2014 Isonomia – Epistemologica
Rivista online di Filosofia
Università degli Studi di Urbino Carlo Bo

Membrane Computing: from biology to computation and back

Paolo Milazzo
University of Pisa
milazzo@di.unipi.it

Abstract

Natural Computing is a field of research in Computer Science aimed at reinterpreting biological phenomena as computing mechanisms. This allows unconventional computing architectures to be proposed in which computations are performed by atoms, DNA strands, cells, insects or other biological elements. Membrane Computing is a branch of Natural Computing in which biological phenomena of interest are related with interactions between molecules inside cells. The research in Membrane Computing has lead to very important theoretical results that show how, in principle, cells could be used to solve any (computable) computational problem with performances that cannot be obtained by conventional computers. However, the implementation of a cell-based computational architecture seems not easily achievable. On the other hand, models of Membrane Computing have found an alternative application to the description of biological systems, with the aim of developing simulators and other analysis tools for the study of biological problems.

1. Introduction

Results obtained in the first half of the last century in the Theory of Computability tell us that not all computational problems are *computable*, namely can be solved by a rational mechanizable computational procedure. Moreover, computable computational problems are in turn divided into *tractable* and *intractable* problems, depending on how the number of computational steps (i.e. elementary operations) necessary to solve an

Paolo Milazzo
"Membrane Computing: from biology to computation and back"

Università degli Studi di Urbino Carlo Bo
http://isonomia.uniurb.it/epistemologica

instance of the problem increases when the size of (a suitable representation of) the instance itself increases (*asymptotic complexity*).

For example, computing the minimum value of a sequence of n integer numbers is a computational problem that can be solved by an algorithm executing a number of computational steps proportional to n. (It is enough to read one by one the values of the sequence and compare each value with the minimum of the previous ones.) Instead, the ordering of a sequence of n integers is a problem that can be solved by algorithms executing at most a number of steps proportional to $n \cdot log(n)$. The game *Tower of Hanoi*[1], is a computational problem the solution of which requires a number of steps (moves) proportional to 2^n.

Usually, a computational problem is considered as tractable if its asymptotic complexity can be expressed as a polynomial function. Computational problems with exponential complexity (like the tower of Hanoi) are hence considered as intractable. In practice, this means that even the most efficient supercomputer would require times in the order of billions of years to solve non-small instances of these problems.

An interesting class of computational problems is that of *NP-complete* problems. For the solution of NP-complete problems only algorithms with exponential complexity are known[2]. On the other hand, there is no formal proof that a polynomial solution does not exist for them[3]. It is worth noting also that NP-complete problems can be reduced one another, and hence a polynomial solution for one of them could be used to solve efficiently all of them. See Garey & Johnson (1979) for a review on the topic of NP-completeness.

Examples of NP-complete problems are determining whether a Hamiltonian path or cycle exists in a given graph, determining the satisfiability of a given boolean formula (SAT) and the well-know Sudoku puzzle. A feature of these problems (and of all the other NP-complete problems) is that given a candidate solution of one instance of one of them it is possible to *check in polynomial time whether such a candidate solution is*

[1] Consisting of a structure of three rods in which n disks of different sizes have to be moved from the first rod to the third one. Disks have to be moved one by one from one rod to an adjacent one without placing a small disk over a bigger one.

[2] This is not completely true. There exist solutions of a few NP-complete problems the complexity of which is subexponential, but superpolynomial. See e.g. Deĭneko *et al.* (2006) where an algorithm to solve the Hamiltonian cicle problem in planar graphs is proposed with an asymptotic complexity proportional to $2^{\sqrt{n}}$.

[3] Determining whether NP-complete problems can be solved in polynomial time is one of the major open problems in Computer Science (known as $P=NP$).

actually a solution or not. For instance, in the case of Sudoku checking whether a candidate solution is correct is intuitively rather easy: it is enough to check whether each row, column and 3x3 square of the matrix contains all numbers from 1 to 9.

The non-existence of efficient solutions to NP-complete problems has not been proved. However, the (supposed) high complexity of such problems and the possibility of easily checking candidate solutions has made some NP-complete problems very useful to develop technologies of significant socio-economic impact, in particular in the context of *cryptography*.

An approach to the efficient solution of intractable problems (in particular NP-complete ones) consists in considering new computing architectures that are not based on the simple sequential execution of a number of computational steps. An idea could be to exploit infrastructures based on *parallel computing* technologies and programming methodologies. Such technologies and methodologies allow performances of programs solving several computational problems to be improved by distributing the computational workload among several different processors working (almost) independently. Anyway, parallel computing is not a solution that in general allows reducing significantly the asymptotic complexity of the considered computational problems. This is due to the fact that computational resources are in practice always limited (e.g. by the number of available processors or by the complexity of orchestrating the activities of processors) whereas the size of an instance of a computational problem can be arbitrarily large. Even the best parallel algorithm applied to solve a large enough instance of a (non-trivial) computational problem will encounter the problem of saturation of the available computational resources.

In order to efficiently solve intractable problems parallel computing architectures with unbounded parallel computational resources would be needed. Several research efforts have been aimed at finding such ideal computing architectures in *unconventional* contexts. In particular, the research field known as *Natural Computing* (Rozemberg *et al.*, 2012) aims at reinterpreting biological phenomena as (parallel) computing mechanisms. Natural Computing includes several different approaches that differ in the kind of considered biological phenomena (e.g. related with DNA, with the interactions between molecules and membranes, with the interactions among cells, etc...). All of these approaches actually deal with *complex systems* where an extremely high (virtually unbounded) number of entities (e.g. DNA strands, molecules, cells, etc...) interact each other autonomously (hence in parallel). The final aim of Natural Computing is hence to find

valid interpretations as computational steps for the interactions among the considered biological entities. This would allow the extremely high level of parallelism of biological systems to be exploited to solve computational problems without the limitations of traditional parallel computing architectures.

Membrane Computing (Păun, 2000 and 2002) is a branch of Natural Computing in which biological phenomena of interest are related with interactions between molecules inside cells, where cells are internally organized in separate compartments given by a *membrane structure*. Models of computation defined in the context of Membrane Computing are called *membrane systems* (or *P systems*, by the name of their inventor Ghorghe Păun). Such models consider different ways of combining basic biological phenomena and provide suitable interpretations of such phenomena as computing mechanisms. The result is that many of these combinations of biological phenomena turn out to have computational capabilities analogous to those of a Turing machine, namely they are *Turing-complete*. Moreover, it has also been shown that some membrane systems exploiting phenomena of membrane division allow the intrinsic parallelism of biological systems to be completely exploited. Indeed, they have been shown to be able to solve NP-complete problems in polynomial time (Zandron *et al.*, 2001).

The computing capacities and the theoretical properties of membrane systems have been deeply investigated, and are still the subject of a very active research field. However, after more than ten years of research no biological implementation of any algorithm based on membrane systems has been proposed. The main difficulty in bringing membrane systems from theory to practice is that the biological mechanisms exploited in the theory are assumed to be very precise when in practice they are extremely subject to stochastic noise, environmental conditions, interferences with other mechanisms, and so on.

Very recently, research on membrane systems has found a way to bring such systems back to reality that was probably unexpected at the beginning. The proposal of such a way back is related with the recent developments in molecular and cellular biology. In these fields new high-throughput analysis techniques (e.g. in genome sequencing) are providing a huge amount of genomic and biochemical data the interpretation of which requires new modelling and analysis approaches at a system level (*systems biology*). The ability of membrane systems to deal with biological mechanisms in a simple and formal way makes them suitable to be used for the aims of systems biology, namely as models of biological processes. It is now rather common to see membrane systems used as a formal notation for the description and

analysis of cellular pathways. More surprisingly, membrane systems found successful practical application not only in the context of systems biology, but also in other biological contexts such as ecosystem modelling. Indeed, membrane systems turn out to be suitable for modelling population dynamics in general, independently from the specific context (cell biology, ecology, socio-economic, etc...).

2. From Biology to Computation: Membrane Systems as Models of Computation

Membrane systems (or P systems), initially proposed and developed in Păun (2000) and Păun (2002), are mathematical objects representing *distributed computing devices* inspired by the structure and the functioning of living cells. The key elements of a membrane system are:

i. Membranes (that create compartments used to distribute computations);
ii. Multisets (abstractions of chemical solutions that are used as data);
iii. Evolution rules (abstractions of chemical reactions that are used as programs).

Membranes of a membrane system have a hierarchical structure with a top-level membrane representing, for instance, the external membrane of a cell. Each membrane of a membrane system contains a multiset of objects, a set of evolution rules, and possibly other membranes. An example of membrane system represented as a diagram[4] is shown in figure 2.1, where membranes are depicted as boxes (identified by a natural number), objects as lower-case letters and evolution rules as pairs of multisets of objects separated by an arrow.

[4] We do not show the mathematical representation of membrane systems since it is not essential for the aims of this paper.

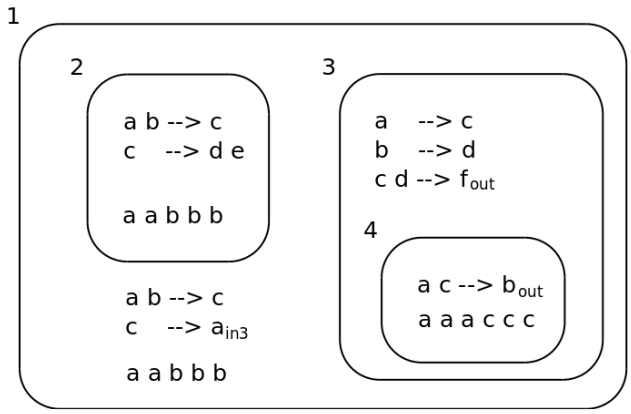

Fig. 2.1 Example of membrane system consisting of four membranes, eight evolution rules and sixteen objects. Subscript *in3* means that the object produced by the rule is sent inside membrane 3. Subscript *out* means that the object produced by the rule is sent into the outer membrane.

An evolution rule in a membrane can be applied only to objects in the same membrane, and the effect of application of a rule is the replacement of the objects mentioned in its left-hand side with the objects mentioned in its right-hand side. Objects produced by the rule may remain in the same membrane, or be sent out of the membrane, or be sent into the inner membranes. In the original definition of membrane systems (Păun, 2000), evolution rules are applied with *maximal parallelism*, namely it cannot happen that a rule is not applied when the objects needed for its triggering are available. Although other forms of parallelism have been considered (see e.g. Bernardini *et al.*, 2005 and Ciobanu *et al.*, 2007), maximal parallelism is the most distinguishing feature of membrane systems, as it allows Turing-completeness to be achieved even when very simple forms of evolution rules are considered (e.g. catalytic rules[5]). Maximal parallelism has the following implications: (i) more than one rule can be applied (on different objects) in the same computation step and (ii) each rule can be applied more than once in the same step (on different objects).

A *computation* of a membrane system is a sequence of steps in which at each step evolution rules are applied according to maximal parallelism.

[5] Rules with just two objects in their left-hand side in which one of the two is left unchanged by the rule, namely is present also in the right-hand side.

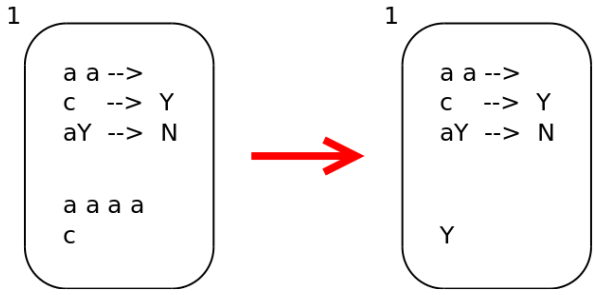

Fig. 2.2 Example of computation of a simple membrane system that tests whether a given natural number *n* is even. Case *n=4*.

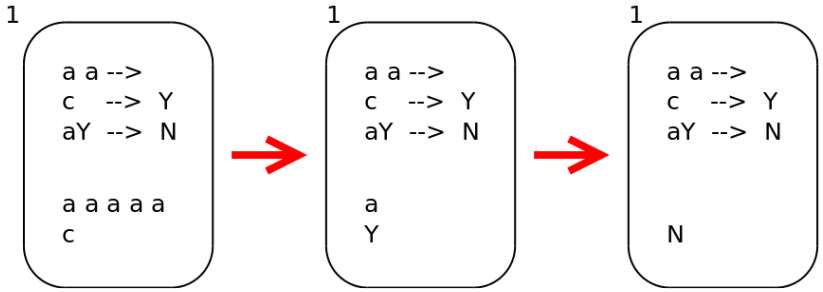

Fig. 2.3 Example of computation of a simple membrane system that tests whether a given natural number *n* is even. Case *n=5*.

Simple examples of computations of a membrane system are shown in figures 2.2 and 2.3. The membrane system considered in both figures consists of a single membrane. The aim of such a system is to check whether a given natural number *n* is even. The input number *n* is encoded as a multiset of objects containing *n* copies of object *a*. The result of the computation is represented by the multiset of objects in the *final configuration* of the system, namely in the situation reached at the end of the computation when there are no further applicable rules. In particular, the multiset in the final configuration will contain either *Y* or *N* depending on whether *n* is even or odd, respectively.

Figure 2.2 shows a computation in the case of *n* even, in particular *n=4*. In this case the computation consists of a single step in which all instances of *a* disappear by a maximally parallel application of the first rule, which replaces pairs of instances of *a* by the empty multiset. At the same time, by maximal parallelism we have that also the second rule has to be applied, transforming the *c* into *Y*. The system reaches a configuration in which Y is

present and no further rule can be applied, hence the answer of the system is that the input value *n* is even.

On the other hand, figure 2.3 shows a computation in the case of n odd, in particular *n=5*. The first step is as in the previous state, but with the difference that one instance of *a* is not deleted (since they are deleted in pairs by the first evolution rule). As a consequence, the configuration reached after the first step is such that the third evolution rule can be applied. So the system performs one more step transforming *a* and *Y* in *N*. This time the configuration obtained is final, and the answer of the systems is that the input value *n* is odd.

This is how membrane systems compute, and this example, although simple, shows how important is the role of maximal parallelism for computing.

Programming membrane systems for non-trivial examples is very difficult since evolution rules are very simple. Variants and extensions of membrane systems obtained by considering different types of evolution rules have been proposed such as membrane systems with rule *priorities*, with *promoters* and *inhibitors*, with *dissolution* of membranes, with *symport/antiport* rules and with *active membranes*. A complete description of all of these variants can be found in Păun (2000). For example, an evolution rule with promoters is a rule having the form $\boldsymbol{u} \rightarrow \boldsymbol{v}|_{\boldsymbol{p}}$ where $\boldsymbol{u}$ and $\boldsymbol{v}$ are multisets of objects used as in a usual evolution rule, and $\boldsymbol{p}$ is a multiset of objects (called promoters) the presence of which enables the application of the rule. Moreover, objects mentioned as promoters (i) are not removed from the multiset the rule is applied to, and (ii) enable the parallel application of the rule as much as possible independently of how many copies of $\boldsymbol{p}$ are present in the system (one copy of $\boldsymbol{p}$ is enough to apply the rule in parallel as many times as possible). An example of a computation of a membrane system in which an evolution rule with promoters is used is shown in figure 2.4.

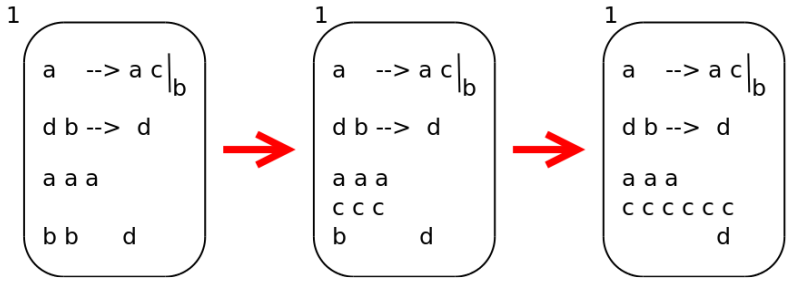

Fig. 2.4 Example of computation of a membrane system computing the product $n \times m$, where n is represented by the number of copies of object a, and m by the number of copies of object b. The result is given by the number of copies of object c in the final configuration.

All of the mentioned variants and extensions of membrane systems are Turing-complete. Most of them, however, are not able to solve NP-complete problems substantially more efficiently than Turing machines. The only exceptions (among the mentioned variants) are membrane systems with active membranes. Such system allow *membrane division* rules to be used having the form $[\boldsymbol{u}]_i \rightarrow [\boldsymbol{v}]_j[\boldsymbol{w}]_k$ where $\boldsymbol{u}$, $\boldsymbol{v}$ and $\boldsymbol{w}$ are multisets of objects and i, j and k are membrane indexes. The meaning of a membrane division rule is that a membrane with index i and containing the multiset of objects $\boldsymbol{u}$ (and any set of evolution rules) can be rewritten into two separate membranes with indexes j and k, containing multiset of objects $\boldsymbol{v}$ and $\boldsymbol{w}$, respectively, and both with the same set of evolution rules as the one in the original membrane i.

Membrane systems with active membranes have been proved to be able to solve NP-complete problems in polynomial time (Zandron *et al.*, 2001). This has been obtained by exploiting the ability of membrane division rules to produce 2^n different membranes in n subsequent steps. Indeed, when such rules are applied with maximal parallelism they can double the number of available membranes at each computation step. This feature of membrane division rules can be used to initialize (in polynomial time) as many different membranes as are the different potential solutions of the considered NP-complete problem. Now, each membrane can check (in parallel with all the other membranes) whether a given potential solution is correct, and this, as we discussed above, can be done in polynomial time. The computed solution is hence chosen among the potential solutions resulted to be correct, and the complexity of the whole procedure turns out to be globally polynomial.

3. From Computation to Biology: Membrane Systems as Models of Biological Systems

In the last few years there has been a strong technological improvement in molecular and cellular biology. A huge amount of data is now available on the structure of biological entities in cells. However, the functioning of such entities is still largely unknown. Cells are complex systems and they have to be studied as such. This is the aim of the new research field of systems biology in which computational models play an essential role: modelling and analysis tools allow researchers (i) to unambiguously formulate hypotheses on the behaviour of the biological phenomena of interest (by developing models), and (ii) to validate such hypotheses *in silico* by comparing results of simulations of their models with data obtained by observing the phenomena of interest.

Several modelling notations have been proposed to make descriptions of biological phenomena unambiguous and computer-friendly, for example by Danos & Laneve (2004), Regev *et al.* (2004) and Barbuti *et al*. (2006). In addition there exist notations that are variants of membrane systems, for example *Metabolic P Systems* proposed by Manca (2010) and *Dynamical Probabilistic P Systems* proposed by Pescini et al. (2006). Membrane systems have a simple notation that captures the essential elements of cellular processes (see for instance the model in figure 3.1). In order to properly describe the dynamics of cellular processes over time, quantitative extensions of membrane systems have been considered in which evolution rules are associated with reaction kinetics information. This enables the translation of membrane systems models into ordinary differential equations (ODEs) and into stochastic models, which in turn enables analysis of models by means of simulations (and not only).

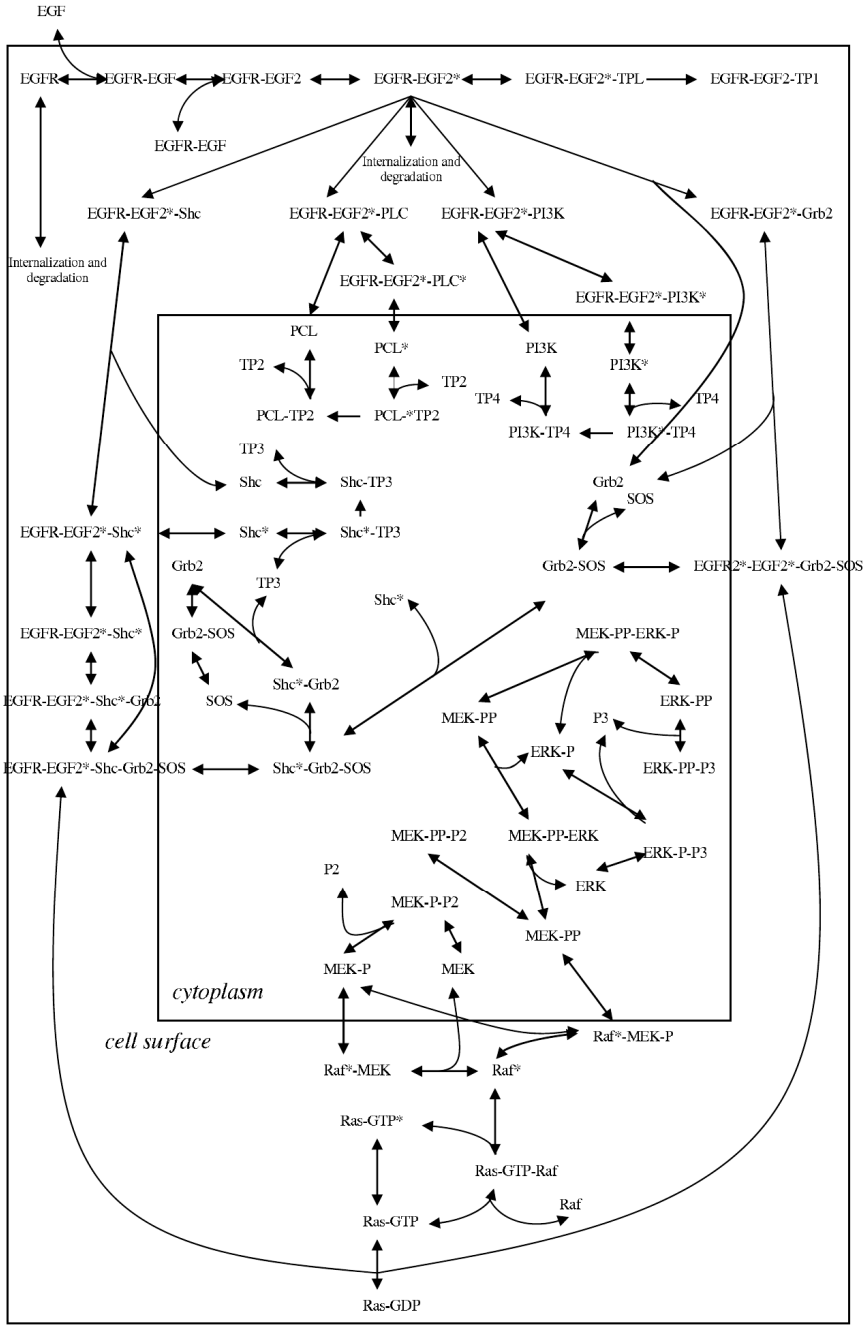

Fig. 3.1 Membrane system modelling the EGF signalling pathway, from Pérez-Jiménez & Romero-Campero (2005).

The simplicity of the membrane systems notation suggested their application also to the modelling of other kinds of system. In particular, they have been successfully applied to the modelling of population dynamics and ecosystems. The idea is that individuals of a population can be modelled as objects whereas actions and interactions can be modelled by means of evolution rules. Moreover, the morphology of the population territory can be (roughly) modelled by means of membranes. If a finer description is necessary of the territory and of the positions in it of the population individuals, the *spatial* extension of membrane systems proposed by Barbuti *et al*. (2006) can be used.

Maximal parallelism is particularly suitable to describe the behaviour of individuals in ecosystems. Indeed, it is often the case that the biology of the modelled species is such that the development of individuals consists in a sequence of clearly defined stages. Moreover, also the life of adult individuals is often organized as a repetition of periodic seasons in which all of the individuals are involved in a specific activity (e.g. reproductive seasons, hibernation, etc...). These situations in which the dynamics of the population is organized in stages or seasons in which all of the individuals are involved in some specific activity can be modelled by maximally parallel applications of evolution rules describing the fates of the individuals at the end of each stage or season.

The modelling of populations and ecosystems by means of membrane systems follows the *Individual Based Modelling (IBM)* approach (Grimm & Railsback, 2005), although only partially. Evolution rules describe the activities and the interactions from an individual viewpoint. Hence, the dynamics of the population emerges from events described at the individual level. However, individuals with the same observable characteristics (e.g. same sex and age class) are indistinguishable in the model since they are described by two instances of the same object.

Examples of evolution rules for population dynamics are as follows. Assume M and F to be objects representing adult male and female individuals, respectively. Moreover, let O to represent an offspring and P a predator. Mating and birth event can be modelled as $MF \rightarrow MFO$, growth of a female offspring as $O \rightarrow F$, death of an adult male as $M \rightarrow$, and predation as $PM \rightarrow P$.

Computational models of populations and ecosystems allow the factors governing population growth and extinction to be better understood. Moreover, they also allow the dynamics of an endangered population to be predicted in order to plan control policies or reintroduction/reinforcement actions. An example of use of membrane systems as a modelling tool for

ecosystem is the paper by Cardona *et al.*, (2011). In such a paper a membrane system is used to model the ecosystem related to the Bearded Vulture in the Pyrenees: an endangered species feeding on bone remains of wild and domestic ungulates. Results of simulation of the model compared with field data are used to discuss causes of vulture extinction.

4. Conclusions

Membrane Computing is a research area in which many interesting theoretical results have been obtained. Such results represent important contributions for Computer Science, in particular in the areas of Theory of Computability and Formal Languages Theory. However, the implementation of new computing devices constituted by biological material and based on the computing mechanisms studied in Membrane Computing seems still far from been possible. Membrane systems, however, have found applications that were probably unexpected at the beginning. In fact, they turn out to be suitable as model notation for several different classes of biological systems. The evolution of research in Membrane Computing is hence an example of how foundational research if well conducted can lead not only to important theoretical results, but also to applications often initially not foreseen.

References

Barbuti, R., Maggiolo-Schettini, A., Milazzo, P., & Troina, A., 2006, «A Calculus of Looping Sequences for Modelling Microbiological Systems», *Fundamenta Informaticae*, vol.72, n. 1, pp. 21-35.

Barbuti, R., Maggiolo-Schettini, A., Milazzo, P., Pardini, G., & Tesei, L., 2011, «Spatial P systems», *Natural Computing*, vol. 10, n. 1, pp. 3-16.

Bernardini, F., Romero-Campero, F. J., Gheorghe, M., Perez-Jimenez, M. J., Margenstern, M., Verlan, S., & Krasnogor, N., 2005, «On P Systems with Bounded Parallelism», In: *Symbolic and Numeric Algorithms for Scientific Computing*, pp. 8-pp, IEEE.

Cardona, M., Colomer, M. A., Margalida, A., Palau, A., Pérez-Hurtado, I., Pérez-Jiménez, M. J., & Sanuy, D., 2011, «A Computational Modeling for Real Ecosystems Based on P Systems», *Natural Computing*, vol. 10, n. 1, pp. 39-53.

Ciobanu, G., Pan, L., Păun, G., & Pérez-Jiménez, M. J., 2007, «P Systems with Minimal Parallelism», *Theoretical Computer Science*, vol. 378, n. 1, pp. 117-130.

Danos, V., & Laneve, C., 2004, «Formal Molecular Biology», *Theoretical Computer Science*, vol. 325, n. 1, pp. 69-110.

Deĭneko, V.G., Klinz, B., Woeginger, G.J., 2006, «Exact Algorithms for the Hamiltonian Cycle Problem in Planar Graphs», *Operations Research Letters*, vol. 34, n.3, pp. 269–274.

Garey, M.R. & Johnson, D. S., 1979, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman.

Grimm, V., & Railsback, S. F., 2005, *Individual-based Modeling and Ecology*, Princeton university press.

Manca, V., 2010, «Metabolic P Systems», Scholarpedia, vol. 5, n. 3, 9273.

Păun, G., 2000, «Computing with Membranes», *Journal of Computer and System Sciences*, vol. 61, n.1, pp. 108-143.

Păun, G., 2002, *Membrane Computing. An Introduction*, Springer-Verlag, Berlin.

Pérez-Jiménez, M. J., & Romero-Campero, F. J., 2005, «A Study of the Robustness of the EGFR Signalling Cascade using Continuous Membrane Systems», In: *Mechanisms, Symbols, and Models Underlying Cognition*, Springer Berlin Heidelberg, pp. 268-278.

Pescini, D., Besozzi, D., Mauri, G., & Zandron, C., 2006, «Dynamical Probabilistic P Systems», *International Journal of Foundations of Computer Science*, vol. 17, n. 1, pp. 183-204.

Regev, A., Panina, E. M., Silverman, W., Cardelli, L., & Shapiro, E., 2004, «BioAmbients: an Abstraction for Biological Compartments», *Theoretical Computer Science*, vol. 325, n. 1, pp.141-167.

Rozenberg, G., Back, T., & Kok, J. (Eds.), 2012, *Handbook of Natural Computing*, Springer Verlag

Zandron, C., Ferretti, C., & Mauri, G., 2001, «Solving NP-complete Problems Using P Systems with Active Membranes», *Unconventional Models of Computation*, pp. 289-301, Springer London.