Psim: A Computational Platform for Metabolic P Systems

Luca Bianco

Cranfield University
Cranfield Health
Silsoe, Bedfordshire, MK45 4DT, UK
E-mail: L.Bianco@cranfield.ac.uk

Summary. Although born as unconventional models of computation, P systems can be conveniently adopted as modelling frameworks for biological systems simulations. This choice brings with it the advantage of producing easier to be devised and understood models than with other formalisms. Nevertheless the employment of P systems for modelling purposes demands for biologically meaningful evolution strategies as well as for complete computational tools to run simulations on. In previous papers an evolution strategy known as *metabolic algorithm* has been presented, here a simulation tool called *Psim* (current version 2.4) is discussed and a case study of its application is given as well.

1 Introduction

Membranes play a prominent role in living cells [1, 20]. In fact, membranes not only act as a separation barrier indispensable to create different environments within cells boundaries, but they can also physically constitute a kind of "working board" whereby enzymes can activate and perform their duties on substrates. Other examples of the crucial role of membranes within cells are their ability to perform selective uptakes and expulsion of chemicals as well as their being the interface of the cell with the surrounding environment allowing communication with neighboring cells.

P systems originate from the recognition of this important role of membranes and, by abstracting from the functioning and structure of living cells, they provide a novel computation model rooted in the context of formal language theory [33, 35].

P systems investigations are nowadays focused on several research lines that make the field "a fast Emerging Research Front" in computer science (as stated by the Institute for Scientific Information). In particular, theoretical investigations on the power of the computational model have been carried on and important results have been achieved so far in order to characterize the computational power of many elements of P systems (such as objects and membranes) and, from a complexity viewpoint, P systems have been employed as well in the solution of NP hard

problems. For a constant up to date bibliography of P systems we refer the reader to [39].

Parallel to these lines some more practical investigations are under way too. These studies exploit the resemblance of P systems to biological membranes in order to develop computational models of interesting biological systems. P systems seem to be particularly suitable to model biological systems, due to their direct correspondence of many elements (namely membranes, objects-chemicals and rules-reactions), even in their basic formulation, with real biological entities building the system to be modelled. Moreover, many extensions have been proposed to the standard formulation of P systems, such as some biologically relevant communication mechanisms [28, 36, 11], energy account [37] and active membranes [34] among others, which show the flexibility of the model. In this way, discrete mathematical tools can be used to represent interesting biological realities to be investigated. A further step is that of simulating all systems described in this way to get more information about their internal regulatory mechanisms and deeper insights into their underlying elements.

Although born as a non-conventional model of computation inspired by nature, P systems can therefore be employed as a simulation framework in which to embed the in silico simulation of interesting biological systems. The strength of this choice is, as said, the advantage that P systems share with biological systems many of their features and this leads to easy-to-devise and easy-to-understand descriptions of the studied realities. In fact, the membrane construct in P systems has a direct counterpart into biological membranes: objects correspond to all chemicals, proteins and macromolecules swimming in the aqueous solution within the cell and, eventually, rewriting rules represent biochemical reactions taking place in the controlled cellular environment. Other formalisms have been employed as modelling and simulation frameworks too, such as Π calculus [29], Petri nets [38] and Ambient calculus [10], but in their case the very same notions of membranes, chemicals and reactions need to be reinterpreted and immersed in the particular representation formalism in a less immediate way.

Nevertheless, the employment of P systems as a modelling framework for biological systems posed, from a purely computational viewpoint, some new challenges to cope with, such as the identification of suitable, biologically meaningful, strategies for system evolution and the development of new automatic tools to describe, simulate and analyze the investigated system.

In previous works a novel strategy for systems evolution, called *metabolic algorithm* has been introduced [6, 27, 8], an hybrid (deterministic-stochastic) variant of which has been proposed as well [5]. Other strategies of evolution are known, such as *Dynamical Probabilistic P systems* [32, 31] and *Multi-compartmental Gillespie's algorithm* [30, 2].

Here we will focus on the metabolic algorithm in its deterministic version which has been confronted with the dynamics of several systems (a collection of case studies can also be found in [4]). Some examples of investigated systems by means of the metabolic algorithm are the Belousov-Zhabotinsky reaction (in the Brus-

selator formulation) [6, 8], the Lotka-Volterra dynamics [6, 27, 7, 14], the SIR (Susceptible-Infected-Recovered) epidemic [6], the leukocyte selective recruitment in the immune response [16, 6], the Protein Kinase C activation [8], circadian rhythms [12] and mitotic cycles in early amphibian embryos [26]. In order to cope with the demand of computational tools to simulate the dynamics of P systems, we developed a first simulator called *Psim* [6], which has now been extended with several new features as will be explained later on. The new version of the simulator is freely available for download at [15].

The remaining of the discussion will firstly introduce (section 2) some theoretical aspects of the simulation framework we developed and some recent advances will be mentioned too. Section 3 will then be devoted to the newer version of the simulator itself and a practical case study will be given as well in such a way to show to the reader how to set up a simulation with the interface of Psim.

2 MP systems

MP systems (Metabolic P systems) [21, 26, 24, 23] are a special class of P systems [33, 35], introduced for expressing the dynamics of metabolic (or, more generally speaking, biological) systems. Their dynamics is computed by means of a deterministic algorithm called *metabolic algorithm* which transforms populations of objects according to a *mass partition* principle, based on suitable generalizations of chemical laws.

A definition of MP systems follows, as given in [4].

Definition 1 (MP system). *An MP system of level $n-1$ (i.e., with $n \in \mathbb{N}$ membranes) is a construct:*

$$\Pi = (T, \mu, Q, R, F, q_0)$$

in which:

- *T is a finite set of symbols (or objects) called the* alphabet*;*
- *μ is the hierarchical membrane structure, constituted by n membranes, labeled uniquely from 0 to $n-1$, or equivalently, associated in a one-to-one manner to labels from a set L of $n-1$ distinct labels;*
- *Q is the set of the possible* states *reachable by the MP system. Each element $q \in Q$ is a function $q : T \times L \longrightarrow \mathbb{R}$, from couples objects-membranes to real values. A value $q(X, l)$, with $X \in T$ and $l \in L$ gives the amount of objects X inside a membrane labeled l, with respect to a conventional unit measure (grams, moles, individuals, ...);*
- *R is the finite set of* rewriting rules*. Each $r \in R$ is specified according to the boundary notation [3]. In other words, each rule r has the form $\alpha_r \longrightarrow \beta_r$, where α_r, β_r are strings defined over the alphabet T enhanced with indexed parenthesis representing membranes. As an example, an hypothetical rule can have the form: $\alpha[_1\beta \longrightarrow \gamma[_1\delta$, with $\alpha, \beta, \gamma, \delta \in T^*$, meaning that α and β are*

respectively changed in γ and δ, where all objects within α and δ are outside membrane labeled 1*, whereas elements of β and γ are placed inside membrane* 1*;*

- *F is the set of* reaction maps*, each $f_r \in F$ is a function uniquely associated to a rule $r \in R$, defined as $f_r : Q \longrightarrow \mathbb{R}$ and, given a certain state q, it produces $f_r(q)$ that is a real number specifying the strength of rule r in acquiring objects;*
- *$q_0 \in Q$ is the* initial state *of the system. It specifies the initial amount of all the species throughout the various compartments of the system.*

Since encodings like [9] show that the membrane structure can be flattened by augmenting the alphabet size, the definition of the membrane structure μ is not very important in this context and the choice to employ *0-level* MP systems in the remaining of the discussion is not limiting from a theoretical point of view. Moreover, dealing with 0-level MP systems ends up in a easier discussion, in fact all states $q \in Q$ do not need the specification of a membrane label and in this way they have the simpler form: $q : T \longrightarrow \mathbb{R}$. For this reason, in the following whenever the term MP system will be used, the more correct term *0-level MP system* has to be implicitly assumed.

The dynamics of MP systems has been calculated, starting from the initial state q_0 by means of an evolution strategy called *metabolic algorithm* [6, 27, 8], which is substantially different from the well known *non-deterministic and maximally parallel* paradigm followed by standard P systems. More precisely, the perspective of MP systems is to model systems at a population level rather than at an objects level. In this way, nothing can be precisely said about individuals but the investigation is focused on the macroscopic dynamics which assumes a deterministic flow in spite of individual behaviors.

2.1 The metabolic algorithm: hints

Without entering into many details (which can be found anyway in [6, 27, 8, 25]), the metabolic algorithm is a deterministic strategy for MP systems evolution based on mass partition among rules of all elements in the alphabet T.

In very general terms, the metabolic algorithm can be summarized in the following main points [26]:

- Reactants are distributed among all the rules, as the system evolves, according to a "competition" strategy.
- If some rules compete for the same reactant, then each of them obtains a portion of the available substance that is proportional to its reaction strength (*reactivity*) in that state.
- The reactivity of a rule in a certain state measures the capability of the rule to acquire its reactants. It is calculated by the evaluation of the reaction map corresponding to the rule due and it depends on the state of the system, that is defined as the concentration and localization of all substances in the considered instant.

- The evolution strategy determines the reaction unit of all rules, that is, the unitary amount of substance which is dealt by the rule. The stoichiometry is used then to obtain the consumed and/or produced amount of substances for each rule.

An example may be useful to clarify the concepts yet introduced. Let us suppose that in a given instant, four rules, namely r_1, r_2, r_3 and $r_4 \in R$, need molecules of a species A (with A belonging to the alphabet T) as reactant (see Figure 1), then a partition strategy for A is necessary.

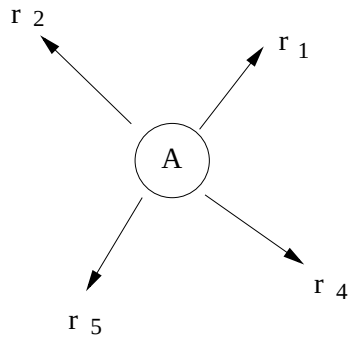

Fig. 1. Competition for object A between rules r_1, r_2, r_4 and r_5.

A real number called *reactivity* represents the strength of the rule (i.e. the rule's capability of obtaining matter to work on), given by the value assumed by a function uniquely associated to the rule called *reaction map*, in the considered state. For example, with respect to Figure 1, if we denote with a, b and c the concentrations of species A, B, C respectively (in a state q not specified for the sake of simplicity), then the reactivities of rules r_1, r_2, r_4 and r_5, which ask for A molecules, can be:

$$f_1 = 200 \cdot a, f_2 = 0.5 \cdot a^{1.25} \cdot b^{-1}, f_4 = a^{1.25} \cdot (b + c)^{-1} \text{ and } f_5 = 10$$

where the choice of reaction maps f_i, $i = \{1, 2, 4, 5\}$ is completely arbitrary in this example.

We define the quantity

$$K_{A,q} = \sum_{i=1,2,4,5} f_i(q)$$

as the *total pressure* on A in the state q (the intuitive idea is that all reaction maps of rules competing for a certain species give the force that pushes that species to react).

Then, for each of the competing rules r_j we consider the *partial pressure* (or *weight*) of r_j on type A as

$$w_{A,q}(r_j) = \frac{f_j(q)}{K_{A,q}}$$

(again the idea behind this is that the strongest the force pushing an element to follow a particular reaction channel, compared to other reaction channels, the more matter will follow that path).

Note that, in general, the quantity $K_{X,q}$ is defined for each couple (X, q) where $X \in T$ and q is a possible state of the system, moreover a weight $w_{X,q}(r)$ has to be calculated for each triple (X, q, r) where X and q are respectively, as before, an element of the biological alphabet and a state of the system, while $r \in R$ is a rule in which the element X appears as a reactant (i.e., according to the terminology adopted above, $X \in \alpha_r$).

Getting back to the example discussed above, it should be easy to see that the partial pressure of r_1 on A is

$$w_{A,q}(r_1) = \frac{200a}{200a + 0.5a^{1.25}b^{-1} + a^{1.25}(b+c)^{-1} + 10}$$

while the same pressure due to r_2 results to be equal to

$$w_{A,q}(r_2) = \frac{0.5a^{1.25}b^{-1}}{200a + 0.5a^{1.25}b^{-1} + a^{1.25}(b+c)^{-1} + 10}$$

and the other weights can be calculated analogously. The weights calculated so far determine the partition factors of the amount of species A, available in the state q, among the rules which need objects A as reactants.

Now to calculate the reaction unit of a particular rule (i.e. the amount of reactant that can be dealt by the rule) we simply need to multiply the partial pressure of the rule on the reactant by the real amount of reactant present into the system at the considered state. For example, the reaction unit of rule r_1 (or equivalently, the amount of A that that *can be* assigned to rule r_1) turns out to be $w_{A,q}(r_1) \cdot a = \frac{0.5a^{2.25}b^{-1}}{200a+0.5a^{1.25}b^{-1}+a^{1.25}(b+c)^{-1}+10}$.

In this way, if r_1 is a rule of the form $A \to X$, no matter what element is represented by $X \in T$, then the amount of A associated to r_1 is exactly $u_1 = w_{A,q}(r_1) \cdot a$ and the effect of r_1's application is the loss of u_1 units of A and the acquisition of the same number of units of X.

In the case of cooperative rules (i.e. rules with more than just one reactant) things are a little bit more complicated since we need to take into account the real availability of all reactants taking part to the reaction. That is, for each X belonging to the reactants of a certain rule r we need firstly to compute the quantities $w_{X,q}(r) \cdot x$ and, since we have to respect species availability, the reaction unit associated to the rule is then computed as the minimum of those quantities. If we suppose that a rule r_1 has the form $AAB \to X$, then the reaction unit

$$u_1 = min(\frac{1}{2} w_{A,q}(r_1) \cdot a \ , \ w_{B,q}(r_1) \cdot b)$$

where the term $\frac{1}{2}$ in the first element of the minimum is due to the fact that A appears twice in the stoichiometry of the rule.

In general terms, the metabolic algorithm is an effective procedure for calculating in each state of the system a reaction unit u_i for all rules $r_i \in R$ by using a partition strategy that employs particular functions f_i associated in a 1-1 manner to rules. After this calculation, the evolution of the system can be obtained in a straightforward way by consuming and producing species in a quantity given by rules' reaction units and by following the stoichiometry of the system.

Assuming an ordering on objects and on rules, let us denote with M the $m \times n$ stoichiometric matrix associated to an MP system having m symbols and n rules (in which the $c_{i,j}$ element of M denotes the gain or the loss of the ith object due to rule j[1]) and with U_q the $[u_1 \ \cdots u_n]^T$ vector of the reaction units in a state of the system q, calculated as mentioned above.

Then, as pointed out in [25] the transition from one state q to the following one is done by means of the *delta operator* ($\Delta_x(q)$) which is a m-sized vector giving the variation of each species in the transition from state q to the next state q'. In particular

$$\Delta_x(q) = M \times U_q$$

stating that the delta operator can be obtained as the product of the stoichiometric matrix M by the reaction units vector of state q, U_q.

Since each row i of $\Delta_x(q)$ gives the variation on the ith object, then if we think of a state q as a vector containing the concentration of all the m species at the corresponding instant, then the next state can easily be calculated as

$$q' = q + \Delta_x(q) = q + M \times U_q.$$

Just to exemplify the last concepts discussed, we can think about an alphabet $T = \{A, B, C\}$ and focus on a rule set comprising the following four rewriting rules:

$$\begin{aligned} r_1 &: A\ B \longrightarrow C \\ r_2 &: B\ B \longrightarrow A \\ r_3 &: C \quad\ \longrightarrow A \\ r_4 &: C \quad\ \longrightarrow B \end{aligned}$$

then, assuming the lexicographic order on elements of the alphabet, we can obtain the following stoichiometric matrix:

$$M = \begin{bmatrix} -1 & 1 & 1 & 0 \\ -1 & -2 & 0 & 1 \\ 1 & 0 & -1 & -1 \end{bmatrix}$$

in which, the first row corresponds to the object A and states that we lose one conventional unit of A due to rule r_1, we get one A both from rule r_2 and r_3 and finally r_4 does not affect A concentration at all.

[1] It is the difference between the number of occurrences of the ith symbol among products and among reactants of jth rule.

Then, let us suppose to be in a state q, described by the vector of concentrations $q = [10\ 32\ 20]^T$ (i.e. we have 10 units of A, 32 of B and 20 of C) and that the corresponding reaction units vector $U_q = [7\ 12\ 5\ 9]^T$ (i.e. reaction r_1 moves 7 mass units, r_2 12, r_3 5 and finally r_4 9). In this way it is possible to calculate the next state q' which turns out to be described by the following vector:

$$q' = q + M \times U_q = \begin{bmatrix} 10 \\ 32 \\ 20 \end{bmatrix} + \begin{bmatrix} -1 & 1 & 1 & 0 \\ -1 & -2 & 0 & 1 \\ 1 & 0 & -1 & -1 \end{bmatrix} \times \begin{bmatrix} 7 \\ 12 \\ 5 \\ 9 \end{bmatrix} = \begin{bmatrix} 20 \\ 10 \\ 13 \end{bmatrix}$$

describing the amount of all species at that particular instant.

In previous papers [26] a convenient and intuitive formalism for representing MP systems called *MP graphs* has been proposed. In particular, MP graphs are bipartite graphs describing both the stoichiometry (i.e. the shape of the rules) and the regulative part of MP systems that need to be effectively calculated in order to obtain the dynamics of the system (i.e. the reaction maps). According to what said above, MP graphs represent all the information needed to simulate MP systems by means of the metabolic algorithm. An example of MP graphs, as produced by the simulator we developed, will be shown later on.

2.2 Generalizing the metabolic algorithm

According to the formulation of the dynamics given in the previous section, the metabolic algorithm is a strategy that given a particular state q provides the system with the corresponding reaction units vector U_q which is used to calculate the transition to the state $q + 1$. As discussed in [25], other strategies can be considered whose aim is to produce a reasonable mass partition among all rules of an MP system, or in other words that give a different U_q for each state q of the system.

This view leads to the definition of several *metabolic algorithms* instead of a single one and the definition of MP systems can be generalized accordingly.

Based on the definition given in [22], Definition 1 can be easily generalized, in very general terms, in the following way:

Definition 2 (Generalized MP systems). *A 0-level (generalized) MP system is a 6-tuple:*

$$\Pi = (T, Q, R, V, q_0, \phi)$$

in which:

- T *is a finite set of symbols (or objects) called the* alphabet*;*
- V *is a finite set of* variables*;*
- Q *is the set of the possible* states *reachable by the MP system. Each element* $q \in Q$ *is a function* $q : T \cup V \longrightarrow \mathbb{R}$*;*
- R *is the finite set of* rewriting rules*;*

- $q_0 \in Q$ *is the* initial state *of the system;*
- ϕ *is the* strategy of evolution, $\phi : Q \longrightarrow \mathbb{R}^n$ *with* $|R| = n$.

Note that nothing is said about the cardinality of the set of variables V and they are not necessarily associated in a one-one manner to rules of R. Moreover, the strategy of evolution ϕ, given a state q has to be defined in such a way that it outputs the n-tuple providing the reaction unit vector of the system, or following the terminology used above, $\phi(q) = U_q$.

Complete freedom is left in the implementation of the strategy of evolution, whose only constraints are that given a state it has to provide the reaction unit vector corresponding to that state, which will be used to calculate the evolution of the system by means of the matrix product recalled in the previous section. As will be mentioned in the following, the specification of a fully customizable strategy of evolution will be one of the prominent features of the new version of the simulator Psim that has been implemented within the MNC Group of the University of Verona.

3 Psim

Based on the theoretical framework expressed in previous sections, a simulator called *Psim* (P systems simulator) has been developed to cope with the problem of calculating the dynamics of biological systems. An early version of Psim has been developed previously [6], with which the newer version shares the same philosophy, though extending some of its concepts and enhancing the simulation environment with many features.

The present release of Psim (version 2.4) has been developed in response to the need of an effective and easy to use tool to calculate the dynamics of MP systems by means of the metabolic algorithm. Its implementation has moreover followed some flexibility and extensibility principles which led to a tool that can be easily extended and integrated with other tools. In this way Psim, thanks to its immediate setup (nothing needs actually to be done provided a Java virtual machine is installed on the computer that is meant to run Psim) and to the easy user interface, can be used by people without a strong background in programming and a deep knowledge in the field of computer science. On the other hand, the extendability provided, as we will see, by means of the plugins mechanism, allows people with stronger expertise in programming to build their own tools to complement and integrate the main core of Psim.

Some features of this tool, which is implemented by using the Java programming language, are listed below:

- friendly user interface which is born to be easy-to-use and to interact with people not necessarily having a strong computer science background. Its immediacy can be found in the input side, which can be specified by means of a transposition of the concept of MP graphs into a point and click graphical

interface. Moreover, the same simplicity principle holds for the output side as well, which is basically constituted by a graph containing the temporal evolution of all the species constituting the system (both on a temporal scale and on the phase plan space);

- plugins architecture: the interaction with the system can either be done manually or by means of some specifically designed plugins which, thanks to the plugins support offered by the simulation engine, can interact with the simulation engine itself. More specifically, three different kinds of plugins can be devised and implemented in Java as well. *Input plugins* can be used to implement various sources for the data to run the simulation on (let us think to some specific pathways databases like KEGG for instance); *output plugins* conversely, can be used to observe and analyze in various ways the results obtained from a simulation and can therefore give some meaningful insights into the simulated dynamics. Moreover, they can be used to export simulation data into particular formats. Finally, *experiment plugins* can directly control and intimately interact with the simulation engine, by controlling the execution flow, checking some properties and changing some experimental conditions. This kind of plugins can be very useful in tasks like model optimizations and stability analysis;
- extreme flexibility. The simulation tool we propose is based on a simulation engine which is designed to accept the definition of new evolution strategies for the calculation of the systems dynamics. At the only price of the implementation of some specific interfaces, the developer has the chance to define his own simulation strategies and to design a customized library of *metabolic algorithms* to calculate the systems evolution;
- models portability has been implemented by using the standard XML language and some extensions towards the SBML language are being considered too;
- cross platform applicability, thanks to the choice of Java, Psim can be run on all platforms supporting the Java virtual machine architecture.

An aspect deserving a special emphasis here is the possibility offered by the simulation engine, to specify custom evolution strategies. Getting back to the definition of generalized MP systems, the architecture of the simulator allows the specification of a fully customized ϕ function. A set of evolution strategies can be devised by developing in Java a specific class implementing a particular interface provided by the main engine. Several different strategies can be handled simultaneously by the simulator that gives the chance to decide which simulation strategy employ in the simulation process. This gives the tool a very high level of flexibility and power as well as the plugins mechanism does. Since plugins can interact with the simulation engine at a different levels, such as input, output but also at the simulation level too, they can be used for various reasons within the simulator and this again gives users plenty of ways to improve the system and to extend its functionalities.

3.1 A case study

In this section we show an application of the Psim computational tool for the simulation of the well known mitotic oscillator as found in early amphibian embryos [18, 19, 26].

Mitotic oscillations are a mechanism exploited by nature to regulate the onset of mitosis, that is the process of cell division aimed at producing two identical daughter cells from a single parent. More precisely, mitotic oscillations concern the fluctuation in the activation state of a protein produced by cdc2 gene in fission yeasts or by homolog genes in other eukaryotes. The model here considered focuses on the simplest form of this mechanism, as it is found in early amphibian embryos. Here, the progressive accumulation of the cyclin protein leads to the activation of cdc2 kinase. This activation is achieved by a bound between cyclin and cdc2 kinase forming a complex known as M-phase promoting factor (or *MPF*). The complex triggers mitosis and degrades cyclin as well; the degradation of cyclin leads to the inactivation of the cdc2 kinase that brings the cell back to the initial conditions in which a new division cycle can take place.

Goldbeter [18, 19] proposed a minimal structure for the mitotic oscillator in early amphibian embryos in which the two main entities are cyclin and cdc2 kinase. According to this model, depicted in Figure 2, the signalling protein cyclin is produced at a constant rate v_i and it triggers the activation (by means of a dephosphorylation) of cdc2 kinase, passing from the inactive form labelled M^+ to the active one, denoted by M. This modification is reversible and the other way round is performed by the action of another kinase (not taken into account in the model) that brings M back to its inactive form M^+. Moreover, active cdc2 kinase (M) elicits the activation of a protease X^+ that, when in the active (phosphorylated) form (X), is able to degrade the cyclin. This activation, as the previous one, is reversible as stated by the arrow connecting X to X^+.

The set of differential equations devised by Goldbeter produces an oscillatory behavior in the activation of the three elements M, C, X that repeatedly go through a state in which cells enter in a mitotic cycle (see Figure 3).

The goal of the case study showed here is to obtain a description and a simulation of the very same model of mitotic oscillations by means of the simulator Psim.

In general, there is no unique way to translate a differential equation system in terms of a metabolic P system, therefore we choose to obtain it by the application of the MP-ODE transformation [13]. The resulting MP system is reported here:

$$\Pi = (T, \mu, \mathcal{R}, F, q_0)$$

where:

- The alphabet: $T = \{A, C, X, X^+, M, M^+\}$
- The membrane structure: $\mu = [_0\ ]_0$;

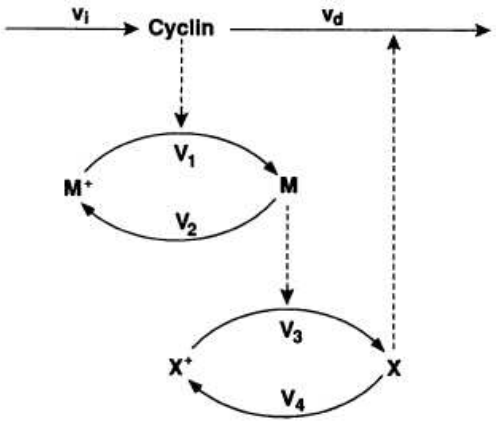

Fig. 2. The mitotic oscillator model by A. Goldbeter, from [18].

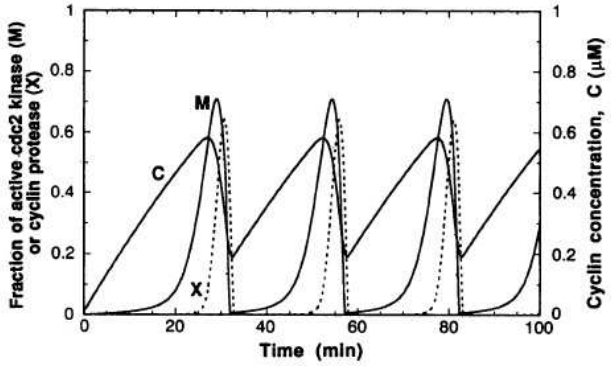

Fig. 3. Dynamics of the mitotic oscillator from [18].

- The set of rules is $\mathcal{R} = \{r_1, r_2, ..., r_{10}\}$, where:
  $r_1 : A \rightarrow A\ C$
  $r_2 : C \rightarrow X$
  $r_3 : X \rightarrow \lambda$
  $r_4 : C \rightarrow \lambda$
  $r_5 : C \rightarrow M\ C$
  $r_6 : M^+ \rightarrow \lambda$
  $r_7 : M \rightarrow M^+$
  $r_8 : X^+ \rightarrow X\ M$
  $r_9 : M \rightarrow \lambda$
  $r_{10} : X \rightarrow X^+$
  in which all symbols have the meaning described before (and A is a kind of well to draw substance C out from). Moreover, for every symbol in the system, we

have introduced an inertia rule (i.e., a rule having the form $Y \rightarrow Y$, for each $Y \in T$ to model the inertia of the system), omitted in this set of rules.

- The set of reaction maps is $F = \{Fr_1, Fr_2, ..., Fr_{10}\}$, where:
  $Fr_1 = k_1$
  $Fr_2 = k_2 \frac{x}{k_3+c}$
  $Fr_3 = k_2 \frac{c}{k_3+c}$
  $Fr_4 = k_3$
  $Fr_5 = k_5 \frac{m^+}{(k_6+c)(k_7+m^+)}$
  $Fr_6 = k_5 \frac{c}{(k_6+c)(k_7+m^+)}$
  $Fr_7 = k_8 \frac{1}{(k_9+m)}$
  $Fr_8 = k_{10} \frac{m}{(k_{11}+x^+)}$
  $Fr_9 = k_{10} \frac{x^+}{(k_{11}+x^+)}$
  $Fr_{10} = k_{12} \frac{1}{(k_{13}+x)}$
- The initial state q_0 of the single membrane system is defined by:
  $q_0(A) = 1.3$;
  $q_0(C) = 0.01$;
  $q_0(X) = 0.01$;
  $q_0(X^+) = 0.99$;
  $q_0(M) = 0.01$;
  $q_0(M^+) = 0.99$;
  in which we deal with concentrations of species, rather than with objects, and in this way the initial amounts are real numbers;

where, for each element of T the reaction map of inertia rules is set to 1600.

We start the modeling session by opening the Psim's main interface showed in Figure 4. This window allows the user to manage all the experiment's stages. In particular the main possible choices involve:

1. modelling the system, setting substances, initial conditions, reaction maps and rules;
2. starting the simulation;
3. displaying output charts.

The first step to consider while setting up a system's simulation is the specification of the corresponding MP graph. In what follows, some steps towards the creation of an MP graph modelling the mitotic oscillator are presented.

After clicking on the *New Experiment* label of the *File* menu, a window like the one depicted in the Figure 5 appears. This is the main window of the graphical interface allowing the user to input in a easy way the MP graph components by simply dragging them from the upper toolbar to the bottom white panel. This task is performed by using the following toolbar icons:

- The *blue circle*: adds a new *type node* that stores the name of a substance, its initial number of molar units and its inertia value (as explained in previous papers, inertias are a way to represent the fact that not all reactants need to

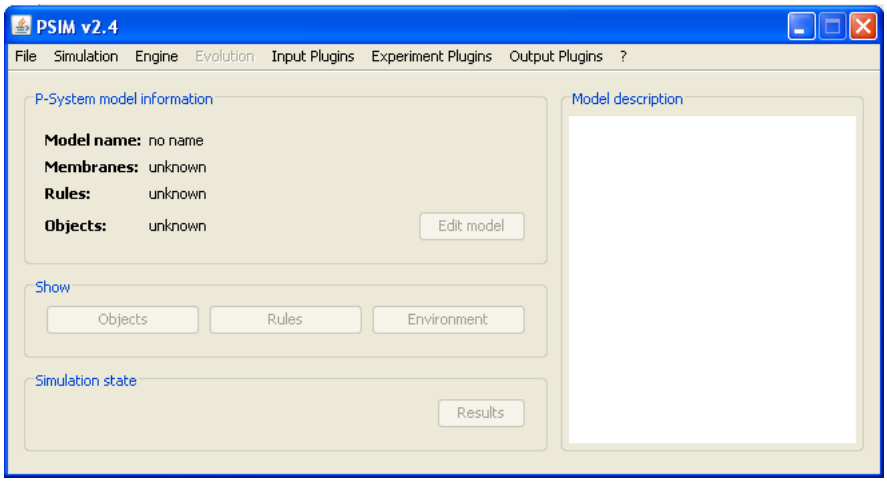

Fig. 4. The Psim's main interface

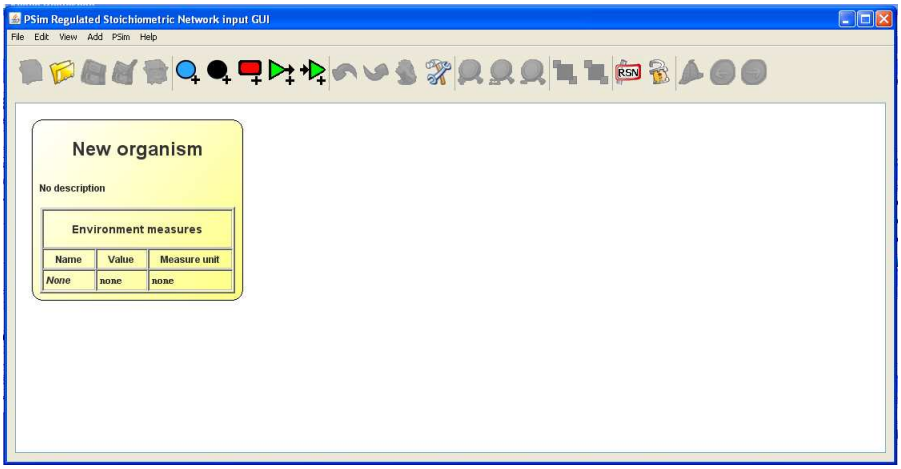

Fig. 5. Psim's input interface

react at a certain instant, they are a sort of resistance opposed to species to performing reactions).

- The *black circle*: adds a new *metabolic reaction node* that represents a reaction channel between interacting substances and stores the name of a reaction rule.
- The *red rectangle*: adds a new *reactivity node* building the regulatory part of MP graphs. In the simulator, reactivity nodes store the reactivity map function corresponding to the connected rule and, if necessary, a boolean guard function for the rule activation.

- The *green triangles*: add *input gates* and *output gates* nodes that identify rules which respectively introduce new matter in the membrane or expel part of it from the system.

After the insertion of the nodes in the white panel the user can specify their internal parameters and connect the nodes by drawing arcs among them. The best way to accomplish this task is to start by defining the *type node* parameters and the *metabolic reaction node* parameters by double clicking on the corresponding nodes and filling in the window's field that automatically appears (Figure 6). Importantly enough, a parser has been implemented to check the consistency of inserted parameters and to alert the user if any irregularity arose.

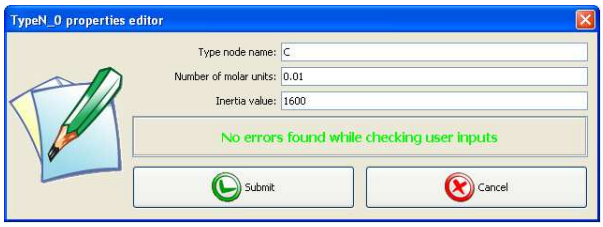

Fig. 6. Insertion of the *type node* parameters

At this point, one can connect the *type nodes* and the *metabolic reaction nodes* with each other by drawing arcs among them with the simple use of the mouse. This is a very important step because it allows to represent the stoichiometric part of the system by means of the MP graph topology (Figure 7).

As an example, let us consider the reaction $r_2 : C \rightarrow X$. Within the input graphical interface it is represented by the $R2$ black circle that is connected by means of black arcs to the C and the X blue circles, representing the corresponding substances; the direction of the arrow represents the substance flow of the reaction.

A further modeling step is needed to add the *reactivity nodes* describing the regulatory part of the system. This can be done by first linking every *type node*, that affects a reaction map, with the corresponding *reactivity nodes* (as showed in Figure 8). Finally the reactivity map function of every reactivity node is specified by using the linked *type nodes* and the *environmental measures* as variables or constants (as reported in Figure 9). Figure 8 represents the final mitotic oscillator MP graph as produced by the Psim GUI.

This completes the modelling stage and the next logical step is to start the simulation of the specified system. This is done by clicking on the rightmost icon of the upper toolbar (the rightward arrow). The click causes a small window to pop out, in which it is possible to set the number of steps the simulation will span (Figure 10). A possible choice for this system is to run 150000 steps. By click on the *Start* button the dynamics computation begins.

When the simulation is finished the system prompts that results are available and ready to be visualized by the *Psim chart visualization form* (Figure 11). Using

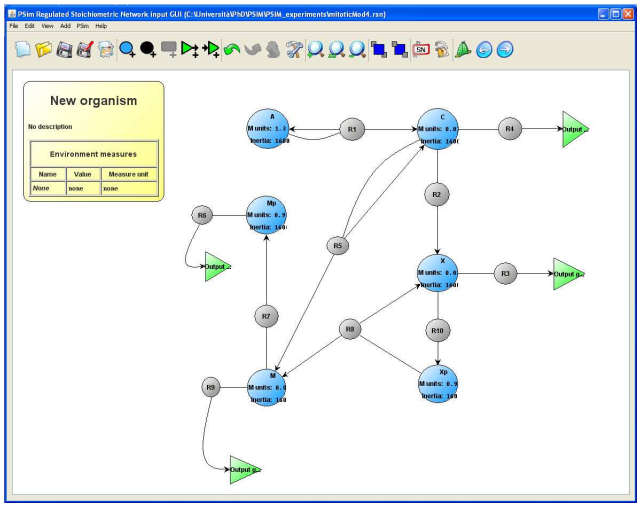

Fig. 7. Adding *type nodes*, *metabolic reaction nodes* and drawing arches among them

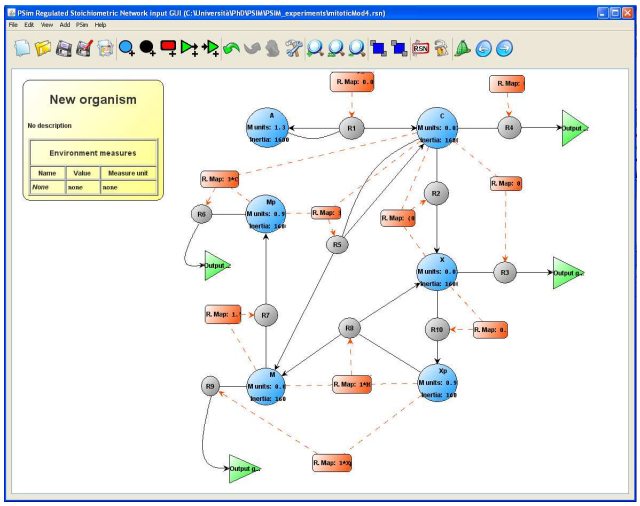

Fig. 8. An MP graph that models the mitotic oscillator

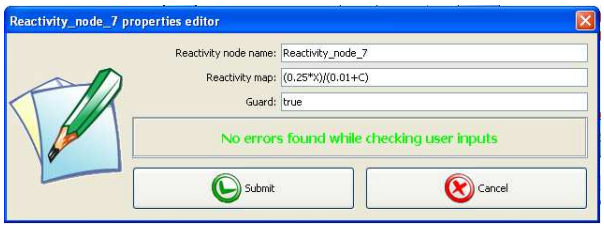

Fig. 9. Reactivity node input parameter window

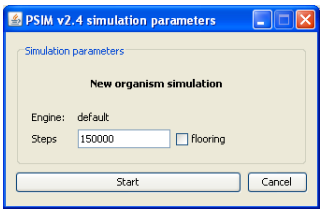

Fig. 10. Set the number of steps for the simulation to 150000

the bottom panel check boxes it is possible to decide elements to be displayed. In the considered case oscillations of cyclin C (red line), active cdc2 kinase M (blue line) and active protease M (green line) are displayed but the phase space plot can be drawn as well.

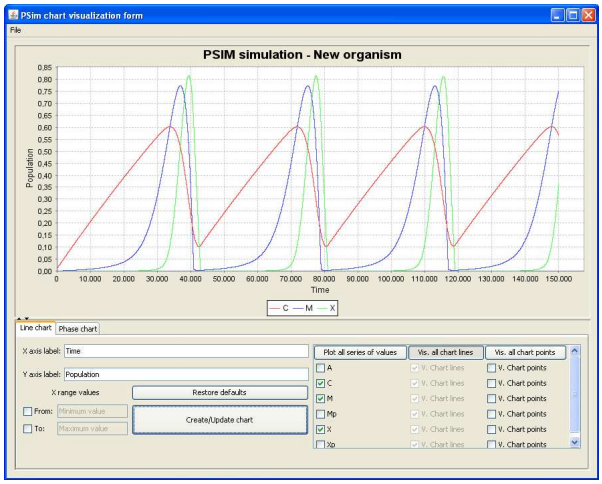

Fig. 11. Oscillations of the mitotic systems as calculated by Psim.

We finally highlight an important mechanism of the Psim platform: plugins extensibility. As already mentioned, plugins allow the user to enhance the main Psim computing core with powerful functionalities for the data import, export, control and analysis. An example under development is the *experiment plugin* that stores the experiment state (concentration of the substances and environmental measures) every x steps, where x is a parameter set by the user before the computation starts. This plugin could save, for instance, an XML file for every state, allowing the user to export the experiment samples in an standard way.

A software developer generates the plugin code (basically some Java classes) relying on the Psim's *JavaDoc* documentation obtainable at [15] which lists the *experiment plugin* methods to be mandatorily implemented. Plugin classes are

meant to be archived in a Jar file and placed in a proper *plugins* directory. Provided this, at the following start up Psim will automatically find and load all the plugins contained in the *plugins* directory. The user can find all available experiment plugin statements in the main interface *Experiment Plugin* menu (Figure 4) in the form of a list. By clicking on the relative label its possible to activate the plugin that will be run at each step of the subsequent simulation and will save the state every x steps chosen by the user filling in a plugin popup window.

The plugin just described yields a set of XML files but the same principles can be extended also to the other kinds of plugins (*input*, *output*, *engine plugins*).

A particular mention is deserved by *engine plugins* that allow to implement new simulation strategies which can be different from the *metabolic algorithm* described above. This gives the simulation tool a very high flexibility as well as extendability as discussed previously.

4 Conclusion and further work

P systems can be useful frameworks to embed biological systems models in. This demands for some modifications to the classical definition of P systems and particularly a biologically meaningful evolution strategy is needed. In previous papers an essentially deterministic strategy, called metabolic algorithm, for the calculation of biological systems dynamics has been provided as well as an extension of the classic model of P systems, known as MP systems, focused on the dynamics of bio-systems. Moreover, all data needed for the simulation of MP systems dynamics can be provided by means of a graphical formalism known as MP graphs.

The basics of MP systems have been briefly revisited in this paper and based on them a simulation tool called Psim has been highlighted, together with a case study of a well known and previously investigated model of mitotic cycles in early amphibians. Psim v. 2.4 is the latest release of the MP systems simulator developed within the MNC Group of the University of Verona and it has very interesting features such as the plugin mechanism and the meta-engine architecture which give the tool an high level of extendability and personalization. In particular, plugins can be useful to perform several tasks such as data import/export, control of the simulation flow, output of dynamics obtained and analysis of the results among others. Moreover, the meta-engine architecture of the simulator allows users to define their own evolution strategies by implementing some fixed interfaces of the simulator.

In the future we plan to enrich the core of this simulation tool by implementing a series of plugins such as the one described above to have a snapshot of the state of the system in particular instants. Other plugins under investigation involve some automatic procedures for parameter estimation given suitable observations of the reality to be modelled. Finally, we plan to employ this simulation tool for the calculation of the dynamics of systems not already modelled and in this respect the possibility to devise ad-hoc evolution strategies can be very important to tackle some specific issues related with the particular reality to be modelled.

References

1. B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, and P. Walter. *Biology of the Cell.* Garland Science, New York, 4th edition, 2002.
2. F. Bernardini, M. Gheorghe, N. Krasnogor, R.C. Muniyandi, M.J. Pérez-Jiménez, and F.J. Romero-Campero. On P Systems as a modelling tool for biological systems. In R. Freund, G. Lojka, M. Oswald, and Gh. Paŭn, editors, *Pre-Proceedings of the 6th International Workshop on Membrane Computing (WMC6)*, pages 193–213, 2005.
3. F. Bernardini and V. Manca. Dynamical aspects of P systems. *BioSystems*, 70:85–93, 2002.
4. L. Bianco. *Membrane Models of Biological Systems.* PhD thesis, Verona University, 2007.
5. L. Bianco and F. Fontana. Towards a Hybrid Metabolic Algorithm. In H. J. Hoogeboom, G. Păun, and G. Rozenberg, editors, *7th International Workshop on Membrane Computing, WMC07 Leiden. Selected Papers*, number 4361 in LNCS, pages 183–196. Springer-Verlag, Berlin, 2006.
6. L. Bianco, F. Fontana, G. Franco, and V. Manca. P systems for biological dynamics. In G. Ciobanu, G. Păun, and M. J. Pérez-Jiménez, editors, *Applications of Membrane Computing.* Springer, Berlin, 2006.
7. L. Bianco, F. Fontana, and V. Manca. Reaction-Driven Membrane Systems. In K. Chen L. Wang and Y.S. Ong, editors, *Advances in Natural Computation, First International Conference, ICNC 2005 (part II)*, Proceedings of "First International Conference in Natural Computation (ICNC 2005)", pages 1155–1158. Springer, 2005.
8. L. Bianco, F. Fontana, and V. Manca. P systems with reation maps. *International Journal of Fondations of Computer Science*, 16(1), 2006.
9. L. Bianco and V. Manca. Encoding-Decoding Transitional Systems for classes of P Systems. In Freund et al. [17], pages 135–144.
10. L. Cardelli and A. D. Gordon. Mobile Ambients. In M. Nivat, editor, *Proceedings of the First international Conference on Foundations of Software Science and Computation Structure. LNCS 1378, Springer, 1998*, pages 140–155.
11. M. Cavaliere. Evolution-Communication P Systems. In G. Păun, G. Rozenberg, A. Salomaa, and C. Zandron, editors, *Membrane Computing: International Workshop, WMC-CdeA 2002, Curtea de Arges,Romania, August 19-23, 2002. Revised Papers.*, volume 2597 of *Lecture Notes in Computer Science*, pages 134–145, Curtea de Arges, Romania, July 2003. Springer-Verlag, Berlin.
12. F. Fontana, L. Bianco, and V. Manca. P systems and the Modeling of Biochemical Oscillations. In Freund et al. [17], pages 199–208.
13. F. Fontana and V. Manca. Discrete Solution of Differential Equations by Metabolic P Systems. *Submitted to TCS*, 2006.
14. F. Fontana and V. Manca. Predator-prey Dynamics in P Systems Ruled by Metabolic Algorithm. *BioSystems, Accepted*, 2006.
15. The Center for BioMedical Computing Web Site. Url: http://www.cbmc.it.
16. G. Franco and V. Manca. A membrane system for the leukocyte selective recruitment. In A. Alhazov, C. Martin-Vide, and G. Păun, editors, *4th Workshop on Membrane Computing*, volume 2933 of *Lecture Notes in Computer Science*, pages 180–189. Springer-Verlag, 2004.
17. R. Freund, G. Paun, G. Rozenberg, and A. Salomaa, editors. *Membrane Computing, 6th International Workshop, WMC 2005, Vienna, Austria, July 18-21, 2005, Revised Selected and Invited Papers*, volume 3850 of *Lecture Notes in Computer Science.* Springer, 2006.

18. A. Goldbeter. A Minimal Cascade Model for the Mitotic Oscillator Involving Cyclin and cdc2 Kinase. *PNAS*, 88(20):9107–9111, 1991.
19. A. Goldbeter. *Biochemical Oscillations and Cellular Rhythms. The molecular bases of periodic and chaotic behaviour*. Cambridge University Press, New York, 2004.
20. H. F. Lodish, A. Berk, P. Matsudaira, C. Kaiser, M. Krieger, M. Scott, L. Zipursky, and J. E. Darnell. *Molecular Cell Biology*. Scientific American Press, New York, 5th edition, 2004.
21. V. Manca. Topics and problems in metabolic p systems. In *Proc. of the Fourth Brainstorming Week on Membrane Computing (BWMC4)*, 2006.
22. V. Manca. Discrete Simulations of Biomolecular Dynamics. *Submitted*, 2007.
23. V. Manca. Metabolic Dynamics by MP Systems. In *InterLink ERCIM Workshop, Eze, France, May 10-12*, 2007.
24. V. Manca. Metabolic P systems for Biochemical Dynamics. *Progress in Natural Sciences, Invited Paper*, 2007.
25. V. Manca. The Metabolic Algorithm for P Systems Principles and Applications. *Submitted*, 2007.
26. V. Manca and L. Bianco. Biological Networks in Metabolic P Systems. *BioSystems, Accepted*, 2006.
27. V. Manca, L. Bianco, and F. Fontana. Evolutions and oscillations of P systems: Theoretical considerations and applications to biochemical phenomena. In G. Mauri, G. Păun, M. J. Pérez-Jiménez, G. Rozenberg, and A. Salomaa, editors, *Membrane Computing, International Workshop, WMC5, Milano, Italy, 2004, Selected Papers*, number 3365 in LNCS, pages 63–84. Springer-Verlag, Berlin, 2005.
28. C. Martin-Vide, G. Păun, and G. Rozenberg. Membrane systems with carriers. *Theoretical Computer Science*, 270:779–796, 2002.
29. R. Milner. *Communicating and Mobile Systems: The π Calculus*. Cambridge University Press, Cambridge, England, 1999.
30. M. J. Pérez-Jiménez and F. J. Romero-Campero. P systems: a new computational modelling tool for systems biology. *Transactions on Computational Systems Biology VI, Lecture Notes in Bioinformatics, 4220*, pages 176–197, 2006.
31. D. Pescini, D. Besozzi, and G. Mauri. Investigating local evolutions in dynamical probabilistic p systems. In *In G. Ciobanu, Gh. Paun, Pre-Proc. of First International Workshop on Theory and Application of P Systems, Timisoara, Romania, September 26-27*, pages 83–90, 2005.
32. D. Pescini, D. Besozzi, G. Mauri, and C. Zandron. Dynamical probabilistic P systems. *International Journal of Foundations of Computer Science*, 17(1):183, 2006.
33. G. Păun. Computing with membranes. *J. Comput. System Sci.*, 61(1):108–143, 2000.
34. G. Păun. P Systems with Active Membranes: Attacking NP-Complete Problems. *Journal of Automata, Languages and Combinatorics*, 1(6):75–90, 2001.
35. G. Păun. *Membrane Computing. An Introduction*. Springer, Berlin, Germany, 2002.
36. G. Păun and A. Păun. The power of communication: P systems with symport/antiport. *New Generation Computing*, 20(3):295–306, 2002.
37. G. Păun, Y. Suzuki, and H. Tanaka. P systems with energy accounting. *Int. J. Computer Math.*, 78(3):343–364, 2001.
38. W. Reisig. *Petri Nets, An Introduction*. EATCS, Monographs on Theoretical Computer Science. 1985.
39. The P Systems Web Site. Url: http://psystems.disco.unimib.it.