_O, t_L\}$.

The solutions inside and outside the cell are described by considering an external *environment* (in short, *Env*) and an inner *region* (in short, *Reg*): the environment is made of solutes (symbols from a given alphabet V_{chem}) and water molecules (each denoted by a symbol $w \notin V_{chem}$); the internal region consists of objects over the same alphabet of the environment, and we assume that no other processes take place inside the cell. The semi-bracket notation is used to denote the membrane (labelled with the tension parameter $t \in Tension$) which separates the external environment and the internal region, that is: $Env\ [_t\ Reg$.

Transitions among tension values are due to the changes in the pressure applied to the patch membrane and simulated in P systems by means of a new type of evolution rules. Namely, an *in vitro environmental rule* describes a change in the pressure parameter p due to external actions, which can happen at any time in the environment and cannot be controlled by any component of the system. We write $\langle p, apply\rangle [_t \longrightarrow_{prob} [_{t'}$ for some $p \in \mathbf{R}, t, t' \in Tension, prob \in [0,1] \subset \mathbf{R}$, to denote any environmental rule which introduces the action of the parameter p, has consequences on the membrane tension value and is applied according to the associated probability value *prob*.

We give an example of *in vitro* evolution rules and their meaning:

$$\langle p, apply\rangle [_{t_C} \longrightarrow_{prob=0.01} [_{t_{CE}} \quad \text{for some } p \ll 40$$

$$\langle p, apply\rangle [_{t_C} \longrightarrow_{prob=0.99} [_{t_C} \quad \text{for some } p \ll 40$$

If the membrane tension is equal to t_C and the applied pressure has a value $p \ll 40$mmHg, then the conformation of the MscL is more likely to remain unchanged (second rule) because the applied suction is not enough to trigger the channel activation; though, we also model the passage to the expanded state with a very low probability (first rule).

The complete set of evolution rules and corresponding explanation can be found in [2]. Moreover, there are reported the simulations *in silico* performed by means of the complex systems simulators *EdnaCo* [5]. The observed quantities emerging from the simulations (that is, they were not explicitly programmed in the simulation) are the tension, the conductance and the current) . Obtained results appear to be in line with the general biological phenomena and thus offer biologists a challenge to verify results by actual laboratory experiments. See [2] for output pictures of simulations and for further details.

Moreover, note that the P model presented in [2] was constructed for a single MscL, anyway it can be considered consistent with the analysis of a population of mechanosensitive channels. In this case, it suffices to consider the same model for many channels, but using different sets of probability values associated to rules, as well as, possibly, to different membrane tension values.

2.3 Towards a fuzzy P model for mechanosensitive channels

Thanks to the promising results of simulation *in silico*, and in order to give a more realistic and fine description of the functioning of MscL, we propose to extend the *in vitro* P model by including some fuzzy tools. From Section 2.1 we know real values or interval of values of the parameter tension (and conductivity) when the channel is closed, closed expanded, and so on, and in Section 2.2 we have seen how to attach a label t to the membrane to denote the current state of the channel. The values of t in the set $Tension$ have been considered consistent with the real values of membrane tension measured in dyne/cm, but no continuous transition has been assumed among channel conformation. More precisely, one takes care only of discrete time steps, hence the channel state is initially closed (with membrane label t_C and some real value of t_C in $[0, 10)$) and, at the next step, it might be closed expanded (with membrane label t_{CE} and value equal to 10). Actually, in the cell there is a gradual transition from one conformation of the channel to the next one, hence it might be better (and closer to reality) to study these situations together with the introduction of *membership functions*, which describe the parameters tension and conductivity. Two possible sets of membership functions for the case study are depicted in Figure 2 and 3.

For simplicity, here we used triangular and trapezoid membership functions. In general, a membership function is a function $\mu_F : U \to [0, 1]$ that assigns to every $u \in U$ a degree of membership $\mu_F(u) \in [0, 1]$ to F, where U is a universe of objects and F is a fuzzy set. The latter is completely determined by the set of tuples $F = \{(u, \mu_F(u)) \mid u \in U\}$. Like a crisp set, a fuzzy set can be used to describe the value of a variable. In fuzzy set theory, the variable is linguistic and its values are described both qualitatively by a linguistic term (the symbols $C, CE, SO1, SO2, \ldots$) and quantitatively by the corresponding membership function. This knowledge representation is used in fuzzy rule-based inference. In order to define this technique and to relate it with a P model, we need to introduce the following definitions.

An *atomic fuzzy proposition* has the form

$$X \text{ is } F$$

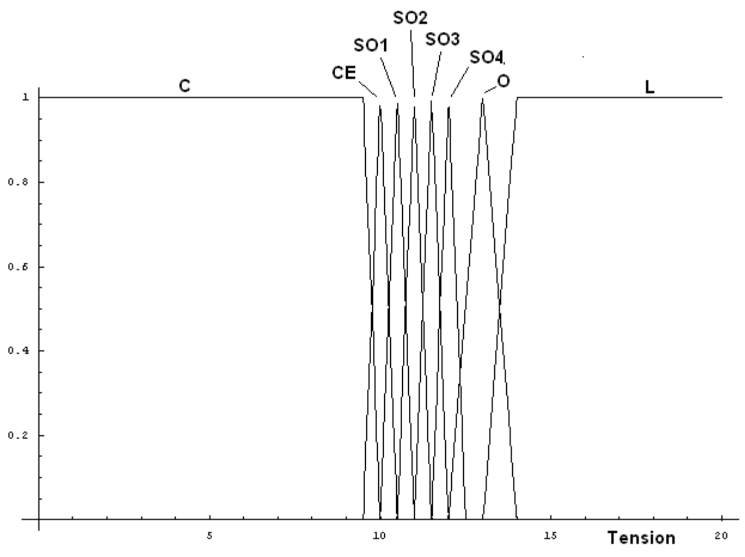


Figure 2: An example of membership functions for the tension.

where X is the linguistic variable, 'is' stands for 'has the property of being' and F is a fuzzy set that describes a property. Based on this definitions and the use of *linguistic connectives* as 'and', 'or' and 'not' one can construct more complex fuzzy propositions.

A *fuzzy rule* is symbolically expressed as

if <fuzzy proposition> then <fuzzy proposition>.

The first proposition (the antecedent) describes an observed condition, while the second one (the consequent) describes a conclusion that depends on the antecedent. An example of fuzzy rule is:

if X is H then Y is A.

The basic steps of fuzzy rule-based inference are the following (see [13, 4] for more details):

1. Fuzzy matching: Calculate the degree to which the input data match the condition of the fuzzy rules.
2. Rule base processing: Calculate the rule's conclusion base on its matching degree and combination in a final conclusion.

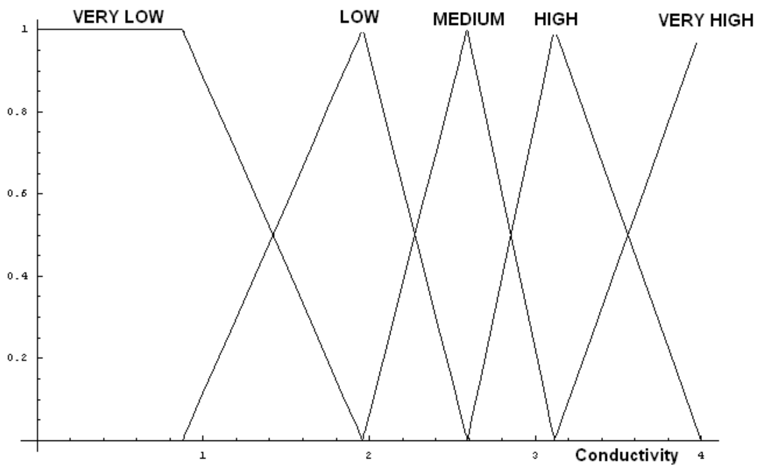


Figure 3: An example of membership functions for the conductivity.

3. Defuzzification: For applications that need a crisp output this step converts a fuzzy conclusion in a crisp one.

Usually, in the theory of fuzzy control, given a set of input and output linguistic variables we obtain a fuzzy set as the result of the inference (that is, the conclusion). In the case of mechanosensitive channels, the tension can be considered as an input linguistic variable, while the conductivity can be considered as an output variable. Moreover, another expected output should be the control action that triggers the application of evolution rules defining the P model for MscL. Hence we are looking for an higher level of description: how to integrate the standard fuzzy approach with P systems?

3 The case of membrane transport proteins

The phospholipid bilayer of cellular membranes is essentially impermeable to most water-soluble molecules and ions. Hence, the passage across membranes of many biochemical substances (amino acids, glucide, ions) has to be mediated by transmembrane proteins, which are usually selective with respect to the transported substances. On the contrary, there exist other substances (gases and small uncharged molecules) which can directly cross the phospholipid bilayer by passive diffusion, down their concentration gradients. In this section we are only interested in transmembrane proteins.

The three major types of transport proteins are *ATP-powered pumps*, *ion channels* and *transporters* (uniporter, symporter, antiporter), which all exhibit a high specificity for the transported substances and differs with respect to the rate of transport and to the mechanism of action. ATP-powered pumps use the energy of ATP hydrolysis to move ions or small molecules against a chemical concentration gradient or electrical potential (the process is known also as active transport). Ion channels simultaneously transport multiple water molecules or many (specific) ions down their concentration or electric potential gradients, at a very rapid rate. Some of them are usually open (for instance, the potassium-specific channel), others are usually closed and open only in response to specific signals. In contrast, transporters bind only one (or a few) molecules at the same time, then a conformational change of the protein allows the transport of such molecules across the membrane. Among transporters, uniporters move one molecule at a time down its concentration gradient, while symporters and antiporters couple the passage of one type of molecule (or ion) against its concentration gradient to the passage of a different type of molecule (or ion) down its concentration gradient. For more notions about membrane transport proteins the reader can consult [8, 2].

In all cases of transport proteins, the rate and extent of ion transport is influenced by the ion concentrations on the external and internal sides of the membrane, and by the electric potential that exists across the membrane, as well as by the biological structure and chemical properties of the proteins. On the other side, the ionic gradients and electric potential across the membrane drive many biological processes. For instance, the conduction of an electric impulse down the axon of neurons is mediated by opening and closing of sodium, potassium and calcium channels; in most cells, an increase in cytosolic calcium concentration is a fundamental regulatory signal, while sodium concentration gradient power the uptake of amino acids and other molecules.

From these considerations it is clear how important would be, from the biological point of view, an investigation of the global behavior of a population of transport proteins and the dynamics of transported molecules. For instance, consider the possibility of analyzing a population of channels of the same type (e.g., a population of sodium-potassium ATPases) or of different types, possibly "competing" for the transported molecules or ions (e.g., a population of calcium ATPases, calcium-sodium antiporters, sodium-potassium ATPases, potassium channels). The natural phases of investigation for this problem consist in first defining a good model for each transport protein of interest, and then designing a software simulator which allows the study of flux dynamics of transported molecules. This kind of study is consistent with the guidelines of Systems Biology [6, 58].

In this work we are mainly interested in finding a good framework for modelling biological structures and processes. As reported in Section 2, P systems have been proved valid for the modelling and the simulation of a very particular type of transmembrane channel, and our aim in Section 2.3 consisted in extending the known P model with fuzzy tools and techniques, in order to gain the highest resemblance to reality. In the case of transport proteins, whose functionality depends on concentration gradients and voltage, a fundamental aspect of the biological reality has to considered: the *"locality"*, which stands for the local physical conditions and for the notion of nearness. Indeed, in a cell the concentration of ions or molecules is not uniformly distributed all over the cellular membrane, but there can exist small local zones with a higher concentration with respect to others with a lower concentration. Thus, some elements in the population might be more active, others working at lower rates, others even resting, according to (1) the position of each (type of) transport protein in such areas (characterized by high or low concentration gradients and voltage), (2) the respective position of the surrounding transport proteins (of the same or different type) and (3) the distance existing between each transport protein and the protein-specific molecules to be transported.

To define appropriate models for simulating the behavior of channels under different local conditions, it might be useful to approach the analysis with the help of fuzzy techniques, in the same direction of Section 2.3.

References

[1] B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, P. Walter. *Molecular biology of the cell.*, 4th edition, Garland Science, New York, 2002.

[2] I.I. Ardelean, D. Besozzi, M.H. Garzon, G. Mauri, S. Roy, P system models for mechanosensitive channels, submitted, 2004.

[3] D. Besozzi, G. Ciobanu, A P system description of the sodium-potassium pump, submitted, 2004.

[4] D. Driankov, H. Hellendoorn and M. Reinfrank. *An Introduction to Fuzzy Control.* Springer-Verlag, Berlin, 1993.

[5] M. Garzon, D. Blain, A. Neel, Virtual Test Tubes for Biomolecular Computing *Journal of Natural Computing*, 3:4 (2004), in press.

[6] T. Ideker, T. Galitski, L. Hood, A new approach to decoding life: systems biology, *Annual Reviews Genomics Hum. Genet.*, 2, 2001, 343–372.

[7] H. Kitano, Systems biology: a brief overview, *Science*, 295, 2002, 1662–1664.

[8] H. Lodish, A. Berk, S. L. Zipursky, P. Matsudaira, D. Baltimore, J. Darnell. *Molecular cell biology*, 4th edition, W.H. Freeman and Co., New York, 2000.

[9] Gh. Păun, Computing with membranes, *Journal of Computer and System Sciences*, 61(1), 2000, 108–143 (See also *Turku Center for Computer Science-TUCS Report* No 208, 1998, www.tucs.fi).

[10] Gh. Păun, *Membrane Computing. An Introduction*, Springer-Verlag, Berlin, 2002.

[11] S.I. Sukharev, M. Betanzos, C.S. Chiang, H.R. Guy, The Gating Mechanism of the Large Mechanosensitive Channel MscL, *Nature*, 409, 720–724, 2001.

[12] S.I. Sukharev, W.J. Sigurdson, C. Kung, F. Sachs, Energetic and Spatial Parameters for Gating of the Bacterial Large Conductance Mechanosensitive Channel, Mscl, *J. Gen. Physiol.*, 1999, 113, 525–539.

[13] J. Yen and R. Langari. *Fuzzy Logic. Intelligence, Control, and Information.* Prentice-Hall, Upper Saddle River, 1999.

Metabolic algorithms and signal transduction dynamical networks (Abstract)

Luca Bianco, Vincenzo Manca

University of Verona
Department of Computer Science
strada Le Grazie, 15
37134 Verona, Italy
E-mail: {`bianco@sci., vincenzo.manca@`}`univr.it`

The *Group for Models of Natural Computing* (MNC group) in Verona developed *Psim*, a simulator of P systems. It's based on the implementation of the *metabolic algorithm* which is developed in Java and proves to be a cross platform application. In the first part of the talk we describe the algorithm and some de tails of the java simulation engine.

Then we focus our attention on the importance of *reactivity coefficients* as means to regulate the policy of rules' application and give an example of the algorithm application to the famous Belousov-Zhabotinsky reaction.

The crucial importance of reactivity coefficients in the metabolic algorithm lead us to the formulation of the *inverse oscillation problem* and we propose a strategy of resolution of this problem, based on cycling dynamics.

We conclude the discussion by outlining one of the problems we are actually facing: the simulation of *signal transduction networks*. This kind of networks describe protein-protein interactions, whose importance is crucial in the understanding of all regulating mechanisms of living cells (and in general of living organisms). From a mathematical viewpoint they turn out to be representable as graphs. Every node contains a certain amount of one actor of the regulating network (e.g. proteins, substrates, protein complexes). Arcs represent the possible set of interactions between contiguous nodes (e.g. activation, inhibition, production, degradation).

Two main kinds of investigations could be carried out on signal transduction networks: a *statical* one, whose aim is to measure some topological parameters of the network; a *dynamical* one, whose purpose is to calculate the behaviour of the system as time elapses, according to some initial conditions.

Due to the very high number of nodes and arcs constituting those networks, the development of reliable simulation tools for the investigation of their dynamic is very important (networks composed by thousands of nodes and arcs are frequent in biological interesting mechanisms).

A fuzzy approach to membrane computing with approximate copies

Jaume Casasnovas, Manuel Moyà, Joe Miró, Francesc Rosselló

Department of Mathematics and Computer Science,
University of the Balearic Islands,
07122 Palma de Mallorca (Spain)
E-mail: {`jaume.casasnovas,cesc.rossello`}`@uib.es`

1 Introduction

Membrane computing is a formal computational paradigm, invented in 1998 by Gh. Păun [5], that rewrites multisets of objects within a spatial structure inspired by the membrane structure of living cells and according to evolution rules that are reminiscent of the processes that take place inside cells. In this paper we use techniques based on fuzzy sets to develop a general membrane computing model that takes into account the imperfection of the reactives involved in computations. I.e., the fact that the actual objects used in computations, as well as the actual output of the latter, need not be exact copies of the reactives that are assumed to be used in the computations or to be produced by them but only approximate copies. This is a generalization of our previous work [1]. Other fuzzy approaches to membrane computing have been proposed in [3, 4, 6].

This work has been partially supported by the Spanish DGES, project BFM2003-00771.

2 The model

Given an alphabet V, we denote by V^* the set of words over V. Given a word $\underline{u} \in V^*$, we denote by $|\underline{u}|$ the length of $\underline{u}$ and, given a letter $v \in V$, by $|\underline{u}|_v$ the number of occurrences of v in w.

A *fuzzy subset* of a set X is a mapping from X to the unit interval $I = [0, 1]$. Such a fuzzy subset *finite-valued* when its image is a finite subset of I. For every fuzzy subset $\varphi : X \to I$, its *t-cut*, for every $t \in [0, 1]$, is

$$\varphi_t = \{x \in X \mid \varphi(x) \geqslant t\}.$$

Roughly described, a fuzzy P-system will be a structure similar to a crisp P-system, supported on a membrane structure that defines regions whose contents evolve following rules that created, destroy and move reactives. Now, we shall use *reactives* as "ideal definitions" of chemical compounds, and hence they are fuzzy subsets of X: for every reactive $v : X \to [0,1]$, we understand that $v(x) = t$ denotes that the object $x \in X$ is a copy of $v \in V$ with a degree t of exactitude. So, $v(x) = 1$ means that x is an *exact copy* of the reactive v, and $v(x) = 0$ means that x cannot represent in any way the reactive v. We shall say that an object $x \in X$ is *similar* to a reactive $v \in V$ when $v(x) > 0$ and we shall assume in this paper that each object in X is similar to at most one reactive.

As in the crisp case, fuzzy P-systems will be supported by a membrane structure. Recall that a *membrane structure* μ is a finite rooted tree whose nodes are called *membranes*. We shall always denote by M the set of membranes of a membrane structure. This tree represents a hierarchical structure of nested membranes, with the edges representing the relation 'being directly inside.' The tree's root 1 is then called the *skin membrane*. We expand every membrane structure μ by adding a new node to it labelled *env* and an arc going from 1 to *env*; let $\overline{\mu}$ denote the resulting tree and $\overline{M}$ its set of nodes $M \cup \{env\}$. This new node *env* is called the *environment*, because it surrounds the skin membrane.

We understand that every $m \in \overline{M}$ defines a *region* K_m. At each moment, every such region contains a set of objects. Now, he reactives being fuzzy sets, the content of these regions at each moment will be formally described by means of an $\overline{M}$-indexed family of *fuzzy multisets* over a set V of reactives. These fuzzy multisets specify, for every $v \in V$ and for every value $t \in]0,1]$, how many objects in each region K_m are copies of the reactive v with degree of accuracy t. They are *ac-fuzzy multisets* in the sense of [2].

A *configuration* for this fuzzy P-system, with set of membranes M and set of reactives V, will be a family of fuzzy multisets $(F_m)_{m\in\overline{M}}$ over V,

$$F_m : V \times I^+ \to \mathbb{N}_\infty, \qquad m \in \overline{M}.$$

Each such mapping F_m specifies, for every $v \in V$ and for every $t \in I^+$, how many objects exist in the region K_m such that $v(x) = t$ in a configuration.

Now, *fuzzy P system* is a structure

$$\Pi = (V, \mu, m_{out}, (S_m)_{m\in M}, (\mathcal{R}_m)_{m\in M})$$

where:

- V is the finite set of *reactives* used by the membrane system.
- μ is a membrane structure, with set of membranes M.
- $m_{out} \in M$ is the *output membrane*.
- $(S_m)_{m\in\overline{M}}$ is a family of finite fuzzy multisets over V, called the *initial configuration*, which describes the initial content of all regions K_m.

- For every membrane $m \in M$, $\mathcal{R}_m$ is a finite set of *evolution rules* associated to the membrane m.

  Each evolution rule in $\mathcal{R}_m$ has the form

  $$R = ((\underline{c}; \underline{a} \to \underline{(b,m)}), \tau, \phi),$$

  where:

  — $\underline{c} \in V^*$ represents the *catalysts* necessary for the reaction represented by the rule to take place; the (possibly inexact copies of these) catalysts used to trigger the application of this rule are not modified in any way by this application.

  — $\underline{a} \in V^*$ represents the reactives that are processed by the reaction represented by the rule; we shall call them the *active reactives* of this rule. These reactives are *spent*, destroyed, when the rule is applied.

  For simplicity, we assume that, for every $v \in V$, if $|\underline{c}|_v > 0$, then $|\underline{a}|_v = 0$: i.e., any reactive that is a catalyst of a rule cannot be an active reactive of this rule —although it could be an active reactive for some other rule in the same region or in another one.

  — $\underline{(b,m)} \in (V \times (M \cup \{env\}))^*$ represents the reactives that are produced by the reaction represented by the rule, together with the region where each one of them is placed: every symbol (b', m') in this word means that a (possibly inexact copy of a) new reactive b' is produced in the region $C_{m'}$ when the reaction represented by this rule takes place.

  We assume that, for every symbol (b', m') appearing in the word $\underline{(b,m)}$, the membrane m' is adjacent to m in the expanded tree $(M, E)^{(env)}$, i.e., m' is directly included in m or m is directly included in m'. If $m' = env$, so that $m = 1$, the object represented by b' is moved to the environment, leaving the membrane system and never coming back (notice that no rule is defined in the environment).

  — $\tau : V \to [0,1]$ is a *threshold* function that determines the degree of similarity to every reactive appearing in $\underline{c}$ or $\underline{a}$ necessary for an object to be considered as such a reactive to the effect of triggering an application of this rule.

  We impose on τ the condition that $\tau(v) > 0$ for every reactive that is a catalyst or an active reactive of this rule. On the other hand, for simplicity, we do not impose any threshold on the reactives that are not either catalysts or active in a given rule: i.e., we assume that, if $v \in V$ is such that $|\underline{c}|_v = 0$ and $|\underline{a}|_v = 0$, then $\tau(v) = 0$.

  — $\phi :]0,1]^{|\underline{c}|} \times]0,1]^{|\underline{a}|} \to]0,1]$ is a function that determines the value of similarity of all objects produced by the reaction to the reactives supposed to be obtained (as specified by the word $\underline{(b,m)}$) in terms of the similarity of the actual objects used in it to the catalysts and active reactives of the rule.

Given a configuration $(F_m)_{m\in M}$,for every $m \in M$ we shall denote by $F_m[v]$ the set of those values $t \in]0,1]$ such that $F_m(v,t) > 0$: these are non-zero degrees of exactitude of the copies of v that exist in C_m in the moment described by the configuration.

An evolution rule

$$R = (\underline{c}; \underline{a} \to \underline{(b,m)}, \tau, \phi)$$

in $\mathcal{R}_{m_0}$ *can be triggered* in a configuration $(F_m)_{m\in M}$ when, for every $v \in V$,

$$\sum_{t \geqslant \tau_{in}(v)} F_{\varepsilon(m_0)}(v,t) \geqslant |\underline{c} \cdot \underline{a}|_v.$$

This means that there are more copies in K_{m_0} within the degree of accuracy required by the threshold functions, than the specified quantities.

When a rule

$$R = (\underline{c}; \underline{a} \to \underline{(b,m)}, \phi, \tau)$$

in $\mathcal{R}_{m_0}$ can be triggered in a configuration $(F_m)_{m\in M}$, an *application* of it modifies this configuration into a new configuration $(F'_m)_{m\in M}$, which we call the *result* of this specific application. This new configuration is obtained in the following way. To simplify the notations, let

$$\begin{array}{rl} K(R) & = \{v \in V \mid |\underline{c}|_v > 0\} \\ A(R) & = \{v \in V \mid |\underline{a}|_v > 0\} \\ B_{m'}(R) & = \{v \in V \mid |\underline{(b,m)}|_{(v,m')} > 0\}, \quad m' \in M; \end{array}$$

recall that, by assumption, $K(R) \cap A(R) = \emptyset$. For every $v \in K(R) \cup A(R)$, let

$$\ell(v) = |\underline{c}|_v + |\underline{a}|_v$$

and for every $m' \in M$ and for every $v \in B_{m'}(R)$, let

$$r_m(v) = |\underline{(b,m)}|_{(v,m)}.$$

Now, $(F'_m)_{m\in M}$ is obtained by performing the following steps:

(1) For every $v \in K(R) \cup A(R)$, which are the reactives that are either catalysts or active for R, we choose $\ell(v)$ objects in C_{m_0} with degree of similarity with v at least $\tau(v)$. Formally, to do it, for every $v \in K(R) \cup A(R)$, if

$$F_m[v] \cap [\tau(v), 1] = \{t_{v,1}, \ldots, t_{v,h_v}\}, \qquad \text{with } t_{v,1} < \cdots < t_{v,h_v},$$

then we form a vector

$$\iota(v) = (\overbrace{t_{v,1}, \ldots, t_{v,1}}^{p_{v,1}}, \ldots \overbrace{t_{v,h_v}, \ldots, t_{v,h_v}}^{p_{v,h_v}})$$

such that $0 \leqslant p_{v,j} \leqslant F_{m_0}(v, t_{v,j})$ for every $j = 1, \ldots, h_v$ and $\sum_{j=1}^{h_v} p_{v,j} = \ell(v)$.

This corresponds to choosing $p_{v,1}$ objects x such that $v(x) = t_{v,1}$, $p_{v,2}$ objects x such that $v(x) = t_{v,2}$, and so on, up to $\ell(v)$ objects. These objects (or rather, the number of objects within each degree of similarity with v) are chosen in a non-deterministic way: forming a different such vector would correspond to a different application of the rule and hence it could lead to a different result. Notice also that the actual objects are unrelevant, only their degree of similarity with v.

(2) We remove from C_{m_0} the chosen inexact copies of the active reactives of R. Formally, we define, for every $m \in M$, a mapping $\widetilde{F}_m : V\times]0,1] \to \mathbb{N}$ as follows: $\widetilde{F}_m = F_m$ for every $m \neq m_0$, and

- for every $v \in A(R)$,

$$\begin{aligned}&\widetilde{F}_{m_0}(v, t_{v,j}) = F_{m_0}(v, t_{v,j}) - p_{v,j} \text{ for every } t_{v,j} \in F_m[v] \cap [\tau(v), 1]\\ &\widetilde{F}_{m_0}(v, t) = F_{m_0}(v, t) \text{ if } t \notin F_m[v] \cap [\tau(v), 1]\}\end{aligned}$$

- for every $v \notin A(R)$, $\widetilde{F}_{m_0}(v, t) = F_{m_0}(v, t)$ for every $t \in]0, 1]$.

Notice in particular that $\widetilde{F}_{m_0}(v, -)$ is not modified for any catalyst of the rule. This corresponds to the fact that catalysts of a reaction are not modified by the reaction: they only must be there for the reaction to be triggered.

(3) Let

$$t_{app} = \Phi((\iota(v))_{v\in K(R)\cup A(R)}) \in]0, 1].$$

Notice that t_{app} depends on the rule (the mapping F) as well as on how much the chosen objects were similar to the necessary reactives. Now, to every region C_m with m adjacent to m_0 and and for every $v \in B_m$, we add $r_m(v)$ copies x of v with degree of exactitude t_{app}; notice that, by assumption, these objects have degree of similarity 0 with any other reactive $v' \neq v$.

Formally, this defines, for every $m \in M$, a mapping $F'_m : V\times]0, 1] \to \mathbb{N}$ as follows:

- for every $m' \in M$ and for every $v \in B_{m'}(R)$,

$$\begin{aligned}&F'_{m'}(v, t_{app}) = \widetilde{F}_{m'}(v, t_{app}) + r_{m'}(v)\\ &F'_{m'}(v, t) = \widetilde{F}_{m'}(v, t) \quad \text{ if } t \neq t_{app}\end{aligned}$$

- for every $m' \in M$ and for every $v \notin B_m(R)$, $F'm'(v, t) = \widetilde{F}_{m'}(v, t)$ for every $t \in [0, 1]$.

This last configuration $(F'_m)_{m\in M}$ is the result of this application of R.

Notice that a given rule may admit several applications to a given configuration, yielding different results, depending on the objects chosen in the first step. Notice moreover that the resulting objects have a non-zero similarity with the expected reactives that depends on the rule as well as on the input objects.

Now, as in the classical case, a *transition* for a membrane system Π consists of a maximal symultaneous application of rules: all steps (1) corresponding to rules being applied are perfomed symultaneously, then all steps (2) and finally all steps (3). The rules applied in a given transition are chosen in a non-deterministic way (or, in more involved models, in some regulated way), but so that for every m no further rule in $\mathcal{R}_m$ can be triggered simultaneously to them. In particular, a given rule can be triggered several times in the same transition, provided enough (inexact) copies of the catalysts and active reactives are available.

A finite sequence of transitions between configurations of a fuzzy P-system Π, starting with the initial configuration, is called a *computation* with respect to Π. A computation *halts* when it reaches a *halting configuration* where no rule can be triggered.

Given a halting computation C with halting configuration $(H(C)_m)_{m\in\overline{M}}$, the (crisp) multiset over I^+ *associated* to it is

$$\begin{array}{rccl} H_C: & I^+ & \to & \mathbb{N} \\ & t & \mapsto & \sum_{v\in V_{out}} H(C)_{m_{out}}(v,t) \end{array}$$

Thus, for every $t \in I^+$, $H_C(t)$ is the number of objects in the output region that, at the end of the computation, are copies of some output reactive with degree of exactitude t.

Then, the *output* of a halting computation C will be the fuzzy subset of $\mathbb{N}$

$$\begin{array}{rccl} Out_{\Pi,C}: & \mathbb{N} & \to & I \\ & n & \mapsto & \bigvee\{t \mid H_C(t) = n\} \end{array}$$

In words, $Out_{\Pi,C}(n)$ is the greatest degree of exactitude t in I for which, at the end of the computation C, there exist n objects in the output region that are copies of some output reactive with degree of exactitude t.

Finally, the fuzzy set of natural numbers *generated* by a fuzzy membrane system Π is the join of all the outputs of halting computations with respect to Π. This is the mapping $Gen_\Pi : \mathbb{N} \to I$ defined by

$$Gen_\Pi(n) = \bigvee_{C \text{ halting}} Out_{\Pi,C}(n), \qquad n \in \mathbb{N}.$$

Thus,

$$\begin{aligned} Gen_\Pi(n) &= \bigvee\Big\{\bigvee\{t \in I^+ \mid H_C(t) = n\} \mid C \text{ halting}\Big\} \\ &= \bigvee\{t \in I^+ \mid H_C(t) = n \text{ for some halting computation } C\}. \end{aligned}$$

Notice that, I being finite, this supremum is actually a maximum, and that if $H_C(t) \neq n$ for every halting computation C, then $Gen_\Pi(n) = \bigvee\emptyset = 0$.

Theorem 1 *A set of finite valued fuzzy natural numbers is r.e. if and only if it is generated by a fuzzy P system.*

The finite-valuedness of the fuzzy subsets of $\mathbb{N}$ generated by our fuzzy P-systems is due to the finiteness of the sets of rules and the initial configuration.

To end this paper, we would like to point out that, although formally correct, our specific approach has a drawback from the fuzzy mathematics point of view. The association to a multiset $H : I^+ \to \mathbb{N}$ of the fuzzy subset of $\mathbb{N}$

$$\begin{array}{rccl} \mathbb{C}(H): & \mathbb{N} & \to & I \\ & n & \mapsto & \bigvee\{t \mid H(t) = n\} \end{array}$$

that underlies our definition of the output of a halting computation with respect to a fuzzy P-system is not additive in any natural sense, and in particular it cannot be considered a fuzzy cardinality [2]. We have tried to use some specific simple fuzzy cardinalities in this step, and we have obtained that the resulting fuzzy P-systems did not generate all finite-valued recursively enumerable fuzzy subsets of $\mathbb{N}$, but we have not ruled out the possibility of using some other, cunningly chosen, fuzzy cardinality.

Our current research agenda includes this problem, as well as the problem of getting rid of the assumption used in this paper that an object can only be similar to one reactive.

References

[1] J. Casasnovas, J. Miró, M. Moyà, F. Rosselló, "An approach to membrane computing under inexactitude".

[2] J. Casasnovas, F. Rosselló, "Counting the contents of fuzzy membranes... and related problems." These proceedings.

[3] A. Obtulowicz, Mathematical models of uncertainty with a regard to membrane systems, *Brainstorming Week on Membrane Computing*, Tarragona, February 2003, TR 26/ 03, URV, 2003, 241–246, and *Natural Computing*, 2, 3 (2003), 251–263.

[4] A. Obtulowicz, General multi-fuzzy sets and fuzzy membrane syste ms, *Pre-proceedings of Fifth Workshop in Membrane Computing, WMC5*, Milano, Italy, 2004, 316–326.

[5] Gh. Păun, "Computing with membranes", *J. of Comp. and Syst. Sci.* **61** (2000) 108–143.

[6] A. Syropoulos, "Fuzzyfying P Systems", submitted (2003).

Counting the contents of fuzzy membranes... and related problems

Jaume Casasnovas, Francesc Rosselló

Department of Mathematics and Computer Science,
University of the Balearic Islands,
07122 Palma de Mallorca (Spain)
E-mail: {jaume.casasnovas,cesc.rossello}@uib.es

The content of a membrane in a configuration of an 'exact' P system is described by a multiset. Recall that a (crisp) *multiset* over a set of *types* X is simply a mapping $d : X \to \mathbb{N}$. The usual interpretation of a multiset $d : X \to \mathbb{N}$ is that it describes a set consisting of $d(x)$ "exact" copies of each type $x \in X$. In particular, it is assumed that the set described by the multiset does not contain any element that is not a copy of some $x \in X$, or rather that we do not care about these elements, and that an element of it cannot be a copy of two different types.

Now, the uncertainty in an 'inexact' P system may arise at the level of the (lack of) crispness of its membranes' contents, and this can be represented using fuzzy multisets of different kinds. For instance, we could understand that the objects are imperfect, approximate copies of the reactives purportedly involved in its reactions. This would lead us to multisets describing, for every reactive v and for every degree of approximation t, how many elements there are in the membrane that are approximate copies of v with (or within) degree of approximation t. We could also understand that our lack of knowledge of the system refers to the number of copies of the (now, exact) reactives in each membrane. This would lead us to multisets describing, for every reactive v and for every $n \in \mathbb{N}$, the degree of certainty of there being n copies of v in the membrane. And so on.

Even using these generalized kinds of multisets, the basic processes of P systems based on them would be still removing, creating and moving objects within the system, and the final result of a computation would be still obtained by counting (in some way) the objects in some membrane. This calls for the development and study of *cardinalities* to 'count' the kind of fuzzy multisets used in this context.

We consider here two types of cardinalities: *scalar*, assigning to each multiset a positive real number, and *fuzzy*, assigning to each multiset a *fuzzy natural* *number*, a fuzzy subset of $\mathbb{N}$ with certain properties. Both may have their interest in different types of P systems. Fuzzy membrane systems with scalar cardinalities would produce computable (in the membrane sense) subsets of $\mathbb{R}^+$, while fuzzy membrane systems using fuzzy cardinalities would produce computable sets of fuzzy natural numbers.

The results on scalar and crisp cardinalities of fuzzy multisets of the first type discussed above (fuzzy multisets of approximate copies) contained in Sections 4 and 6 of this note were proved in [3]. The rest of this note is devoted to discuss work currently in progress. Previous studies of fuzzy cardinalities of fuzzy multisets include [1, 2, 4, 5]

This work has been partially supported by the Spanish DGES, project BFM2003-00771.

1 Fuzzy natural numbers

A *generalized natural number* is a mapping $\nu : \mathbb{N} \to [0,1]$; the set of generalized natural numbers is, then, $[0,1]^{\mathbb{N}}$. The *support* of a generalized natural number $\nu : \mathbb{N} \to [0,1]$ is the set

$$Supp(\nu) = \{n \in \mathbb{N} \mid \nu(n) > 0\}.$$

We can include $\mathbb{N}$ into $[0,1]^{\mathbb{N}}$ in several ways. For instance:

- By associating to every $n \in \mathbb{N}$ the generalized natural number $\overline{n} : \mathbb{N} \to [0,1]$ defined by $\overline{n}(n) = 1$ and $\overline{n}(m) = 0$ for every $m \neq n$.
- By associating to every $n \in \mathbb{N}$ the generalized natural number $\widehat{n} : \mathbb{N} \to [0,1]$ defined by $\widehat{n}(m) = 1$ if $m \leqslant n$ and $\widehat{n}(m) = 0$ if $m > n$.
- By associating to every $n \in \mathbb{N}$ the generalized natural number $\widetilde{n} : \mathbb{N} \to [0,1]$ defined by $\widetilde{n}(m) = 0$ if $m < n$ and $\widetilde{n}(m) = 1$ if $m \geqslant n$.

Notice that $\overline{0} = \widehat{0} \neq \widetilde{0}$ and that $\overline{n} = \widehat{n} \wedge \widetilde{n}$.

It has been agreed that the 'sum' of generalized natural numbers corresponds to the following operation $\oplus$ on $[0,1]^{\mathbb{N}}$, called the *extended sum*: for every $\nu, \mu \in [0,1]^{\mathbb{N}}$,

$$(\nu \oplus \mu)(k) = \bigvee \{\nu(i) \wedge \mu(j) \mid i + j = k\} \text{ for every } k \in \mathbb{N}.$$

This extended sum of generalized natural numbers is associative, commutative, its neutral element is $\overline{0}$, and it extends the sum of natural numbers for each one of the embeddings described above. Moreover, the extended sum of two increasing (resp., decreasing) generalized natural numbers is again increasing (resp., decreasing).

We shall not use all generalized natural numbers but a certain subset of them.

A generalized natural number is *convex* when $\nu(k) \geqslant \nu(i) \wedge \nu(j)$ for every $i \leqslant k \leqslant j$. By a *fuzzy natural number* we shall understand a convex generalized natural number.

We shall say that a fuzzy natural number has a *summit* when it takes its greatest value in, and only in, an element $n_0 \in \mathbb{N}$, and that it has a *plateau* when it takes its greatest value in, and only in, all elements of an interval $\{n_0, n_0+1, \ldots, n_0+k\} \subseteq \mathbb{N}$. Every fuzzy natural number has a summit or a plateau, and it increases to the left of it and decreases to the right of it.

Every increasing or decreasing generalized natural number is convex, and the extended sum of two convex generalized natural numbers is again convex.

It would also be natural to impose the finiteness of the support of fuzzy natural numbers, which would entail that at some distance to the left and, specially, to the right of the summit of the generalized natural number, it is defined 0. We shall not do it here (mainly because the generalized natural numbers $\widetilde{n}$ do not have a finite support), although in some contexts this restriction appears in a natural way: see Section 4.

The extended sum of two fuzzy natural numbers (resp., with finite support) is again a fuzzy natural number (resp., with finite support).

We shall denote by $\overline{\mathbb{N}}$ the set of all fuzzy natural numbers. The embeddings $\mathbb{N} \hookrightarrow [0,1]^{\mathbb{N}}$ described above are embeddings $\mathbb{N} \hookrightarrow \overline{\mathbb{N}}$.

We shall use fuzzy natural numbers as models of ‘imprecisely known’ natural numbers, taking as ‘exact natural numbers’ the images of one of these embeddings $\mathbb{N} \hookrightarrow \overline{\mathbb{N}}$. So, for instance, our knowledge of a quantity that lies ‘around 5’ will be represented by a fuzzy natural number with a summit in 5, or a plateau around 5.

2 Fuzzy multisets of uncertain quantities

As we mentioned in the introduction, a natural definition of fuzzy multiset assigns to each element of a set of types an imprecisely known natural number. Since we are advocating here for the use of fuzzy natural numbers (actually, of some suitable subset of them; see the next section) as models of the latter, this leads us to the following definition.

Definition 1 *A* fuzzy multiset of uncertain quantities*, a* uq-fuzzy multiset *for short, over a set of* types X *is a mapping* $F : X \to \overline{\mathbb{N}}$*. Such a uq-fuzzy multiset is* finite *if its* support

$$Supp(F) = \{x \in X \mid F(x) \neq \overline{0}\}$$

is a finite subset of X*.*

For every uq-fuzzy multisets F, G over a set X, their *sum* is the uq-fuzzy multiset $A + B$ defined pointwise by

$$(A+B)(x) = A(x) \oplus B(x), \qquad \text{for every } x \in X.$$

A *scalar cardinality* of uq-multisets would measure their size by means of positive real numbers, assigning moreover to each crisp multiset its usual cardinal. Such a scalar cardinality could be obtained by taking any morphism of monoids

$$\alpha : \overline{\mathbb{N}} \to \mathbb{R}^+$$

that preserves the chosen embedding $\mathbb{N} \hookrightarrow \overline{\mathbb{N}}$, and then defining

$$Sc_\alpha(F) = \sum_{x \in Supp(F)} \alpha(F(x)).$$

Actually, if we define abstractly a *scalar cardinality* of uq-fuzzy multisets as a mapping that sends every uq-fuzzy multiset to a positive real number, preserves the sums and extends the usual cardinality of crisp multisets, then it is not difficult to prove that all such scalar cardinalities are obtained in this way.

Anyway, the natural definition of the cardinal $\mathbb{C}$ of a finite uq-fuzzy multiset F over a set of types X assigns to each one of them a fuzzy natural number:

$$\mathbb{C}(F) = \bigoplus_{x \in Supp(F)} F(x) \in \overline{\mathbb{N}}.$$

This also extends the usual cardinality of crisp multisets.[3]

The main problem with uq-fuzzy multisets comes from a handicap of fuzzy natural numbers. In membrane processes, we must be able to compare and to subtract multisets. Now, the natural definition of $F \leqslant G$ for two uq-fuzzy multisets F and G should be

$$F(x) \leqslant G(x) \text{ in } \overline{\mathbb{N}}, \text{ for every } x \in X,$$

And when $F \leqslant G$, the natural definition of their difference $G - F$ should be

$$(G - F)(x) = G(x) - F(x) \text{ in } \overline{\mathbb{N}}, \text{ for every } x \in X.$$

But, what are these order and subtraction in $\overline{\mathbb{N}}$?

3 Subtracting fuzzy natural numbers

As we see, the use of fuzzy natural numbers to describe our imprecise knowledge of the number of reactives in a membrane at a given moment of a process poses two problems: comparison and subtraction. Given two fuzzy natural numbers μ and ν, if μ is larger than ν, how can we subtract ν from μ, finding a fuzzy natural number $\mu - \nu$ such that

$$\nu \oplus (\mu - \nu) = \mu?$$

[3]It is not difficult to define abstractly *fuzzy cardinality* of uq-fuzzy multisets and to characterize them as we do it for the fuzzy cardinalities of another type of fuzzy multisets in Section 6. We shall not do it here.

Would it be uniquely determined?

And, actually, to begin with, what does 'larger' mean in $\overline{\mathbb{N}}$? There are some proposals in this connection [7]. All of them translate in some sense the intuitive idea that if $\nu \leqslant \mu$, then the 'increasing' and the 'decreasing' branches of ν should lie to the left of those of μ, respectively. But they do not yield a well-defined subtraction.

Then, if we want (and we want!) to describe uncertainly known quantities of reactives in a membrane as fuzzy natural numbers, we need to know how to subtract them.

First of all, notice that if the rules in the membrane system remove crisp quantities of reactives, then we only need to compare natural numbers, embedded in $\overline{\mathbb{N}}$ as we had decided to do it, with fuzzy natural numbers, and to subtract a natural number from a fuzzy natural number. Let's take a glance at the embeddings given a the beginning.

- Assume that we take the embedding $n \mapsto \overline{n}$. For every $\nu \in \overline{\mathbb{N}}$ and $n \in \mathbb{N}$, and for every $m \in \mathbb{N}$, we have that, if $\nu - \overline{n}$ is defined, then, for every $i = 0, \ldots, m$,

  $$(\nu - \overline{n})(i) \wedge \overline{n}(m-i) = \begin{cases} 0 & \text{if } m - i \neq n, \text{ i.e., if } i \neq m - n \\ (\nu - \overline{n})(i) & \text{if } m - i = n, \text{ i.e., if } i = m - n \end{cases}$$

  which implies that

  $$\bigvee_{i=0,\ldots,m} (\nu - \overline{n})(i) \wedge \overline{n}(m-i) = \begin{cases} 0 & \text{if } m < n \\ (\nu - \overline{n})(m-n) & \text{if } n \leqslant m \end{cases}$$

  In particular, if it has to be $\nu(m)$, we have that $\nu(m) = 0$ for every $m < n$. This leads us to the following definition-result:

  Proposition 1 *For every $\nu \in \overline{\mathbb{N}}$ and $n \in \mathbb{N}$, we define that*

  $$\overline{n} \leqslant \nu \text{ if and only if } \nu(m) = 0 \text{ for every } m < n.$$

  And if $\overline{n} \leqslant \nu$, then we define $\nu - \overline{n} \in \overline{\mathbb{N}}$ by $(\nu - \overline{n})(i) = \nu(n+i)$ for every $i \in \mathbb{N}$.

  With these definitions, if $\overline{n} \leqslant \nu$, then the fuzzy natural number $\nu - \overline{n}$ is the only one such that $\overline{n} \oplus (\nu - \overline{n}) = \nu$.

- Assume now that we take the embedding $n \mapsto \widehat{n}$. In this case more involved discussion proves the following result.

  Proposition 2 *For every $\nu \in \overline{\mathbb{N}}$ and $n \geqslant 1$, there exists some $\nu - \widetilde{n} \in \overline{\mathbb{N}}$ such that $\widetilde{n} \oplus (\nu - \widetilde{n}) = \nu$ if and only if ν has a plateau of at least $n + 1$ elements.*

  And when ν has such a plateau $\{n_0, \ldots, n_0 + k_0\}$, then taking $(\nu - \widetilde{n})(i) = \nu(i)$ for every $i < n_0 + k_0$ and $(\nu - \widetilde{n})(i) = \nu(i + n)$ for every $i \geqslant n_0 + k_0$ we obtain a fuzzy natural number such that $\widehat{n} \oplus (\nu - \widehat{n}) = \nu$, but not all of them.

Thus, we can define

$$\widehat{n} \leqslant \nu \text{ if and only if } \nu \text{ has a plateau of at least } n+1 \text{ elements,}$$

and the subtraction $\nu - \widehat{n}$ as in the last proposition.

Since the fuzzy natural numbers $\widehat{n}$ are decreasing, it is natural to use them when we only consider *decreasing* fuzzy natural numbers. For decreasing fuzzy numbers, the order defined above becomes

$$\widehat{n} \leqslant \nu \text{ if and only if } \nu(0) = \cdots = \nu(n)$$

and then, when $\widehat{n} \leqslant \nu$, taking $\nu - \widehat{n}$ defined by $(\nu - \widehat{n})(i) = \nu(i+n)$ for every $i \in \mathbb{N}$, yields a solution of $\widehat{n} \oplus (\nu - \widehat{n}) = \nu$.

- A similar situation happens with the embeddings $n \mapsto \widetilde{n}$. We leave the details to the reader.

In the general situation, if we want to remove uncertain quantities of reactives from a membrane where we have other uncertain quantities of reactives, we need to give some answer to the following question.

Open question. To identify a meaningful and general enough subset $\overline{\mathbb{N}}'$ of $\overline{\mathbb{N}}$ where a meaningful (possibly partial) order $\leqslant$ can be defined in such a way that, for every μ, ν belonging to this subset, if $\nu \leqslant \mu$, then there exists one distinguished element $\mu - \nu$ such that $\nu \oplus (\mu - \nu) = \mu$. Moreover, $\mathbb{N}$ should be embedded into $\overline{\mathbb{N}}'$ in some way.

This would mean, of course, that we would allow the "uncertain quantities of reactives" to lie only in $\overline{\mathbb{N}}'$, i.e., to define *uq-fuzzy multisets* as mappings $X \to \overline{\mathbb{N}}'$.

For instance, if we restrict ourselves to *decreasing fuzzy natural numbers*, then the solution of the general problem is the following. We define

$$\nu \leqslant \mu \text{ if and only if } |\nu^{-1}(t)| \leqslant |\mu^{-1}(t)|, \text{ for every } t \in]0,1],$$

and then, a subtraction satisfying the desired property can be defined as follows: if $\nu \leqslant \mu$ in this sense, then we consider the mapping $H(\mu,\nu) :]0,1] \to \mathbb{N}$ defined by

$$H(\mu,\nu)(t) = |\mu^{-1}(t)| - |\nu^{-1}(t)| \quad \text{for every } t \in]0,1],$$

and then

$$(\mu - \nu)(n) = \bigvee \{t \in [0,1] \mid \sum_{t' \geqslant t} H(\mu,\nu)(t') \geqslant n\}.$$

This is a decreasing fuzzy natural number such that $\mu \oplus (\nu - \mu) = \nu$, but it is the only one.

These order and subtraction were first described by A. Obtułowicz in [6], and can be obtained as a slight modification of a particular case of a general construction that we shall discuss in Section 5.

If $\nu = \widehat{n}$, for some n, then, with this order, $\widehat{n} \leqslant \mu$ if and only if $\mu(0) = \cdots = \mu(n) = 1$, and then the subtraction agrees with the one described above in the particular case of subtracting $\widehat{n}$ from decreasing fuzzy natural numbers.

If we restrict ourselves to *increasing fuzzy natural numbers*, then our problem also has a solution. The order and the corresponding subtraction can be obtained again as a particular case of the aforementioned general construction we shall give later.

In general, in Section 5 we shall show a method to produce families of fuzzy natural numbers where an order and a subtraction can be defined. We do not know whether some of them is natural enough to be used in practice: perhaps a nice subfamily of one of them will work. The interesting fact is that our families arise as another type of fuzzy cardinalities of multisets.

4 Fuzzy cardinalities of finite multisets on $]0,1]$.

A (*crisp*) *multiset* over $]0,1]$ is a mapping $A :]0,1] \to \mathbb{N}$. A multiset A over $]0,1]$ is *finite* if its *support*

$$Supp(A) = \{t \in]0,1] \mid A(t) > 0\}$$

is a finite subset of $]0,1]$. We shall denote the set of all finite multisets over $]0,1]$ by $FMS(]0,1])$, and by $\perp$ the *null multiset*, defined by $\perp(t) = 0$ for every $t \in]0,1]$.

For every $A, B \in FMS(]0,1])$, their *sum* $A + B$ is the multiset

$$(A + B)(t) = A(t) + B(t), \qquad \text{for every } t \in]0,1].$$

We shall denote by n/t the multiset sending t to n and every $t' \neq t$ to 0.

A fuzzy cardinality of a finite multiset A over $]0,1]$ is a fuzzy natural number that measures how many elements has A.

Definition 2 *A* fuzzy cardinality *on $FMS(]0,1])$ is a mapping $\mathbb{C} : FMS(]0,1]) \to \overline{\mathbb{N}}$ that satisfies the following conditions:*

(i) *For every $A, B \in FMS(]0,1])$, $\mathbb{C}(A + B) = \mathbb{C}(A) \oplus \mathbb{C}(B)$.*

(ii) *For every $A, B \in FMS(]0,1])$ and for every $i, j \in \mathbb{N}$ such that $i > \sum_{t \in Supp(A)}(A)$ and $j > \sum_{t \in Supp(A)}(B)$, $\mathbb{C}(A)(i) = \mathbb{C}(B)(j)$.*

(iii) *If $Supp(A) \subseteq \{1\}$, then $\mathbb{C}(A)(i) \in \{0,1\}$ for every $i \in \mathbb{N}$ and, moreover, if $n = A(1)$, then $\mathbb{C}(A)(n) = 1$.*

(iv) *If $t, t' \in]0,1]$ are such that $t \leqslant t'$, then*

$$\mathbb{C}(1/t)(0) \geqslant \mathbb{C}(1/t')(0) \quad \text{and} \quad \mathbb{C}(\perp)(1) \leqslant \mathbb{C}(1/t)(1) \leqslant \mathbb{C}(1/t')(1).$$

Let us explain the meaning as well as some motivations for each one of these conditions:

- Condition (i), *additivity*, generalizes to fuzzy natural numbers the additivity of the classical cardinal of a crisp multiset.
- Condition (ii) implements the idea that the elements t not belonging to the support of a finite multiset A should not affect the cardinality of A.
- Condition (iii) requires that, on each multiset of the form $n/1$, with $n \in \mathbb{N}$, any fuzzy cardinality must take values only in $\{0, 1\}$, and the value 1 on the specific number n. If in this property we restrict the type of cardinalities we accept for $n/1$, then the overall set of cardinalities is restricted.
- Condition (iv) captures the restriction that the value of the cardinality of singletons on 0 must decrease and their value on 1 must increase with the element of their support.

The bracket fuzzy cardinality defined in the next example will play a key role henceforth.

Example 3 *Let us consider the function*

$$\begin{array}{rccl} [\ ]: & FMS(]0,1]) & \rightarrow & [0,1]^{\mathbb{N}} \\ & A & \mapsto & [A] \end{array}$$

where, for every $A \in FMS(]0,1])$,

$$\begin{array}{rccl} [A]: & \mathbb{N} & \rightarrow & [0,1] \\ & i & \mapsto & [A]_i \end{array}$$

is defined by

$$[A]_i = \bigvee\{t \in [0,1] \mid \sum_{t' \geqslant t} A(t') \geqslant i\}.$$

It is clear that $[A]$ is decreasing and that $[A]_i = 0$ for every $i > \sum_{t \in Supp(A)}(A)$, and hence $[A] \in \overline{\mathbb{N}}$ for every $A \in FMS(]0,1])$. It turns out that this mapping $A \mapsto [A]$ is a fuzzy cardinality on $FMS(]0,1])$, which we shall call the bracket *cardinality.*

Definition 3 *Let $f : [0,1] \rightarrow [0,1]$ be an increasing mapping such that $f(0) \in \{0,1\}$ and $f(1) = 1$ and let $g : [0,1] \rightarrow [0,1]$ be a decreasing mapping such that $g(0) = 1$ and $g(1) \in \{0,1\}$.*

Let $\mathbb{C}_{f,g} : FMS(]0,1]) \rightarrow \overline{\mathbb{N}}$ be the mapping defined as follows: for every $A \in FMS(]0,1])$ and $i \in \mathbb{N}$,

$$\mathbb{C}_{f,g}(A)(i) = f([A]_i) \wedge g([A]_{i+1}).$$

The key theorem in this section is the following.

Theorem 4 *A mapping $\mathbb{C} : FMS(]0,1]) \to \overline{\mathbb{N}}$ is a fuzzy cardinality if and only if $\mathbb{C} = \mathbb{C}_{f,g}$ for some increasing mapping $f : [0,1] \to [0,1]$ such that $f(0) \in \{0,1\}$ and $f(1) = 1$ and some decreasing mapping $g : [0,1] \to [0,1]$ such that $g(0) = 1$ and $g(1) \in \{0,1\}$.*

The last theorem allows us to call the mapping $\mathbb{C}_{f,g}$, for every f, g as in Definition 3, the *fuzzy cardinality generated by f and g*. It provides an explicit description of all fuzzy cardinalities in terms of the bracket cardinality.

Proposition 5 *$\mathbb{C}_{f,g}(A)$ is increasing for every $A \in FMS(]0,1])$ if and only if f is the constant mapping 1, in which case $\mathbb{C}_{f,g}(A)(k) = g([A]_{k+1})$ for every $A \in FMS(]0,1])$ and $k \in \mathbb{N}$.*

Proposition 6 *$\mathbb{C}_{f,g}(A)$ is decreasing for every $A \in FMS(]0,1])$ if and only if g is the constant mapping 1, in which case $\mathbb{C}_{f,g}(A)(k) = f([A]_k)$ for every $A \in FMS(]0,1])$ and $k \in \mathbb{N}$.*

Since, for every f, g as in Definition 3 and, for every $A \in FMS(]0,1])$ and $k \in \mathbb{N}$,

$$\mathbb{C}_{f,g}(A)(k) = f([A]_k) \wedge g([A]_{k+1}),$$

we deduce the following result.

Corollary 7 *Every fuzzy cardinality on $FMS(]0,1])$ is the meet of an increasing fuzzy cardinality and a decreasing fuzzy cardinality: namely $\mathbb{C}_{f,g} = \mathbb{C}_{f,1} \wedge \mathbb{C}_{1,g}$.*

The equality in the last statement and the fact that, for every A, $\mathbb{C}_{f,1}(A)$ is decreasing and $\mathbb{C}_{1,g}(A)$ is increasing, easily entail that, in the non-trivial cases when neither f nor g are the constant mapping 1, there exists an $n_0 \in \mathbb{N}$ such that

$$\mathbb{C}_{f,g}(A)(i) = \begin{cases} \mathbb{C}_{1,g}(A)(i) & \text{if } i < n_0 \\ \mathbb{C}_{f,1}(A)(i) & \text{if } i \geqslant n_0 \end{cases}$$

These give the increasing and decreasing branches of $\mathbb{C}_{f,g}(A)$.

5 Subtracting fuzzy natural numbers revisited

Let $\leqslant$ denote the partial order on $FMS(]0,1])$ defined pointwise by

$$A \leqslant B \text{ if and only if } A(t) \leqslant B(t) \text{ for every } t \in]0,1].$$

If $A \leqslant B$, then their *difference* $B - A$ is the multiset defined pointwise by

$$(B - A)(t) = B(t) - A(t) \text{ for every } t \in]0,1].$$

Proposition 8 *Let $\mathbb{C}$ be a fuzzy cardinality on $FMS(]0,1])$. If $A, B \in FMS(]0,1])$ are such that $A \leqslant B$, then*

$$\mathbb{C}(A) \oplus \mathbb{C}(B - A) = \mathbb{C}(B).$$

This makes us return to the open question that we posed in Section 3. In view of Proposition 8, a possible answer to it would be to take, for any fuzzy cardinality $\mathbb{C}$ on $FMS(]0,1])$,

$$\mathbb{N}_{\mathbb{C}} = \{\mathbb{C}(A) \in \overline{\mathbb{N}} \mid A \in FMS(]0,1])\},$$

to define on this set the partial order

> $\nu \preccurlyeq \mu$ if and only if there exist $A, B \in FMS(]0,1])$ such that $\nu = \mathbb{C}(A)$, $\mu = \mathbb{C}(B)$ and $A \leqslant B$,

and then to define, for every $A, B \in FMS(]0,1])$ such that $A \leqslant B$,

$$\mathbb{C}(B) - \mathbb{C}(A) = \mathbb{C}(B - A).$$

This poses, of course, several technical questions. Is $\preccurlyeq$ a wel-defined partial order? Is the subtraction in $\mathbb{N}_{\mathbb{C}}$ well-defined, in the sense that if $A, A', B, B' \in FMS(]0,1])$ are such that $\mathbb{C}(A) = \mathbb{C}(A')$, $\mathbb{C}(B) = \mathbb{C}(B')$, $A \leqslant B$, and $A' \leqslant B'$, does it always happen that

$$\mathbb{C}(B' - A') = \mathbb{C}(B - A)?$$

We conjecture that the answer is in general positive, but we have not been able to prove it. Anyway, we have the following result.

Proposition 9 *Let f, g be mappings as in Definition 3. If f and g are injective, then, for every $A, B \in FMS(]0,1])$, if $\mathbb{C}_{f,g}(A) = \mathbb{C}_{f,g}(B)$, then $A = B$.*

Thus, if we restrict the set of cardinalities to those generated by *bijective* mappings f and g, then the answers are indeed positive (although we still do not know whether $\mathbb{C}(B - A)$ is the only fuzzy natural number whose extended sum with $\mathbb{C}(A)$ yields $\mathbb{C}(B)$).

The third question is the characterization of the sets $\mathbb{N}_{\mathbb{C}}$. In this connection, we have the following results.

Proposition 10 *For every $\nu \in \overline{\mathbb{N}}$ there always exist injective mappings f, g as in Definition 3 such that $\nu \in \overline{\mathbb{N}}_{\mathbb{C}_{f,g}}$. But, given $\mu, \nu \in \overline{\mathbb{N}}$, there need not exist a fuzzy cardinality $\mathbb{C}$ such that $\mu, \nu \in \overline{\mathbb{N}}_{\mathbb{C}}$.*

Theorem 11 *Let f, g be two bijective mappings as in Definition 3, and let $t_0 \in [0,1]$ be the only point where they cross, i.e., such that $f^{-1}(t_0) = g^{-1}(t_0)$.*

For every $\nu \in \overline{\mathbb{N}}$, we have that $\nu \in \overline{\mathbb{N}}_{\mathbb{C}_{f,g}}$ if and only if one of the following two conditions holds:

(a) $\nu(n) \leqslant t_0$ for every $n \in \mathbb{N}$, and ν has a plateau.

(b) There is only one point n_0 such that $\nu(n_0) > t_0$, and then the values of $\nu(n_0 - 1)$, $\nu(n_0)$ and $\nu(n_0 + 1)$ are linked through some specific conditions (namely, there exist $t_1 < t_0 < t_2$ such that $\nu(n_0 - 1) = g(t_2)$, $\nu(n_0 + 1) = f(t_1)$ and $\nu(n_0) = f(t_2) \wedge g(t_1)$).

Besides cardinalities $\mathbb{C}_{f,g}$ generated by bijective mappings, we could also take $\mathbb{C}$ to be the bracket cardinality, or the increasing cardinality $A \mapsto 1-[A]_{i+1}$. The first one yields all decreasing fuzzy multisets μ with finite support and $\mu(0) = 1$, while the second one yields all increasing fuzzy multisets with finite support. In this cases, the subtraction is also well defined through the construction provided above.

6 Fuzzy multisets of approximate copies

Let us consider now fuzzy multisets describing sets containing approximate copies of the types. In a first approach, by such a *fuzzy multiset* over a set of *types* X we would understand a mapping $\overline{A} : X \times [0,1] \to \mathbb{N}$. Such a fuzzy multiset would be understood to describe a set consisting of, for each $x \in X$ and for every $t \in [0,1]$, $\overline{A}(x,t)$ copies of x with degree of similarity t to it.

We shall impose two restrictions on this interpretation of a fuzzy multiset. First, the set described by the fuzzy multiset does not contain any element that is not a copy of some $x \in X$ with some non-negative degree of similarity —or rather, we do not care about them. This is a natural condition. Second, we assume that if an element of the set is an inexact copy of x with degree of similarity $t > 0$, then it cannot be an inexact copy of any other type in X with a non-negative degree of similarity. This is a strong condition, and we shall return on it at the end of this section. These two conditions entail that, for every $x \in X$, the value $\overline{A}(x,0)$ must be equal to $\sum_{w\in X-\{0\}} \sum_{t>0} \overline{A}(w,t)$ and in particular that the restriction of $\overline{A}$ to $X \times \{0\}$ is determined by the restriction of $\overline{A}$ to $X\times]0,1]$. This leads us finally to the following definition.

Definition 4 *A* fuzzy multiset of approximate copies*, an* ac-fuzzy multiset *for short, over a set X is a mapping $\overline{A} : X\times]0,1] \to \mathbb{N}$,, i.e., a mapping*

$$\overline{A} : X \times MS(]0,1]).$$

Such an ac-fuzzy multiset is finite *if its* support

$$Supp(\overline{A}) = \{x \in X \mid \overline{A}(x) \neq \perp\}$$

is a finite subset of X and, for every $x \in Supp(\overline{A})$, $\overline{A}(x) \in FMS(]0,1])$.

We shall denote the sets of all ac-fuzzy multisets and of all finite fuzzy multisets over X by $\mathcal{FMS}(X)$ and $\mathcal{FFMS}(X)$, respectively.

Given two ac-fuzzy multisets $\overline{A}, \overline{B}$ over X, their *sum* $\overline{A} + \overline{B}$ is the ac-fuzzy multiset over X defined pointwise by

$$(\overline{A} + \overline{B})(x) = \overline{A}(x) + \overline{B}(x) \text{ for every } x \in X.$$

For every $x \in X$ and $A \in MS(]0,1])$, we shall denote by A/x the fuzzy multiset over X defined by $(A/x)(x) = A$ and $(A/x)(y) = \perp$ for every $y \neq x$.

Notice that if A is finite, then A/x is also finite, and that, for every $\overline{A} \in \mathcal{FFMS}(X)$,

$$\overline{A} = \sum_{x \in Supp(\overline{A})} (\overline{A}(x))/x.$$

The partial order $\leqslant$ on $\mathcal{FMS}(X)$ is defined by

$$\overline{A} \leqslant \overline{B} \text{ if and only if } \overline{A}(x) \leqslant \overline{B}(x) \text{ for every } x \in X.$$

If $\overline{A} \leqslant \overline{B}$, then their *difference* $\overline{B} - \overline{A}$ is the fuzzy multiset defined pointwise by

$$(\overline{B} - \overline{A})(x) = \overline{B}(x) - \overline{A}(x) \text{ for every } x \in X.$$

The scalar cardinality of a finite fuzzy multiset $\overline{A}$ is a real number that measures the overall size of the set described by $\overline{A}$.

Definition 5 *A* scalar cardinality *on $\mathcal{FFMS}(X)$ is a mapping $Sc : \mathcal{FFMS}(X) \to \mathbb{R}^+$ that satisfies the following conditions:*

(i) $Sc(\overline{A} + \overline{B}) = Sc(\overline{A}) + Sc(\overline{B})$ for every $\overline{A}, \overline{B} \in \mathcal{FFMS}(X)$.

(ii) $Sc((1/1)/x) = 1$ for every $x \in X$.

A scalar cardinality Sc on $\mathcal{FFMS}(X)$ is homogeneous *when it satisfies the following extra property:*

(iii) $Sc(M/x) = Sc(M/y)$ for every $x, y \in X$ and $M \in FMS(]0,1])$.

Next proposition provides a description of all scalar cardinalities on $\mathcal{FFMS}(X)$.

Proposition 12 *A mapping $Sc : \mathcal{FFMS}(X) \to \mathbb{R}^+$ is a scalar cardinality if and only if for every $x \in X$ there exists a mapping $f_x :]0,1] \to \mathbb{R}^+$ with $f_x(1) = 1$, such that, for every fuzzy multiset $\overline{A}$ over X,*

$$Sc(\overline{A}) = \sum_{x \in X} \sum_{t \in Supp(\overline{A}(x))} f_x(t)\overline{A}(x)(t).$$

Moreover, the mappings $(f_x)_{x \in X}$ are uniquely determined by Sc, and Sc is homogeneous if and only if $f_x = f_y$ for every $x, y \in X$.

Now, a fuzzy cardinality of a fuzzy multiset measures the size of the set it describes by means of a fuzzy natural number.

Definition 6 *A* fuzzy cardinality *on $\mathcal{FFMS}(X)$ is a mapping $\mathbb{C} : \mathcal{FFMS}(X) \to \overline{\mathbb{N}}$ that satisfies the following conditions:*

(i) For every $\overline{A}, \overline{B} \in \mathcal{FFMS}(X)$, $\mathbb{C}(\overline{A} + \overline{B}) = \mathbb{C}(\overline{A}) \oplus \mathbb{C}(\overline{B})$.

(ii) For every $x \in X$, the mapping

$$\begin{array}{rccl} \mathbb{C}(\ /x): & FMS(]0,1]) & \to & \overline{\mathbb{N}} \\ & M & \mapsto & \mathbb{C}(M/x) \end{array}$$

is a fuzzy cardinality on $FM(]0,1])$

A fuzzy cardinality is homogeneous *when it satisfies the following further condition:*

(iii) For every $x, y \in X$, $\mathbb{C}(\ /x) = \mathbb{C}(\ /y)$.

Proposition 13 *A mapping $\mathbb{C} : \mathcal{FFMS}(X) \to \overline{\mathbb{N}}$ is a fuzzy cardinality if and only if for every $x \in X$ there exists an fuzzy cardinality $\mathbb{C}_x : FMS(]0,1]) \to \overline{\mathbb{N}}$ such that*

$$\mathbb{C}(\overline{M}) = \bigoplus_{x \in X} \mathbb{C}_x(\overline{M}(x)).$$

Moreover, the family $(\mathbb{C}_x)_{x \in X}$ is uniquely determined by $\mathbb{C}$, and $\mathbb{C}$ is homogeneous if and only if $\mathbb{C}_x = \mathbb{C}_y$ for every $x, y \in X$.

Thus, homogeneous scalar and fuzzy cardinalities understand fuzzy multisets as a sum of crisp multisets, one on every type $x \in X$, and "count" this sum. Arbitrary scalar and fuzzy cardinalities "count" each multiset on each $x \in X$, possibly using a different cardinality for every $x \in X$, and then add up these results.

Adding a fuzzy multiset to a fuzzy multiset corresponds to the extended sum of their cardinalities. As far a removing a fuzzy multiset from another fuzzy multiset, we still have the following result.

Corollary 14 *Let $\mathbb{C}$ be a fuzzy cardinality on $\mathcal{FFMS}(X)$. If $\overline{A}, \overline{B} \in \mathcal{FFMS}(X)$ are such that $A \leqslant B$, then*

$$\mathbb{C}(\overline{A}) \oplus \mathbb{C}(\overline{B} - \overline{A}) = \mathbb{C}(\overline{B}).$$

Therefore, the fuzzy cardinal of $\overline{B} - \overline{A}$ can be seen as the subtraction of the cardinal of $\overline{A}$ to that of $\overline{B}$. Notice anyway that now we are subtracting fuzzy multisets, and the subtraction of cardinals is a consequence of this operation.

We should remove our working hypothesis that an object can only be similar to only one reactive in order to cover more general situations, where objects can be similar to different reactives or even where reactives can be similar themselves. This would affect our constructions in two ways. The first one is that the values of a fuzzy multiset over a set X on elements of the form $(x, 0)$ would no longer be entailed by the rest of values, and thus multisets would have to be defined as mappings

$$F : X \to MS([0,1]).$$

This is conceptually easy, although technically involved, to cope with.

But if we remove this hypothesis, then the order for fuzzy multisets and their subtractions become something darker. It is not the same to have an object similar to x and to y than two objects, one similar to x and the other similar to y: in the first case, when we remove one single object we get the null multiset, in the second case, not.

Our results on cardinalities of ac-fuzzy multisets deal with abstract objects and therefore they are formally correct in this new setting, but they are not sound. For instance, Proposition 14 is a direct consequence of additivity, but it should not held in this setting. Thus, cardinals of these fuzzy multisets would have to be handled in a completely different way, and uncertain P systems with membranes' contents described by these multisets would be more difficult to define.

References

[1] P. Bosc et al, About difference operation on fuzzy bags. *Proceedings IPMU 2002*, 1541–1546.

[2] P. Bosc et al, About Zf, the Set of Fuzzy Relative Integers, and the Definition of Fuzzy Bags on Zf. *Proceedings IFSA2003*, 95–102.

[3] J. Casasnovas, F. Rosselló, Scalar and fuzzy cardinalities of crisp and fuzzy multisets. Submitted (2004).

[4] M. Delgado, D. Sanchez, M. J. Martín-Bautista, M.A. Vila, A probabilistic definition of a nonconvex fuzzy cardinality. *Fuzzy Sets and Systems* 126 (2002) 177–190.

[5] Delgado M. et al, On a Characterization of Fuzzy Bags. *Proceedings IFSA2003*, 119–126.

[6] A. Obtułowicz, General multi-fuzzy sets and fuzzy membrane systems. *Pre-proceedings of the WMC 2004*.

[7] M. Wygralak, *Vaguely defined objects, Representations, fuzzy sets and non-classical cardinality theory*. Kluwer Academic Press (1996).

Modelling Biological Processes in P Systems: Handling Imprecision and Constructing New Models
Abstract

Matteo Cavaliere

Research Group on Natural Computing
Department of Computer Science and Artificial Intelligence
University of Sevilla
Avda. Reina Mercedes s/n, 41012 Sevilla, Spain
martew@inwind.it

Membrane systems (P systems) are computational devices inspired from cell functioning, [8]. For this reason, it is important to introduce, in the P systems area, instruments to handle the imprecision that arises from (our) partial knowledge of many cellular phenomena.

We present two possible ways to tackle imprecision in P systems.

A first way consists in handling imprecision by using known mathematical devices (for instance, probability). This approach has been considered in [1, 2]. There has been shown how, using a model of P systems called *evolution-communication*, [3], enriched with probabilities, it is possible to simulate simple (and important) biological processes that occur in living cells. In particular, in [2] has been presented the modelling of respiration in escherichia coli and the modelling of respiration-photosynthesis interaction in cyanobacteria.

A second way to handle imprecision in P systems can be considered more "drastic": one tries to construct P systems that work independently from imprecision. This approach has been presented in [4], where time imprecision has been considered. In particular, in [4], have been introduced *timed P systems* and *time-free P systems.*

The motivation for these models comes from the following consideration: a standard feature of P systems is that each rule is executed in exactly one (clock) step; this mathematical assumption does not have a corresponding counterpart in cell biology, where different chemical reactions might take different times to be executed. Therefore, it is natural to consider a model of P systems (timed P systems) where to each rule is associated a certain time of execution.

On the other hand, we want to avoid "problems" that could derive from time imprecision; for this reason, it would be extremely useful to investigate P systems producing always the same result independently from the time of executions of the rules. In this respect, a special model of P systems, called *time-free*, has been introduced and investigate in [4].

Formally, a P system is called time-free when it produces always the same set of vectors of natural numbers independently from the time of executions of its rules. In this way, time-free P systems can be considered "safe" against time imprecision.

A third way to handle time imprecision would be to mix the first two approaches: one tries to construct time-free P systems that are "safe" against "controlled" imprecision. In this respect, in [4, 6] has been considered the class of *partially time-free P systems.*

Preliminary results concerning time-free P systems have been obtained in [4, 5, 6, 7]. There, several computational results have been obtained but many (interesting) open problems still need to be afforded.

References

[1] I.I. Ardelean, M. Cavaliere, Playing with a Probabilistic P System Simulator: Mathematical and Biological Problems, *First Brainstorming Week on Membrane Computing*, TR 26/03 URV, 2003, Tarragona, Spain, 37–45.

[2] I.I. Ardelean, M. Cavaliere, Modelling Biological Processes by Using a Probabilistic P System Software, *Natural Computing*, 2, 2, 2003, 173–197.

[3] M. Cavaliere, Evolution–Communication P Systems, *Membrane Computing* (Gh. Păun, G. Rozenberg, A. Salomaa, C. Zandron, eds.), LNCS 2597, Springer-Verlag, Berlin, 2003, 134–145.

[4] M. Cavaliere, Toward Asynchronous P Systems, *Pre-Proceeding of Fifth Workshop On Membrane Computing, WMC5*, Milano, Italy, 2004, 161–173.

[5] M. Cavaliere, V. Deufemia, On Time-Free P Systems, manuscript, 2004.

[6] M. Cavaliere, D. Sburlan, Time-Independent P Systems, *Membrane Computing. International Workshop WMC5*, Milano, 2004, LNCS, Springer-Verlag, to appear.

[7] M. Cavaliere, D. Sburlan, Time and Synchronization in Membrane Systems, submitted, 2004.

[8] Gh. Păun, *Membrane Computing – An Introduction.* Springer-Verlag, Berlin, 2002.

Quantum Energy–based P Systems

Alberto Leporati, Dario Pescini, Claudio Zandron

Dipartimento di Informatica, Sistemistica e Comunicazione
Università degli Studi di Milano – Bicocca
Via Bicocca degli Arcimboldi 8, 20126 Milano, Italy

e-mail: {leporati,pescini,zandron}@disco.unimib.it

Abstract

Energy–based P systems have been recently defined as P systems in which the amount of energy manipulated and/or consumed during computations is taken into account. In this paper we propose two quantum versions of energy–based P systems. Both versions are defined just like classical energy–based P systems, but for objects and rules. Objects are represented as pure states in the Hilbert space $\mathbb{C}^d$, whereas the definition of rules differs between the two models. In the former, rules are defined as bijective functions — implemented as unitary operators — which transform the objects from the alphabet. In the latter, rules are defined as generic functions which map the alphabet into itself. Such functions are implemented using a generalization of the Conditional Quantum Control technique, and may yield to non–unitary operators. Finally, we address some problems and outline some directions for future work.

1 Introduction

P systems (also called *membrane systems*) have been introduced in [14] as a new class of distributed and parallel computing devices, inspired by the structure and functioning of living cells. The basic model consists of a hierarchical structure composed by several membranes, embedded into a main membrane called the *skin*. Membranes divide the Euclidean space into *regions*, that contain some *objects* (represented by symbols of an alphabet) and *evolution rules*. Using these rules, the objects may evolve and/or move from a region to a neighboring one. The rules are applied in a nondeterministic and maximally parallel way: all the objects that may evolve are forced to evolve. A *computation* starts from an initial configuration of the system and terminates when no evolution rule can be applied. The result of a computation is the multiset of objects contained into an *output membrane* or emitted from the skin of the system.

In what follows we assume the reader is already familiar with the basic notions and the terminology underlying P systems. For a systematic introduction, we refer the reader to [15]. The latest information about P systems can be found in [18].

Energy–based P systems have been defined in [16] as P systems in which the amount of energy manipulated and/or consumed during computations is taken into account. A given amount of energy is associated to each object of the system. Moreover, instances of a special symbol e are used to denote free energy units occurring into the regions of the system. These energy units can be used to transform objects, using appropriate rules. The rules are defined according to conservativeness considerations. An object can always be transformed into another object having the same energy. On the other hand, if the transformed object has a different energy then the required (resp., exceeding) free energy units are taken from (resp., released to) the region where the rule is applied. We assume that the application of rules consumes no energy. This means, in particular, that objects can be moved (without altering them) between the regions of the system without energy consumption. A special case of energy–based P systems are *conservative* P systems, where the amount of energy entering the system with the input values is completely returned with the output values at the end of the computation.

Formally, an energy–based P system (of degree $m \geq 1$) is a construct

$$\Pi = (A, \varepsilon, \mu, e, w_1, \ldots, w_m, R_1, \ldots, R_m, i_{\text{in}}, i_{\text{out}})$$

where:

- A is an alphabet; its elements are called *objects*;
- $\varepsilon : A \to \mathbb{N}$ is a linear mapping that associates to each object $a \in A$ the value $\varepsilon(a)$ (also denoted by ε_a), which can be thought of as the "energy value of a". If $\varepsilon(a) = \ell$, we also say that object a *embeds* ℓ units of energy. Precisely, if $A = \{a_1, a_2, \ldots, a_d\}$ then for all $i \in \{1, 2, \ldots, d\}$ it holds $\varepsilon(a_i) = \varepsilon(a_1) + (i-1)\delta$ for an appropriate integer value $\delta > 0$. Hence, the energy values considered in the system are equispaced by the quantity δ. By adding "dummy" symbols into the alphabet (that is, symbols which never appear in the system during the computations), we can always assume $\delta = 1$ without loss of generality;
- μ is a hierarchical membrane structure consisting of m membranes. For the sake of clarity, we will label membranes with mnemonic identifiers which recall their function;
- $e \notin A$ is a special symbol that denotes one *free energy* unit, that is, one unit of energy which is not embedded into any object;
- w_i, for all $i \in \{1, \ldots, m\}$, specify the multiset (over $A \cup \{e\}$) of objects initially present in region i;

- R_i, for all $i \in \{1, \ldots, m\}$, is a finite set of evolution rules over A associated with region i. Only rules of the following types are allowed:

$$ae^k \to (b,p) \quad , \qquad a \to (b,p)e^k \quad , \qquad e \to (e,p)$$

  where $a, b \in A$, $p \in \{\text{here}, \text{in}(\textit{name}), \text{out}\}$ and k is a non negative integer;

- i_{in} is an integer between 1 and m and specifies the input membrane of Π;

- i_{out} is an integer between 0 and m and specifies the output membrane of Π. If $i_{\text{out}} = 0$ then the environment is used for the output, that is, the output value is the multiset of objects (over A) emitted from the skin.

A special attention is due to the definition of rules. The meaning of rule $ae^k \to (b,p)$, with $a, b \in A$, $p \in \{\text{here}, \text{in}(\textit{name}), \text{out}\}$, and k a positive integer number, is the following: the object a, in presence of k free energy units, is allowed to be transformed into object b. If $p = \text{here}$ then the new object b remains in the same region; if $p = \text{out}$ then b exits from the current membrane. Finally, if $p = \text{in}(\textit{name})$ then b enters into the membrane labelled with *name*, which must be a child of the current membrane in the membrane hierarchy.

The meaning of rule $a \to (b,p)e^k$, when k is a positive integer number, is analogous. The object a is allowed to be transformed into object b by releasing k units of free energy. As above, the new object b may optionally move one level up or down into the membrane hierarchy. The k free energy units can now be used by another rule to produce "more energetic" objects from "less energetic" ones.

When $k = 0$ the rule $ae^k \to (b,p)$ is written as $a \to (a,p)$, and simply moves (if $p \neq \text{here}$) the object a upward or downward into the membrane hierarchy, without acquiring nor releasing any free energy unit. Analogously, rules $e \to (e,p)$ simply move (if $p \neq \text{here}$) one unit of free energy upward or downward into the membrane hierarchy.

A further constraint is that each rule must be "conservative", in the sense that the amount of energy occurring on the left side of the rule must be the same as the amount of energy which occurs on the right side.

With a little abuse of notation, when the pair (x,p), with $x \in A \cup \{e\}$ and $p \in \{\text{here}, \text{in}(\textit{name}), \text{out}\}$, appears into a rule we will write x_p. Also, if $p = \text{in}(\textit{name})$ and no confusion arises we will usually write just the name of the membrane. Moreover, instead of writing e^k we will sometimes explicitly write k instances of e. It is also understood that the position of e^k (that is, on the left or on the right of the symbol from A) either into the left or into the right side of a rule is uninfluent. Finally, when the position p of an object which occurs in the right side of a rule is "here" we will omit to write it.

A *configuration* of Π is the tuple $(M_1, \ldots, M_m)$ of multisets (over $A \cup \{e\}$) of objects contained in each region of the system. $(w_1, \ldots, w_m)$ is called the *initial configuration*. For two configurations $(M_1, \ldots, M_m)$, $(M'_1, \ldots, M'_m)$ of Π we write $(M_1, \ldots, M_m) \Rightarrow (M'_1, \ldots, M'_m)$ to denote a *transition* from $(M_1, \ldots, M_m)$ to $(M'_1, \ldots, M'_m)$. The reflexive and transitive closure of $\Rightarrow$ is denoted by $\Rightarrow^*$. A *final configuration* is a configuration where no rule can be applied.

A *computation* is a sequence of transitions between configurations of Π, starting from the initial configuration. A computation is *successful* if and only if it reaches a final configuration or, in other words, it *halts*. It is understood that the multiset (over A, that is, not considering free energy units) of objects which occur in $w_{i_{\text{in}}}$ are the *input values* for the computation. Analogously, the multiset (over A) of objects occurring in the output membrane (or emitted from the skin if $i_{\text{out}} = 0$) in the final configuration is the *output* of the computation. A non–halting computation produces no output.

Since energy is an additive quantity, it is natural to define the *energy of a multiset* as the sum of the amounts of energy associated to each instance of the objects which occur into the multiset. Analogously, the energy of a configuration is the sum of the amounts of energy associated to each multiset which occurs into the configuration. A *conservative computation* is a computation where each configuration has the same amount of energy. A *conservative energy–based P system* is an energy–based P system that performs only conservative computations.

Energy–based P systems are by no means the first model of P systems which involve energy. We recall in particular [1, 8, 17, 9]. In [12] it is shown that energy–based P systems are able to simulate the Fredkin gate. By allowing different objects of the alphabet to embed the same amount of energy, in [13] the simulation is extended to reversible Fredkin circuits. Moreover it is shown that the simulating P systems can be made *self–reversible* (that is, able to perform both "forward" and "backward" computations) and conservative. This result shows that (non–uniform families of) energy–based P systems are able to perform universal computations.

2 Quantum versions of energy–based P systems

In this section we propose two quantum versions of energy–based P systems. From now on, quantum energy–based P systems will simply be called *quantum P systems* for short. Both versions are defined just like classical energy–based P systems, but for objects and rules. The objects are represented as pure states in the Hilbert space $\mathbb{C}^d$, whereas the definition of rules differs between the two models. In the former, rules are defined as bijective functions — implemented as unitary operators — which transform the objects from the alphabet. In the latter, rules are defined as generic functions which map the alphabet into itself. Such functions are implemented using a generalization of the Conditional Quantum Control technique, and may yield to non–unitary operators.

Before delving into the details of quantum P systems, let us recall some basic notions of quantum computing. From an abstract point of view a quantum computer can be considered as made up of interacting parts. The elementary units (memory cells) that compose these parts are two–levels quantum systems called *qubits*. A qubit is typically implemented using the energy levels of a two–levels atom, or the two spin states of a spin–$\frac{1}{2}$ atomic nucleus, or a polarization photon. The mathematical description — independent of the practical realization — of a single qubit is based on the two–dimensional complex Hilbert space $\mathbb{C}^2$. The Boolean truth values 0 and 1 are represented in this framework by the unit vectors of the canonical orthonormal basis, called the *computational basis* of $\mathbb{C}^2$:

$$|0\rangle = \begin{bmatrix}1\\0\end{bmatrix} \qquad\qquad |1\rangle = \begin{bmatrix}0\\1\end{bmatrix}$$

Qubits are thus the quantum extension of the classical notion of bit, but whereas bits can only take two different values, 0 and 1, qubits are not confined to their two basis (also *pure*) states, $|0\rangle$ and $|1\rangle$, but can also exist in states which are coherent superpositions such as $\psi = c_0 |0\rangle + c_1 |1\rangle$, where c_0 and c_1 are complex numbers satisfying the condition $|c_0|^2 + |c_1|^2 = 1$. A qubit in this state is not simply in state $|0\rangle$ or $|1\rangle$, nor is it in an intermediate state; rather the qubit is in both states *simultaneously* and a mere act of measurement alters this state. Indeed, performing a measurement on a qubit in the above superposition will return 0 with probability $|c_0|^2$ and 1 with probability $|c_1|^2$; the state of the qubit after the measurement (*post–measurement state*) will be $|0\rangle$ or $|1\rangle$, depending on the outcome.

Let us stress that in axiomatic quantum mechanics a pure state is described by a one–dimensional subspace of the involved Hilbert space, whose vectors are *representatives* of this state. Thus, two unit vectors $|\psi\rangle$ and $|\varphi\rangle$ describe (belong to) the same state if and only if they differ of a phase factor, that is, if and only if there exists a real value $\vartheta \in [0, 2\pi)$ such that $|\psi\rangle = e^{i\vartheta} |\varphi\rangle$.

A *quantum register* of size n (also called an *n–register*) is mathematically described by the Hilbert space $\otimes^n \mathbb{C}^2 = \underbrace{\mathbb{C}^2 \otimes \ldots \otimes \mathbb{C}^2}_{n \text{ times}}$, representing a set of n qubits labelled by the index $i \in \{1, \ldots, n\}$. An *n–configuration* (also *pattern*) is a vector $|x_1\rangle \otimes \ldots \otimes |x_n\rangle \in \otimes^n \mathbb{C}^2$, usually written as $|x_1, \ldots, x_n\rangle$, considered as a quantum realization of the Boolean tuple $(x_1, \ldots, x_n)$. Let us recall that the dimension of $\otimes^n \mathbb{C}^2$ is 2^n and that $\{|x_1, \ldots, x_n\rangle : x_i \in \{0, 1\}\}$ is an orthonormal basis of this space called the n–register *computational basis.*

Unlike the situation of the classical wired computer, where voltages of a wire go over voltages of another, in quantum computers something different happens. Each qubit of a given n–register is prepared in some particular pure state ($|0\rangle$ or $|1\rangle$) in order to realize the required n–configuration $|x_1, \ldots, x_n\rangle$, quantum realization of an input Boolean tuple of length n. Then, a linear operator $G : \otimes^n \mathbb{C}^2 \to \otimes^n \mathbb{C}^2$ is applied to the n–register. The application of G has the effect of transforming the n–configuration $|x_1, \ldots, x_n\rangle$ into a new n–configuration $G(|x_1, \ldots, x_n\rangle) = |y_1, \ldots, y_n\rangle$, which is the quantum realization of the output tuple of the computer. In other words, G transforms the vectors of the n–register computational basis into vectors of the same basis. Let us stress that in particular such operator G changes the state $|x_i\rangle$ (with $x_i \in \{0, 1\}$) of each qubit of the register into a new state $|y_i\rangle$ (with $y_i \in \{0, 1\}$) of the same qubit, and we interpret such modifications as a computation step performed by the quantum computer.

The action of the operator G on $\Phi = \sum c^{i_1 \cdots i_n} |x_{i_1}, \ldots, x_{i_n}\rangle$, expressed as a linear combination of the elements of the n–register basis, is obtained by linearity: $G(\Phi) = \sum c^{i_1 \cdots i_n} G(|x_{i_1}, \ldots, x_{i_n}\rangle)$. We recall that linear operators which act on n–registers can be represented as order 2^n square matrices of complex entries. Usually such operators, as well as the corresponding matrices, are required to be unitary. In particular, this implies that the implemented operations are logically reversible (an operation is *logically reversible* if its inputs can always be deduced from its outputs).

All these notions can be easily extended to quantum systems which have $d > 2$ pure states. In this setting, the d–valued versions of qubits are usually called *qudits* [10]. As it happens with qubits, a qudit is typically implemented using the energy levels of an atom or a nuclear spin. The mathematical description — independent of the practical realization — of a single qudit is based on the d–dimensional complex Hilbert space $\mathbb{C}^d$. In particular, the pure states $|0\rangle, \left|\frac{1}{d-1}\right\rangle, \left|\frac{2}{d-1}\right\rangle, \ldots, \left|\frac{d-2}{d-1}\right\rangle, |1\rangle$ are represented by the unit vectors of the canonical orthonormal basis, called the *computational basis* of $\mathbb{C}^d$:

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}, \quad \left|\frac{1}{d-1}\right\rangle = \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \\ 0 \end{bmatrix}, \quad \cdots, \quad \left|\frac{d-2}{d-1}\right\rangle = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \\ 0 \end{bmatrix}, \quad |1\rangle = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}$$

As before, a *quantum register* of size n can be defined as a collection of n qudits. It is mathematically described by the Hilbert space $\otimes^n \mathbb{C}^d = \underbrace{\mathbb{C}^d \otimes \ldots \otimes \mathbb{C}^d}_{n \text{ times}}$. An *$n$–configuration* is now a vector $|x_1\rangle \otimes \ldots \otimes |x_n\rangle \in \otimes^n \mathbb{C}^d$, simply written as $|x_1, \ldots, x_n\rangle$, for x_i running on $L_d = \left\{0, \frac{1}{d-1}, \frac{2}{d-1}, \ldots, \frac{d-2}{d-1}, 1\right\}$. An n–configuration can be viewed as the quantum realization of the "classical" tuple $(x_1, \ldots, x_n) \in L_d^n$. The dimension of $\otimes^n \mathbb{C}^d$ is d^n and the set $\{|x_1, \ldots, x_n\rangle : x_i \in L_d\}$ of all n–configurations is an orthonormal basis of this space, called the *n–register computational basis*. Notice that the set L_d can also be interpreted as a set of truth values, where 0 denotes falsity, 1 denotes truth and the other elements indicate different degrees of indefiniteness.

In our definition of quantum P systems, all the elements of the model (multisets, the membrane hierarchy, input and output membrane, configurations, computations, and so on) are defined just like the corresponding elements of a classical energy–based P system, but for objects and rules. The objects are represented by the pure states of a quantum system. Hence, if the alphabet contains $d \geq 2$ elements, then without loss of generality we can put $A = \left\{|0\rangle, \left|\frac{1}{d-1}\right\rangle, \left|\frac{2}{d-1}\right\rangle, \ldots, \left|\frac{d-2}{d-1}\right\rangle, |1\rangle\right\}$, that is, $A = \{|a\rangle : a \in L_d\}$. In a possible physical realization, we can think of a quantum system which is able to assume the above pure states. As stated above, such system will also be able to assume as a state any superposition of the kind:

$$c_0 |0\rangle + c_{\frac{1}{d-1}} \left|\frac{1}{d-1}\right\rangle + \ldots + c_{\frac{d-2}{d-1}} \left|\frac{d-2}{d-1}\right\rangle + c_1 |1\rangle$$

with $c_0, c_{\frac{1}{d-1}}, \ldots, c_{\frac{d-2}{d-1}}, c_1 \in \mathbb{C}$ such that $\sum_{i=0}^{d-1} \left|c_{\frac{i}{d-1}}\right|^2 = 1$. A multiset is simply a collection of quantum systems, each in its own state. In the most general setting, two or more quantum systems may become *entangled*, either because they are prepared in this way as input values for a computation, or because they are the result of the application of an operator on them. When two or more quantum systems are entangled, the state of each single system is tied to the state of the other systems. So, if we perform a measurement on a single system of an entangled pair, such operation will affect also the state of the other system. Formally, two or more quantum systems are entangled if their global state cannot be factorized as the tensor product of the states of the single systems. For example, $\frac{1}{\sqrt{2}}\left(|10\rangle - |01\rangle\right)$ is an entangled quantum state of the Hilbert space $\mathbb{C}^2 \otimes \mathbb{C}^2$.

Now let us turn to rules. As stated above, in this paper we propose two versions of quantum P systems. In the first version, rules are simply defined as unitary operators which transform the state of their input qudits. This is analogous to what happens with reversible logic gates, which act according to their truth table. If a rule acts on n quantum systems, we say that it computes an (n, d)–function, that is, a function $f : A^n \to A^n$. Notice that, since rules are defined by unitary operators, they are logically reversible. This means that f is a permutation on the set A^n.

As an example let us assume $d = 3$, so that $A = \left\{|0\rangle, \left|\frac{1}{2}\right\rangle, |1\rangle\right\}$. If we want to write the unitary operator which realizes the function $f : A \to A$ such that:

$$f(|0\rangle) = \left|\frac{1}{2}\right\rangle, \qquad f\left(\left|\frac{1}{2}\right\rangle\right) = |0\rangle, \qquad f(|1\rangle) = |1\rangle$$

we can operate as follows. The unitary matrix is an order $d^n = 3^1 = 3$ square matrix having complex entries. Each row and each column is associated with an element of A^n (in this case, A). If $f(|x_1, \ldots, x_n\rangle) = |y_1, \ldots, y_n\rangle$, then the element of the matrix whose row and column is associated with $|x_1, \ldots, x_n\rangle$ and $|y_1, \ldots, y_n\rangle$ respectively, is put to 1. All the other elements in the same row are put to 0. Since f is bijective, also all the other elements in the same column are put to 0. Continuing with the example, the unitary matrix which corresponds to the function $f : A \to A$ described above is:

$$U_f = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Of course there exist also *genuine quantum functions*, i.e., functions that have no classical counterpart and are thus characterized by the fact that some input pure states (tensor product of vectors of the computational basis of $\mathbb{C}^d$) are transformed into non–trivial superpositions of pure states. An example of an operation of this kind is the $\sqrt{\textsc{Not}}$ function, acting on the states of a single qubit, that can be thought of as the realization of a 1–register. Another genuine quantum gate is the *Hadamard* gate, also acting on quantum registers of size 1.

Formally, the map $H : \mathbb{C}^2 \to \mathbb{C}^2$ is described by the following order 2 unitary matrix:

$$H := \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

and the corresponding action on qubits is given by:

$$\begin{cases} H\,|0\rangle = \frac{1}{\sqrt{2}}\,|0\rangle + \frac{1}{\sqrt{2}}\,|1\rangle \\ H\,|1\rangle = \frac{1}{\sqrt{2}}\,|0\rangle - \frac{1}{\sqrt{2}}\,|1\rangle \end{cases}$$

Our second proposal for quantum P systems is more complicated, but is in some sense closer to physics laws. Objects are defined as in the first proposal, whereas the definition of rules is directly inspired from energy–based P systems. However, as we will see, differently from a classical energy–based P system, in a quantum P system free energy units are not just symbols which move into the system and cooperate with objects when the rules are applied. In quantum P systems, free energy units are true quanta of energy (for example, photons) which are necessary to transform a "low" energy state of a system to a higher energy state. On the other hand, when a quantum system decays to a lower energy state, it is understood that the difference of energy between the two states is released in the form of energy quanta (e.g., photons).

In order to become more formal, let us consider the set $\mathcal{E}_d = \left\{\varepsilon_0, \varepsilon_{\frac{1}{d-1}}, \varepsilon_{\frac{2}{d-1}}, \ldots, \varepsilon_{\frac{d-2}{d-1}}, \varepsilon_1\right\} \subseteq \mathbb{R}$ of real values; we can think to such quantities as energy values. To each element $v \in L_d$ (and hence to each object $|v\rangle \in A$) we associate the energy level ε_v; moreover, let us assume that the values of $\mathcal{E}_d$ are all positive, equispaced, and ordered according to the corresponding objects: $0 < \varepsilon_0 < \varepsilon_{\frac{1}{d-1}} < \cdots < \varepsilon_{\frac{d-2}{d-1}} < \varepsilon_1$. If we denote by $\Delta\varepsilon$ the gap between two adjacent energy levels then the following linear relation holds:

$$\varepsilon_v = \varepsilon_0 + \Delta\varepsilon\,(d-1)\,v \qquad\qquad \forall\, v \in L_d \tag{1}$$

Notice that it is not required that $\varepsilon_0 = \Delta\varepsilon$.

If $\underline{x} = |x_1, \ldots, x_n\rangle \in A^n$ is an n–configuration, we define the *amount of energy associated to* $\underline{x}$ as $E_n(\underline{x}) = \sum_{i=1}^{n} \varepsilon_{x_i}$, where $\varepsilon_{x_i} \in \mathcal{E}_d$ is the amount of energy associated to the i–th element $|x_i\rangle$ of the configuration. Let us remark that the map $E_n : L_d^n \to \mathbb{R}^+$ is indeed a family of mappings parameterized by n, the size of the input. We say that a rule is *conservative* if, for any input configuration $\underline{x} = |x_1, \ldots, x_n\rangle \in A^n$, the corresponding output configuration $\underline{y} = |y_1, \ldots, y_m\rangle \in A^m$ is such that $E_n(\underline{x}) = E_m(\underline{y})$. Rules are again defined as (n, d)–functions, that is, functions of the kind $f : A^n \to A^n$. The difference with respect to the first proposal is that such functions are not necessarily bijections on A^n: they can be arbitrary mappings. This means that the linear operators which realize such functions are not necessarily unitary. Hence, we need a method to build and describe them.

In this paper we present a quantum realization of rules using an extension of the *Conditional Quantum Control* technique introduced in [2]. The technique is used to write quantum operators which describe the behavior of classical rules. Such operators are sums of “local” operators, each of which is a tensor product of suitable compositions of the operators $a^\dagger$ and a, which are the finite dimensional versions of creation and annihilation operators usually found in quantum mechanics. An equivalent formulation is also given, using spin–rising (J_+) and spin–lowering (J_-) operators.

In the following sections we interpret the d energy levels of a quantum system by the truncated quantum harmonic oscillator. Moreover, we introduce the creation and annihilation operators on $\mathbb{C}^d$, and we show how they can be used to transform the state of a single quantum system, as required by the rules which occur in energy–based P systems. An alternative description is also given, using spin–rising and spin–lowering operators. Finally, we show how the linear operators which correspond to rules can be built, using both a “brute force” approach and an extension of the Conditional Quantum Control.

3 A mathematical model for quantum rules

3.1 The d–levels single system Hamiltonian

In describing a computation device it is important, from the point of view of quantum mechanics, to give the Hamiltonian operator for the physical system that constitutes the computing machinery. As it is well known, the Hamiltonian operator describes the energy of the quantum system and allows one to derive its time evolution.

The quantum realization of d–valued one–input/one–output rules can be done by considering single quantum systems whose Hamiltonian on $\mathbb{C}^d$ is:

$$H = \begin{bmatrix} \varepsilon_0 & 0 & \dots & 0 \\ 0 & \varepsilon_0 + \Delta\varepsilon & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \varepsilon_0 + (d-1)\Delta\varepsilon \end{bmatrix} \tag{2}$$

The energy eigenvalues $\varepsilon_k = \varepsilon_0 + k\Delta\varepsilon$ of H, starting from the ground energy state ε_0 and equispaced by the quantum of energy $\Delta\varepsilon$, are the ones of the infinite dimensional quantum harmonic oscillator truncated at the $(d-1)$-th excited level (see Figure 1).

The unit vector $|H = \varepsilon_k\rangle = \left|\frac{k}{d-1}\right\rangle$, for $k \in \{0, 1, \dots, d-1\}$, is the eigenvector of the state of energy $\varepsilon_0 + k\Delta\varepsilon$. The spectral resolution of the above truncated harmonic oscillator Hamiltonian (2) is:

$$H = \sum_{k=0}^{d-1} (\varepsilon_0 + k\Delta\varepsilon) P_{\varepsilon_k}$$

where each orthogonal projection $P_{\varepsilon_k} = P_{\frac{k}{d-1}}$ is the quantum realization of the sharp event “a measure of the system energy yields the value $\varepsilon_0 + k\Delta\varepsilon$”.

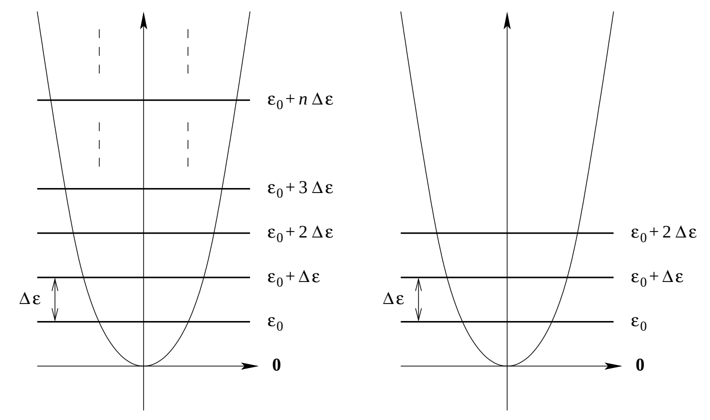


Figure 1: Energy levels of the infinite dimensional (on the left) and of the truncated (on the right) quantum harmonic oscillator

We can now introduce the creation and annihilation operators on the d–dimensional Hilbert spaces $\mathbb{C}^d$. Formally, *creation* and *annihilation* operators on the Hilbert space $\mathbb{C}^d$ are respectively defined as:

$$a^\dagger = \begin{bmatrix} 0 & 0 & \cdots & 0 & 0 \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & \sqrt{2} & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \sqrt{d-1} & 0 \end{bmatrix} \qquad a = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & \sqrt{2} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \sqrt{d-1} \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix}$$

The operators $a^\dagger$ and a are non–Hermitian, adjoints of each other, and satisfy the following commutation and anticommutation relations, respectively:

$$[a, a^\dagger] = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & \cdots & 0 & 1-d \end{bmatrix} \qquad [a, a^\dagger]_+ = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \\ 0 & 3 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & d-3 & 0 \\ 0 & 0 & \cdots & 0 & d-1 \end{bmatrix}$$

Thus, if one excludes the case $d = 2$ where the boson anticommutation rule is satisfied, neither the fermion commutation rule $[a, a^\dagger] = \mathrm{Id}$ nor the anticummutation rule $[a, a^\dagger]_+ = \mathrm{Id}$ of the infinite dimensional case hold.

From these formulas it follows that the action of $a^\dagger$ on the vectors of the canonical orthonormal basis of $\mathbb{C}^d$ is the following:

$$a^\dagger \left| \frac{k}{d-1} \right\rangle = \sqrt{k+1} \left| \frac{k+1}{d-1} \right\rangle \qquad \text{for } k \in \{0, 1, \ldots, d-2\}$$
$$a^\dagger \left| 1 \right\rangle = \mathbf{0}$$

whereas the action of a is:

$$a \left| \frac{k}{d-1} \right\rangle = \sqrt{k} \left| \frac{k-1}{d-1} \right\rangle \qquad \text{for } k \in \{1, 2, \ldots, d-1\}$$
$$a \left|0\right\rangle = \mathbf{0}$$

Using $a^\dagger$ and a we can introduce the following operators:

$$N = a^\dagger a = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & d-1 \end{bmatrix} \qquad aa^\dagger = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \\ 0 & 2 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & d-1 & 0 \\ 0 & 0 & \cdots & 0 & 0 \end{bmatrix}$$

The eigenvalues of the self–adjoint operator N are $0, 1, 2, \ldots, d-1$, and the eigenvector corresponding to the generic eigenvalue k is $\left|N = k\right\rangle = \left|\frac{k}{d-1}\right\rangle$. This corresponds to the notation adopted in [10], where the qudit base states are denoted by $\left|0\right\rangle, \left|1\right\rangle, \ldots, \left|d-1\right\rangle$, and it is assumed that a qudit can be in a superposition of the d base states:

$$c_0 \left|0\right\rangle + c_1 \left|1\right\rangle + \ldots + c_{d-1} \left|d-1\right\rangle$$

with $c_i \in \mathbb{C}$ for $i \in \{0, 1, \ldots, d-1\}$, and $|c_0|^2 + |c_1|^2 + \ldots + |c_{d-1}|^2 = 1$.

One possible physical interpretation of N is that it describes the *number of particles* of physical systems consisting of a maximum number of $d-1$ particles. In order to add a particle to the k particles state $\left|N = k\right\rangle$ (thus making it switch to the "next" state $\left|N = k+1\right\rangle$) we apply the creation operator $a^\dagger$, while to remove a particle from this system (thus making it switch to the "previous" state $\left|N = k-1\right\rangle$) we apply the annihilation operator a. Since the maximum number of particles that can be simultaneously in the system is $d-1$, the application of the creation operator to a full $d-1$ particles system does not have any effect on the system, and returns as a result the null vector. Analogously, the application of the annihilation operator to an empty particle system does not affect the system and returns the null vector as a result.

Another physical interpretation of operators $a^\dagger$ and a, by operator N, follows from the possibility of expressing the Hamiltonian (2) as follows:

$$H = \varepsilon_0 \,\mathrm{Id} + \Delta\varepsilon\, N = \varepsilon_0 \,\mathrm{Id} + \Delta\varepsilon\, a^\dagger a$$

In this case $a^\dagger$ (resp., a) realizes the transition from the eigenstate of energy $\varepsilon_k = \varepsilon_0 + k\,\Delta\varepsilon$ to the "next" (resp., "previous") eigenstate of energy $\varepsilon_{k+1} = \varepsilon_0 + (k+1)\,\Delta\varepsilon$ (resp., $\varepsilon_{k-1} = \varepsilon_0 + (k-1)\,\Delta\varepsilon$) for any $0 \le k < d-1$ (resp., $0 < k \le d-1$), while it collapses the last excited (resp., ground) state of energy $\varepsilon_0 + (d-1)\,\Delta\varepsilon$ (resp., ε_0) to the null vector.

The collection of all linear operators on $\mathbb{C}^d$ is a d^2–dimensional linear space whose canonical basis is:

$$\{E_{x,y} = \left|y\right\rangle \left\langle x\right| \;:\; x, y \in L_d\}$$

Since $E_{x,y}\,|x\rangle = |y\rangle$ and $E_{x,y}\,|z\rangle = \mathbf{0}$ for every $z \in L_d$ such that $z \neq x$, this operator transforms the unit vector $|x\rangle$ into the unit vector $|y\rangle$, collapsing all the other vectors of the canonical orthonormal basis of $\mathbb{C}^d$ into the null vector. For $i, j \in \{0, 1, \ldots, d-1\}$, the operator $E_{\frac{i}{d-1},\frac{j}{d-1}}$ can be represented as an order d square matrix having 1 in position $(j+1, i+1)$ and 0 in every other position:

$$E_{\frac{i}{d-1},\frac{j}{d-1}} = (\delta_{r,j+1}\delta_{i+1,s})_{r,s=1,2,\ldots,d}$$

Each of the operators $E_{x,y}$ can be expressed, using the whole algebraic structure of the associative algebra of operators, as a suitable composition of creation and annihilation operators.

We can use the whole algebraic structure (in particular, the composition operation) of the associative algebra of operators to express any such operator (i.e., any order d complex matrix) as a linear combination of suitable compositions of creations and annihilations. Precisely, if we denote by $A^{p,q,r}_{u,v}$ the expression

$$\underbrace{v \cdots v}_{r}\underbrace{v^* \cdots v^*}_{q}\underbrace{v \cdots v}_{p}u \tag{3}$$

where $u, v \in \{a^\dagger, a\}$, v^* is the adjoint of v, and p, q, r are non negative integer values, then for any $i, j \in \{0, 1, \ldots, d-1\}$ we can express the operator $E_{\frac{i}{d-1},\frac{j}{d-1}}$ in terms of creation and annihilation as follows:

$$E_{\frac{i}{d-1},\frac{j}{d-1}} = \begin{cases} \frac{\sqrt{j!}}{(d-1)!}A^{d-2,d-1-j,0}_{a^\dagger,a^\dagger} & \text{if } i = 0 \\ \frac{\sqrt{j!}}{(d-1)!}A^{d-1,d-1-j,0}_{a,a^\dagger} & \text{if } i = 1 \text{ and } j \geq 1 \\ \frac{\sqrt{i!}}{(d-1)!\sqrt{j!}}A^{d-2-i,d-1,j}_{a^\dagger,a^\dagger} & \text{if } (i = 1,\ j = 0 \text{ and } d \geq 3) \text{ or} \\ & \quad (1 < i < d-2 \text{ and } j \leq i) \\ \frac{\sqrt{j!}}{(d-1)!\sqrt{i!}}A^{i-1,d-1,d-1-j}_{a,a} & \text{if } (i = d-2,\ j = d-1 \text{ and } d \geq 3) \\ & \quad \text{or } (1 < i < d-2 \text{ and } j > i) \\ \frac{1}{\sqrt{(d-1)!j!(d-1)}}A^{d-1,j,0}_{a^\dagger,a} & \text{if } i = d-2 \text{ and } j \leq d-2 \\ \frac{1}{\sqrt{(d-1)!j!}}A^{d-2,j,0}_{a,a} & \text{if } i = d-1 \end{cases}$$

3.2 The angular momentum interpretation of qudits

As it is well known, for a fixed integer $d \geq 2$ the angular momentum based on the Hilbert space $\mathbb{C}^d$ consists of the triple of self–adjoint operators $\mathbf{J} = (J_x, J_y, J_z)$. Moreover, for $j = \frac{d-1}{2}$, the real value $j(j+1)$ is an eigenvalue of the operator $\mathbf{J}^2 = J_x^2 + J_y^2 + J_z^2$. The matrix representation of the z component of this angular momentum with respect to the orthonormal basis of its eigenvectors is:

$$J_z = \begin{bmatrix} \frac{d-1}{2} & 0 & \ldots & 0 & 0 \\ 0 & \frac{d-3}{2} & \ldots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \ldots & \frac{3-d}{2} & 0 \\ 0 & 0 & \ldots & 0 & \frac{1-d}{2} \end{bmatrix}$$

Thus, the z component of the angular momentum can assume d possible eigenvalues:

$$m = \frac{d-(2k+1)}{2} \qquad\qquad \text{for } k \in \{0,1,\dots,d-1\}$$

with corresponding eigenvectors:

$$\left| J_z = \frac{d-(2k+1)}{2} \right\rangle = \left| \frac{k}{d-1} \right\rangle \tag{4}$$

Let us consider the two operators J_+ and J_- on the Hilbert space $\mathbb{C}^d$ which are obtained from the general angular momentum operators as:

$$J_+ = J_x + iJ_y \qquad\qquad J_- = J_x - iJ_y$$

The operators J_+ and J_- are non–Hermitian, adjoints of each other, and satisfy the canonical commutation relation $[J_+, J_-] = 2J_z$. In matrix form they can be expressed as follows:

$$J_+ = \begin{bmatrix} 0 & \sqrt{d-1} & 0 & \cdots & 0 & 0 \\ 0 & 0 & \sqrt{2(d-2)} & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \sqrt{2(d-2)} & 0 \\ 0 & 0 & 0 & \cdots & 0 & \sqrt{d-1} \\ 0 & 0 & 0 & \cdots & 0 & 0 \end{bmatrix}$$

and

$$J_- = \begin{bmatrix} 0 & 0 & \cdots & 0 & 0 & 0 \\ \sqrt{d-1} & 0 & \cdots & 0 & 0 & 0 \\ 0 & \sqrt{2(d-2)} & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & \sqrt{2(d-2)} & 0 & 0 \\ 0 & 0 & \cdots & 0 & \sqrt{d-1} & 0 \end{bmatrix}$$

That is, for $r, s \in \{1,2,\dots,d\}$, the element in position (r,s) of matrices J_+ and J_- is, respectively:

$$(J_+)_{r,s} = \sqrt{r(d-r)}\delta_{r,s-1}$$
$$(J_-)_{r,s} = \sqrt{s(d-s)}\delta_{r,s+1}$$

As it is well known, the action of operators J_+ and J_- on the vectors of the orthonormal basis of $\mathbb{C}^d$ formed by the eigenvectors of J_z is the following:

$$J_+ \left| J_z = m \right\rangle = \sqrt{j(j+1) - m(m+1)} \left| J_z = m+1 \right\rangle \qquad \text{for } m = -j, \dots, j$$

and

$$J_- \left| J_z = m \right\rangle = \sqrt{j(j+1) - m(m-1)}\, \left| J_z = m-1 \right\rangle \qquad \text{for } m = -j, \ldots, j$$

Thus, we can interpret these operators as follows: the application of J_+ has the effect of changing the z component of the angular momentum to the next value. If applied to a system which has already a maximum value of J_z, J_+ leaves the system unchanged and returns as a result the null vector. Analogously, the application of J_- has the effect of switching the system to the previous value of the z component of the angular momentum. If applied to a system which has already a minimum value of J_z, J_- does not affect the system and returns as a result the null vector. Usually, J_+ and J_- are called the *spin–rising* and the *spin–lowering* operators, respectively.

The actions of J_+ and J_- on the vectors of the qudit orthonormal basis are the following:

$$J_+ \left| \frac{k}{d-1} \right\rangle = \sqrt{k(d-k)} \left| \frac{k-1}{d-1} \right\rangle \qquad \text{for } k \in \{1, 2 \ldots, d-1\}$$
$$J_+ \left| 0 \right\rangle = \mathbf{0}$$

and

$$J_- \left| \frac{k}{d-1} \right\rangle = \sqrt{(k+1)(d-(k+1))} \left| \frac{k+1}{d-1} \right\rangle \qquad \text{for } k \in \{0, 1, \ldots, d-2\}$$
$$J_- \left| 1 \right\rangle = \mathbf{0}$$

In particular, let us note that J_+ switches a qudit to the *previous* element in L_d, whereas J_- switches it to the *next* element. The effect of operator J_+ is depicted on the left side of Figure 2 for a spin–1 system on the Hilbert space $\mathbb{C}^3$. On the right side of the same figure the annihilation action of the same operator on a three–levels system is given for comparison with the previous behavior. A similar figure with respect to J_- can be drawn showing its spin–1 annihilation action with respect to the eigenstate creation behavior.

Let us note also that in the Boolean case (that is, when $d = 2$) it holds:

$$a^\dagger = J_- = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} \qquad \text{and} \qquad a = J_+ = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$$

Therefore it holds also $N = J_- J_+$ and $N' = J_+ J_-$, whereas in general, for $d > 2$, such equalities do not hold.

We conclude this section by presenting the expressions that allow one to obtain the operators $E_{\frac{i}{d-1}, \frac{j}{d-1}}$ in terms of spin–rising and spin–lowering. Let us consider the formal expression (3) applied to $u, v \in \{J_+, J_-\}$; moreover, let:

$$c_{r,s} = \frac{\prod_{k=r}^{s} \sqrt{k(d-k)}}{\prod_{k=1}^{d-1} k(d-k)}$$

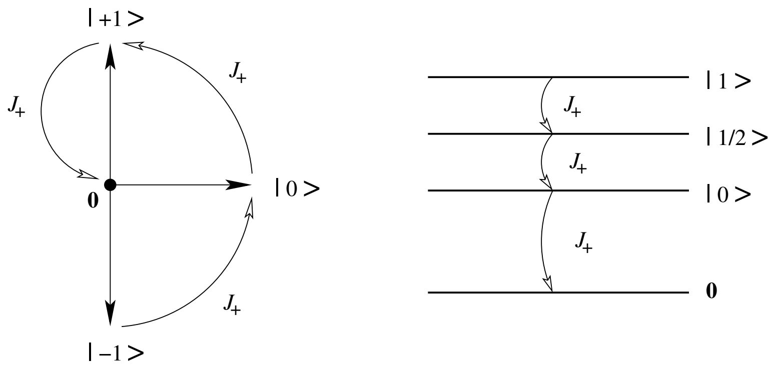


Figure 2: The effect of the spin–rising operator on a spin–1 system and the corresponding annihilation on three–level eigenstates

where s, r are two non negative integers. Then, for $i, j \in \{0, 1, \ldots, d-1\}$ it holds:

$$E_{\frac{i}{d-1},\frac{j}{d-1}} = \begin{cases} c_{1,j}\ A^{d-2,d-1-j,0}_{J_-,J_-} & \text{if } i = 0 \\ c_{2,j}\ A^{d-1,d-1-j,0}_{J_+,J_-} & \text{if } i = 1 \text{ and } j \geq 1 \\ c_{j+1,i}\ A^{d-2-i,d-1,j}_{J_-,J_-} & \text{if } (i = 1,\ j = 0 \text{ and } d \geq 3) \text{ or} \\ & \quad (1 < i < d-2 \text{ and } j \leq i) \\ c_{i+1,j}\ A^{i-1,d-1,d-1-j}_{J_+,J_+} & \text{if } (i = d-2,\ j = d-1 \text{ and } d \geq 3) \text{ or} \\ & \quad (1 < i < d-2 \text{ and } j > i) \\ c_{2,d-1-j}\ A^{d-1,j,0}_{J_-,J_+} & \text{if } i = d-2 \text{ and } j \leq d-2 \\ c_{1,d-1-j}\ A^{d-2,j,0}_{J_+,J_+} & \text{if } i = d-1 \end{cases}$$

4 Quantum realization of rules

Now that we have a mathematical model to interpret objects as vectors of the Hilbert space $\mathbb{C}^d$, and the quantum version of rules as linear operators $G : \otimes^n\mathbb{C}^d \to \otimes^n\mathbb{C}^d$ which implement (n, d)–functions, let us address the following problem.

Problem 1 *Given the truth table of an (n, d)–function $f : L^n_d \to L^n_d$, describe the linear operator $G_f : \otimes^n\mathbb{C}^d \to \otimes^n\mathbb{C}^d$ that provides a quantum realization of the rule as a formula containing only the linear operators* Id_2, *$a^\dagger$, a and the algebraic operations $+$, $-$, $\circ$, $\otimes$.* ■

If the (n, d)–function f is reversible then we already know how to build the corresponding (unitary) operator G_f. With the techniques we will introduce, it is also immediate to write a formula which describes G_f. However, we are interested to give a quantum description of *all* possible (n, d)–functions and thus, as we will see in the following, generally the operators we will obtain are not necessarily unitary.

In the next sections we expose two methods that can be used to build any (n, d)–function: the so called "brute force" method, and an extension of the Conditional Quantum Control method, originally proposed by Barenco, Deutsch, Ekert and Jozsa in [2].

4.1 The "brute force" method

We can write the global operator G_f as a sum of so called *local operators*, by a "brute force" procedure, where each local operator corresponds to a single row of the table which describes the (n, d)–function. Precisely, in order to translate the generic table row:

$$(x_1, x_2, \ldots, x_n) \mapsto (y_1, y_2, \ldots, y_n)$$

meaning that the input n–tuple $(x_1, x_2, \ldots, x_n)$ is transformed by the function into the output n–tuple $(y_1, y_2, \ldots, y_n)$, we build the "local" operator:

$$E_{x_1,y_1} \otimes E_{x_2,y_2} \otimes \cdots \otimes E_{x_n,y_n}$$

where $E_{x,y} := |y\rangle\langle x|$, with $x, y \in L_d$, is the operator that transforms the single qudit vector $|x\rangle$ into the vector $|y\rangle$, and returns the null vector if it is applied to any other vector of the computational basis of $\mathbb{C}^d$.

Hence, the operator $E_{x_1,y_1} \otimes E_{x_2,y_2} \otimes \cdots \otimes E_{x_n,y_n}$ transforms the input vector $|x_1, x_2, \ldots, x_n\rangle \in \otimes^n \mathbb{C}^d$ into the output vector $|y_1, y_2, \ldots, y_n\rangle \in \otimes^n \mathbb{C}^d$, whereas it collapses all the other input vectors of the n–register computational basis to the null vector.

Since each operator $E_{x,y}$ can be expressed as an appropriate composition of creation and annihilation (resp., spin–rising and spin–lowering) operators, we can conclude that every local operator is a tensor product of suitable compositions of creation and annihilation (resp., spin–rising and spin–lowering) operators.

4.2 The generalized "Conditional Quantum Control" method

Let us now introduce a method derived from *Conditional Quantum Control* [2].

The quantum realization of a "controlled behavior" can be obtained by making use of the operators $P_X = E_{X,X} = |X\rangle\langle X|$, for $X \in \Big\{0, \frac{1}{d-1}, \frac{2}{d-1}, \ldots, \frac{d-2}{d-1}, 1\Big\}$. For simplicity, let us first consider the case of a $(2, 2)$–function. For a reason that will be clear in a moment, we call *control qubit* and *target qubit* the first and the second input, respectively. If we want to realize a linear operator performing the condition: "if the control qubit is $|1\rangle$ then the operator O_1 is applied to the target qubit (and the control qubit is left unchanged)", then we can build the operator $N \otimes O_1$, where $N = E_{1,1} = |1\rangle\langle 1|$ checks for the condition "the control qubit is $|1\rangle$" and O_1 is the operator which acts on the target qubit $|x_2\rangle$. Note that if the control qubit is $|0\rangle$ then the operator $N \otimes O_1$ produces the null vector of $\mathbb{C}^2 \otimes \mathbb{C}^2$. Similarly, $N' \otimes O_0$, with $N' = E_{0,0} = |0\rangle\langle 0|$ realizes the condition "if the control qubit is $|0\rangle$ then the operator O_0 is applied to the target qubit $|x_2\rangle$ (and the control qubit is left unchanged)".

Notice that the method we are using here is a generalization of the Conditional Quantum Control method introduced in [2]. In fact recall that in Conditional Control $(n, 2)$–functions, 2^k functions $\delta_0, \ldots, \delta_{2^k-1}$ are stored in the memory of the control unit, the function δ_a being bijectively associated to the control input configuration labelled by the integer number $a \in \{0, \ldots, 2^k - 1\}$ (see Figure 3). In [2] these functions are described through unitary operators $U_0, U_1, \ldots, U_{2^k-1}$, defined on the Hilbert space $\otimes^{n-k}\mathbb{C}^2$; here we drop the requirement that such operators, as well as the global operator defined on $\otimes^n\mathbb{C}^2$, be unitary. Moreover, in the following we apply this method to realize d–valued operators.

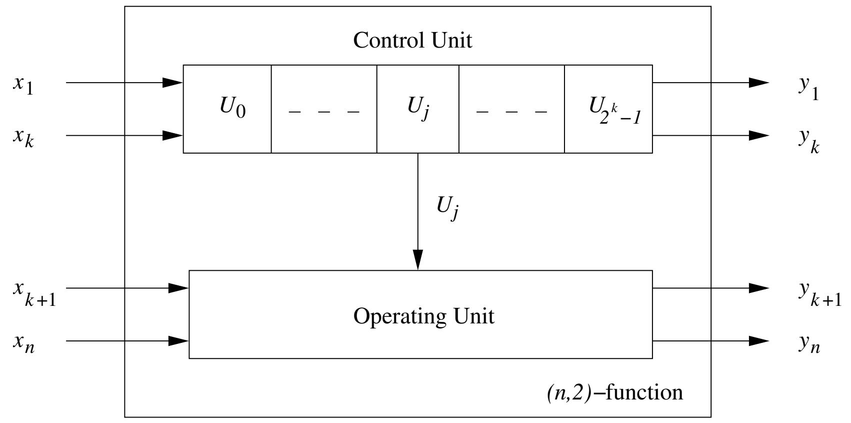


Figure 3: Conditional Quantum Control: the function is divided into a *control unit* and an *operating unit*. The input values of the control unit select a prescribed operator to be applied to the input values of the operating unit

Thus, when a configuration $|x_1, \ldots, x_k\rangle$ is fed to the control unit of a Conditional Control function two things happen:

1. the control configuration $|x_1, \ldots, x_k\rangle$ is returned unchanged into the output values of the control unit, and

2. the (non necessarily unitary) operator U_a bijectively associated to the control configuration is selected and applied to the input configuration $|x_{k+1}, \ldots, x_n\rangle$ of the operating unit, producing the output configuration $U_a\,|x_{k+1}, \ldots, x_n\rangle$.

The global operator on $[\otimes^k\mathbb{C}^2] \otimes [\otimes^{n-k}\mathbb{C}^2]$ which describes the behavior of the function can thus be written as:

$$P_0 \otimes U_0 + P_1 \otimes U_1 + \ldots + P_{2^k-1} \otimes U_{2^k-1} \tag{5}$$

where $P_a = E_{a,a} = |a\rangle\langle a|$ is the orthogonal projection of the Hilbert space $\otimes^k \mathbb{C}^2$ which selects the a-th control configuration and collapses to the null vector all the other control configurations, and U_a is the corresponding operator on $\otimes^{n-k}\mathbb{C}^2$ which has to be applied to the target configuration. Making use of Dirac notation, expression (5) can be equivalently written as (see [2]):

$$|0\rangle\langle 0| \otimes U_0 + |1\rangle\langle 1| \otimes U_1 + \ldots + |2^k-1\rangle\langle 2^k-1| \otimes U_{2^k-1} \tag{6}$$

A further extension of the Conditional Quantum Control method to the d–valued case is immediate. If the (n,d)–function under consideration can be divided as a k–input/k–output *control unit* and an $(n-k)$–input/$(n-k)$–output *target* (also *operating*) *unit*, then any input configuration $|x_1,\ldots,x_k,x_{k+1},\ldots,x_n\rangle$ can be splitted into a *control configuration* $|x_1,\ldots,x_k\rangle$ and a *target configuration* $|x_{k+1},\ldots,x_n\rangle$. The control configuration is returned unchanged on the k output values of the control unit; as a side effect, it selects one of the d^k (non necessarily unitary) operators $U_0, U_1, \ldots, U_{d^k-1}$, defined on the Hilbert space $\otimes^{n-k}\mathbb{C}^d$, stored into the control unit. The selected operator is applied to the target configuration in order to produce the output values of the target unit. The global operator that describes the behavior of the (n,d)–function has now the form:

$$P_0 \otimes U_0 + P_1 \otimes U_1 + \ldots + P_{d^k-1} \otimes U_{d^k-1} = \sum_{X=0}^{d^k-1} P_X \otimes U_X$$

where $P_X = E_{X,X} = |X\rangle\langle X|$ is the orthogonal projection of the Hilbert space $\otimes^k\mathbb{C}^d$ which selects the X-th control configuration, and collapses to the null vector all the other configurations. If many of the operators U_i are identical then this expression is much shorter than the one obtained with the brute force method. On the other hand, it is clear that the method derived from Conditional Quantum Control cannot be used to describe every conceivable (n,d)–function, since there are functions which cannot be divided into a control unit and an operating unit.

5 Some problems and directions for future work

In this section we address some problems we have encountered while trying to define quantum P systems. Since this is a work in progress, still in its early stage, we would like to share these problems with the community. We hope in this way to generate stimulating discussions on appropriate ways to define quantum P systems.

A first problem which comes to mind when speaking about quantum systems concerns the localization of objects. How can we be sure that an object will stay for a long time into a region surrounded by a membrane? Indeed, one notable feature of quantum systems is the so called "tunnel effect", thanks to which in every moment we have a positive probability that the object spontaneously (i.e., without the application of any rule) leaves the current region. How should we manage this situation? How can we control the computation (that is, the behavior of the system) under the assumption that every object can be anywhere with a non–zero probability?

The above problem is exacerbated by the fact that in classical P systems the objects can be moved as the effect of the application of a rule. Precisely, in classical energy–based P systems we can have a rule of the kind:

$$ae^k \to (b, p)$$

where $p \in \{\text{here}, \text{in}(name), \text{out}\}$ denotes the position of the new object b. This means that the rule has to transform the object a into the object b, using k units of free energy, and move b according to the prescribed position. In our current definition of quantum energy–based P systems we have only addressed the transformation of a into b. The object a will be represented as a pure state $\left|\frac{\ell}{d-1}\right\rangle$ of $\mathbb{C}^d$, with $\ell \in \{0, 1, \ldots, d-1-k\}$. Analogously, the object b will be represented as the pure state $\left|\frac{\ell+k}{d-1}\right\rangle$. The presence of k free energy units makes the transition from $\left|\frac{\ell}{d-1}\right\rangle$ to $\left|\frac{\ell+k}{d-1}\right\rangle$ possible. Currently, the movement of $\left|\frac{\ell+k}{d-1}\right\rangle$ to the new position p is assumed to occur in the same way as in classical P systems. However, this would imply the existence of a "magic" transportation mechanism that, notwithstanding the tunnel effect, is able to pick up and move a quantum system exactly as desired.

Another problem in the definition of a quantum P system derives from the fact that the presence of k units of free energy enables the transition from any state $\left|\frac{\ell}{d-1}\right\rangle$ of $\mathbb{C}^d$, with $\ell \in \{0, 1, \ldots, d-1-k\}$, to the state $\left|\frac{\ell+k}{d-1}\right\rangle$. Indeed, it is tempting to translate the classical rule $ae^k \to (b, p)$ into the quantum rule:

$$\left|\frac{\ell}{d-1}\right\rangle e^k \to \left(\left|\frac{\ell+k}{d-1}\right\rangle, p\right) \tag{7}$$

Now assume that a given region contains two rules of this kind, possibly with different values of ℓ and k. The presence of a big number of free energy units (at least as many as the maximum of the two values of k) activates both rules. This occurs even if $k = 1$ for both rules, and the region contains one free energy unit. However, in this last case one rule or the other is correctly applied in a nondeterministic way, as it happens in classical P systems. When $k > 1$ for both rules, and the region contains some free energy units, we must avoid the undesirable situation in which some free energy units modify the first object and the remaining free energy units modify the second object. This occurs because when we have written the quantum rule (7) we have implicitly assumed that the k units of free energy must act *simultaneously.*

One possible solution to this problem would be to allow only transitions which involve just one unit of free energy. However, we should check whether these systems are computationally equivalent to the more general ones (we conjecture that the answer is affirmative).

The solution we have adopted in this paper involves the linear operators $E_{x,y} = |y\rangle\langle x|$, with $x, y \in L_d$. The classical rule $ae^k \to (b, p)$ is translated as:

$$\left(E_{\frac{\ell}{d-1},\frac{\ell+k}{d-1}}, p\right) \equiv \left(\left|\frac{\ell+k}{d-1}\right\rangle \left\langle\frac{\ell}{d-1}\right|, p\right)$$

In this way, the object $\left|\frac{\ell}{d-1}\right\rangle$ can only be transformed into $\left|\frac{\ell+k}{d-1}\right\rangle$, and this transformation requires k units of free energy to be performed.

Another observation concerning the use of quantum rules is the following. In a classical P system, a rule of the kind $ae^k \to (b, p)$ is applied *simultaneously* to every occurrence of a in the region, provided that enough free energy units are present. Clearly, this is a mathematical abstraction. In a real quantum system, there will be a subsystem — whose behavior is described by an appropriate linear operator — which is devoted to transform an instance of a into an instance of b, using k units of free energy. Hence, only one of such transformations at the time is possible. How does this affect the computational power of quantum P systems? Notice that P systems were originally inspired by the functioning of living cells, and in living cells we have the same problem: a prescribed biological mechanism is devoted to transform one (or a limited number of) instance(s) of a into one (or the corresponding number of) instance(s) of b. Hence, even in classical P systems the possibility to simultaneously apply a rule to every instance of a is a mathematical abstraction.

Concerning the power of quantum P systems we note that, in analogy with other models of quantum computers, there is the possibility to initialize the system with a multiset of objects whose state is a superposition of classical (that is, pure) states. As a result, the computation will transform such input multiset to an output multiset which is obtained by linearity as a superposition of the results of the computation on every single classical state. As usual, when we measure the state of the systems which occur into the output multiset we will obtain a pure state as a result, according to the probability distribution which is induced by the coefficients of the superposition. Another interesting aspect of quantum P systems is their behavior when some quantum systems from the input multiset are in an entangled state.

Of course we advocate the study of the computational power of quantum P systems, by comparing them both against their classical counterparts and other quantum computational models. In particular, it would be useful to define a quantum version of counter machines, since they have proven to be a useful and powerful tool in the classical setting.

Last, but not the least, we pose the problem to write the linear operator which describes an entire quantum P system, starting from the linear operators which describe the single rules. Such global operator is important, from the point of view of Quantum Computing, for the existence of a physical system that behaves like the prescribed P system. Moreover, such operators are related with the Hamiltonian of the system, which describes the internal energy of the system. The difficulty of writing the global operator is due to the fact that in quantum P systems, at each computation step a maximally parallel set of rules is *nondeterministically* chosen and applied. This contrasts the deterministic nature of the application of the global operator to the global state of the quantum P system. One possible solution, in order to always select a maximally parallel set of rules, would be to distribute the free energy contained into each region to the rules of the region, sorted in increasing order with respect to the required amount of free energy units. However, this would be equivalent to introducing a priority between the rules of each region. We ask ourselves whether this is appropriate or not.

We hope that all these problems will generate stimulating discussion on quantum P systems. In this sense, they can all be considered a starting point for future work.

6 Conclusions

In this paper we have introduced two quantum versions of energy–based P systems. Both versions are defined just like classical energy–based P systems, but for objects and rules. Objects are represented as pure states in the Hilbert space $\mathbb{C}^d$, whereas the definition of rules differs between the two models. In the former, rules are defined as bijective functions — implemented as unitary operators — which transform the objects from the alphabet. In the latter, rules are defined as generic functions which map the alphabet into itself. Such functions are implemented using a generalization of the Conditional Quantum Control technique, and may yield to non–unitary operators. Finally, we have addressed some problems and outlined some directions for future work.

Acknowledgments

We gratefully thank Gheorghe Păun, who prompted us to define quantum P systems.

References

[1] G. Alford. Membrane systems with heat control. In *Pre–Proceedings of the Workshop on Membrane Computing*, Curtea de Arges, Romania, August 2002.

[2] A. Barenco, D. Deutsch, A. Ekert, R. Jozsa. Conditional Quantum Control and Logic Gates. *Physical Review Letters*, **74**, 1995, pp. 4083–4086.

[3] P. Benioff. Quantum Mechanical Hamiltonian Models of Discrete Processes. *Journal of Mathematical Physics*, **22**, 1981, pp. 495–507.

[4] P. Benioff. Quantum Mechanical Hamiltonian Models of Computers. *Annals of the New York Academy of Science*, **480**, 1986, pp. 475–486.

[5] D. Deutsch. Quantum Theory, the Church–Turing Principle, and the Universal Quantum Computer. *Proceedings of the Royal Society of London*, **A 400**, 1985, pp. 97–117.

[6] R. P. Feynman. Simulating Physics with Computers. *International Journal of Theoretical Physics*, **21**, No. 6–7, 1982, pp. 467–488.

[7] R. P. Feynman. Quantum Mechanical Computers. *Optics News*, **11**, 1985, pp. 11–20.

[8] R. Freund. Energy–Controlled P Systems. In *Membrane Computing*, Proceedings of the International Workshop WMC–CdeA 2002, Curtea de Arges, Romania, August 2002, LNCS 2597, Springer, 2002, pp. 247–260.

[9] P. Frisco. The conformon–P system: a molecular and cell biology–inspired computability model. *Theoretical Computer Science*, **312**, 2004, pp. 295–319.

[10] D. Gottesman. Fault–tolerant quantum computation with higher–dimensional systems. *Chaos, Solitons, and Fractals*, **10**, 1999, pp. 1749–1758.

[11] J. Gruska. *Quantum Computing.* McGraw–Hill, 1999.

[12] A. Leporati, C. Zandron, G. Mauri. Simulating the Fredkin Gate with Energy–Based P Systems. *Journal of Universal Computer Science*, **10**, No. 5, 2004, pp. 600–619. A preliminary version is contained in [16], pp. 292–308.

[13] A. Leporati, C. Zandron, G. Mauri. Universal families of Reversible P Systems. In *Machines, Computation and Universality* (MCU 2004), Saint–Petersburg, Russia, September 21–26, 2004. To appear in Lecture Notes in Computer Science.

[14] G. Păun. Computing with membranes. *Journal of Computer and System Sciences*, **1**, No. 61, 2000, pp. 108–143. See also Turku Centre for Computer Science – TUCS Report No. 208, 1998.

[15] G. Păun. *Membrane Computing. An Introduction.* Springer–Verlag, Berlin, 2002.

[16] G. Păun, A. Riscos Nuñez, A. Romero Jiménez, F. Sancho Caparrini (Eds.). Second Brainstorming Week on Membrane Computing, Seville, Spain, February 2–7, 2004. Department of Computer Sciences and Artificial Intelligence, University of Seville TR 01/2004.

[17] G. Păun, Y. Suzuki, H. Tanaka. P Systems with energy accounting. *International Journal Computer Math.*, **78**, No. 3, 2001, pp. 343–364.

[18] The P systems Web page: `http://psystems.disco.unimib.it/`

A Typology of Imprecision

Solomon Marcus

Romanian Academy, Mathematics
Bucureşti, Romania
E-mail: solomon.marcus@imar.ro

1 Introduction

For about 300 years, scientific language takes its distance in respect to ordinary language, just because the latter is no longer able to face the requirements of rigor and precision of the former. In these conditions, it could look strange to make of imprecision a goal of scientific investigation. The explanation of this paradox is that there is no precise definition of precision, while the difference between mathematics and the other fields is that mathematics is dealing with exact approximations, while in absence of mathematics we cope with . . . approximate exactness.

When we relate mathematics to precision, we don't have in view its object of investigation, but only the nature of its approach. We are looking for a precise approach to a world of imprecision. As a matter of fact, poetry too involves a combination of precision and imprecision, as Baudelaire observed: the psychological states determined by a work of art are imprecise, but the means used to obtain this work are precise.

2 Abstraction, Approximation, and Generality

The first type of imprecision is *abstraction*, the move from five apples, five boxes, five stones to "five", with no other specification. The imprecise nature of abstraction is related to the imprecision related to its various ways of instantiation. In absence of abstraction, mathematics is not possible.

The second type (historically speaking) of imprecision considered in mathematics was *approximation*, from its elementary form (approximation

of an irrational number by rational ones), to approximation of the sum of a series by a finite sum, then to approximation of a function by another one (typical example: approximation of a continuous function by polynomials), approximation of the solution of a differential equation, etc. The traditional approximation is that where we are looking for finite approximations of countable infinite structures, then for countable approximations of non-countable infinite structures, etc. The novelty brought by computer science and by the modern mathematics is the increasing importance of approximation of finite structures by means of infinite ones. Typical in this respect is grammatical inference. In formal language theory, the simplest example is the fact that the finite language of powers of x from 1 to n cannot be generated by essentially less than n Chomskian rules (of a bounded size, e.g., regular), while the infinite language of all powers of x going from 1 to infinite needs only two rules in order to generate it by a Chomskian grammar.

The next types of imprecision were *generality* and *genericity*; may be they are concomitant with *approximation*, but their typical form is related to the birth of algebra, when we replace 5 by x. Let us observe also the importance of generality and genericity in natural languages.

3 Randomness, Vagueness (Fuzziness), and Ambiguity

The next type of imprecision coming into the attention of mathematics was (in the 15th and the 16th century) *randomness*. It gave rise to probability theory, developed by Pascal, Bernoulli, Laplace and others and reaching its modern form in the 20th century, with Kolmogorov.

Already the Greek antiquity, then the modern times brought into attention the *paradox* (*aporia*), which, interpreted as a transgression of one of the principles of Aristotelian logic, is a kind of imprecision, because it involves the simultaneity of two opposite states (in case of transgression of the principle of non-contradiction), or the lack of clarity, when the principle of identity is transgressed, between identity and alterity. Similarly for the transgression of principle of excluded middle, as it appears already in non-Euclidean geometry.

"Vagueness" was the title of a famous article by Black (1937) and we adopt the hypothesis according to which its mathematical model is Zadeh's fuzziness (1965). About their equivalence, we bring as argument the view proposed by Frege (1903): "The concept must have a sharp boundary. To the concept without a sharp boundary there would correspond an area that does not have a sharp boundary-line all around".

Surprisingly, *ambiguity*, one of the most frequent form of imprecision, still has no general mathematical representation. The word is so popular that it may be a label for any type of imprecision. A reason of this gap could be the fact that "ambiguity" itself is very ambiguous; it may mean "non-specificity"; "dissonance" or "confusion"; "loss of information", etc. (see, in this respect, Klir, 1987). Specific types of ambiguity were however investigated; see Empson (1930) for literary-artistic ambiguity and Marcus, ed. (1981, 1983) for contextual ambiguity, while ambiguity in formal grammars is presented in Rozenberg-Salomaa (eds.) (1997).

4 Learn to Combine Various Types of Imprecision

For a combination of some of the above considered types of imprecision, consider the statement “If it will not rain and if it will not be cold, I will wait for you tomorrow around 5 p.m., near the corner of the University”. We have here randomness (rain), fuzziness (cold), approximation (around 5 p.m.), and ambiguity (if the building of the University has several corners). This is the typical situation with imprecision. Life situations bring together several types of imprecision and we have to learn to cope with them in combination, not only each of them on its own, as it happens in most cases.

5 Negligibility, Indiscernibility and Roughness

Negligibility concerns the global behavior of a set, of a function, of a class of sets or of functions or of another entity for which the local-global distinction makes sense. Negligibility refers usually to cardinality, measure or topology, but other types are also possible. Examples: A real monotonous function defined on the real interval $[a, b]$ is continuous in each point of $[a, b]$, except a countable set; it is differentiable in each point of $[a, b]$, except a set of Lebesgue measure zero. Any real function defined on $[a, b]$, which is in $[a, b]$ the limit of a sequence of continuous functions is continuous in each point of $[a, b]$, except a set of first Baire category (i.e., which is a countable union of rare sets). In the first example, the exceptional set is negligible in respect to cardinality, in the second example it is negligible in respect to Lebesgue measure, in the third example it is negligible in respect to Baire category (which is of a topological nature). In the theory of formal languages, “regular” may be considered negligible in respect to “non-regular context-free”, while “context-free” may be considered negligible in respect to “non context-free, but context-sensitive”, negligibility being here considered in respect to Chomsky hierarchy of languages. Finite sub-languages of a language L are negligible in respect to L if L is regular, but infinite. It was proved by Marcus–Păun that some theorems concerning convexity of sets, where negligibility is related to measure or to topology, have their counterpart in problems related to convexity in formal languages, where negligibility is considered just in respect to Chomsky hierarchy. Negligibility is considered also in the theory of recursive functions, mainly in respect to topology.

Indiscernibility was approached by Pawlak (1982) by means of his concept of a rough set, today a very well-known topic in Computer Science and in Artificial Intelligence. It is considered in respect to an information system, consisting of a set of objects and a set of criteria (such as size, color, material, weight etc.). Each criterion has various possible values; for instance, color may be green, red, yellow, blue, black, white etc. To each object we associate its values in respect to the considered criteria. Selecting a value for each criterion, we may consider the set A of objects with the respective values; it will be approximated from its interior by the set $int(A)$ of objects having with certainly the respective values and from its exterior by the set $ext(A)$ of objects which possibly have the respective values. We may always assert that $int(A)$ is contained in $ext(A)$; their ordered pair is a rough set. If $int(A) = ext(A)$, then we have a usual set. Taking into account that this approach is based on the idea of similarity (two objects having the same values in respect to all criteria are similar) and observing that similarity is a tolerance relation (i.e., reflexive and symmetric) rather than an equivalence relation, as it is considered in Pawlak's approach, Marcus (1993) considered tolerance rough sets (see synonymy in natural languages and the relation "smaller than" considered by Zeeman in its topology of the brain).

Rough sets showed their relevance in approaching other types of imprecision, such as fuzziness (Pawlak–Skowron 1993), evidence (Shafer 1976; Skowron 1993), inconsistency (Grzymala-Busse 1992) and vagueness (Pawlak 1992).

6 Plausibility, Possibility, Credibility, Uncertainty

The first three of them are favored topics in Logic and in Artificial Intelligence, as it can be seen, for instance, in Klir(1987), Klir and Folger (1988). Uncertainty is among the most fashionable topics; see, for instance, Kline (1980), Klir (1987), Klir and Folger (1988), Smithson (1989). Kline's slogan "Mathematics: the loss of certainty" was preceded by Heisenberg's uncertainty principle (1927) and has been followed by a similar slogan by Prigogine ("La fin des certitudes"). Both physics and mathematics are involved here. Gödel's incompleteness theorem seems to be a basic motivation of this pessimistic view. As a matter of fact, it is not the loss of certainty the right phenomenon we have to face here, but the fact that the previous feeling of certainty was determined by our ignorance; as soon as we became aware of how things happen, the mistake was no longer possible. The same thing is valid in respect to certainty in physics.

But we can go further and ask: is certainty the natural state of human psychic? The answer is rather negative. Human beings are characterized by a state of tension, of restlessness stimulating him to learn more and more and to be more and more creative, in order to diminish the gap between their limitations and life's and world's mystery.

7 Absence of Cohesion and Lack of Coherence

Within a social group G, the link between two parts A and B of G can be evaluated by the product between the cardinal number of the common part of A and B and the cardinal number of the symmetric difference of A and B (Bunge 1971). In this product, the first factor refers to the common participation, while the second one to the heterogeneity of A and B. From another direction, related to language aspects and to linguistics, we observe that cohesion refers rather to the syntactic aspect, while coherence refers to the semantic one. We can speak of the cohesion of a text if its parts are organically related. The first idea coming in mind is the topological notion of connectedness. Connectedness of what? Lipsky (1974), Saloni–Trybulec (1974) and Brainerd (1977) proposed as a mathematical model of the cohesion of a statement s, the connectedness of the dependency and subordination graph G associated to s (see, in this respect, the last chapter in our *Algebraic Linguistics*, Academic Press, New York, 1967). If we proceed in this way, then the lack of cohesion is measured by the smallest positive integer n such that there exist, in G, n arcs with the property that if we delete them, then the obtained graph is connected. When $n = 0$, the statement s has the cohesion property. The proposition (having as its mathematical model a rooted tree) is the simplest example of linguistic cohesion.

We can extend this approach to obtain a model of lack of coherence, if we use an idea by Irena Bellert (1970), permitting to associate to the statement s not only a syntactic graph, as above, but also a semantic graph; it can be obtained by defining a relation of semantic dependency. A term a of s is semantically dependent of a term b of s if the semantic interpretation of a depends on the semantic interpretation of b. The reflexive and transitive closure of this relation leads to the semantic graph H of s. If H is connected, then s is coherent. If not, then a measure of the lack of coherence of s is given by the smallest positive integer n such that there exist n arcs of H with the property that if we delete them from H, then the remaining graph is connected.

For a global view on textual cohesion and textual coherence see S. Marcus, "Textual cohesion and textual coherence", *Revue Roumaine de Linguistique*, 25, 2 (1980), 101–112.

8 Measures of Graduality

The graduality $f(A)$ of a set A, conceived as something opposed to the idea of precise set A (as in Cantorian set theory), should fulfill some intuitive requirements: $f(A) = 0$ iff A is precise; if B is more gradual than A, then $f(A) < f(B)$ or $f(A) = f(B)$; $f(A)$ takes the maximal value iff A is maximally gradual (but let us observe that it may happen that maximality does not exist). A. de Luca and S. Termini (1972) define a measure of graduality taking as a model the entropy:

$$f(A) = -\sum_{x \in X} (m(A,x) log_2(m(A,x)) + (1 - m(A,x)) log_2(1 - m(A,x))),$$

where, if A is not more gradual than B, then $m(A,x) \leq m(B,x)$ when $m(B,x) \leq 1/2$, and $m(A,x) \leq m(B,x)$ when $m(B,x) \geq 1$, for any $x \in X$. Maximal graduality corresponds to the degree of belongingness equal to $1/2$ for any $x \in X$.

Another measure was proposed by A. Kaufmann (1975):

$$f(A) = \sum_{x \in X} (|m(A,x) - m(C,x)|),$$

where C is a precise set for which $m(C,x)$ is equal to zero if $m(A,x) \leq 1/2$, and $m(C,x) = 1$ if $m(A,x) > 1/2$.

A larger class of measures of graduality was proposed by S.G. Loo (1977). According to R.R. Yager (1979), the most natural way to express graduality of A is to require the absence of a sharp distinction between A and its complementary set $c(A)$. If A is gradual, then $c(A)$ is given by a mapping $c : [0,1] \to [0,1]$, associating to each value $m(A,x)$ a value $c(m(A,x))$ expressing the degree of belongingness of x to $c(A)$. The mapping c is required to be non-increasing, to take value 1 in origin and value zero in 1. This means that, if A is precise, then c becomes the classical complementary set. Sometimes it is also required to c to be continuous and involutive: $c(c(a)) = a$ for any a between 0 and 1. See for more G.J. Klir, "Where do we stand on measures of uncertainty, ambiguity, fuzziness, and the like", *Fuzzy Sets and Systems*, 24 (1987), 140–160.

It seems that there are distinctions which are not yet considered. It may happen that a graduality is not associated to a phenomenon of imprecision, in respect to the distinction between a property and its negation. The property of water to be liquid is not vague (fuzzy) in Zadeh's sense, because the move from liquid to non-liquid is exactly at zero or at 100 degrees. However, the same move is gradual, the water at 70 degrees is nearer to gaseous state than the water at 60 degrees. By contrast, an explosion of a plain during a flight, following the clash with another plain, is not at all gradual.

9 Confidence, Plausibility and Ignorance

The already mentioned paper by Klir (1987) reminds some other proposed measures of various types of imprecision. Starting from a basic probability m associating to each subset A of X a number $m(A)$ between 0 and 1, such that $m(O) = 0$ and $\sum_{A \subseteq X} m(A) = 1$, one can say that $m(A)$ defines the degree of confidence that a specific element of X belongs to A. The corresponding "measure of confidence" is given by a mapping f associating to each part A of X a number $f(A)$ between 0 and 1, such that $f(A) = \sum_{B \subseteq A} m(B)$. The number $f(A)$ gives the total degree of confidence that a considered element belongs to A or to an arbitrary subset of A.

The "measure of plausibility" is given by a mapping g associating to each part A of X a number $g(A)$ between 0 and 1, such that $g(A) = \sum_{B \cap A \neq \emptyset} m(B)$.. There is a link between the measure of confidence and the measure of plausibility: $g(A) = 1 - f(X - A)$.

The "total ignorance" is expressed by $m(X) = 1$ and $m(A) = 0$ for any A different from X; therefore, $f(X) = 1$ and $f(A) = 0$ for any A different from X and $g(O) = 0, g(A) = 1$ for any non-empty A. As a particular case of the measure of plausibility, we get the "measure of possibility". For bibliographic references and for more details, see Klir (1987).

10 Hartley, Shannon, Renyi, Higashi, Klir on Ambiguity

We have already pointed out the ambiguity of the phenomenon of ambiguity. We will call into attention a few cases when the mathematics of ambiguity was successful.

A first approach belongs to Hartley (1928) and refers to ambiguity as non-specificity. It is given by $A(N) = k.log_b N$, where N is the total number of variants involved in a system, while k is a strictly positive constant. For $k = 1$ and $b = 2$, the measure $A(N)$ of non-specificity is evaluated in bits. Alfred Renyi has shown that the mapping $A(N)$ measuring the ambiguity involved in the selection of an element in a set can be structurally characterized by three properties: $A(N \times M) = A(N) + A(M)$ (additivity) $(N, M = 1, 2, 3, ...)$; $A(N) \leq A(N + 1)$ (monotonicity); $A(2) = 1$ (normalization).

Other measures of non-specificity were proposed by M. Higashi and G.J. Klir (1982, 1983). Given a normalized distribution $p = (p(x); xinX)$, $max(p(x); xinX) = 1$, of possibilities on X, the measure of non-specificity is given by the integral from 0 to 1 of the logarithm in base 2 of the cardinal number of the section $c(p, x)$ associated with x.

The measure of the classical ambiguity conceived as dissonance or confusion is just Shannon's entropy in respect to a probability distribution.

Ambiguity as loss of information has been investigated by J.L. Dolby (1977). For more details, see Klir (1987).

11 Contextual Ambiguity in Languages and in Medical Diagnosis

Given a finite non-empty alphabet A and a language L on A, the word x on A contextually dominates the word y on A in respect to L if for any two words u and v, such that $uxv \in L$, we have $uyv \in L$. In other words, x contextually dominates y in respect to L if any context accepting $x \in L$ accepts also $y \in L$. The interpretation of this fact is: when x contextually dominates y in respect to L, the contextual ambiguity of x in respect to L is not larger than the contextual ambiguity of y in respect to L. Things become very clear when x and y are of length one, i.e, elements of A, interpreted as the vocabulary of a natural language. Then, in French, for instance, we observe that 'beau' contextually dominates 'douce', but the converse is not true, because 'douce nuit' is well-formed in French, while 'beau nuit' is no longer well-formed. The relation of reciprocal domination is an equivalence relation, in respect to which strings on A are organized in equivalence classes, called 'distributional classes'. To any natural language we can associate the graph of its distributional classes. However, since a natural language is not a precise set (as a matter of fact, its status in respect to the typology of imprecision is not clear, because, for instance, it should be both finite and infinite; see Charles Hockett), we limit discussion to a precise subset of a natural language, i.e., to what is called one of its levels of grammaticality. We also limit the discussion to the distributional classes of the vocabulary of the language, in order to cope with a finite graph. In the graph G so obtained, we have a line from the vertex x to the vertex y if any element in the distributional class x contextually dominates the elements in the distributional class y. The number of different lines leading to the same vertex a is a measure of the contextual ambiguity of words in a, called their "index of ambiguity".

In the monograph *Contextual ambiguities in natural and in artificial languages* (ed. S. Marcus), volume 1, 1981, volume 2, 1983, Communication and Cognition, Ghent, Belgium, contextual ambiguity is examined in English, French, Hungarian and Romanian and in Fortran IV, Assiris and Pascal. It was shown that in the considered programming languages contextual ambiguity is very poor, while in natural languages it is very rich.

In English, in respect to a level of grammaticality higher than that of noun groups and of verb groups, there are 14 types of contextual ambiguity, represented, in the increasing order of their index of ambiguity, by words like 'receive'(index=1), 'isolate'(index=2), 'often'(3), 'death'(4), 'rain'(5),'see'(6), 'cry'(7), 'iron'(8), 'hang'(9), 'round'(10), 'fly'(11), 'run'(12), 'wash'(13), 'call'(14).

The same mathematical model can be applied to medical diagnosis, if the alphabet A is formed by various possible symptoms of a disease. Words on A include in this case various possible clinical examinations (syndroms). We can define the degree of ambiguity of a diagnosis, having its lower limit in the case of pathognomonic symptoms, able to indicate with no ambiguity a specific disease. See, for more, Eugen Celan - Solomon Marcus, Le diagnostique comme langage, *Cahiers de Linguistique Theorique et Appliquee*, 10, 2 (1973), 163–173.

12 Eubulides, Zadeh and Sugeno on Graduality

The imprecision resulting from absence of a clear (sharp) border between a property and its negation, between a set and its complement has been observed already by the old Greeks; see Eubulides' paradox of the heap of grains and of baldness. However, Zadeh's (1965) fuzziness, where the characteristic function of a set A contained in the total set X is replaced by a mapping associating to each element in X a number in the compact interval $[0, 1]$ (value representing the degree of belongingness to A), does not capture successfully Eubulides' examples. Both fuzziness and roughness remain irrelevant in respect to these old paradoxes, as we have already shown (Marcus 1999).

An alternative to Zadeh's approach was proposed by M. Sugeno (1977), who considers, for each element $x \in X$, a mapping $g(x)$ associating to every part A of X a value $g(x, A) \in [0, 1]$, representing the degree of belongingness of x to A. We have $g(x, O) = 0, g(x, X) = 1$, while for A contained in B $g(x, A)$ is not larger than $g(x, B)$.

In all its variants, the definition of graduality uses non-graduality, i.e., sets which are no longer gradual. For instance, the set of points where Zadeh's mapping takes a definite value is no longer fuzzy, it is a set in the classical sense. If however we allow for it to be fuzzy, then we move the problem to the next level. The last meta-level will always be no longer fuzzy.

Another difficulty is related to the behavior of the values 0 and 1. For most properties, there is no way to assign these values. How could we assign the values 0 and 1 in the case of a property such as 'beautiful' or 'clever'? How can we assign the value 1 in the case of 'non-bald'?

13 Numerical Negligibility and the Crisis of Classical Probability

We have in view a history beginning with Leibniz, who refers to infinitely small quantities,different from zero, but smaller than any number of the form 1/n, where n is a strictly positive integer. Mathematicians in the XVIIIth and XIXth centuries were not able to make meaningful Leibniz's proposal, so they replaced it with another version, where the infinitely small is no longer a fixed quantity, but a function which, in some point, has zero as its limit. One can introduce also the order of an infinitely small. The function f is infinitely small of order n at the point a if the ratio between $f(x)$ and the power n of $x - a$ has the limit zero when x is approaching a. A fundamental problem in mathematical analysis is to make the error of approximation of a function to be infinitely small of order as high as we want. Higher is this order, better is the approximation. A typical example is the approximation of a continuous function by polynomial functions.

The above phenomenon concerns numerical negligibility of various orders. The numerical negligibility conceived by Leibniz received a coherent interpretation from Abraham Robinson, with his non-standard analysis, in the '60th of the XXth century. He conceived a universe larger than the universe of real numbers, under the form of a totally ordered field larger than the field of real numbers; in contrast with the latter, the former is no longer archimedean (i.e., it is no longer true that for any two positive elements a and b there exists a natural number n such that $na > b$). It can be observed that Leibniz's infinitely small is just the negation of Archimede's axiom. To give only one example of a situation requiring just a non-standard approach, let us consider the classical notion of probability, where the probability of a number in $[0, 1]$ to be in a subinterval of length $m < 1$ is just equal to m. Then, what is the probability of the number $x \in [0, 1]$ to be equal to 1/2? Since 1/2 is contained in subintervals of length m with m as small as we want, it follows that the only coherent solution is to give to the probability of x to be equal to 1/2 the value zero. This happens if we remain in the framework of classical analysis and classical probability theory. But we cannot be satisfied with this 'solution', because it is in contradiction with the intuitive fact that x can be really equal to 1/2. The shortcoming of probability theory, conceived as a measure, is just this gap between impossibility and zero probability; they should be equivalent, but, unfortunately, impossibility implies zero probability, while the converse is not true. The solution is to consider that the probability of x to be equal to $1/2$ is just an infinitely small in the Leibnizian sense, which is possible in the non-Archimedean framework conceived by Robinson.

14 Randomness and Its Intuitive Base

At the beginning of the XXth century, Emile Borel defined an infinite random sequence r on the binary alphabet $\{0, 1\}$ by the property that each of the binary digits 0 and 1 has the same probability of appearance in r. More precisely, denoting by $p(0, n)$ ($p(1, n)$) the probability to have zero (one) in the prefix of length n of r, the limit of $p(0, n)$ ($p(1, n)$) when n tends to infinite is equal to 1/2 (1/2 respectively). Similarly one can define the randomness of an infinite sequence on an arbitrary finite alphabet. This type of randomness was called by Borel 'normality'. Real numbers in their decimal writing are infinite sequences on the alphabet $\{0, 1, 2, ..., 9\}$. The basic result obtained by Borel was that almost all real numbers are normal. 'Almost' means here 'except a set of Lebesgue measure zero'. On the other hand, it was shown later, by J.C. Oxtoby and S. Ulam (*Annals of Math.*, 42, 1941, 874–920) that the set of non-normal real numbers, which is negligible in respect to Lebesgue measure (according to Borel's theorem) is no longer negligible in respect to Baire category. In other words, normality is an exceptional property, in the sense that normal numbers form a set of first Baire category (i.e., it is a countable union of rare sets), while non-normal numbers no longer have this property. This fact shows how misleading can be, from an intuitive viewpoint, the mathematical terminology.

There is also another trap of theorems like those by Borel and Oxtoby-Ulam: they give an information of a global nature, having no counterpart from a local viewpoint. For instance, for most familiar numbers such as square root of 2 or number π we are ignorant whether they are or not normal. This feature is characteristic for most, if not all theorems involving global negligibility.

Defining randomness as normality proved to be unsatisfactory, as it can be seen on the sequence obtained by writing, in lexicographic order, all finite sequences of length 1, 2, 3, ... on a given alphabet. The resulting infinite sequence will be obviously normal, despite the fact that it was written according to a very clear rule. Such a situation is in conflict with the most primitive representation of randomness, as absence of rule. The considered example shows also why a stronger requirement such as to require to all sequences of the same length on the considered alphabet to have the same probability of appearance in the considered infinite sequence is also not acceptable as a mathematical model of randomness. Richard von Mises (1919) proposed then to impose Borel's property (the law of large numbers) also to some subsequences, according to some rules. But this restriction too proved to be insufficient. Some next proposals (A. Wald, 1937; A. Church, 1940) tried to impose to the selection rules proposed by von Mises a constructive character. The critical analysis of all these attempts, made by Michiel van Lambalgen (Von Mises's definition of random sequences reconsidered, *J. of Symbolic Logic*, 52, 1987, 3, 725–755) reaches the conclusion that in the framework of classical mathematics (Platonistic, as he calls it) there is no satisfactory possibility to formalize randomness. Trying however to recuperate von Mises' intuitions, Lambalgen finds that the most acceptable solution is to adopt the approach based on sequential tests, proposed by P. Martin-Loef (1966) and studied further by C.P. Schnorr (1971). If an infinite sequence is random according to Martin-Loef, then almost all its subsequences are also random.

Progress in this direction is looking for more and more weaker conditions, that however still refuse to be sufficient conditions of randomness. Pure(total) randomness seems to can be only approximated, asymptotically approached.

What is, intuitively speaking, randomness? equal preference, absence of rule, total imprevisibility, highest possible complexity? Can they be simultaneously satisfied? It seems that the answer is negative.

15 Disorder as Entropy versus Order as Information

In a thermodynamic perspective, information was identified, in the second half of the XIXth century, with order and organization, as opposed to disorder, chaos and entropy. The notion of entropy, introduced by Clausius and reconsidered by Boltzmann and Helmholtz, leads to the evaluation of the thermodynamic order as the difference between the maximum possible entropy and the real entropy; just this order expresses the thermodynamic meaning of information. The second principle of thermodynamics indicates the trend of the physical world towards the increasing of entropy, i.e., of disorder. But, as Prigogine showed, within this ocean of increasing entropy the human being creates an island of decreasing entropy. According to George Birkhoff, the artistic beauty of an object is given by the ratio between its order and its complexity. According to Karl Popper, a statement says about the empirical reality just what it interdicts to it. Both Birkhoff and Popper wrote in the thirties of the XXth century, i.e., 20 years before Shannon's information theory, based on the same philosophy. Information means reduction of disorder.

16 Randomness for Finite and for Infinite Strings

Roughly speaking, according to Kolmogorov and Chaitin, a finite string x over a finite non-empty alphabet A is random if no computer program describing x is shorter than x. This definition is related to a way to look at the algorithmic complexity of x, in respect to which randomness expresses the highest possible complexity of x. Larger is the difference between the length of x and the length of the shortest possible computer program describing x, smaller is the algorithmic complexity of x. When this difference is zero, x is random.

Surprisingly, this way to look at complexity is similar to the way a specific type of poetry is considered by some literary critics, as a text in which nothing can be deleted, added or modified and no abstract is possible. So, poetry corresponds to highest complexity.

There is a huge discrepancy between the global and the local behavior of randomness of finite strings. On the one hand, in some sense (which is specified mathematically; see the above idea of global negligibility) most finite strings are random; on the other hand, no instantiation of a random finite string is possible. By analogy, we may have an idea of this phenomenon if we think to what means to consider an arbitrary triangle; as soon as you try to represent it on a piece of paper, it is no longer arbitrary. It is interesting to observe the basic difference between The Kolmogorov- Chaitin approach and Shannon's approach in his 'information theory'. The former is dealing with individual entities, while the latter considers the global aspect. Shannon's theory makes sense in respect to a probability distribution in a system having various possible states, while Kolmogorov and Chaitin refer to individual strings over A.

Starting from randomness in finite strings, we may obtain a natural approach to randomness of an infinite string s over A. We define the randomness of s by requiring the randomness of all prefixes of s.

Infinite random strings on A have nice properties, that may be considered far from our intuitive expectations: If s is random, then any possible finite string over A occurs infinitely many times in s. This means, for instance, if A is the alphabet of English, that s will include infinitely many times the whole work of Shakespeare and of any other English writer. But this fact shows that global randomness implies local non-randomness. On the other hand, if we start with a random finite string x and we consider the infinite string obtained by concatenation of x with itself infinitely many times, $xxx...x...$, then we obtain a periodic infinite string, i.e., a non-random string.

17 The Analytic Approach to Deterministic Chaos

Besides the traditional chaos, associated with probabilistic systems, there is also the deterministic chaos, associated with deterministic systems. Small differences of initial conditions of a dynamical system may lead to huge differences in the behavior of the system. The analytic approach to this phenomenon is possible in the following way:

Let I be a real compact interval and let f be a continuous mapping from I into I. A point $p \in I$ is considered periodic for f if there exists a strictly positive integer n such that the value of the iterated of order n of f in p is equal to $f(p)$. The smallest n with this property is considered the period of p. Given a strictly positive number a, f is considered a-chaotic if there exists a perfect subset S (i.e., a set which is identical to the set of its accumulation points) of I with the property that, for any two distinct points x and y of S and for any periodic point p of f, the the following properties take place: the difference between the iterates of order n of f in x and y has, in absolute value, its inferior limit (when n tends to infinite) equal to zero, while its superior limit remains larger than or equal to a; the difference between the iterates of order n of f in the points x and p has, in absolute value, its superior limit (when n tends to infinite) larger than or equal to a. As it was shown by K. Jankova and J. Smital (A characterization of chaos, *Bull. of Australian Math. Soc.*, 34 (1986), 283–292), this notion is equivalent to that considered previously by Li and J. Yorke in 1975, in order to approach a problem in biology.

As it was shown by J. Smital(Chaotic functions with zero topological entropy, *Trans. Amer. Math. Soc.*, 297 (1986), 269–282), any continuous function f which is chaotic for no a strictly positive has the following property: for any $x \in I$ and for any a strictly positive, there is a periodic point p, such that the difference between the iterates of order n of f in x and p has, in absolute value, its superior limit (when n tends to infinity) strictly inferior to a. In other words, for a continuous mapping f from I into I which is not chaotic, any trajectory can be approximated by cycles. Practically, this behavior cannot be distinguished from the asymptotic periodicity of the trajectories. Non-chaotic continuous mappings can serve as deterministic mathematical models of some real processes.

18 Smale's Horseshoe

Almost any system with a chaotic behavior includes as one of its components a certain dynamical system (or its continuous variant) discovered by Stephen Smale and known under the name 'Smale horseshoe'. Chronologically, this was one of the first dynamical systems where the sensible dependency on the initial conditions has been understood in a rigorous and complete way.

Let us consider a mapping f defined on the interval $[0, 1]$ and associating to each $x \in [0, 1]$ the fractional part of $2x$, i.e., the difference between $2x$ and the largest integer which is not larger than $2x$. The mapping f defines one of the simplest dynamical systems; looking at its functioning, we will be nearer to the understanding of the 'paradox of the deterministic randomness'. Let us represent x in base 2. We have $f(x) = 2x$ if $x < 1/2, f(x) = 0$ if $x = 1/2, f(x) = 2x - 1$ if $1/2 < x < 1$ and $f(1) = 0$. It follows, for instance, that $f(0, 11) = 0.1,\ f(0, 111) = 1.11 - 1 = 0.11$ and, by induction, if $x = 0.11...1$, where 1 appears $n+1$ times after coma, then $f(x) = 0.11...1$, where 1 appears n times after coma. In other words, the mapping f moves with one digit at right the position of the coma and replaces with zero the first occurrence of 1.

This dynamical system is defined by the iterative application of the mapping f, iteration made possible by the fact that the values of f are situated in the interval of definition of f. The system has inputs and outputs. To an input x between 0 and 1 corresponds an output $f(f...f(x)...)$, where the number of left parentheses is equal to the number of right parentheses and both are equal to the number of applications of the mapping f. Let us consider the input $x = 0.11...1$, where 1 appears 30 times after coma. With each application of f, the number of occurrences of 1 after coma diminishes with one, so, after 30 iterations of f, the 'initial condition', under the form of the input x, will be completely neutralized. In other words, after a sufficient large iterations of f, the result is no longer dependent on the starting point (the initial state of the system) and the behavior gets a random aspect.

This is the type of mechanism explaining the behavior of most chaotic dynamical systems.

19 Chaos in the Evolution of a Population

Let us refer to the evolution of a population of a given species (human beings included). Its evolution in time is described by a mapping iteratively applied to some initial data, related to the state of the considered population at a given initial moment. The first application gives the evolution of the population after one year, n iterations gives the evolution after n years. But how should we choose the form of the mapping? Accepting that the population is growing with some percentage, the same every year, we are lead to a linear function, i.e., of the form $f(x) = ax$, where a is a constant defining the rate of growth. If, for instance, $x = 1000$ and the rate is equal to 1.1, then, after a year, the population will be 1.100; after one more year, 1210 etc. This was the scenario proposed by THomas Robert Malthus in respect to the population growth. There is no room in this scenario for various economic, social, psychological, moral parameters. Already in the first part of the XXth century, Vito Volterra has investigated, by means of the theory of differential equations, the evolution of some species. Researchers agree now that the mapping f should be selected in order to fulfill the following three conditions: rapid growth if the considered population is small; reduction of the growth until some values near to zero, if the population is of an intermediary size; diminution, if the population is very large. A function satisfying these requirements could be $f(x) = ax(1 - x)$. In this case, the population will be expressed by a number between 0 and 1.

To the function above one associates an equation with differences, called the 'logistic equation', considered successively with various modifications, in order to adapt it to various situations. Most authors agreed that, after some growth and some oscillation, a population has a trend of stability around an equilibrium value. THis idea is firmly expressed by J. Maynard Smith ("Mathematical ideas in biology", 1968), according to which populations get stable or oscillates with 'a regular enough periodicity' around an equilibrium point.

From another direction, related to meteorology (Lorenz) and further developed by Smale and Yorke, we reach the mapping already considered (Smale horseshoe). On the other hand, Robert May tried to investigate a population of fishes, by means of the already mentioned logistic equation. Its numerical investigation revealed surprising results. At the frontier between stability and oscillation. For $a = 2.7$, the population proved to be equal to 0.6292. By successive increasing the value of the parameter a, the population is increasing. As soon as the parameter a crosses the value 3, the imaginary population of fishes considered by May begins to oscillate, some times with a period of two years, other times with a period of four years. But beyond some point this periodicity becomes chaos. James Yorke has investigated this behavior in a paper with the significant title "Period three implies chaos" (published in *American Math. Monthly*).

$*$

Many other things remain to be said. Unfortunately, we have to stop here. For a common denominator of many of these types of imprecision, under the form of conjugate pairs, and for more precise bibliographic references, see our papers: Solomon Marcus: Imprecision, between variety and uniformity: the conjugate pairs. In J.J. Jadacki, W. Strawinski, eds. *In the World of Signs*, Poznan Studies in the Philosophy of the Sciences and the Humanities 62, 1998, 59–72, Rodopi, U.S.A. Solomon Marcus, *Controverses in science and in engineering* (in Romanian) Technical Publ. House, Bucharest, 1990.

An approximate algorithm for NP-complete optimization problems exploiting P-systems

Taishin Y. Nishida

Faculty of Engineering
Toyama Prefectural University
Kosugi-machi, 939-0398 Toyama, Japan
E-mail: nishida@pu-toyama.ac.jp

Abstract

A new approximate algorithm for optimization problems, called membrane algorithm, are proposed, which is an application of G. Păun's membrane computing or P-system. Membrane algorithm consists of several membrane separated regions and a subalgorithm and a few tentative solutions of the optimization problem to be solved in every region. Subalgorithms improve tentative solutions problem simultaneously. Then the best and worst solutions in a region are sent to adjacent inner and outer regions, respectively. By repeating this process, a good solution will appear in the innermost region. The algorithm terminates if a terminate condition is satisfied. A simple terminate condition is the number of iterations, while a little sophisticated condition becomes true if the good solution is not changed during a predetermined period. Computer experiments show that the algorithm solves the traveling salesman problem as good as simulated annealing algorithm.

Key words: approximate algorithm, traveling salesman problem, P-system

1 Introduction

Studies on approximate algorithms for NP-complete problems are a very important issue in computer science because ([1, 2, 6]):

- There are thousands of NP-complete problems.
- Almost all NP-complete problems correspond to practical problems.
- There are very few (I think no) expectations for P = NP, or strictly solving NP-complete problems in polynomial time.

We suggest a new approximate algorithm for solving NP complete optimization problems. The algorithm uses P-system paradigm [4]. Then it is called *membrane algorithm*. Membrane algorithm borrows nested membrane structures, rules in membrane separated regions, and transporting mechanisms through membranes from P-systems. Membrane algorithm remakes these components to solve NP-complete optimization problems approximately.

In the next section, the outline of membrane algorithm is explained. Details membrane algorithm are defined in order to solve the traveling salesman problem approximately in Section 3. The section also describes results of computer experiments under the definitions. An advanced membrane algorithm is mentioned in Section 4.

2 The outline of membrane algorithm

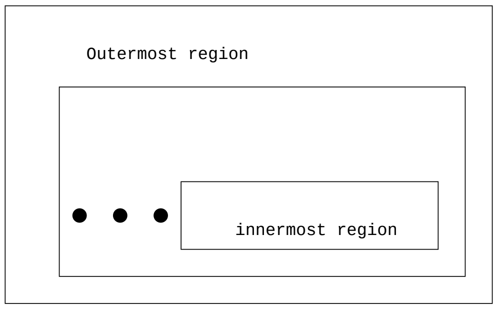


Figure 1: Membrane structure of the suggested algorithm.

Here we explain the new algorithm, called *membrane algorithm*. Membrane algorithm consists of three different kinds of components:

1. A number of regions which are separated by nested membranes (Figure. 1).
2. For every region, a subalgorithm and a few tentative solutions of the optimization problem to be solved.
3. Solution transporting mechanisms between adjacent regions.

After initial settings, membrane algorithm works as follows:

1. For every region, the solutions are updated by the subalgorithm at the region, simultaneously.
2. In every region, the best and worst solutions, with respect to the optimization, are sent to the adjacent inner and outer regions, respectively.
3. Membrane algorithm repeats updating and transporting solutions until a terminate condition is satisfied. A simple terminate condition is the number of iterations, while a little sophisticated condition becomes true if the good solution is not changed during a predetermined period.

The best solution in the innermost region is the output of the algorithm.

Membrane algorithm can have a number of subalgorithms which are any approximate algorithm for optimization problems, for example, genetic algorithm, tabu search, simulated annealing, local search, and so forth. The algorithm is expected to be able to escape from local minima by using a subalgorithm which likes random search at outer regions. On the other hand, the algorithm can improve good solutions in the inner regions by a subalgorithm which likes local search. So, assigning appropriate subalgorithms for a given problem, performance of the algorithm will be excellent.

Because the subalgorithms are separated by membranes and communications occur only between adjacent regions, membrane algorithm will be easily implemented in parallel, distributed, or grid computing systems. This is the second superior point of the algorithm.

3 First experiment of membrane algorithm solving traveling salesman problem

In this section we fix components of membrane algorithm to solve traveling salesman problem (TSP for short). Then we implement and experiment the algorithm on a computer.

3.1 Details of the algorithm

Let m be the number of membranes and let region 0 be the innermost and region $m-1$ be the outermost regions, respectively.

An instance of TSP with n nodes contains n pairs of real numbers (x_i, y_i) $(i = 0, 1, \dots, n-1)$ which correspond to points in the two dimensional space. The distance between two nodes $v_i = (x_i, y_i)$ and $v_j = (x_j, y_j)$ is the geometrical distance $d(v_i, v_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$. A solution is a list of nodes $(v_0, v_1, \dots, v_{n-1})$. The *value* of a solution $v = (v_0, v_1, \dots, v_{n-1})$ denoted by $W(v)$ is given by

$$W(v) = \sum_{i=0}^{n-2} d(v_i, v_{i+1}) + d(v_{n-1}, v_0).$$

For two solutions u and v, v is better than u if $W(v) < W(u)$. The solution which has the minimum value in all possible solutions is said to be the *strict solution* of the instance. A solution which has a value close to the strict solution is called an approximate solution.

The algorithm has one tentative solution in region 0 and two solutions in regions 1 to $m-1$.

We use a tabu search as the subalgorithm in the innermost region, region 0. Tabu search searches a neighbour of the tentative solution by exchanging two nodes in the solution. In order to avoid appearing the same solution twice, tabu search has a tabulist which consists of nodes already exchanged. Nodes in the tabulist are not exchanged again. Tabu search resets the tentative solution and the tabulist if one of the three conditions occurs:

1. The value of the neighbouring solution is less than that of the tentative solution. The neighbouring solution becomes the tentative solution.

2. The value of the best solution in the region 1 is less than that of the tentative solution. The best solution in the region 1 becomes the new tentative solution.

3. Neighbour search exceeds a predetermined turns (in this case $\frac{n}{5}$). The tentative solution remains. Only tabulist is reset.

In case 3, no improvement occurs. But tabu search tries to search other neighbours, since there are many unsearched neighbours.

The tentative solutions in regions 1 to $m-1$ (there are two solutions in each region) are improved by a subalgorithm summerized below:

1. If the two solutions have the same value, then a part of one solution (which is selected probabilisticly) is reversed.

2. Recombinates the two solutions and makes two new solutions.

3. Modifies the two new solutions by point mutations. In the i-th region, a mutation occurs under probability $\frac{i}{m}$.

Obviously the subalgorithm described above resembles genetic algorithms. But the subalgorithm always recombinates the two solutions in a region while genetic algorithms randomly select solutions to be recombinated. If the two solutions in a region are identical, recombination makes no new solutions. The step 1 avoids this case and introduces a new solution using reverse operation, which is a kind of mutation.

The overall algorithm looks like:

1. Given an instance of TSP.

2. Randomly makes one tentative solution for region 0 and two tentative solutions for every region 1 to $m-1$.

3. Repeats 3.1 to 3.3 for d times (d is given as a parameter).

   3.1 Modify tentative solutions simultaneously in every region using the subalgorithm at the region.

   3.2 For every region i ($1 \leq i \leq m-2$), sends the best solution of the solutions in the region (old solutions and modified solutions) to region $i-1$ and the worst solution to region $i+1$. (In region 0, sends the worst solution to region 1 and in region $m-1$, sends the best solution to region $m-2$.)

   3.3 For every region 1 to $m-1$ erases solutions but the best two.

4. Outputs the tentative solution in region 0 as the output of the algorithm.

In the above algorithm, steps 3.2 and 3.3 correspond to solution transporting mechanisms between adjacent regions.

3.2 Computer experiments

Table 1: Results of membrane algorithm and a simulated annealing (SA) for the benchmark problem eil51 (51 nodes). Membrane algorithm repeat step 3 40000 times. The number of trials of membrane algorithm is 10. Membrane 2, 10, 30, and 50 stand for the algorithms with 2, 10, 30, and 50 regions, respectively.

Algorithm	Membr. 2	Membr. 10	Membr. 30	Membr. 50	SA
Best	440	437	433	429	430
Average	544	450	442	435	438
Worst	786	457	450	444	445

Table 2: Results for benchmark problem kroA100 (100 nodes). 100000 iterations and 10 trials.

Algorithm	Membr. 2	Membr. 10	Membr. 30	Membr. 50	SA
Best	24524	22319	21770	21651	21369
Average	32973	23422	23200	22590	21763
Worst	49667	24862	23940	24531	22564

We have implemented the algorithm using Java programming language. By using Java, modifications of the algorithm have been easily tested on a computer. For example, we have implemented several recombination methods and have found that edge exchange recombination (EXX) [3] exhibits the best performance.

Tables 1 and 2 show results of the program for TSP benchmark problem eil51[4] and kroA100[5] from TSPLIB [5]. Results of simulated annealing from [7] are also shown in the tables.

Figure 2 shows changes of the average value of solutions for kroA100 problem solved by membrane algorithm with 50 membranes. One can see that the algorithm converges to considerable good solutions in a few steps, about 2000 to 3000 steps.

[4]The value of the optimum solution is 426.

[5]The value of the optimum solution is 21282.

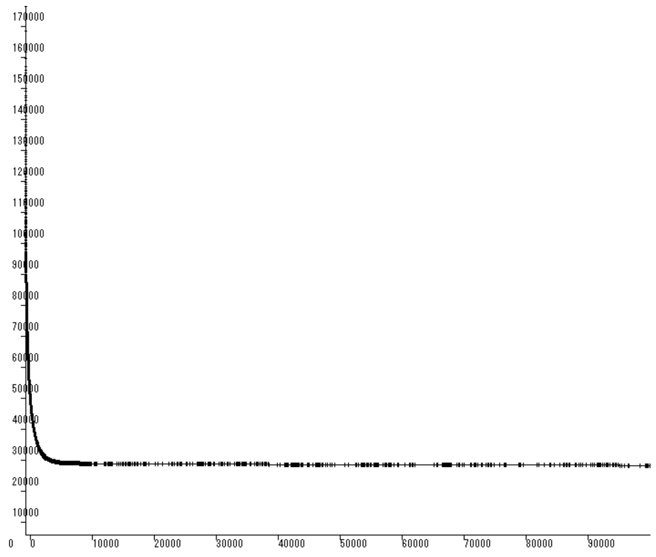


Figure 2: Changes of the average value of solutions for kroA100 problem solved by membrane algorithm with 50 membranes.

4 An improved membrane algorithm

In this section we discuss an improved membrane algorithm, called compound membrane algorithm, which corresponds to a tissue P-system.

Compound membrane algorithm has two phases (Figure 3). In the first phase, a number of membrane algorithms make good solutions from randomly generated initial solutions. The good solutions, in turn, become the initial solutions of the second phase. And a better solution is obtained.

We examine compound membrane algorithm with the following parameters:

- Number of membrane algorithms in the first phase is 100.
- All membrane algorithms have 50 membranes.
- Each membrane algorithm in the first phase terminates if the best solution does not improved during 500 iterations[6].
- The membrane algorithm in the second phase terminates if the best solution does not improved during 5000 iterations[6].

[6]These numbers are selected according to the feature that membrane algorithm converges fast (Figure 2).

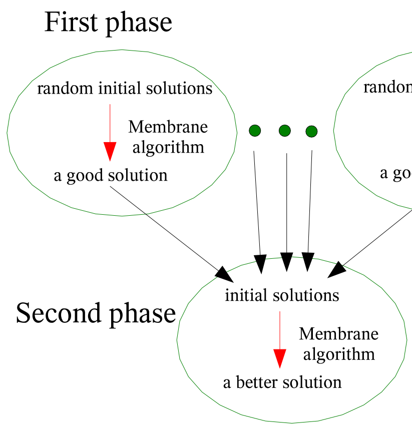


Figure 3: Compound membrane algorithm.

Table 3: Results of compound membrane algorithm. Trials of compound and simple membrane algorithms are 10.

	eil51			kroA100		
	compound	membrane 50	SA	compound	membrane 50	SA
best	429	429	430	21431	21651	21369
average	431	435	438	21616	22590	21763
worst	435	444	445	21816	24531	22564

Results of computer experiments of compound membrane algorithm are shown in Table 3. We can see that compound membrane algorithm always outputs almost strict solutions.

On a single processor, computation time of compound membrane algorithm, of course, is much longer than that of simple membrane algorithm. But, because membrane algorithms in the first phase work completely independent, compound membrane algorithm will easily be implemented on distributed computing system and computation time will be twice as short as that of simple membrane algorithm.

5 Conclusion

We have proposed and implemented a new algorithm, called membrane algorithm, for solving NP-complete optimization problems. Computer experiments have shown that membrane algorithm gets as good approximate solutions for

TSP as simulated annealing algorithm. Convergence of membrane algorithm is fast. An improved membrane algorithm, compound membrane algorithm, always gives almost strict solutions for TSP.

References

[1] C. A. Floudas and P. M. Pardalos (eds), *Encyclopedia of Optimization* (Kluwer, Dordrecht, 2001).

[2] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, (Freeman, 1979).

[3] K. Maekawa *et. al.*, A solution of traveling salesman problem by genetic algorithm (in Japanese), *SICE*, **31**, 598–605, 1995.

[4] Gheorghe Păun, Computing with membrane, *Journal of Computer and System Sciences*, **61**, 108–143, 2000.

[5] Gerhard Reinelt, TSPLIB, URL http://www.iwr.uni-heidelberg.de/group/comopt/software/TSPLIB95/

[6] Arto Salomaa, *Computation and Automata*, (Cambridge University Press, Cambridge, 1985).

[7] M. Yoneda, URL http://www.mikilab.doshisha.ac.jp/dia/research/person/yoneda/research/2002_7_10/SA/07-sareslut.html

Fuzzy P systems and fuzzy rule-based decisionmaking systems Abstract

Adam Obtułowicz

Institute of Mathematics
Polish Academy of Sciences
e-mail: `adamo@impan.gov.pl`

The application of Petri nets to model rule-based decisionmaking systems, known since late 70s of XX century (cf. [1], [2], [4]), and the relationship of P systems and Petri nets described in [3] give rise to the following conclusion explained in the present lecture and containing the proposals of the future more detailed investigations.

P systems, eventually their modifications or fuzzy counterparts can be used for modelling rule-based decisionmaking by a hierarchically organized system of many single (separate) rule-based decisionmaking systems modelled by Petri nets[7], respectively such that

— the single rule-based decisionmaking systems, belonging to the whole hierarchically organized system are associated to (or placed in) hierarchically organized ambients (or the regions of membranes which form a membrane structure—a tree), respectively,

— the outputs of the single rule-based decisionmaking system associated to an ambient (or placed in the region of a membrane) m may coincide only with some places of the Petri net modelling the single rule-based decisionmaking system associated to an ambient (or a membrane) immediately neighbouring with m, i.e. the ambient immediately containing m or an ambient immediately contained in m.

It seems that it is more natural and less elaborate to formulate the decision rules of such hierarchically organized system in the manner of evolution rules of P system and then to transform the obtained P system to an appropriate

[7]where the places and the transitions of Petri nets correspond to decision conditions and decision rules, respectively.

hierarchically organized system of Petri nets, like in [3], in order to investigate, for instance, reachability problem by using the known methods of Petri net theory.

References

[1] Shyi-Ming Chen, Jyh-Sheng Ke, Jin-Fu Chang, *Knowledge Representation Using Petri Nets*, IEEE Transactions on Knowledge and Data Engineering, vol. 2, No. 3, September 1990, pp. 311–319.

[2] C. G. Looney, *Fuzzy Petri Nets for Rule-Based Decisionmaking*, IEEE Transactions on Systems, Man, and Cybernetics, vol. 18, No. 1, February 1988, pp. 178–183.

[3] A. Obtułowicz, *Mathematical models of uncertainty with a regard to mem brane systems*, Natural Computing, 2, 2003, pp. 251–263.

[4] W. Pedrycz, H. Camargo, *Fuzzy timed Petri nets*, Fuzzy Sets and Systems, 140, 2003, pp. 301–330.