Chapter 3
P Systems for Biological Dynamics

Luca Bianco, Federico Fontana, Giuditta Franco, Vincenzo Manca

University of Verona
Department of Computer Science
strada Le Grazie 15, 37134 Verona, Italy
{bianco,fontana,franco}@sci.univr.it, vincenzo.manca@univr.it

Summary. P systems have clear structural analogies with the cell. However, certain difficulties arise when one attempts to represent a biomolecular process using these systems. This chapter suggests some ways to overcome such difficulties and to provide P systems with further functionalities aimed at increasing their versatility in the modeling of biomolecular processes. Concepts from state transition dynamics are taken to put P systems in a general analysis framework for dynamical discrete systems. An explicit notion of environment is proposed to provide P systems with a regulatory and constraining agent, as real biomolecular processes must deal with. The chapter focuses on a new rewriting strategy inspired by biochemistry, in which reactivities play a central role in driving the rules as it happens during biochemical reactions. Tests on an algorithm implementing rewriting with reactivities, realized on a simulator called *Psim*, show the capability of this algorithm to express several processes with precision, particularly those presenting oscillatory phenomena. Finally, an analysis of the process of leukocyte recruitment is also performed using *Psim*.

1 Introduction

We know that the simplicity of a computing device does not limit its power to solve problems, assumed that its design criteria follow certain specifications and enough space is provided to represent the data and to store the processing rules, as well as enough time is left to the device to compute the solution. This fact became clear after the Turing machine was designed and the proofs of universality and equivalence with other computing machines and formalisms (most of them being as simple as the architecture proposed by Alan Turing) were given. So, why a modern computer is in practice so different from a Turing machine, provided that it has the same computational power?

The answer is obvious: a modern computer architecture is much better interfaced with the external world than a Turing machine, and, consequently, a task can be more easily implemented over it. Algorithms and features that

would need an indefinite time to be shaped to run on a Turing or a Von Neumann machine, are rapidly implemented over a modern computer instead. On the other hand, Turing machines and equivalent "ideal" architectures proved to be invaluable for crossing the bridge between computer theory and practice. In some sense, we can say that ideal machines cannot be avoided, although any practical application of them must move through the existence of more elaborate systems.

In 1998 *P systems* were presented as a new model of computation [27]. We argue that this model can be considered and developed in such a way to become an analogue of Turing machines, playing the role of a mathematically idealized model for biological systems. The following discussion will provide specific arguments for this claim.

Before P systems, some other classes of rewriting systems have already shown the ability of expressing specific biological phenomena [32, 20, 13]. P systems move a step further: they have clear structural analogies with the cell, in particular they model several features of the biological membranes (for this reason they are often referred to as *membrane systems*). Moreover, the transitions happening in these systems recall certain evolution processes that take place in a living cell.

From a formal viewpoint, P systems satisfy a result of universality even in their basic definition [27]. In this sense they have all the computational power needed to capture a biomolecular process – provided that we are able to arrange it into an algorithmic procedure. In addition to this, the similarities existing between P systems and (at least some aspects of) biological cells might suggest that P systems are also able to represent the same process in a meaningful way, that is, not only to compute it as any universal machine would do, but also to provide potential insight on the biological mechanisms determining and controlling the process via the observation of the transitions of the system.

Unfortunately, for most of the classes of P systems considered so far, this is true only to some extent. Modeling specific biological activities inside a P system is not an easy task. A lot of alternative constructs derived from the basic definition of P system have been proposed, sometimes capturing crucial aspects of the biology of cells such as *thickness*, *polarity*, *transport* via *symport* and *antiport* rules, *catalysis*, *dissolution*, *polarity*, *permeability*, *inhibition*, *promotion*, communication via *carriers* and *energy* [29, 26, 25], sometimes importing paradigms coming from other formal systems having biological implications too, such as *splicing* and object–structuring (in form of strings) [30]. All these alternative constructs exhibit properties of universality, hence by all means they represent a first, necessary attempt to move P systems closer to the world of bio-molecules meanwhile preserving their computational power.

Nevertheless there are some aspects, that are crucial in almost any study of biomolecular processes, that the traditional formulations of P systems do

not take into major account (at least not so explicitly to turn into versatile constructs for biological applications):

- *Dynamics* of bio-systems. The halting of a P system tells that a computation has terminated successfully, but the dynamical behavior of biomolecular processes has a major relevance in the description of the processes themselves. This means that two or more processes that terminate with identical configurations may move through completely different transitions. Thus, in the context of living organisms it is more appropriate to consider the dynamical patterns of the "life" evolving in a given environment. The knowledge and classification of these patterns is a preliminary task for understanding or influencing some behaviors (possibly harmfully) for specific purposes.
- *Environmental energy* and *resources*. The resources available in the environment play a major role in the control a biomolecular process. The existence in the environment of elements, which can act as catalysts or provide the energy needed for the bio-chemical elements to react, can radically change the nature of a process. In particular, an environment which periodically feeds the system with resources can transfer properties of periodicity to the system as well.
- *Asynchronous* system control. Biomolecular mechanisms are the result of many individual local reactions, each of those being formed by processes whose extension is limited in time and space. These processes interact to each other by means of specific communication strategies, in a way that they finally exhibit a (sometimes surprising) overall coordination. In this sense, and despite this coordination, biomolecular processes are by all means asynchronous.

Clearly, these aspects are closely related one to the other: shifting the focus on the system dynamics means that less attention is payed to the final configuration of the system; meanwhile, the continuous control of the resources needed by the process to evolve is fundamental to drive the system dynamics along a specific trajectory. The environment itself has the role of a "supervisor" in the process control, since it becomes responsible of a sort of external input, whose effects in the system propagate via local reactions.

How today's P systems deal with the points just outlined? About the first point, we know that P systems are intended to "consume" the available resources in a maximally parallel way during the rewriting of symbols. Imposing this property, all the symbols that are present in the system at a given configuration become potential resources: they are consumed as many as possible, and new symbols are produced in consequence of that action. In other words, maximal parallelism constrains the system to consume all the available resources during a transition.

We know that we can regulate the system evolution by adding auxiliary symbols (and correspondent cooperative rules which use such symbols) or, alternatively, by providing the system with *priority* constraints on the rules,

to form (sometimes complex) relationships of precedence for the rewriting rules. But, this modification of the system structure is not guaranteed to have a correspondent biological counterpart. So, we are looking for an alternative, more biologically founded strategy for the regulation of parallelism.

About the second point, we know that non terminating processes are of key importance in the study of *periodicity* and *quasi–periodicity*, two aspects whose detection is important for understanding many biological processes [33]. From this viewpoint, many existing types of P systems do not provide versatile tools to handle periodicity.

The question of resource availability (this issue leads us to the third point) is unavoidable in the study of biomolecular phenomena. P systems, in their native definition, do not take any energetic constraint into account. So, it is not unfeasible for them to use indefinitely large amounts of resources to perform a computation: clearly, this cannot happen in nature. Further types of P systems have been proposed in which energy is considered as a constraining factor, although in those systems the environment, intended as a place where to exchange resources, does not play a central role [31, 12].

Finally, P systems are organized in a way that their evolution is synchronous, i.e., a global clock triggers the production of new symbols inside all membranes. This limits their versatility in modeling asynchronous phenomena.

Coming back to our initial considerations on ideal and practical computing devices, it is our opinion that P systems are still at an initial development stage. On the one hand, their simplicity does not limit their computational power and, in fact, this simplicity has allowed to prove important results of universality. On the other hand, much research still has to be done to get them closer to the world of (especially, but not only) biomolecular applications, meanwhile keeping them theoretically well–founded. Turing machines have found in modern PC's their applied counterpart: it is time, now, to look for ways that turn P systems into *real* biomolecular computing devices.

The lot of research aimed at proving the well–foundedness of the various types of P systems (see, e.g., [28]) is the background of the research presented in this chapter. However, we have focused our effort in addressing some novel theoretical and practical issues especially oriented to biomolecular computing – this applicative direction is followed also by other groups [6, 3, 1].

First, we consider a new perspective (for many aspects still in progress) according to which P systems are cast in a dynamical framework. In this perspective, we will characterize the transitions and the space state of a discrete system using *state transition dynamics* [23]. Then we will introduce *P systems with boundary rules* (shortly, PB systems) and *PB systems with environment* (shortly, PBE systems) [5, 4] as constructs in which the concept of *environment cycle* is proposed to represent cyclic biological behaviors; meanwhile it applies at an operative level the definition of periodicity and quasi–periodicity of state transition dynamics to P systems.

Next, we propose to observe the rules of a rewriting system from a different viewpoint: not only they produce new symbols starting from existing ones; they will also be part of a system reproducing a reaction, in which the application of every rule changes the relative amounts of reacting substances present before and after the production. Moreover, such relative amounts will influence the *reactivity* of the rules in a way that their application will be dependent on the substance concentration, as it normally happens in biochemical phenomena. An application of this model to some known bio-chemical problems – so far described in terms of differential equations – is proposed, in particular in the simulation of the *Brusselator*, a simplified model of the Belousov–Zabotinskii (BZ) reaction having great relevance in biochemistry [33, 15, 37, 24].

Finally, we look at an extended version of a P system in action, aimed at simulating some mechanisms happening in the human immune system when it activates the *leukocyte selective recruitment* against an inflammatory process.

Closing this section, we want to stress once again the importance that the dynamic characterization and the environment have in our vision of a class of P systems especially intended to serve as biomolecular computers:

- the system dynamics, as a way to capture recurrent (or however determined) behaviors by "clues", that are left even by those biomolecular processes that cannot be decoded due to their apparently chaotic behavior;
- the environment, as an entity that regulates parallelism, alters an otherwise unavoidable terminal state, provides resources, and acts as a de-localized control for the system; as we have seen, all these issues are intimately connected with the dynamics of our system.

Although this survey does not pretend to define a comprehensive application framework for membrane systems, nevertheless the authors hope that the issues proposed herein will suggest possible points of investigation from where to carry on some more applied research on P systems. In the following of this chapter we will constantly refer to the basic definitions and notation on P systems and on *multisets* introduced by Păun [28].

2 The Dynamics of Discrete Systems

Continuous systems are often described in terms of differential equations. A common strategy to figure out such equations consists in writing down equilibrium conditions for *infinitely small* physical units such as time units, $\mathrm{d}t$, and spatial volume units, $\mathrm{d}\boldsymbol{s}$. From there, a classic approach to discrete system modeling consists in picking up the continuous phenomenon (i.e., the differential problem describing it) and then producing a discrete model of it according to a given discretization method. Finally a simulation is run, provided that the discrete model respects certain stability conditions. This approach is largely used in the practice of system modeling.

A discrete model differs from the continuous phenomenon it comes from. Sometimes this discrepancy can be arbitrarily reduced, that is, the model precision is proportional to the granularity with which the continuous phenomenon is reproduced in the discrete domain.

Rather, here we are interested in those physical phenomena whose characterization (that is, the information needed to describe them) is *inherently* discrete. In this case a discrete model can represent the physical phenomenon completely. This is the case of many biomolecular processes (think, for instance, to DNA replication [30]).

For the above reasons – particularly for the last one – we are especially interested in discrete systems regardless of any specific relationship with a continuous system, and any prior argument on the precision of the discrete solution against the continuous one. Furthermore, in most cases of biological interest the discrete paradigm can be extended even to the values the system assumes during its evolution, in a way that numerical values are conveniently substituted by symbols.

The *state transition dynamics* formalism considers a system defined in a discrete domain, assuming discrete values. It studies properties such as state orbits and trajectories, periodicity, eventual periodicity and divergence, recurrence of states, attractors and fixed points. By means of this analysis we are able not only to characterize such properties, but also to make important considerations about determinism vs. nondeterminism, and about regularity vs. *chaos*.

To give an idea of the characterization given by state transition dynamics, here we report the most important definitions and results (sometimes in a quite informal presentation). For further details, discussions and mathematical insight we refer to [23] where a general approach to discrete system dynamics has been investigated in its formal and computational aspects.

Definition 1. *A state transition dynamics is a pair* (S, q) *where* S *is a set of states and* q *is a function from* S *into its power set:*

$$q : S \to \mathcal{P}(S) .$$

By calling *quasi state* any subset $X \subseteq S$, and extending the application of q over quasi states, i.e.,

$$q(X) = \bigcup_{x \in X} q(x) ,$$

then we map quasi states into quasi states by means of q to form *orbits*, and characterize specific *trajectories* along these orbits by means of the following definitions.

Definition 2. *An* X-orbit *is a sequence* $\{X_i\}_{i \in \mathbb{N}}$ *of quasi states such that*

$$\begin{array}{ll} X_0 = X, & i = 0, \\ X_i \subseteq q(X_{i-1}), & i > 0. \end{array} \tag{1}$$

An x-trajectory *is a function* $\xi : \mathbb{N} \to S$ *such that*

$$\begin{aligned} &\xi(0) = x, \\ &\xi(i) \ \in q\big(\xi(i-1)\big), \ \ i > 0. \end{aligned} \tag{2}$$

Let us denote with q^i the composition of q repeated i times, and let us define $q^*(x) = \bigcup_{i \in \mathbb{N}} q^i(x)$. We refer as *flights* and *blackholes* to the following special trajectories:

Definition 3. *An* x*-trajectory is an* x-flight *if it is an injective function on* $\mathbb{N}$*. An* x*-flight is an* x-blackhole *if* $q^*(x) \subseteq \xi(\mathbb{N})$*, where* $\xi(\mathbb{N})$ *is the image set of* ξ*.*

When S is made of symbolic values then the relation $y \in q(x)$ induced by q between two states, x and y, is conveniently expressed using the notation typical of rewriting systems: $x \to y$. Note that we can easily introduce non terminating computations as long as q is total.

It is clear that the notion of a dynamical system defined above is nondeterministic, because any state can transform into a set of possible states – though, an equivalently expressive deterministic system where states are the quasi states of the original system can be figured out. The nondeterministic aspect is essential for the modeling of many phenomena.

We now give a characterization of the evolution in these systems.

Definition 4. *An* X*-orbit is* periodic *if* $q^n(X) = X$ *for some* $n > 0$*. An orbit is* eventually periodic *if* $q^{n+k}(X) = q^k(X)$ *for some* $k, n > 0$*. In this case* k *is called the* transient *and* n *the* period*.*

Definition 5. *An* X*-orbit is* $\Omega\big(f(n)\big)$-divergent *with respect to a function* $\mu : S \to \mathbb{N}$*, called Ljapunov function, if* $\mu\big(q^n(X)\big)$ *has order* $\Omega\big(f(n)\big)$*. A similar definition holds for the order of divergence* $O\big(f(n)\big)$*.*

Definition 6. *A state* x *is a fixed point if the transition relation transforms it into itself deterministically, that is,* $q(x) = \{x\}$*.*

Periodicity and eventual periodicity are properties with a strong computational significance. It can be shown that, in a suitable computational framework where every machine finds a counterpart in a corresponding state transition dynamics, the periodicity decision problem turns out to be computationally equivalent to the termination problem [23]:

Proposition 2.1 *Given a computationally universal class of machines, then the (eventual) periodicity of the related dynamical systems is not decidable.*

Affine to periodicity (but weaker) is recurrence:

Definition 7. *A state* x *is* recurrent *if* $x \in q^n(x)$ *for some* $n > 0$*. A state* x *is* eternally recurrent *if for all* $n > 0$ *such that* $y \in q^n(x)$ *there is* $m > 0$ *such that* $x \in q^m(y)$*.*

A system dynamics is ultimately characterized by its *attractors*, that in very first approximation can be seen as quasi states in which the system must fall in the end. First of all, we say that an Y-orbit is *included* in an X-orbit if the former sequence is contained in the latter sequence, and *eventually included* if it is included in the X-orbit except for a finite number of quasi states.

We call *basin* a set $B \subseteq S$ such that $q(x)$ is included in B for every state $x \in B$. Inside a basin we possibly find an *attracting set* A, i.e., a subset which eventually includes the x-orbit of every state $x \in B$. If A is minimal under set inclusion, i.e., no subsets (even made of a single state) can be removed from A otherwise causing the lost of the attracting property, then our attracting set is an attractor.

A complete characterization of attractors requires more definitions than those recalled in this chapter; we refer to [23] for details. In particular, here we have only outlined the so–called *unavoidable* attracting sets, and corresponding attractors. It can be also shown (here we omit all the intermediate results, along with further definitions) that a state transition dynamics can have three different types of attractors:

1. *periodic attractors*, that is, periodic orbits (fixed point attractors are a special case);
2. *eternally recurrent blackholes*;
3. *complex attractors*, that is, a combination of the two previous cases.

The notion of an attractor opens a wider perspective upon the classical notion of calculus, which seems to fit better with a computational interpretation of biological systems. In fact, these systems do not compute states that encode results (according to the Turing's paradigm), rather they "compute" attractors, or stable regimes satisfying behavior requirements that respect certain conditions for life.

Particularly interesting are chaotic attractors. Life chooses forms approaching chaos, meanwhile tries not to fall into it. While getting closer to this total freedom, simple cycles take the rich and complex forms featured by evolution and adaptation. The expression *at the edge of chaos* [19], in fact, expresses the typical condition in which biological systems explore the space of computable forms moving along a threshold that lies in the middle between biological *status quo* and chaotic evolution, both of them destructive for a species. Though, at the edge of this threshold lies that constructive evolution life is constantly searching for.

The (nondeterministic) notion of orbit turns useful also in an attempt to give a characterization of *discrete chaos*. Looking at chaos from this perspective gives further insight on the meaning of nondeterminism.

Chaotic dynamical systems are characterized by the following features:

- Global recurrence. In a chaotic dynamical system the set of all states is its own attractor, which is also called a *strange* attractor. In other words,

a chaotic behavior is a global property that cannot be decomposed into distinct parts.

- Sensitivity to initial conditions. This requirement implies an exponential divergence of orbits where points that are "near" become exponentially "far" in time. This aspect can be viewed as an explosion of orbits, or as an "informational drift" along the orbits, and it is relative to some Ljapunov function with respect to some measure of distance between states.
- Ubiquitous periodicity. This property refers to the erratic aspect of chaos: orbits are wandering everywhere and forever, that is, explosions of orbits are mixed with orbit implosions in such a way that dynamics returns periodically onto itself, according to their intrinsic recurrence, but these periods are endlessly overlapping each other.

Such features can be expressed in the context of state transition dynamics [23].

The definition of chaos expressed in terms of state transition dynamics allows us to consider chaos of dynamical systems very similarly to the notion of deterministic chaos of continuous systems (logistic maps, Bernoulli shifts, Manneville maps [9, 7]), which, although ruled by very simple dynamics, present evident chaotic behaviors. What makes these systems intrinsically chaotic is the essential role of quasi states in their descriptions. In fact, their dynamical systems are defined on states given by real numbers, but these numbers are always expressed by some of their finite approximations, that is, rational numbers. In this sense, one may view an infinite set of states as a rational number, that is, a quasi state which comprises all the real numbers that, at some level of approximation, share the same rational number. Therefore, the sensitivity to initial conditions corresponds to the exponential growth of a Ljapunov function along the orbits associated to the finite approximations of the states of the system. Analogously, the overlapping of periods in these systems corresponds to the eventual intersections of their periodic orbits, when we consider the quasi states which correspond to the finite representations of their states.

In conclusion, what is called deterministic chaos does not differ from nondeterministic chaos. The difference is only a matter of the way orbits are defined. In deterministic chaos these are introduced by the intrinsic approximation of states; in the nondeterministic chaos of state transition dynamics, the orbits are defined by state transition relations that provide many possible states that can be reached from a single state. But, what is very important to remark is that neither is determinism synonymous of predictability, nor is nondeterminism synonymous of unpredictability. Indeed, a system that is deterministic but chaotic becomes unpredictable, and a nondeterministic system can be predictable in several aspects [14].

Similar behaviors have also been observed in other constructs, such as Kauffman Networks and cellular automata [17, 38, 39]. These systems show that many relevant characteristics of their dynamical behavior are conse-

quences of the relationships existing between the transition function and the state structure. Parameters like *connectivity*, *channeling*, *majority*, *input entropy* and figures taken from Derrida plots might inspire the search for similar quantities in P systems. Also, many concepts of Formal Language Theory can be revisited in the perspective of state transition dynamics: for instance, the languages generated by grammars or recognized by automata are special cases of attractors.

These considerations upon nondeterministic chaos in discrete systems allow to gain insight on those biomolecular behaviors that must be classified as chaotic. In particular, such a notion of chaos might help in identifying orbits from apparently "unreadable" biomolecular process dynamics, as life cycles depict periodicities that are masked or simply got blurred by nondeterministic shifts away from the main trajectories, but leave fingerprints of these trajectories along the way.

Conversely, it could enable to specify richer and more "open" dynamics than those defined by other representations, for instance the dynamics provided by the (deterministic) solution of a differential problem. In section 4 we will see some examples, both formulated in terms of a differential problem and analyzed using a discrete dynamical (string transition) system.

3 Resource Drawing from the Environment

The boundary of a P system against the external world is represented by the skin membrane. In most types of P systems every membrane limits the scope of the rules in a way that, by definition, they can generate and/or consume symbols only in the region delimited by their own membrane, and the skin does not make an exception to this definition. The environment has no special roles as well, but providing the (possibly infinite) amount of resources needed by the system to evolve, and receiving the (possibly indefinitely many) symbol–objects that, once properly decoded, give the result of the computation the system has performed.

PB systems, PBE systems, and PBE systems with resources [5, 4] enrich the P construct, by implementing the idea of giving a more active role to the boundary and to the environment. This idea has a strong foundation in some typical features that are often exhibited by biological systems, such as:

- *periodicity and quasi–periodicity.* Life is always related to temporal cycles where, even if some temporal irreversibility is intrinsic, many parameters change periodically and some basic rhythms are preserved;
- *stability and adaptability.* Biological systems tend, within some limits, to keep their "form" and their basic "behavior" even if their external world changes;
- *growth and degeneration.* A living organism is able to perform a correct life cycle when it maintains some basic oscillating reactions along time.

In this sense, periodical behavior, resource availability, and influence of the environment to the system are in close relationship to each other.

Recalling the basic construct of a P system, and in particular the notion of a *configuration* as a string μ encoding the membrane topology and the multiset contained in every membrane [27], then we define a *PB system* in the following way:

Definition 8. *A P System with Boundary Rules (PB system) is a construct:*

$$\Pi = (V, \mu_0, R, i_O),$$

where:

(i) V *is an alphabet of symbols;*
(ii) μ_0 *is the initial configuration;*
(iii) R *is a finite set of rules of the following two forms:*
- $xx' [_i y'y \rightarrow xy' [_i x'y$, *for* $x, y, x', y' \in V^*$ *and* $1 \leq i \leq m$ *(*communication rules*);*
- $[_i y \rightarrow [_i y'$, *for* $y, y' \in V^*$ *and* $1 \leq i \leq m$ *(*transformation rules*);*

(iv) $i_O \in \{1, \ldots, m\}$ *is the label of the output membrane.*

In addition to basic P systems we have essentially the communication rules of the form $xx' [_i y'y \rightarrow xy' [_i x'y$. By means of these rules we can move objects through membranes: if the membrane i contains the multiset $y'y$ and the multiset xx' is present outside the membrane i, then the multiset x' moves into the membrane i and the multiset y' is sent out from it; clearly, some of these multisets may be empty. The salient fact in the action of the communication rules is that they can "see" the immediate outside of the membrane region they belong to. Indeed, their nature recalls the *antiport* rules from of P systems with *linked transport* [26].

The computational universality of PB systems was proved by showing that they are able, using three membranes, to characterize the recursively enumerable sets of vectors of natural numbers [4].

We give the notion of *environment cycle of period* k as the infinite sequence where k multisets $\beta_0, \beta_1, \ldots, \beta_{k-1}$ occur periodically in time. Now we can define a *PBE system.*

Definition 9. *A PB System with Environment (PBE system) is a construct:*

$$\Pi = (V, \mu_0, R, E, R_E, i_O),$$

where:

(i) V, μ_0, R, i_0 *are as in Definition 8;*
(ii) E *is an environment cycle of period* k*;*
(iii) R_E *is a finite set of rewriting rules on multisets of the form* $x \rightarrow y$, *for* $x, y \in V^*$ *(*environment rules*);.*

The system configuration at time j, i.e., η_j, is related to the previous configuration η_{j-1} by the following relation, in which we make use of the dynamics functions q_E and q_R, related to R_E and R (see Definition 1), respectively mapping the environment configuration $\beta_{j-1}\gamma_{j-1}$ and the (overall) system configuration η_{j-1} at time $j-1$ onto respective new multisets:

$$\eta_j = \beta_j \gamma_j \mu_j = \beta_j \, q_E(\beta_{j-1}\gamma_{j-1}) \, q_R(\eta_{j-1}) \,, \tag{3}$$

where:

- $\beta_j \in E$ is the multiset produced by the environment at time j;
- $\gamma_j = q_E(\beta_{j-1}\gamma_{j-1})$ is the environment configuration at time j resulting from mapping the previous environment configuration, i.e., the multiset $\beta_{j-1}\gamma_{j-1}$, to the multiset γ_j by means of q_E;
- $\mu_j = q_R(\eta_{j-1}) = q_R(\beta_{j-1}\gamma_{j-1}\mu_{j-1})$ is the internal system configuration at time j, resulting by applying q_R to the previous configuration, η_{j-1}.

Relation (3) is initialized with β_0 (the first multiset of the environment cycle), μ_0 (the initial configuration), and $\gamma_0 = \emptyset$.

Since the environment cycle E is periodic, it is not difficult to see that the sequence of configurations $\eta_0, \eta_1, \ldots$ produced by the PBE system, read as a sequence of quasi states, is eventually periodic in the sense of Definition 4. The peculiar aspect of PBE systems is the assumption of a periodic behavior of the environment; the system behavior is obtained consequently.

We now add resources to a PBE system as symbols of a finite set $\{r_i\}_{1 \le i \le h}$, with $h > 0$, hence obtaining a *PBE system with resources*.

Definition 10. *A PBE system with resources is a construct*

$$\Pi = (V, \mu_0, R, E, R_E, i_O),$$

where:

(i) V, μ_0, E, R_E, i_O *are as in Definition 9;*
(ii) the rules in R *have the following form:*

- $xx' [_i \, y' y \, r_j^k \to xy' [_i \, x' y$, *for* $x, y, x', y' \in V^*$, $1 \le j \le h$, $k > 0$, *and* $1 \le i \le m$; *(*communication rules*);*
- $[_i \, y \, r_j^k \to [_i \, y'$, *for* $y, y' \in V^*$, $1 \le j \le h$, $k > 0$, *and* $1 \le i \le m$ *(*transformation rules*).*

Along with resources, *waste objects* can be also introduced in the definition of PBE systems with resources. The interplay between resource and waste objects makes PBE systems with resources satisfy a condition of *non-creativity*: every object produced by some rule is consumed by some other rule. In other words, a non creative system defines a cycle where no object is created or destroyed, and every object is transformed into another one. The only new objects that are introduced in the system are provided by the environment cycle [4].

In the following example we exhibit a simple periodic PBE system with resources.

Example 1. Consider the system

$$\Pi_1 = (V, \mu_0, R, E, R_E, i_O),$$

with:

$$V = \{a, b, r_1, r_2\}, \quad \mu_0 = [_1\, a\, [_2\,]_2\,]_1,$$
$$R = \left\{ [_1 a r_1 \to [_1 ab, \;\; b[_2 r_2 \to [_2 b, \;\; r_1[_1 \to [_1 r_1, \;\; r_2[_1 \to [_1 r_2, \;\; r_2[_2 \to [_2 r_2 \right\}$$
$$E = \{r_1, r_1, r_2^2, \lambda, \lambda\}, \quad R_E = \emptyset, \quad i_O = 1.$$

The behavior of this PBE system is described by the following sequence of transitions:

$$\begin{array}{lll}
\eta_0 = & r_1[_1 a[_2\,]_2]_1 & \to \\
& r_1[_1 r_1 a[_2\,]_2]_1 & \to \\
& r_2^2[_1 r_1 ab[_2\,]_2]_1 & \to \\
& [_1 r_2^2 ab^2[_2\,]_2]_1 & \to \\
& [_1 ab^2[_2 r_2^2]_2]_1 & \to \\
\eta_5 = & r_1[_1 a[_2 b^2]_2]_1 & \to \\
& r_1[_1 r_1 a[_2 b^2]_2]_1 & \to \\
& r_2^2[_1 r_1 ab[_2 b^2]_2]_1 & \to \\
& [_1 r_2^2 ab^2[_2 b^2]_2]_1 & \to \\
& [_1 ab^2[_2 r_2^2 b^2]_2]_1 & \to \\
\eta_{10} = & r_1[_1 a[_2 b^4]_2]_1 & \to \\
& \ldots &
\end{array} \tag{4}$$

In configuration η_5 we note that the observable membrane assumes, again, the value taken during the initial configuration η_0. Since the environment has started a new cycle of production in η_5, then the environment configurations again assume the value taken during the initial configuration, too. Hence, the initial sequence of rules must repeat starting from η_5, and so on at every new 5-step cycle.

In conclusion, the system continues to repeat the same sequence of transitions, cyclically. Actually, it can be proved that the observable sequence $\{X_i\}$ generated by Π_1 is eventually periodic with transient $k_0 = 0$ and period $k = 5$. In fact, the finite observable subsequence read along two configurations, η_{5n} and $\eta_{5(n+1)}$, $n \geq 0$, is always equal to $\{a, a, ab, ab^2, ab^2\}$ [4].

This example shows an intriguing analogy with a fundamental result coming from linear system theory [16]. According to this result, the excitation of a *linear system* with a pure sinusoid always produces, at the system output, another (scaled and time–shifted) sinusoid having the same frequency as the incoming one. In practice a transient is always present before the system goes to a *stationary* condition. After this transient the output becomes purely sinusoidal.

More formally, if we look at the linear system as an operator $\mathcal{F}$ mapping time–domain functions into time–domain functions, or *signals*, then if we inject a purely sinusoidal signal into it,

$$x(t) = A_x \sin(\omega_x t + \phi_x)$$

having *amplitude* A_x, *frequency* ω_x, and *phase* ϕ_x, then the system in its turn responds with a sinusoidal signal having different amplitude and phase, respectively A_y and ϕ_y, but the same frequency:

$$A_y \sin(\omega_x t + \phi_y) = \mathcal{F}\big[A_x \sin(\omega_x t + \phi_x)\big] .$$

It can be shown for linear systems [16] that, in the case of purely sinusoidal excitation, the amplitude of the (sinusoidal) output signal depends on the resonance properties of the system: the closer the frequency ω of the sinusoid to the *resonance frequency* Ω, the larger the amplitude of the output signal is.

Although formally provable, this result has an immediate interpretation in basic system dynamics. If we consider, for example, an extremely simple linear dynamical system such as the pendulum, then it is easy to show that this system has its own natural oscillation frequency that only depends on its structural parameters (i.e., size and mass). This oscillation frequency *is* the resonance frequency of the system and, in fact, making a pendulum oscillate out of its resonance frequency (i.e., forcing it to an unnatural oscillation by repeatedly moving it with the hand, this corresponding to injecting a "sinusoidal" non-resonant signal into the system in the limit of our capability to reproduce a sinusoidal signal with our hands) becomes as harder, as farther the forcing sinusoid is from the resonance frequency. In more technical details, linear system dynamics tells us that in a second order linear system having natural resonance frequency Ω (as the pendulum is), A_y will be as greater, as closer to Ω the frequency of the input signal is. Equivalently, we say that the system resonates at frequency $\omega = \Omega$ [16].

The aforementioned properties hold also when the linear system is defined in the discrete–time domain. In this case it can be shown that a similar relationship involving discrete–time sinusoids exists between the system input and output we have:

$$A_y \sin(\omega_x n + \phi_y) = \mathcal{F}_D\big[A_x \sin(\omega_x n + \phi_x)\big] ,$$

where we have substituted the continuous–time operator $\mathcal{F}$ with the discrete–time operator $\mathcal{F}_D$, and the continuous–time variable t with the discrete–time one, n.

The environment cycle constantly feeds a PBE system with a sequence of multisets that can be seen as a *symbolic* input signal. This signal is by all means periodic (of period k) since it can be univocally encoded by a discrete–time sinusoid having the same period. A general question arises: if a PBE system is constantly fed by a periodic environment cycle, is the system output (i.e., the content of the observable membrane) periodic as well and, if so, has this periodical behavior any affinity with that of a linear system?

Membrane systems seem to have poor affinity with linear systems: it suffices to say that they have structural analogies with the cell, that is definitely a

nonlinear system. Going into more technical details, we can straightforwardly design a P system that, for instance, sends to the environment a periodic sequence made of k_S different symbols repeating forever, after it has been triggered by injecting the symbol S inside the skin, independently of any further symbol injected into the system after the first one. This P system has a clearly nonlinear behavior, as the period of its output is completely determined by the initial injected symbol rather than the symbolic input sequence. Despite this, Example 1 describes a likely linear behavior in which the system resonance is driven by the period of the environment cycle.

These clues, once put together, suggest a possible research direction aimed at studying the system dynamics (in all its aspects of periodicity, chaos and so on) not only through the analysis of the trajectories drawn by the system, as state transition dynamics does, but also by means of an analysis of the *structural properties* of the system. Predicting at least to some extent the dynamical behavior (either close to linearity, nonlinearity, nondeterminism, etc.) of a membrane system by an evaluation of its structure would mark a step forward in the application of P systems as models of biomolecular processes.

4 Oscillatory Biochemical Systems

In this section we focus the attention on some biochemical processes that exhibit periodic behavior. These processes are modeled by means of membrane systems, whose objects are reinterpreted in terms of *concentrations* of biological or chemical elements. Such membrane models are finally implemented on a simulator, called *Psim*, running on a normal PC.

First, we will explain the structure of the simulator. Then, we will illustrate some details of the algorithm it implements. Finally, we will discuss some experimental results obtained by the simulation of three well–known dynamical systems.

4.1 Structure of the Simulator

Psim simulates a P system using the algorithm explained in Section 4.2. Its structure is inspired from some existing software especially designed for the simulation of P systems: the application developed in Prolog by Malita [22], that privileges the execution speed; the simulator written in Lisp by Suzuki and Tanaka [36], particularly useful in long–time simulations of relatively simple systems; the *membrane simulator* developed by Ciobanu and Paraschiv [8], that is able to provide a graphical representation of the whole system along time.

The main features of *Psim* are (1) a flexible definition of the membrane structure via an XML file, (2) a user–friendly interface provided with printing

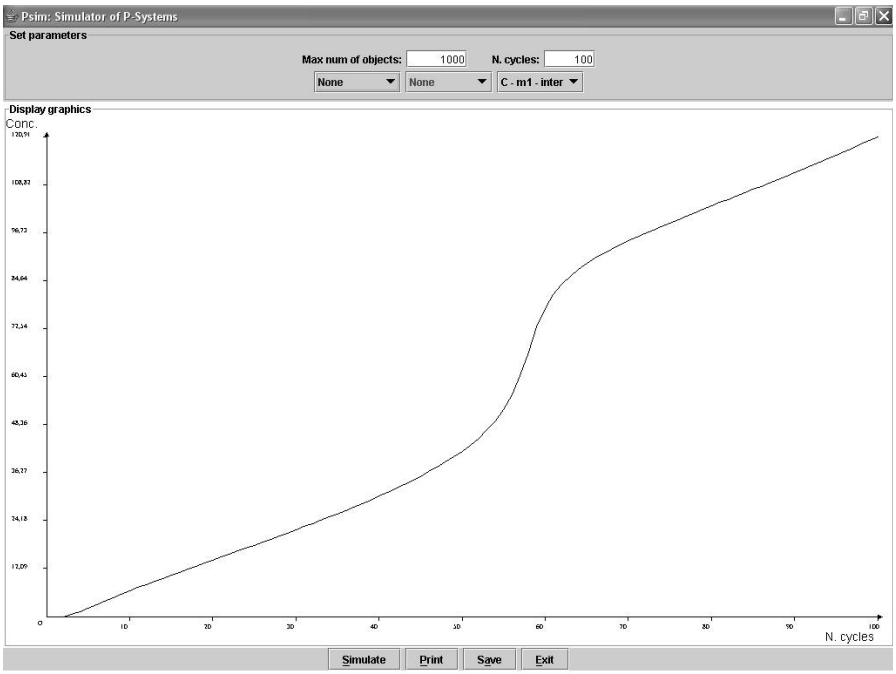

Fig. 1. Screenshot of the simulator.

and graphical capabilities, (3) the possibility to save and then reload intermediate results.

The main screen of the simulator is depicted in Figure 1. It is divided into three frames: the top one contains two text fields, in which the user types the largest number of elements allowed in a membrane, and the number of simulation cycles; it also contains three drop down boxes, allowing the user to select the type of output graph. The frame in the middle displays the computation results. The frame at the bottom contains four buttons, which tell *Psim* to start a simulation, to print results, to save the system state, or to exit.

The membrane structure is encoded into an XML file that is selected at launch time, at the command line, and then loaded by the simulator. A membrane is completely specified by a name, a position in the system and a multiplicity. Each membrane is made of three different regions: an internal region called *in*, an external region called *out*, and, unlikely previous implementations, a third region called *inter*. Similarly to the structure described in Section 5, this is the part of the membrane that can contain receptors: we can imagine it as an intermediate region located between the membrane inside and the outside, separated from these two regions in a way that both of them can see its content. This region, hence, can be used for interchanging objects and for communication and, thus, it can be opened or closed to allow or inhibit

communication, respectively. Each of these parts can contain objects, as we will explain later, and also further membranes in such a way that the user can design more complex structures. In the end *Psim* models a multiset of membranes, each containing three (possibly empty) multisets of objects.

The information on the topological structure of the membranes is followed by the description of the objects that are initially present in the system. These objects are associated to a name, a multiplicity and a reference to a membrane containing them. The first two parameters are attributes of the tag *object* of the XML file, whereas the information about the outer membrane (thus, more generally, the information regarding the topology of the system) is encoded by nesting corresponding *object* tags.

The last part of the input file is the description of the rule set. The syntax describing such rules implements the notation of a PBE system (see Section 3). Each rule is associated to a single membrane, e.g., it describes its evolution by controlling the production of a subset of its objects, and it is specified by a name, a reactivity factor that alters the uniform distribution of probability in the application of the rule, and a non empty list of elements related to it.

These elements are of two kinds: *reactants* and *products*. Each of them is specified by a name, a position within the membrane structure and a *reaction factor* (later on in this chapter we will talk about reactivities more extensively).

Intuitively a rule states that certain reactants take part in a reaction, with proportions given by their stoichiometric coefficients. In this way they generate corresponding products at a rate depending on some chemical and physical factors. According to these ideas every XML tag describing an element of a rule is composed by a reactant part and a product part. Each of these parts has one attribute identifying the type of objects it refers to, one regarding the amount factor of the objects and a pair of attributes specifying the position of that element in the system in terms of the membrane and the relative region inside it. The couple of both these parts says how a reactant transforms itself into a product in terms of type of objects, position within the system, and multiplicity.

This information, together with the reactivity coefficient of a rule, is the only information our algorithm needs to compute the contribution of a reaction in the production of an object over time. It is important to stress, at this time, that with such a syntax it is possible to implement both communication and transformation rules (according to the terminology used in Section 3).

Figure 2 contains a simple XML file that can be used as an input to the simulator. All previously discussed points can be recognized to take part in it.

From a theoretical point of view, and following the terminology proposed in [27], the system we have implemented can describe every family of P systems with $m \geq 1$ membranes, both with or without priorities between rules, catalysts, and i/o. Features like *dissolution*, *electrical charge*, *thickness* and *permeability* have not been implemented yet, but the flexible backbone of our

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE sistema SYSTEM "psim.dtd">

<simulator>
    <membrane id="m1" mult="1">
      <in>
        <object name="E" mult="1"/>
      </in>
    </membrane>

    <rule name="R1" rho="1"  membrane="m1">
        <elem reactant="A" r_mult="1" r_memb="m1" r_zone="out"
         product="C" p_mult="1" p_memb="m1" p_zone="inter"/>
    </rule>
    <rule name="R2" rho="1"  membrane="m1">
        <elem reactant="C" r_mult="1" r_memb="m1" r_zone="inter"
         product="D" p_mult="1" p_memb="m1" p_zone="inter"/>
    </rule>
    <rule name="R3" rho="1"  membrane="m1">
        <elem reactant="D" r_mult="1" r_memb="m1" r_zone="inter"
         product="E" p_mult="1" p_memb="m1" p_zone="in"/>
    </rule>
    <rule name="R5" rho="1"  membrane="m1">
        <elem reactant="E" r_mult="1" r_memb="m1" r_zone="in"
         product="A" p_mult="1" p_memb="m1" p_zone="out"/>
    </rule>
</simulator>
```

Fig. 2. Example of an XML file. It implements a simple set of impulses sent to membrane $m1$ from the outside in the form of concentration of an object E. This concentration repeatedly goes from 0 to its greatest allowed amount, then drops down to 0, with a period of 4 simulation steps.

system would eventually allow their introduction. Formally speaking, we can model systems of the form $P_m(Pri, Cat, i/o, n\pm, n\delta, n\tau)$, where $m > 0$.

The simulator output consists in a series of graphs representing the objects multiplicity along time. The interaction between the user and the simulator is mediated by a simple Graphical User Interface (GUI), that helps the user to write all the parameters needed by the simulation, and to select the graphs that will be displayed after the computations.

Both the simulator and the GUI are written in Java, and in this way they are executed by a Java virtual machine. This makes the simulator cross–platform. However, at this stage of development the system dynamics in our simulator includes only the production and the spatial movement of objects. We want to extend this functionality, allowing the topological structure to change in time.

4.2 The Metabolic Algorithm and Some Applications

The algorithm implemented by the simulator is inspired by a *chemical* reading of the rewriting rules. Due to the biological implications of this type of reading, we called the algorithm *metabolic*.

The reinterpretation of the rewriting rules in the light of a specific application is not new: several researchers have applied rewriting systems to contexts different from the purely abstract one, giving alternative meanings to the rules [2, 35, 37]. In P systems every rule can be seen as a binary relation between strings, mapping the leftward argument into the rightward one. For instance, a rule $r : AB \to CD$ containing symbols defined over an alphabet V states that every occurrence of the object $A \in V$ in the system, once paired with $B \in V$, can be substituted by the new object pair $CD \in V^*$.

If we look at r as a chemical *reaction*, then the leftward objects A and B have the role of *reactants* whereas those on the right are *products*. Following this interpretation, we propose to look at rules as descriptors of the changes in concentration of the reactants into products. In other words, r says that a number of objects of type A and B transforms into objects of type C and D. In this way we deal with populations rather than single objects.

This interpretation needs the introduction of some definitions. Consider a P system on an alphabet $V = \{A, B, C, \dots\}$, provided with a nonempty set R of rewriting rules. Every rule $r : \alpha \to \beta$, with $\alpha, \beta \in V^*$, is associated to a *reactivity coefficient* k_r whose role will be made clear in the following.

For each membrane M we give a maximum number of objects, $|M|$, that cannot be overcome. This parameter is related to the physical properties of M, and we will call it *capacity* of M. From here we agree to define a conventional *molarity unit*:

$$\mu = \nu\,|M|\,,$$

where ν is the *reaction factor*, taking values between 0 and 1 ($\nu = 0.01$ in our experiments), which defines a fraction μ of the membrane capacity as the reactive unit or, simply, *mole*.

Denoting with $|X|$ the number of elements of type X in M, we define the quantity

$$||X|| = \frac{|X|}{\mu} \tag{5}$$

as the number of *moles* of X inside M. This molar formulation for the quantities involved in a reaction leads to the *α–molar concentration*, defined as

the product of the moles of every object [21] in a string $\alpha = \alpha_1 \dots \alpha_{l(\alpha)}$, $l(\alpha)$ being the length of α:

$$||\alpha|| = \prod_{i=1}^{l(\alpha)} ||\alpha_i|| \,. \tag{6}$$

It is now possible to describe an algorithm that translates the rewriting rules into a set of equations defining the *molar variation*, $\Delta||X||$, of every element X in consequence of the application of the rules.

A rule $r : \alpha \to \beta \in R$ acts on the leftward (i.e., reactant) and rightward (i.e., product) objects: the leftward part of r diminishes the concentration of the reactants, whereas the rightward part increases the concentration of the products. Hence, the changes in the amount for an element X in M due to r are equal to:

$$|\beta|_X - |\alpha|_X, \tag{7}$$

where $|\gamma|_S$ indicates the number of occurrences of S contained in γ. Note that this factor is independent of the concentrations; it "syntactically" ties the rule to the object. Note also that the final balance for X can be either negative or positive, depending on the specific reaction.

In chemical terms, r affects the concentration of every element appearing in it by a similar contribution, depending on the concentration of all the reactants at the instant of application. The term $||\alpha||$ takes this aspect into account, according to equation (6). Thus, we can compute the effect $p(X, r)$ of a rule $r : \alpha \to \beta$ on the concentration of X, as

$$p(X, r) = k_r \left(|\beta|_X - |\alpha|_X\right) ||\alpha|| \,, \tag{8}$$

where k_r is the *reactivity coefficient* of the rule. Therefore, $p(X, r)$ is the product of three factors: i) the reactivity k_r; ii) the quantity (7), which plays the role that stoichiometric coefficients have in chemical reactions; and iii) the molar concentration (6) of the reactants.

In general, an object is present in more than one rule. In order to compute the overall molar variation of an object X we have to take the contributions of all rules into account. This is made by summing up their effects on the concentration of X:

$$\Delta||X|| = \sum_{r \in R} p(X, r), \tag{9}$$

where R is the set of rules in our P system.

Hence, after the application of a set of rules our algorithm updates the number of moles of an object X according to the following assignment:

$$||X|| := ||X|| + \Delta||X|| \,. \tag{10}$$

The multiplicity of X is updated accordingly:

$$|X| := |X| + \mu\, \Delta||X|| \,. \tag{11}$$

Let us now see a concrete example of this translation from rewriting rules to *metabolic equations*. Consider the following set of rules:

$$\begin{array}{l} r1 : AC \quad \to AB \\ r2 : BC \quad \to A \\ r3 : BBB \to BC \end{array} \tag{12}$$

each of them associated to a reactivity coefficient, respectively k_{r1}, k_{r2}, and k_{r3}. We want to calculate the variation in the multiplicity of every object in the system caused by the rules.

If we apply equation (9) to each object, then we obtain the following system of metabolic equations:

$$\begin{array}{rcl} \Delta||A|| & = & 0 \cdot k_{r1}||AC|| + 1 \cdot k_{r2}||BC|| + 0 \cdot k_{r3}||BBB|| \\ \Delta||B|| & = & +1 \cdot k_{r1}||AC|| - 1 \cdot k_{r2}||BC|| - 2 \cdot k_{r3}||BBB|| \\ \Delta||C|| & = & -1 \cdot k_{r1}||AC|| - 1 \cdot k_{r2}||BC|| + 1 \cdot k_{r3}||BBB|| \end{array} \tag{13}$$

where k_{r1}, k_{r2}, and k_{r3} can be read as "rates" of application of $r1$, $r2$ and $r3$, respectively. As we can see from (13), where all contributions (including the null ones) are represented, it is always possible to figure out an equation for every object of the P system from the corresponding set of rewriting rules. Each of these equations gives the molar variation of the related element as time elapses.

By applying equation (5) we can figure out the finite *differentials* associated to the system (13):

$$\begin{array}{rcl} \Delta a & = & +\mu \cdot \frac{k_{r2}}{\mu^2} \cdot bc \\ \Delta b & = & +\mu \cdot \frac{k_{r1}}{\mu^2} \cdot ac - \mu \cdot \frac{k_{r2}}{\mu^2} \cdot bc - 2\mu \cdot \frac{k_{r3}}{\mu^3} \cdot b^3 \\ \Delta c & = & -\mu \cdot \frac{k_{r1}}{\mu^2} \cdot ac - \mu \cdot \frac{k_{r2}}{\mu^2} \cdot bc + \mu \cdot \frac{k_{r3}}{\mu^3} \cdot b^3 \end{array} \tag{14}$$

in which we have denoted numbers of elements with a, b, c instead of $|A|, |B|, |C|$, respectively. Note that the correspondence between rewriting rules and differential equations is not bi–directional: in general there is no unique way to translate a system of differentials into a set of rewriting rules, whereas the other way round holds.

We want to emphasize some important facts about the coefficients k_r. We have seen that in the molar formulation of rewriting rules they are called reactivities, and their role is to weight each rule's action. Reactivities take several aspects into account. They model many physical parameters of the reaction environment uniformly acting on the rules, such as pressure and temperature. They also model further chemical parameters, not acting uniformly on rules, such as the pH and the presence of catalysts or enzymes, supporting a reaction that could not effectively take place otherwise.

The application of the reactivity factors in a P system should account for the following aspects:

- strategy of application;

- synchronization times;
- degree of parallelism;
- times and degrees of activation;
- reactants' partition;
- energy partition;
- reaction rate.

If we consider all the interconnections existing between the points introduced in the previous list, then it is easy to understand that the tuning of reactivity factors is very important. We think that this aspect needs further investigation, and our future work will proceed along this line. For instance, it is important to notice that a change in the reactivity can be put in relation with the granularity with which we observe a reaction. This type of change has a clear analogy with the tuning of the temporal step that controls the resolution at which we observe the discrete approximation of the solution of a differential equation (for example, obtained using the Euler's method). Furthermore, the discovery of algorithms reproducing specific dynamics that are observable in nature might be useful in the tuning of the reactivities. This is another aspect we want to investigate.

As previously seen, the multiplicity of X is updated according to (10) after each system transition. Unfortunately it might happen that a rule is applied too many times with respect to the reactant allowance, due to a wrong choice of the reactivity coefficients. In other words, the system in principle can consume more reactants than those which are available at a given configuration. This violates the Principle of Mass Conservation.

To account for this, we add in our model a set of constraints that force the system to respect the Principle of Mass Conservation. One possibility is the following: for every object X, before calculating its molar variation $\Delta||X||$ check if the negative contribution on X due to this variation exceeds the amount $|X|$ calculated at the previous step; if so, then stop the computation, else go on. Another possible work–around to a violation of the previously discussed constraints is to decrease each reactivity value to a lower rate and, then, repeat the check.

To clarify this, it is useful to calculate the constraint set on a concrete example. Consider the P system defined by the rule set (12). As discussed before we associate the following constraints to each reactant, A, B, and C – respectively $\mathbf{C}_{|A|}$, $\mathbf{C}_{|B|}$ and $\mathbf{C}_{|C|}$:

$$\begin{array}{l} \mathbf{C}_{|A|} : k_{r1}||AC|| < |A| \\ \mathbf{C}_{|B|} : k_{r2}||BC|| + k_{r3}||BBB|| < |B| \\ \mathbf{C}_{|C|} : k_{r1}||AC|| + k_{r2}||BC|| < |C| \end{array} \tag{15}$$

One may think that the constraint on an object X can be equivalently calculated *after* the updating of $|X|$, by simply checking that it never assumes negative values. Once more, this is the wrong approach. In fact, even if the

balance of positive and negative contributions results in an admissible variation, no one is able in this way to prevent that the amount of X consumed by all the reactions (those including it among their reactants) during a transition exceeds its real amount. So, conditions such as (15) must be checked *before* every transition.

Once a constraint violation has been discovered, there are several ways to react. The investigation on how to do that is in progress. There are some open questions in our model, and our future work will try to give an answer to them. One of such questions deals with the temporal variation of the reactivity parameters: we think that setting these parameters free to vary along time would have a strong impact on the system behavior, enabling it to simulate reactions more precisely.

At this stage the model is time–invariant and the overall dynamical system behavior completely depends on a vector I, sized $n + m$, containing n normalized reactivities $k_1/k_M, \ldots, k_n/K_M$ (k_M being the largest reactivity coefficient) in their turn divided by μ, plus m initial concentrations $a_{01}, \ldots, a_{0m}$ of the elements $a_1, \ldots, a_m$, respectively:

$$I = \left(\frac{k_1}{\mu k_M}, \ldots, \frac{k_n}{\mu k_M}, a_{01}, \ldots, a_{0m} \right) . \tag{16}$$

In other words, every reactivity coefficient can be expressed as a fraction of the molarity unit as well as the largest reactivity, in a way that it is possible to think at every coefficient as the result of two normalizations: i) a *molar* normalization, accounting for the mass and time granularity of the reaction, and ii) a *kinetic* normalization, accounting for the relative strengths of rules.

We have tested the algorithm with *Psim*, by simulating some well–known oscillating biochemical reactions. The first test has been conducted on the Brusselator. This reaction occurs when certain reactants such as sulphuric acid, malonic acid, ferroin and bromate sodium are combined together in presence of a cerium catalyst [33, 15, 37, 24].

After a period of inactivity, the resulting compound starts producing a series of periodic changes in color ranging from red to blue. The corresponding chemical reaction, according to the formula given in [37], can be symbolically described in terms of the following rewriting rules:

$$\begin{array}{ll} r_1 : A & \to X \\ r_2 : BX & \to YD \\ r_3 : XXY & \to XXX \\ r_4 : X & \to C \end{array} \tag{17}$$

Usually, the assumption is made that the reaction is continuously fed by the external environment. To account for this we provide the set (17) with two further generative rules, whose role is to feed the system with reactants A and B:

$$\begin{array}{l} r_5 : \lambda \to A \\ r_6 : \lambda \to B \end{array} \tag{18}$$

Starting from this extended set of rules, and assuming that every rule r has a reactivity k_r, then using the algorithm discussed previously it is possible to obtain the following set of metabolic equations:

$$\begin{array}{l} \Delta||A|| \ = k_{r5} \ - \ k_{r1}||A|| \\ \Delta||B|| \ = k_{r6} \ - \ k_{r1}||BX|| \\ \Delta||C|| \ = k_{r4}||X|| \\ \Delta||D|| \ = k_{r2}||BX|| \\ \Delta||X|| \ = k_{r1}||A|| \ - \ k_{r2}||BX|| + \ k_{r3}||XXY|| \ - \ k_{r4}||X|| \\ \Delta||Y|| \ = k_{r2}||BX|| \ - \ k_{r3}||XXY|| \end{array} \tag{19}$$

each describing the evolution of the concentration of the respective element. The oscillating behavior of the BZ reaction turns, in the abstract system expressed by (17) and (18), into corresponding oscillations in the number of X and Y.

We have encoded the rule set (19) into an XML input file, and fed this file to the simulator. The number of X and Y as functions of time is plotted in Figure 3: the oscillating behavior of these functions is clearly visible. According to the assumptions made in [35], initially all objects have multiplicity equal to zero. Note that it is possible to normalize all the reactivity coefficients by the largest reactivity (k_1 in the example of Figure 3).

Prigogine and Nicolis [24] have studied a simpler dynamics of the BZ reaction in terms of only the objects X and Y. This formulation yields the following system of differential equations:

$$\begin{array}{l} x' = k_1 - k_2 x + k_3 x^2 y - k_4 x \\ y' = k_2 x - k_3 x^2 y \end{array} \tag{20}$$

depending on four parameters: k_1, k_2, k_3, and k_4. By expressing (20) in terms of rewriting rules we obtain the following rewriting system, that can be viewed as a simpler form of the Brusselator:

$$\begin{array}{ll} r1: & \lambda \ \ \ \ \ \ \rightarrow X \\ r2: & X \ \ \ \ \ \ \rightarrow Y \\ r3: & XXY \rightarrow XXX \\ r4: & X \ \ \ \ \ \ \rightarrow \lambda \end{array} \tag{21}$$

The associated molar equations are:

$$\begin{array}{l} \Delta||X|| \ = k_{r1} - \ k_{r2}||X|| + \ k_{r3}||XXY|| \ - \ k_{r4}||X|| \\ \Delta||Y|| \ = k_{r2}||X|| \ - \ k_{r3}||XXY|| \end{array} \tag{22}$$

where, as usual, every rule r has reactivity k_r.

Equations (20) produce interesting behaviors depending on the reactivity coefficients. As outlined in [24], when choosing in particular $k_1 = 100$, $k_2 = 3$, $k_3 = 10^{-4}$ and $k_4 = 1$ they originate the typical oscillating dynamics of the Brusselator.

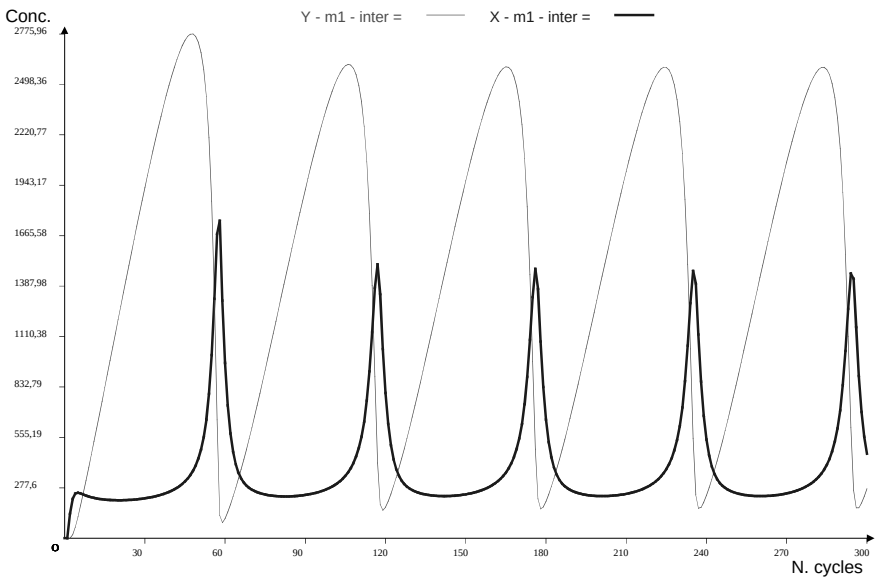

Fig. 3. Oscillations of Belousov–Zhabotinskii reaction model simulated by *Psim* with parameters $k_1 = 0.9$, $k_2 = 0.7$, $k_3 = 0.36$, $k_4 = 0.36$, $k_5 = 0.1$, $k_6 = 0.15$ and $\mu = 1000$ ($|M| = 100000$). Parameters can be rewritten as functions of k_1 in this way: $k_2 = 0.78\,k_1$, $k_3 = 0.4\,k_1$, $k_4 = 0.4\,k_1$, $k_5 = 0.11\,k_1$, $k_6 = 0.17\,k_1$.

These values must be normalized to equate the dynamics obtained with the differential approach to the dynamics obtained with the metabolic algorithm. In fact, let us denote with k_i the parameters characterizing the differential formulation, and with $k_i^{(m)}$ the corresponding parameters in the metabolic algorithm. The translation from the differential to the metabolic parameter – refer also to (14) – is done for every rule according to the following *molar normalization*:

$$k_i^{(m)} = \frac{k_i}{\mu^{1-l(\alpha)}}, \tag{23}$$

where $l(\alpha)$ is the number of reactants in the rule the formula applies to.

As we can see from Figure 4, the simulation of this model of the Brusselator yields the same behavior described in [24] and [35] after molar normalization of the parameters given in [24]. Moreover, by solving (20) with a well–known numerical integration method like Runge–Kutta [18] we have verified that the correspondence between the differential approach and our algorithm also holds when the parameters are chosen in a way that no oscillations are observed. All these correspondences existing between the results from our simulations and those obtained using robust integration methods, like Runge–Kutta, suggest

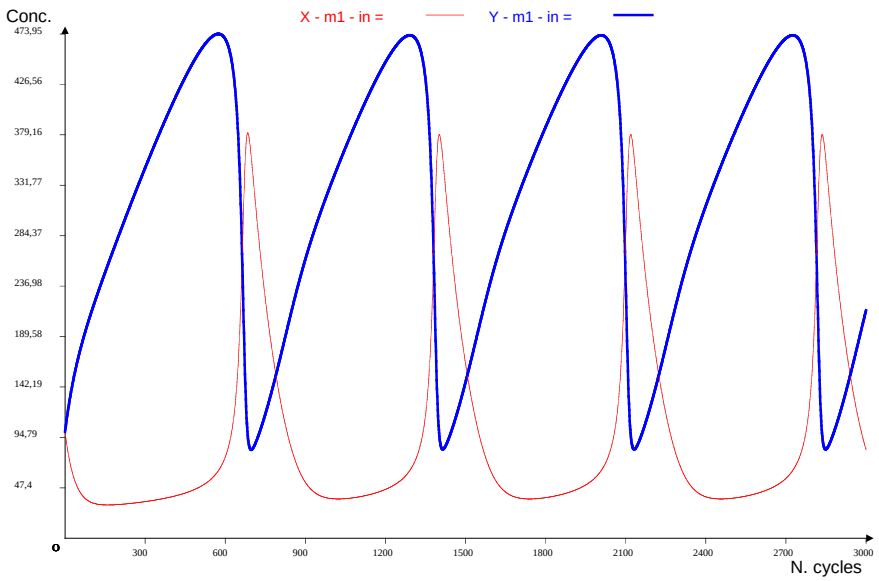

Fig. 4. Oscillations of the simplified Brusselator model simulated by *Psim* with reactivities scaled by a factor 100: $k_1 = 0.001$, $k_2 = 0.03$, $k_3 = 1$, $k_4 = 0.01$, and $\mu = 1000$ ($|M| = 100000$). The initial cardinality of X and Y is set to 100.

that the algorithm implemented by *Psim* can certainly be considered as a reliable candidate for modeling this kind of systems.

The second dynamic system we investigate is a simple predator–prey model [15]. The model is formed by two objects evolving in time: preys, X, and predators, Y. We make the following simplifying assumptions:

- the number of preys increases following a Malthusian model;
- the increase of preys reduces proportionally to the number of predators;
- predators extinguish in absence of preys since in that case they become preys in their turn;
- predators increase proportionally to the number of preys.

This model is described by the Lotka–Volterra differential equations:

$$\begin{aligned} x' &= ax - dxy \\ y' &= exy - by \end{aligned} \tag{24}$$

with initial conditions $x_0 > 0$ and $y_0 > 0$.

We can translate these differential equations into the following set of rewriting rules (recall that it is $x = ||X||$ and $y = ||Y||$):

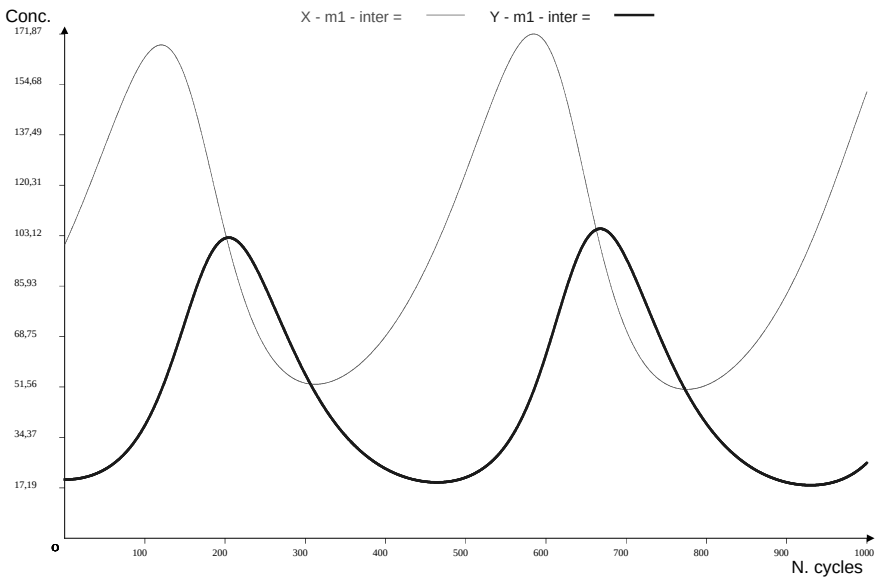

Fig. 5. Oscillations of the predator–prey model simulated by *Psim* with $k_1 = 0.01$, $k_2 = 0.02$, $k_3 = 0.02$ and $\mu = 100$ ($|M| = 10000$).

$$\begin{array}{lll} r1: & X & \rightarrow & XX \\ r2: & XY & \rightarrow & YY \\ r3: & Y & \rightarrow & \lambda \end{array} \tag{25}$$

In this way we obtain the corresponding metabolic equations:

$$\begin{array}{l} \Delta||X|| = k_{r1}||X|| - k_{r2}||XY|| \\ \Delta||Y|| = -k_{r2}||XY|| - k_{r3}||Y|| \end{array} \tag{26}$$

The above rules and objects are contained into a system with just one membrane.

We have simulated this system using an initial amount of 100 preys and 20 predators. The simulation, as we can see from Figure 5, confirmed the oscillating behavior of the number of preys and predators in the predator–pray model described by the Lotka–Volterra equation system.

The last model we discuss in this paragraph is that of an infective disease that spreads over a population, causing death or permanent immunity to the infected people. We make the simplifying assumption that the population is closed (e.g., no births, immigration or emigration are allowed) in a way that the population can be partitioned into three different categories: healthy people, C, ill people, G, and immune people, K.

When an healthy person meets an ill one he (or she) gets ill with a probability expressed by the reactivity of the rule. An ill person has three possibilities: either he dies, or becomes immune forever, or survives indefinitely although being ill. On the other hand, an healthy individual maintains his state as long as he is not in contact with an ill one. This system can be expressed by the following set of rules:

$$\begin{array}{lll} r_1: & CG \rightarrow & GG \\ r_2: & G \rightarrow & K \\ r_3: & G \rightarrow & \lambda \end{array} \tag{27}$$

in which all the symbols have the previously discussed meaning.

It is now possible to write down the set of associated metabolic equations:

$$\begin{array}{l} \Delta||C|| = -\ k_{r1}||CG|| \\ \Delta||G|| = k_{r1}||CG|| \ -\ k_{r2}||G|| \ -\ k_{r3}||G|| \\ \Delta||K|| = -\ k_{r2}||G|| \end{array} \tag{28}$$

with the usual meaning of the reactivity parameters.

The simulation of this system with *Psim* has shown results that agree with those found in the literature. In particular, it has highlighted the existence of a threshold of activation for the epidemic: on the one hand, if the initial healthy population is below a certain amount, then the epidemic does not start and, hence, ill people decrease in number until they vanish; on the other hand, if the initial healthy population is beyond that threshold, then the epidemic activates and the number of ill people grows up until reaching its maximum. Finally this number goes back to zero and, thus, the epidemic vanishes.

In consequence of our choice of the parameters – reported in Figures 6 and 7 – it turns out that the threshold of activation is around 2570 (for more details on its computation we refer, for example, to [18]); we, accordingly, find two kinds of behaviors depending on the initial amount of healthy people: in Figure 6 the case is depicted in which the epidemic does not activate, because of the number of initial healthy people being 2000 and, thus, under the threshold; in Figure 7 the initial number of healthy people is 7000 and the epidemic reaches a maximum, then it vanishes. In both cases the initial number of ill people is equal to 300.

We end this section with some considerations that will guide our future work. In particular, we will try to investigate carefully the molar normalization, whose importance is emphasized by the following propositions.

Let E be a system of metabolic equations derived from a set of rewriting rules, and let $MA(E, \mu)$ be the dynamics we get by applying the metabolic algorithm MA using a molarity unit μ. Let us call E_μ the molar normalization of equations E obtained by replacing every reactivity parameter k_i in E with $k_i^{(m)}$. Finally, let us call dE the differential form obtained by replacing, in E, the finite difference operator Δ with the differential operator, d/dt, and the molar quantities with absolute quantities (that is, by setting $\mu = 1$). If *Euler*

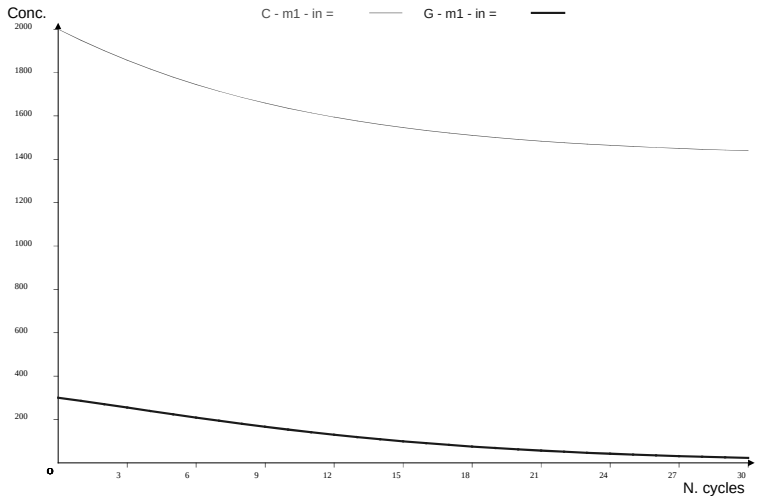

Fig. 6. Inactive epidemic model simulated by *Psim* with $k_1 = 0.3$, $k_2 = 0.1$, $k_3 = 0.12$ and $\mu = 3500$.

is the Euler's approximation method for solving differential equations, then the following proposition is easily proved.

Proposition 4.1 $MA(E, \mu) = Euler(dE_\mu)$.

Nevertheless, scale factor could be essential for a right observation of a system and, in computational terms, scale factor should be relevant for the reliability of numerical simulations. This is the case with Euler's approximation method. In fact in general we have: $Euler(d(E)) \neq Euler(d([E]_\mu))$. In the case of the oscillatory phenomena we have studied – especially in the Brusselator as reported in [24] – we got the following experimental result.

Proposition 4.2 $MA(E, \mu) = Runge\text{–}Kutta(dE)$.

These propositions highlight the well–foundedness of molar normalization. Furthermore, the metabolic algorithm provides a versatile way to add initial conditions in the form of numbers of objects, and nonlinear constraints in the form of equations such as those seen in (15).

We are planning to carry on further theoretical and experimental work, for a better understanding of biological phenomena and for improving the metabolic algorithm by means of a more systematic and practical use of molar normalization. However, it is important to notice that the metabolic approach is likely to form a reliable basis for building up a discrete tool for the simulation of the behavior of P systems. Next steps will be extending the dynamic

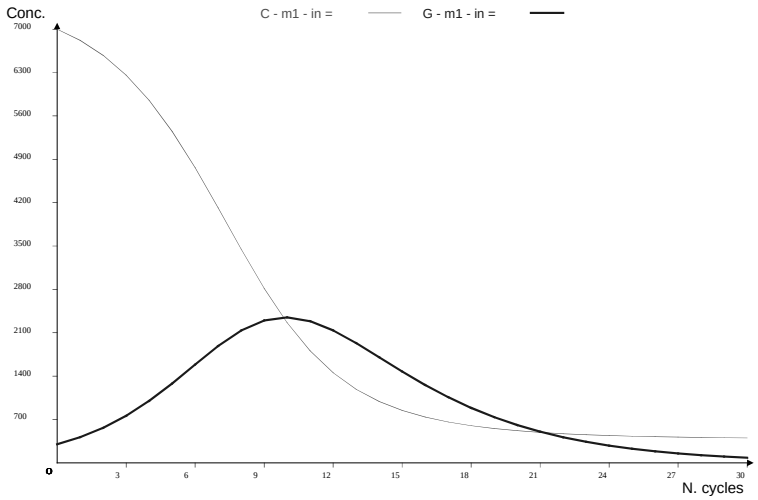

Fig. 7. Active epidemic model simulated by *Psim* with $k_1 = 0.3$, $k_2 = 0.1$, $k_3 = 0.12$ and $\mu = 3500$.

approach to more complex membrane topologies, and to situations where the reaction parameters vary in time under the influence of external factors.

5 P Systems for Immunological Processes

The immune system represents a case of complex adaptive system where the notion of cell membrane is essential. For example, in the description of phenomena such as *lysis* or *opsonization*, that are processes by means of which the bacterial membrane is destroyed and coated, respectively, we must be able to formalize the concept of cell surface. P systems provide a versatile mean to describe the main processes happening in the immune system.

In spite of its complex nature and capability to rapidly adapt against the attack of infectious agents, the immune system can be considered as a typical example of distributed system [34] consisting of many locally interacting components, that provide global protection without any need of centralized control. Moreover the immune system is inherently dynamical, since the individual components circulating throughout the body are continually created and destroyed. A dynamical study of P systems [4] – for example, from the point of view of string transition dynamics, as reported in Section 2 – may help in better understanding the relevant features of immunological processes.

In this section we analyze the basic aspects of the innate immune system in order to discover structures of objects and types of membrane rules useful

to formalize, in terms of membrane systems, the fundamental steps of the primary immunological response. Then, a simplified version of the membrane system for leukocyte selective recruitment [11] is simulated using *Psim*. We will see that using certain parameters the experimental behaviors are well reproduced. A formal description of the adaptive immune system is left to forthcoming research.

A realistic description of the cell needs a richer structure than the one defined for traditional P systems and their variations [28]. For example, we need a representation of membranes that describes the recognition and pairing mechanism between cells by means of "receptor–receptor" bonds. The surface of many (in particular immune) cells is covered with *receptors*, that are complex three–dimensional, electrically charged structures well visible from the outside. The more complementary the structure and charge of receptors belonging to two cells, the more likely the binding between such cells is. The receptor of a pathogen is called *epitope*, and the bond strength between a receptor and an epitope is called *affinity*. Receptors are deemed *specific* because they tightly bind only to a few similar epitope structures or patterns.

The notion of a receptor, crucial in the cellular interplay happening in the immune system, was not considered yet in the basic model of P systems. In order to formally express this aspect [11] here we use $\{\,[_i$ instead of $[_i$ to indicate a membrane, with this notation expressing the presence (between braces and square brackets) of an interstice, intended as a region belonging to the external surface where objects (receptors) are detectable from the outside and, in their turn, can detect objects in the external world. This feature can be considered as an extension of the communication mechanism of PB systems (refer to Section 3) [5] and symport/antiport P systems [26]. In fact, in PB systems $x[_i y$ means that a membrane labeled i can see outside of its boundary, in particular it can see an object x close to its external surface. Here we allow that also *semi–internal* objects are visible from the outside.

A second relevant aspect is the introduction of rules that manage entire membranes (including their content) as objects. This approach is suitable and realistic in order to express phenomena such as *phagocytosis* by macrophages and *diapedesis*. In both these processes, microorganisms are engulfed and consumed by immune system cells, and selected cells have to pass through tissues to attack the infection, respectively.

Moreover, in order to describe *adhesion* between cells, that is, the cellular complex obtained after bonds are created among respective receptors, we use the following rule:

$$\{p[_a\ ]_a\}\,\{q[_b\ ]_b\} \rightarrow \{p\,q[_{a+b}\ ]_{a+b}\}$$

where p represents the receptors of the membrane a, q are the receptors of the membrane b, and $+$ is a special symbol for joining labels.

6 The Architecture of the Immune System

Some replicating pathogens constantly attack the body, and can be harmful if left unchecked. Since different antigens have to be destroyed in different ways, the problem faced by the immune system is to recognize them and to choose the right tools for destroying that particular kind of pathogens. We can associate all the relevant biological properties that characterize the antigens to the symbols of an alphabet A, and all the possible forms of epitopes to the symbols of an alphabet E, so that a particular external microorganism can be represented by $\{r[_s\ ]_s\}^v$, where $s \in A^\star, r \in E^\star, v \in \{0, +, -\}$. The charge associated to the membrane is neutral when the antigen is opsonized, positive when it is replicating, and negative when it is not replicating.

To protect multicellular organisms from foreign pathogens – especially those that replicate such as viruses, bacteria, parasites – the immune system must be capable of distinguishing harmful foreign material from normal constituents of the organism, which we will indicate with a membrane $[_{self}\ ]^v_{self}$ where $v \in \{0, +, -\}$. The charge associated to this membrane is normally positive; it becomes negative when the *self* cell is infected, and neutral when it is immune to the attack of antigens.

The recognition of an antigen as foreign in the immune system can be seen as a problem of pattern recognition implemented by binding. For example, lymphocytes recognize pathogens by forming molecular bonds between pathogen fragments and receptors on the surface of the lymphocyte. The more complementary the molecular shape and electrostatic surface charge between pathogen and receptor, the stronger the bond (and the higher the affinity) is. When an immune system detector binds to an antigen, then we say that the immune system has recognized the pattern encoded by the antigen.

To describe this kind of affinity between cells we define a function R_c that measures the bond strength between receptors. If we call R the set of elements representing all types of receptors (of self cells), then we have:

$$R_c : \{(x, y) \mid x \in R, y \in E\} \to \mathbb{N}$$

This definition can be extended on strings, too. In this case computing the value of R_c becomes a hard problem, as it depends on structural and topological cellular contraints. For this reason we avoid an explicit formulation of R_c for strings.

We need to define an affinity function also between molecules (identified with elements of a set *Mol*) and cell receptors, because a cytokine produces its actions by binding to specific high–affinity cell surface receptors (typically in close proximity to where it is produced):

$$R_m : \{(x, y) \mid x \in R, y \in Mol\} \to \mathbb{N}\,.$$

This is a general definition which, in the context of cytokine molecules, we can restrict to a boolean function since *cytokine receptors* are associated with

the structurally unique family of receptors, termed JAKs, that are expressed by many different cells. So, we can see R_m as the characteristic function on $JAK \times Y$ where $JAK \subset R$ is the subset of symbols associated to JAKs receptors.

The architecture of the immune system is multi–layered, and it is provided with defenses on several levels. The most elementary one is the skin, which is the first barrier to infection. Another barrier is constituted by the physiological conditions, such as pH and temperature, that provide uncomfortable living conditions for foreign organisms [34]. We will describe these two levels as a more internal filter–membrane $[_p \,]_p$ of a membrane system (see Figure 8).

Once pathogens have entered the body they encounter the *innate* immune system and, then, the *adaptive* immune system. Both systems consist of a multiplicity of cells and molecules that interact in a complex manner to *detect* and *eliminate* pathogens. The first system is that part of the immune system we are provided with since our birth. It does not change, or adapt, to specific pathogens and provides a rapid first line of defense to keep an early infection in check, giving the adaptive immune system enough time to prepare a more specific response. The term *adaptive* in fact refers to that part of the immune system that learns how to recognize specific kinds of pathogens, and retains memory of this for speeding up future responses.

Both detection and elimination depend upon chemical bonds. The surfaces of immune system cells are covered with various receptors; some of them chemically bind to pathogens and some bind to other immune system cells or molecules.

The membrane system we propose describes the principal steps of first immune response. It basically contains two nested membranes representing the first two levels of defense before going to the adaptive part of the immune system, that is located in the environment: in Figure 8 the attack of antigens must be seen from the inside to the outside of the system, because the most internal region represents the external world and the most external region represents the last, and more specific body defense. This choice is motivated by the increasing complexity of the processes that must be described.

For the sake of simplicity, in the following we do not close brackets indicating corresponding membranes, if not necessary. The most internal membrane $[_p$ represents the body skin. It is a filter that selects, according to specific criteria, membranes that are labeled by strings in $A^\star$, and allows to exit only membranes which have labels belonging to a language $L \subset A^\star$. The region of the most external membrane $[_I$ represents the phase during which the innate immune system is active. Only the pathogens that need a specific action for being destroyed can pass through the membrane labeled with I. Now we see how the innate immune system attacks the antigens. We first introduce some notation.

Complement molecules, together with macrophage cells, are respectively the primary chemical and the phagocytic response of the immune system in the early stages of infection. We split the complement molecules into two sets,

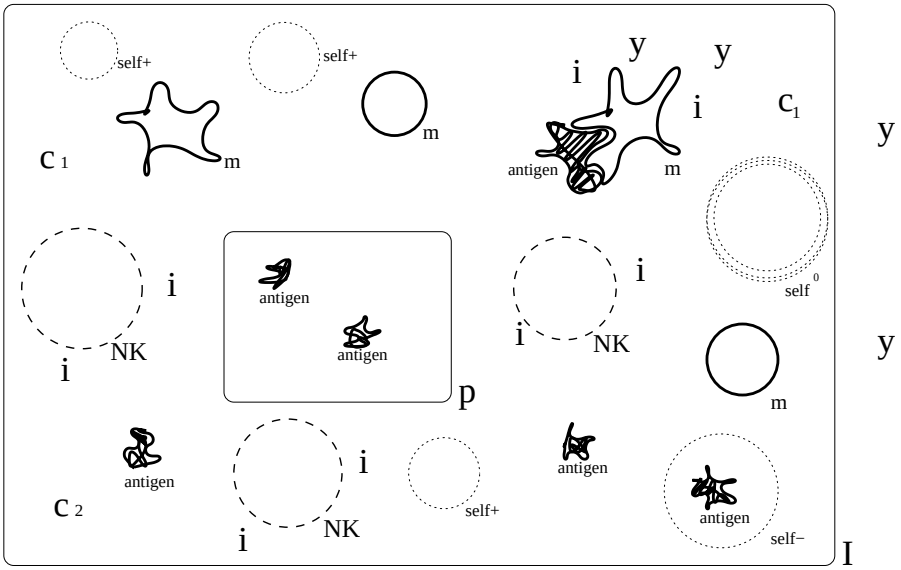

Fig. 8. A membrane system for immune system

C_1 and C_2, because they are involved in two distinct phenomena, respectively lysis and opsonization. Lysis is the process by means of which the complement ruptures the bacterial membrane, this action resulting in the destruction of the bacterium. Opsonization refers to the coating of bacteria with the complement, causing the bacteria to be detected by macrophages. By denoting the non-opsonized antigen with $\{r[_s\,]_s\}^v$, where $v \in \{+,-\}$, and the complement molecules with $c_1 \in C_1$, $c_2 \in C_2$, then we will describe the above phenomena respectively with the following rewriting rules:

$$c_1\ \{r[_s\,]_s\}^v \to c_1\ r',$$
$$c_2\ \{r[_s\,]_s\}^v \to \{r[_s\,]_s\}^0,$$

where r' is a debris resulting from the destruction of the antigen, and the null polarity represents the opsonization state. Self cells have regulatory proteins on their surfaces, preventing complements from binding to them. So, they are protected against the effects of complements.

Macrophages play a crucial role in all stages of the immune response. In order to distinguish them by their two principal functions, i.e., to engulf some specific bacteria or, simply, bacteria opsonized by complement, then we indicate the macrophages with two different types of membranes, respectively:

$$\{\,t\,[_m\,]_m\}\quad [_m\,]_m\,,$$

where $t \in R^\star$ is the string of cell receptors. Actually, macrophages have receptors both for certain kinds of bacteria and for complement, but this abstract

description of macrophages contains all relevant aspects needed by the dynamical system we are describing.

We associate the *cytokine* molecules to symbols belonging to Y. Cytokines are molecules that act as a variety of important signals, and their release activates the next phase of the host defense, called early induced response. They are produced not only by macrophages and other immune system cells, but also by some self cells (which are not part of the immune system) when they are damaged by pathogens. Their major effect is to induce an inflammatory response, associated to some physiological changes (fever) which reduce the activity of pathogens and reinforce the immune response by triggering the production of *acute phase proteins* (APP), substances which bind to bacteria thus activating macrophages or complement.

When infected by viruses certain cells produce *interferons*, a family of cytokines so–called because they inhibit the viral replication. Moreover, they activate certain immune system cells called *Natural Killer* (NK), that kill virus–infected self cells. We classify the interferons as elements of a subset $\mathcal{I} \subset Y$, and the NK cells as membranes labeled by nk:

$$[_{nk}\,]_{nk}$$

NK cells bind to normal host cells, but they are normally not active because healthy cells express molecules that act as inhibitory signals. When some virally infected cells cannot express these signals they are killed by activated NK cells, that release special chemicals that trigger the *apoptosis* (programmed cell death) on an infected cell.

The overall validity of this representation must be verified in terms of capability to explain immunological dynamics. *Psim* applies rewriting rules to objects taking into account their reactivity, i.e., the ability shown by such objects (reactants) to meet (react) together. In particular, the concept of concentration used by *Psim* helps in formalizing a sort of substance (or cellular) adjacency which is not defined in membrane systems. In fact, P systems are topological spaces without a metric on objects. The existence of such a metric, conversely, can be important when dealing with cellular reactions.

Nevertheless, the notation used in membrane systems shows features that are essential in the representation of immunological phenomena. Cells and molecules in fact move around compartments, moreover cells engulf objects and in their turn are engulfed by other cells. Likewise, agents change their internal state, or their state of perception, with respect to the objects they meet, the place where they are, and their state.

7 Innate Immune System with Membranes

In the following we propose a membrane system dealing with the objects and membranes previously introduced, and providing them with suitable rules.

Consider the following P system [10]:

$$(V, \mu, R)$$

where V is the alphabet, μ is the initial configuration (where one can see also the initial membrane structure of the system), and R is the set of rewriting rules that extend the evolution and communication rules [28] by considering also membranes having receptors, along with managing membranes as objects.

Our system has the following alphabet:

$$A \cup E \cup E' \cup R \cup C_1 \cup C_2 \cup Y \cup \{APP, d\},$$

where A contains the biological properties of the antigen, E contains the epitopes in all possible forms, E' contains the primed versions of the symbols in E (this is useful to indicate the epitopes of the bacterium after it has been processed by lysis), R contains the self cell receptors, C_1 and C_2 contain the complement, Y the cytokines, and finally APP and d are two special symbols needed to simulate the the activation of macrophages and the programmed death, respectively.

The system starts from the initial configuration (see Figure 8):

$$\begin{aligned}&[_I [_{self}]^+_{self}{}^{k_s}\ [_{nk}]_{nk}{}^{k_n}\ y\ [_m]_m{}^{k_m}\ \{t_1[_m]_m\}^{k_1}\\ &\quad \ldots \{t_h[_m]_m\}^{k_h}\ [_p\ \{r_1[_{s_1}]_{s_1}\} \ldots \{r_n[_{s_n}]_{s_n}\}\ ]_p\ ]_I,\end{aligned}$$

where $k_s, k_n, k_m, k_1, \ldots, k_h$ are membrane multiplicities. Then, we have the following rules:

- the *starting rule* simulates the entry of some antigens through the body skin:

$$[_p \ldots \{r[_s\ ]_s\}^+ \ldots]_p \to [_p]_p\ \{r[_s\ ]_s\}^+ \quad r \in E^\star, s \in L \quad \text{entrance}$$

- once antigens have entered the body – from now on we will omit to write $r \in E^\star$, $r' \in E'^\star$, $s \in L$, $c_1 \in C_1$, $c_2 \in C_2$ all the times – they induct the primary response by replicating themselves and by infecting self healthy cells (which switch from positive to negative polarity):

$$\{r[_s\ ]_s\}^+ \to \{r[_s\ ]_s\}^+\ \{r[_s\ ]_s\}^+ \qquad \text{replication}$$

$$[_{self}\ ]^+_{self}\ \{r[_s\ ]_s\}^v \to [_{self}\ ]^-_{self} \quad v \in \{+,-\} \qquad \text{infection}$$

- complement, which is present in the region I, thus can act by lysis and opsonization of some kinds of antigens, indicated by a string of the set $\hat{L} \subset L$:

$$c_1\ \{r[_s\ ]_s\}^v \to c_1\ r' \quad s \in \hat{L}, v \in \{+,-\} \qquad \text{lysis}$$

$$c_2\ \{r[_s\ ]_s\}^v \to \{r[_s\ ]_s\}^0 \quad s \in \hat{L}, v \in \{+,-\} \qquad \text{opsonization}$$

- macrophages act as scavenger cells (engulfing debris and opsonizing pathogens), as process antigens (engulfing and digesting antigens, and then presenting fragments of their proteins on the surface), and also, when activated by bindings between receptors, as signal emitters (via cytokines) to the immune system:

$$r' \, [_m \,]_m \to [_m \,]_m \qquad \text{debris remotion}$$

$$\{r[_s \,]_s\}^0 \, [_m \,]_m \to [_m \,]_m \qquad \text{opsonized antigen remotion}$$

$$\{t[_m \,]_m\} \, \{r[_s \,]_s\}^v \to \{t \, s[_{m+s} \,]_{m+s}\} \qquad \text{antigen–macrophage complex}$$

$$\{t \, r[_{m+s} \,]_{m+s}\} \to \{t \, r[_m \,]_m\} \qquad \text{functionality restoration (with trace)}$$

$$\{t \, r[_{m+s} \,]_{m+s}\} \to \{t \, r[_{ms} \,]_{ms}\} \, y \qquad \text{cytokine production by macrophages}$$

  where in the third rule $v \in \{+, -\}$ and $R_c(t, s) > 0$, and in the fifth rule $y \in Y$. In our notation one can distinguish whether the macrophages are activated or not by the presence of the symbol + in their respective label;
- cytokines are probably the most important biologically active group of molecules to identify; with hundreds of known cytokine–like molecules, here it is necessary to restrict the discussion to a few key cytokines and their most important properties, that include starting and maintaining the inflammatory response. However it is clear that, being the common signaling system for cell growth, inflammation, immunity, differentiation and tissue repair processes, cytokines are involved in many, if not all, physiological functions.
  Interferons, and in general cytokines, are produced in the adaptive immune system by T cells and B cells, by activated macrophages, and also by endothelial cells of inflamed tissue:

$$[_{self} \,]^-_{self} \to [_{self} \,]^-_{self} \, y \quad y \in Y \qquad \text{cytokines production by infected cells}$$

  Cytokines increase the number of macrophages and complement by production of APP:

$$y \to y \, APP \qquad \text{APP production by cytokines}$$

$$APP \to c_1 \, c_2 \qquad \text{complement increasing by APP}$$

$$APP \to [_m \,]_m \qquad \text{macrophages increasing by APP}$$

  They also increase the resistance of self cells against bacterial infection (especially mycobacteria and certain viruses):

$$y^k [_{self} \,]^+_{self} \to [_{self} \,]^0_{self} \qquad \text{self cells immunization by cytokines}$$

  where $k \in \mathbb{N}$ is a given number. Moreover, interferons (that are a special class of cytokines) inhibit the virus replication and activate the NK cells, respectively:

$$i\{r[_s\ ]_s\}^+ \to \{r[_s\ ]_s\}^-\ i \in \mathcal{I} \quad \text{virus replication/inhibition}$$

$$i[_{nk}\ ]_{nk} \to [_{nk}\ i\ ]_{nk} \quad i \in \mathcal{I} \quad \text{NK cell activation by interferons}$$

Interferons are also produced by activated NK cells:

$[_{nk} i\ ]_{nk} \to j\ [_{nk} i\ ]_{nk} \quad i, j \in \mathcal{I}$ interferon production by activated NK cells

Finally, activated NK cells induce programmed death in infected cells:

$$[_{nk}\ i\ ]_{nk}\ [_{self}\ ]^-_{self} \to [_{nk}\ i\ ]_{nk}\ [_{self}\ d\ ]^-_{self} \quad \text{programmed death}$$

$$[_{self}\ d\ ]^-_{self} \to d \quad \text{self cell death}$$

$$d \to \lambda$$

These rules are not used like in P systems, but in the way of *Psim*, therefore the process dynamics and effects are regulated at every step by the actual amounts of reactants. Every dissolving membrane delivers its content to the immediately outer membrane [29], and every dividing membrane replicates its content inside the new membrane.

Due to its complexity, the description of the adaptive immune system and its interaction with the innate immune system is left to future work. Priorities between regions (and not between rules) might be established to simulate different macro steps, in a way that some processes of the innate immune system would activate with higher priority than those of the adaptive immune system. This regulation might be performed more accurately by using reactivities. Since cells and substances are sent from the adaptive system to the innate one as macrophages and antibodies acting as complement, and as cytokines, respectively, from now on we will suppose that the environment periodically provides these materials [4].

8 Leukocyte Selective Recruitment

In an organism the first response against an inflammatory process consists in the activation of a tissue–specific recruitment of leukocytes. Activation relies on the complex functional interplay between the surface molecules, that are designed for specialized functions. These molecules are differently expressed by leukocytes circulating in the blood, and by endothelial cells covering the blood vessel.

Leukocyte recruitment in tissues requires extravasation from the blood. Extravasation is made possible by a process of trans–endothelial migration, and three major steps have been identified in the process of leukocyte extravasation: (i) tethering–rolling of free–flowing white blood cells, (ii) activation of the same cells, and (iii) arrest of their movement due to their adherence to endothelial cells. After this arrest a phenomenon of *diapedesis* occurs, that is, leukocytes move from the blood beyond endothelial cells toward the tissue.

A leukocyte cell has some receptors put on its surface, that bind with counter–receptors located on the surface of the endothelial cells. These bonds slow down the initial speed of leukocyte. Moreover some molecules, called chemokines, are produced by the epithelium and by the bacteria that have activated the inflammation process. Chemokines can bind with the leukocyte receptors, so producing signals inside it. Such signals generate on the leukocyte surface new and different receptors that, interacting with the endothelial receptors, strongly slow down the cell speed until it stops (see Figure 9).

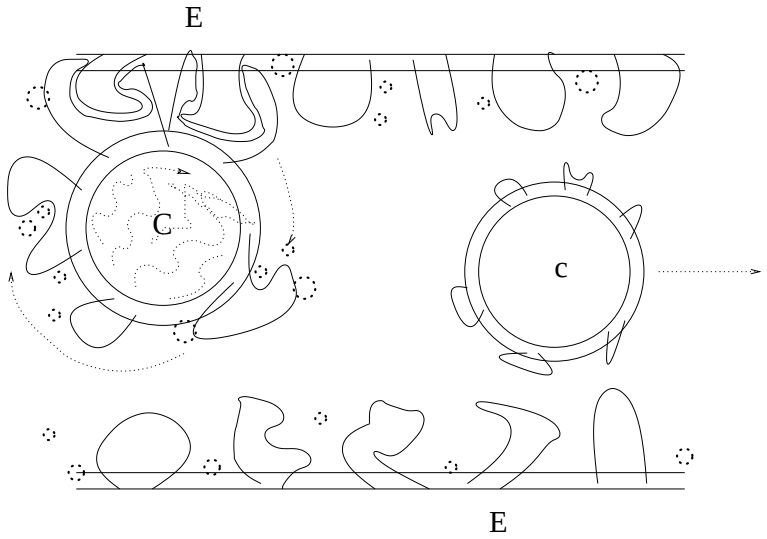

Fig. 9. Leukocyte cell attacked by chemokines and endothelial receptors. C: leukocytes; E: epithelium.

We call A the initial state with fast circulating leukocytes into the blood, B the rolling state, C the activation state, and D the final adhesion state. Therefore, the system evolves through three big phases:

1. $A \rightarrow B$, by means of some receptor–receptor interactions,
2. $B \rightarrow C$, by means of some chemokine–receptor interactions, and
3. $C \rightarrow D$, by means of some receptor–receptor interactions.

(refer to [11] for a description of a membrane system representing this immunological phenomenon; in that paper, ad–hoc notations have been introduced for adding more realism to the model).

Here we analyze, using *Psim*, a simplified model of the system where only a specific kind of leukocytes are present and the dynamics is ruled by only one production of symbols. Accordingly, in this model receptor–receptor bonds are not explicitly represented.

We use two membranes labeled E and L, respectively: E represents the epithelium entered and infected by the bacterium; L represents the leukocyte which interplays with the epithelium by producing or transforming symbols. Initially we have one symbol b inside the membrane labeled E.

The process develops as follows:

- The antigen inside the epithelium produces chemokines and epithelial receptors externally, and copies of itself internally.
- Chemokines become internal signals inside the leukocyte (this process simulates leukocyte activation).
- The internal signals transform into leukocyte receptors, in two steps. When a sufficient number of both epithelial and leukocyte receptors are present in the system, then the elimination of the bacterium is triggered by the production of a symbol v (this is an abstraction of the diapedesis phenomenon [11]).

Consider the alphabet $\{b, c, p, q_1, q_2, s, v\}$. Here we will use the notation adopted with *Psim*, with obvious meaning of the rules presented hereafter. Initially, we have a symbol b representing the bacterium inside E. This bacterium replicates, and all its copies have to be definitely destroyed:

$$k_1 : (b, E, in) \rightarrow (bb, E, in)$$

The presence of b produces symbols c, representing the chemokines outside E, and symbols p, representing the epithelial receptors on the surface of E. We decided for a rate of 3 copies of c and 2 copies of p for each copy of b:

$$k_2 : (b, E, in) \rightarrow (c^3, E, out)(b, E, in)$$
$$k_3 : (b, E, in) \rightarrow (p^2, E, inter)(b, E, in)$$

The chemokines c outside membranes become symbols s, representing internal signals inside membranes labeled with L. Moreover s is 'slowly' transformed into q_2, representing leukocyte receptors, by moving trough a transformation into 5 copies of q_1:

$$k_4 : (c, E, out) \rightarrow (s, L, in)$$
$$k_5 : (s, L, in) \rightarrow (q_1^5, L, inter)$$
$$k_6 : (q_1, L, inter) \rightarrow (q_2, L, inter)$$

Finally, the presence of both p and q_2 – respectively representing epithelial and leukocyte receptors – activates the production of a symbol v inside E, where each copy of b is deleted by the presence of v.

The problem is to regulate the reactivity values $k_1, \ldots, k_8$ to correctly modulate the application of rules, i.e., to neutralize the infection. As one can see in Figures 10, 11, and 12, setting $k_1 = 1$, $k_2 = 1$, $k_3 = 0.8$, $k_4 = 0.7$, $k_5 = 1$, $k_6 = 0.2$, $k_7 = 0.8$, $k_8 = 1$ lets the immune response work sound, as it destroys all copies of replicating b.

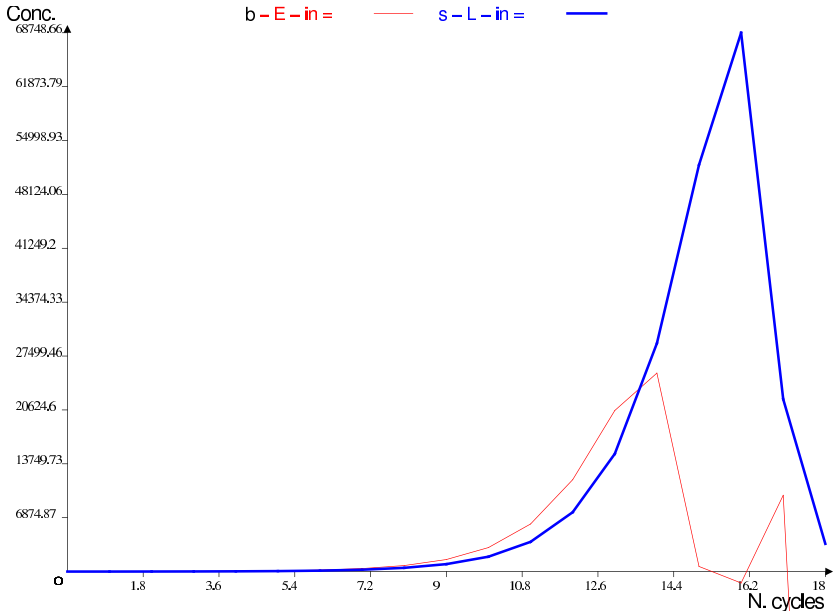

Fig. 10. The amount of b (representing the quantity of infection) inside E (the inflamed epithelial tissue) and s (internal signals as immune response) inside L (leukocyte cell) after 18 applications of the rules; the membrane maximum capacity is equal to 10^6.

Note that the infection decay (e.g., the decreasing amount of b inside E) just corresponds to the increasing activation of leukocytes (e.g., amount of s inside L), as it is well known from immunological studies [34]. Also, when the infection disappears then the production of s slows down quickly, as it can be seen in Figure 10. Moreover, the production of epithelial receptors is higher than the production of chemokines, as it happens in real immunological processes, and the chemokines' behavior follows the amount of b plotted in Figures 11 and 12.

The application of the rules at each computational step is dictated by the principles outlined in Section 4. In this case we have the following differentials:

$$\begin{aligned}
\Delta b &= |b| - \frac{|b||v|}{\mu} \\
\Delta c &= 3|b| - 0.7|c| \\
\Delta p &= 2 * 0.8|b| - 0.8\frac{|p||q_2|}{\mu} \\
\Delta s &= 0.7|c| - |s| \\
\Delta q_1 &= 5|q_1| - 0.2|q_1|
\end{aligned}$$

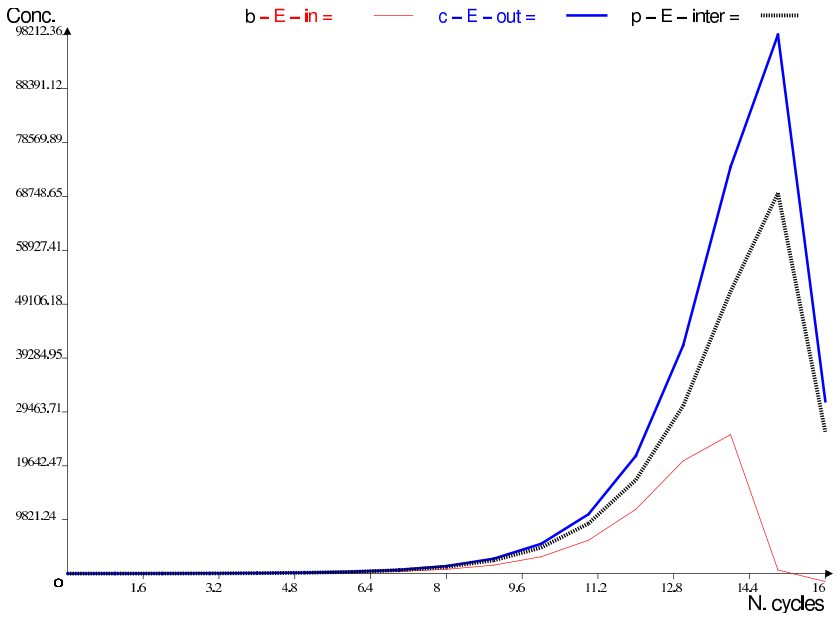

Fig. 11. The amounts of b (antigens), c (chemokines), p (epithelial receptors) after 16 applications of the rules; the membrane maximum capacity is equal to 10^6.

$$\Delta q_2 = 0.2|q_1| - 0.8\frac{|p||q_2|}{\mu}$$

$$\Delta v = 0.8\frac{|p||q_2|}{\mu}$$

9 Conclusion and Future Directions

This chapter describes theoretical facts, as well as experimentation and applications we have recently dealt with during our investigation on the dynamics of P systems. The leading idea of this investigation has been that of looking at P systems not as machines which, in front of an input, produce a corresponding output (provided that they reach a final configuration), but, rather, as systems capable in principle of reproducing a wide spectrum of dynamics, including those pertaining to molecular processes.

If we resolve, for example, the Brusselator reaction expressed by (17) using the *metabolic graph* drawn in Figure 13, then we somehow emphasize in the associated P system a sort of "neuron–like" membrane structure (according to Păun's terminology).

Such a "neural" representation of a P system can be immediately extended to any membrane structure, by making a proper use of labels. In this way any

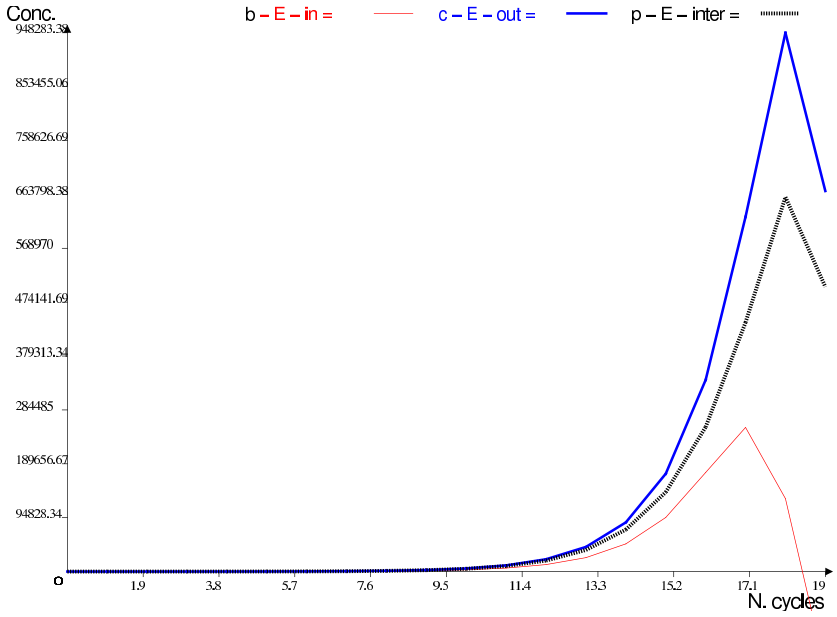

Fig. 12. The amounts of b (antigens), c (chemokines), p (epithelial receptors) after 19 applications of the rules; the membrane maximum capacity is equal to 10^7.

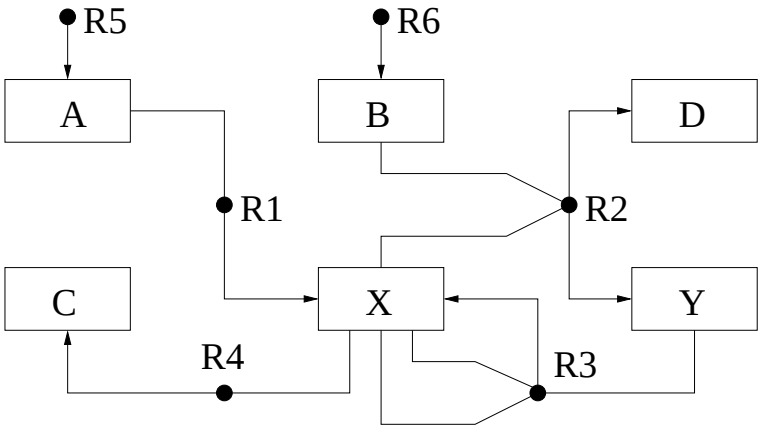

Fig. 13. Brusselator's *Metabolic Graph*.

membrane system takes the aspect of a *dynamical network*, i.e., a graph whose nodes have a state that at every temporal step depends on the state of other nodes, and whose nodes and edges can be added and/or removed dynamically.

It is easy to discover that a dynamics can present oscillatory behaviors only if the associated graph contains cycles. In general, finding parameters that translate into oscillations is not easy. The *inverse oscillation problem* can be stated in the following way: given a metabolic graph, find initial concentrations and reactivity parameters (i.e., a collection of values for I as defined by (16)) that cause oscillations. At the end of Section 3 we have seen that in the case of linear systems the solution is well established.

Future research will investigate on algorithms capable to attack this problem. Related to this investigation will be the search, on metabolic graphs, for system parameters having a dynamical relevance. Though many suggestions on these topics come from the theory of cellular automata and Kauffman networks [17, 38, 39], nevertheless a lot of theoretical analysis and experimental work still has to be done in this direction.

References

1. I.I. Ardelean, D. Besozzi: New Proposals for the Formalization of Membrane Proteins. In G. Păun, A. Riscos–Núñez, A. Romero–Jiménez, and F. Sancho–Caparrini, eds.: *Proc. Second Brainstorming Week on Membrane Computing*, Seville, Spain, February 2004, 53–59.
2. G. Bellin, G. Boudol: The Chemical Abstract Machine. *Theoretical Computer Science*, 96 (1992), 217–248.
3. F. Bernardini, M. Gheorghe: Cell Communication in Tissue P Systems and Cell Division in Population p Systems. In G. Păun, A. Riscos–Núñez, A. Romero–Jiménez, F. Sancho–Caparrini, eds.: *Proc. Second Brainstorming Week on Membrane Computing*, Seville, Spain, February 2004, 74–91.
4. F. Bernardini, V. Manca: Dynamical Aspects of P Systems. *BioSystems*, 70 (2002), 85–93.
5. F. Bernardini, V. Manca: P Systems with Boundary Rules. In *Proc. Int. Workshop on Membrane Computing, WMC-CdeA 2002*, *Lecture Notes in Computer Science* 2597, Springer, Berlin, 2002, 107–118.
6. D. Besozzi, I.I. Ardelean, G. Mauri: The Potential of P Systems for Modeling the Activity of Mechanosensitive Channels in E. Coli. In *Pre–proceedings of Workshop on Membrane Computing, WMC2003*, Tarragona, GRLMC Report 28/03, 84–102.
7. C. Bonanno, V. Manca: Discrete Dynamics in Biological Models. *Romanian Journal of Information Scicnce and Technology*, 5, 1-2 (2002), 45–67.
8. G. Ciobanu, D. Paraschiv: Membrane Software. A P System Simulator. *Fundamenta Informaticae*, 49, 1-3 (2002),61–66.
9. R.L. Devaney: *Introduction to Chaotic Dynamical Systems*. Addison–Wesley, Reading, Mass, 1989.
10. G. Franco, V. Manca: Modeling Some Biological Phenomena by P Systems. In *Proc. of the 2nd Annual Meeting of CE MolCoNet Project*, Wien, Austria, November 2003, 1–3.
11. G. Franco, V. Manca: A Membrane System for the Leukocyte Selective Recruitment. In *Membrane Computing. Intern. Workshop, WMC2003, Tarragona*, *Lecture Notes in Computer Science* 2933, Springer, Berlin, 2004, 181–190.

12. R. Freund: Energy–Controlled P Systems. In G. Păun, C. Zandron, eds.: *Proc. Int. Workshop on Membrane Computing, WMC-CdeA 2002*, *Lecture Notes in Computer Science* 2597, Curtea de Arges, Romania, August 2002, 247–260.
13. T. Head: Formal Language Theory and DNA: An Analysis of the Generative Capacity of Specific Recombinant Behaviours. *Bull. Mathematical Biology*, 49 (1987), 737–759.
14. R.C. Hilborn: *Chaos and Nonlinear Dynamics*. Oxford University Press, Oxford, UK, 2000.
15. D.S. Jones, B.D. Sleeman: *Differential Equations and Mathematical Biology*. Chapman & Hall/CRC, London, UK, 2003.
16. T. Kailath: *Linear Systems*. Prentice–Hall, Englewood Cliffs, 1980.
17. S.A. Kauffman: *The Origins of Order*. Oxford University Press, New York, NY, 1993.
18. J.D. Lambert: *Computational Methods in Ordinary Differential Equations*. J. Wiley & Sons, New York, NY, 1973.
19. C.G. Langton: Computation at the Edge of Chaos: Phase Transitions and Emergent Computation. *Physica D*, 42, 12 (1990).
20. A. Lindenmayer: Mathematical Models for Cellular Interaction in Development. *J. of Theoretical Biology*, 18 (1968), 280–315.
21. B.H. Mahan: *University Chemistry*. Addison Wesley, Reading, MA, 1967.
22. M. Malita: Membrane Computing in Prolog. In *Pre-proceedings of the Workshop on Multiset Processing*, Curtea de Arges, Romania, 2000, 159–175.
23. V. Manca, G. Franco, G. Scollo: String Transition Dynamics – Basic Concepts and Molecular Computing Perspectives. In M. Gheorghe, M. Holcombe, eds.: *Molecular Computational Models – Unconventional Approaches*, Idea Group, London, 2004.
24. G. Nicolis, I. Prigogine: *Exploring Complexity. An Introduction*. Freeman and Company, San Francisco, CA, 1989.
25. A. Păun: On P Systems with Membrane Division. In I. Antoniou, C.S. Calude, and M.J. Dinneen, editors, *Unconventional Models of Computation*, Springer, London, UK, 2000, 187–201..
26. A. Păun, G. Păun: The Power of Communication: P Systems with Symport/Antiport. *New Generation Computing*, 20, 3 (2002), 295–306.
27. G. Păun: Computing with Membranes. *J. Comput. System Sci.*, 61, 1 (2000), 108–143.
28. G. Păun: *Membrane Computing. An Introduction*. Springer, Berlin, Germany, 2002.
29. G. Păun, G. Rozenberg: A Guide to Membrane Computing. *Theoretical Computer Science*, 287 (2002), 73–100.
30. G. Păun, G. Rozenberg, A. Salomaa: *DNA Computing – New Computing Paradigms*. Springer, Berlin, 1998.
31. G. Păun, Y. Suzuki, H. Tanaka: P Systems with Energy Accounting. *Int. J. Computer Math.*, 78, 3 (2001), 343–364.
32. G. Rozenberg, A. Salomaa: *Handbook of Formal Languages*. Springer, Berlin, Germany, 1997.
33. K.S. Scott: *Chemical Chaos*. Oxford University Press, Oxford, UK, 1991.
34. L.A. Segel, I.R. Cohen: *Design Principles for the Immune System and Other Distributed Autonomous System*. Oxford University Press, Oxford, UK, 2001.

35. Y. Suzuki, H. Tanaka: Chemical Oscillation in Symbolic Chemical Systems and Its Behavioral Pattern. In Y. Bar–Yam, ed.: *Proc. International Conference on Complex Systems*, Nashua, NH, September 1997.
36. Y. Suzuki, H. Tanaka: On a Lisp Implementation of a Class of P Systems. *Romanian Journal of Information Science and Technology*, 3, 2 (2000), 173–186.
37. Y. Suzuki, H. Tanaka: Abstract Rewriting Systems on Multisets and Their Application for Modeling Complex Behaviours. In *Proc. of Brainstorming Week on Membrane Computing*, Tarragona, Spain, February 2003, 313–331.
38. S. Wolfram: *Theory and Application of Cellular Automata.* Addison–Wesley, 1986.
39. A. Wuensche: Basins of Attraction in Network Dynamics: A Conceptual Framework for Biomolecular Networks. In G. Schlosser, G.P. Wagner, eds.: *Modularity in Development and Evolution.* Chicago University Press, Chicago, MA, 2003.