Introduction to Psim

Luca Bianco

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

Abstract

Psim is the simulator of P systems developed by the *Group for Models of Natural Computing* (MNC group) in Verona[1]. It's based on the implementation of the metabolic algorithm introduced in [2] and briefly reviewed in Section 1. The algorithm is inspired by the Law of Mass Action. The simulator is developed in Java and in this way it is cross-platform. We start describing formally the algorithm implemented by the simulator and the membrane structure on which it's based. Then we discuss some practical details of its usage by giving the implementation of some concrete models of dynamical systems. We end with a Section devoted to future planned developments of Psim.

1 The metabolic algorithm

In this Section we briefly describe the metabolic algorithm used by Psim to implement the evolution of the P-system it intends to simulate. More details on the metabolic algorithm can be found in [2].

A central problem connected with P systems is the strategy of application of the set of rewriting rules. We think that the classical approach based on the "nondeterministic and maximally parallel" rules' application is not very realistic. For this reason we propose the metabolic algorithm. We want to stress the fact that this algorithm is inspired by the *Law of mass action*. This is a well known chemical law and it states that the driving force of a chemical reaction is directly proportional to the active masses of all the reactants.

Let's start with a concrete example of the translation from rewriting rules to metabolic equations. Consider the following set of rules:

$$\begin{array}{lrcl} r1: & AC & \rightarrow & AB \\ r2: & BC & \rightarrow & A \\ r3: & BBB & \rightarrow & BC \end{array} \tag{1}$$

each of them associated to a reactivity coefficient, respectively k_{r1}, k_{r2}, and k_{r3} that will be explained later on. First of all, we consider symbols as representing *moles* of objects rather than single objects. This means that the first rule, $r1$, says that a mole of reactant

[1]For more information on the composition of the group and on our actual main interests please visit our web page at the url: http://www.di.univr.it

A combined with a mole of the other reactant C produces a mole of product A and a mole of product B. Moles are, in this sense, reaction units and the amount of objects constituting a mole will depend on some modelling choices explained later on. Every rule can affect the concentration (e.g. the number of moles) of every symbol it deals with. For this reason we have to consider the effects of the whole set of rules in order to describe the trend of the concentration of an object in time.

The key point for understanding the metabolic algorithm is to look at rewriting rules of P systems as chemical reactions. In this fashion the left-hand part of a rule contains the reactants while the right one contains the products of the reaction described by the rule. From this viewpoint we can consider the *molar variation* induced by the chemical reactions, modelled by the rewriting rules, on the set objects present into the system. The molar variation of an object due to the application of a rule is defined as the difference in concentration of the object present into the system before the execution of the rule and after it.

For example, let's compute the variation, $\Delta_{r1}||C||$, on object C due to rule $r1$. By looking at rule $r1$, we see that C appears into the reactants part of it but not in the products one. In this sense the reaction transforms object C into something else (e.g. object B): it's not hard to understand that the application of the rule decreases the concentration of C (while simmetrically increasing the concentration of B). This contribution on the reactant C is independent from its concentration. We call this factor *stoichiometric coefficient* of the rule. It's basically defined as the multiplicity of C in the product decreased by the multiplicity of C in the reactant. Following the previously introduced terminology we can say that:

$$\Delta_{r1}||C|| = \ -k_{r1}||A|| \cdot ||C||$$

where the term $||A|| \cdot ||C||$ is the product of the concentration of reactants relative to rule $r1$ (e.g. A and C); this resembles the application of the law of mass action. The stoichiometric factor in this example assumes the value -1 because C is present only in the reactants with cardinality equal to one.

In order to calculate the overall variation on the concentration of C during a step of simulation, we have to sum over all the variations induced by every rule.

Next, we illustrate the variation on other objects according to the rule set (1):

$$\begin{array}{lcl}
\Delta_{r1}||A|| & = & 0\ k_{r1}||A|| \cdot ||C|| \ = \ 0 \\
\Delta_{r2}||A|| & = & k_{r2}||B|| \cdot ||C|| \\
\Delta_{r3}||A|| & = & 0\ k_{r3}||B||^3 \ = \ 0 \\
\Delta_{r1}||B|| & = & k_{r1}||A|| \cdot ||C|| \\
\Delta_{r2}||B|| & = & -k_{r2}||B|| \cdot ||C|| \\
\Delta_{r3}||B|| & = & -2k_{r3}||B||^3 \\
\Delta_{r2}||C|| & = & -k_{r2}||B|| \cdot ||C|| \\
\Delta_{r3}||C|| & = & k_{r3}||B||^3
\end{array} \tag{2}$$

In this way the variation $\Delta||X||$ of every object $X \in \{A,\ B,\ C\}$ turns out to be the sum of all its variations:

$$\begin{array}{lcl}
\Delta|A|| & = & k_{r2}||B|| \cdot ||C|| \\
\Delta||B|| & = & k_{r1}||A|| \cdot ||C|| - k_{r2}||B|| \cdot ||C|| - 2k_{r3}||B||^3 \\
\Delta||C|| & = & -k_{r1}||A|| \cdot ||C| - k_{r2}||B|| \cdot ||C|| + k_{r3}||B||^3
\end{array} \tag{3}$$

In general, let R be a set of rules of a P system and assume that each $r \in R$ is of the form $\alpha \to \beta$. In this way α is the reactant part and β the product part of r. Let's denote with

X an object involved into r; the metabolic algorithm expresses the effect of rule r on X as:

$$\Delta_r||X|| = k_r \, ||\alpha|| \, (|\beta|_X - |\alpha|_X) \tag{4}$$

where k_r is the reactivity coefficient of r, $||\alpha||$ is the product of the masses of all reactants of r, $|\alpha|_X$ is the number of X present into the string α, while $|\beta|_X$ is the number of X present into the string β. The term $(|\beta|_X - |\alpha|_X)$ is the stoichiometric coefficient of X inside r; for example in the case of rule $r3$ and object B, it is $(|\beta|_B - |\alpha|_B) = -2$ as stated in $\Delta_{r3}||B||$ in (2).

In order to obtain the whole effect of a computation step on X we have to consider the sum over all rules of R of their contribution on the concentration of X:

$$\Delta||X|| = \sum_{r \in R} \Delta_r||X||. \tag{5}$$

We would like to spend a few words on the reactivity coefficient of every rule. This is a very important factor that takes into account several interconnected aspects of a reaction like speed, synchronization times, priority between rules and degree of parallelism. It usually assumes values in the interval $(0, 1]$.

The importance of reactivity factors is due to the fact that they give a description of the strategy of application of the rules. Moreover what is important is the ratio between the coefficients of every rule rather their single values. In fact, given the two rules

$$\begin{array}{lcl} r1: & A & \stackrel{k_1}{\rightarrow} \quad B \\ r2: & B & \stackrel{k_2}{\rightarrow} \quad A \end{array} \tag{6}$$

if we set $k_1 = 0.01$ and $k_2 = 0.1$ and we start with comparable amounts of A and B, then we observe a decrease in the concentration of B because it's produced (by rule $r1$) at a rate that's ten time slower than the rate on which it's consumed (by rule $r2$). The opposite holds for A. This simple example stresses once more the importance of the ratio between the reactivity coefficients of the rules and the great influence of the reactivity coefficients on the dynamics of a system. In Psim reactivity coefficients are implemented as fixed constants, but we think that a more realistic model should let them change in time according to some conditions of the system. For this reason we intend to implement their dynamical evolution in time in subsequent releases of the simulator.

Let's conclude with some remarks on the concept of molarity. We have said that $||X||$ denotes the concentration (or more precisely the molar concentration) of the object X. Formally we can write

$$||X|| = \frac{|X|}{\mu} \tag{7}$$

where $|X|$ is the number of objects of type X present into the system and μ is called *molarity unit*. The molarity unit is defined as a constant percentage (say 1%) of the maximum size of a membrane. Note that the size of a membrane is the maximum number of objects that the membrane can hold. More precisely

$$\mu = |M| \cdot \nu \tag{8}$$

where $|M|$ is a parameter of the system describing the maximum size of membrane M and ν is a simulation parameter fixed to 0.01 in our experiments.

According to (7), if we want to calculate the number of objects of type X present in a certain membrane at a given time we have to compute the quantity:

$$|X| = \mu||X|| \tag{9}$$

and according to this equivalence the amount of an object X at every simulation step has to be updated by means of the following assignment:

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

where all symbols have the meaning introduced previously.

2 The membrane structure in Psim

Another very important aspect of a P system is the membrane structure; in fact this feature affects not only the topology of the system but even the syntax of the set of rewriting rules.

Psim implements the *boundary notation* introduced in [1]. According to this notation every membrane can be viewed as being composed of three different regions. The first one is the inside of the membrane and from now on we refer to it as "IN"; the second region is the outside of the membrane, denoted "OUT"; it can be imagined as the space surrounding the membrane delimiting its ray of action. The membrane could catch objects present in this region with an imaginary arm and bring them in its inside, on the other hand the membrane can throw out of its inside some elements putting them in this OUT region. The last region composing a membrane is called "INTER"; it coincides with the phospholipid bilayer that physically build the membrane. This is the region of the membrane that hosts *receptors* and that can be seen as an intermediate region between membrane's inside and outside, whose objects could be viewed from (and thus could interact with) both inside and outside of the membrane.

According to the boundary notation, every rule r associated with a certain membrane labelled by the number i has the form:

$$r : \alpha\{_i\beta[_i\gamma \rightarrow \alpha'\{_i\beta'[_i\gamma' \tag{11}$$

where $\alpha, \beta, \gamma, \alpha', \beta'$ and $\gamma' \in \Sigma^*$, where Σ is the alphabet of the P system. Moreover the indexed bracket $\{_i$ indicates the separation of the region INTER from the region OUT of the membrane i. The squared bracket $[_i$ denotes the separation between IN region and $INTER$.

Using a notation more tied to the syntax of the simulator we can rewrite the rule r expressed in (11) in the following manner:

$$r : \alpha_{\text{OUT},i}\, \beta_{\text{INTER},i}\, \gamma_{\text{IN},i} \rightarrow \alpha'_{\text{OUT},i}\, \beta'_{\text{INTER},i}\, \gamma'_{\text{IN},i} \tag{12}$$

where all the symbols have the usual meaning.

Note that at least one of the strings appearing into a rule in one of the regions indicated might be empty and that the index of the membrane is the same both on the left part of the rule and on the right one. Furthermore we would like to stress the fact that no structure changing in the topology of the membrane system has been allowed until now, but we intend to add this functionality in the near future.

According to what we have seen so far, all systems handled by Psim can be considered as multisets of membranes, each of them having three (e.g. one corresponding to each of the three different regions of a membrane), possibly empty, multisets of objects.

From a formal point of view we can say that Psim is able to describe all families of P systems with $m \geq 1$ membranes, with or without priorities between rules, catalysts, and i/o; according to the terminology discussed in [6, 7] this can be formalized with the expression $P_m(Pri, Cat, i/o, n\pm, n\delta, n\tau)$ where $m > 0$.

We end this introductory Section by outlining the aim that has guided us in developing this simulator. Please note that we are not interested in membrane systems as a machine producing an output in response of a certain input encoded somehow, rather we are interested in the dynamic behaviour of the system. For this reason we have no specification of an output membrane, as in standard P systems; our interest is focused on the dynamic aspects of the evolution of a P system. Following this way of reasoning we are interested in the object concentrations in the compartments of the system as time elapses.

3 The simulator

We describe, in this Section, some technical details on how to install and execute Psim.

The simulator is entirely developed in Java; in this way it's executed by the java virtual machine and can be used under the principal operating systems. In order to execute the java bytecode it's necessary to have the *Java Runtime Environment* installed on the machine. For more information on its latest release and free downloads please refer to the Sun Microsystems web site, at the url:

`http://java.sun.com/j2se/index.jsp.`

Let's start by describing some details on how to install and execute the simulator. Psim doesn't need any particular installation; it just requires the unpacking of the distribution archive respecting the structure of the folders contained inside it.

The simulator is executed with the following simple synopsis:

```
java psim/psim paramfile.xml
```

where `paramfile.xml` is an xml file [9], expressing the topological membrane structure of the P system and the set of rules characterizing each membrane, according to the structure explained later on. Before entering into the details of the xml syntax let's have a look at the interface of the simulator.

Once prompted the execution command, the simulator will start the session by showing its main screen. An example of the main screen of the simulator is depicted in Figure 1. The figure represents the graph of object Y placed into membrane $m1$ in region IN, as expressed by the central drop down element. The simulation took 100 unitary steps, as stated by the number of cycles parameter and confirmed by the length of the graph plotted in the middle of the screen.

The main screen of Psim 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 (Max), and the number of simulation cycles (Cycle); it also contains three drop down boxes, allowing the user to select the type of output graph. To display the trend of an object the user has simply to select the object into the drop down box corresponding to the color he (or she) intends to assign to the plot. The frame in the middle displays the

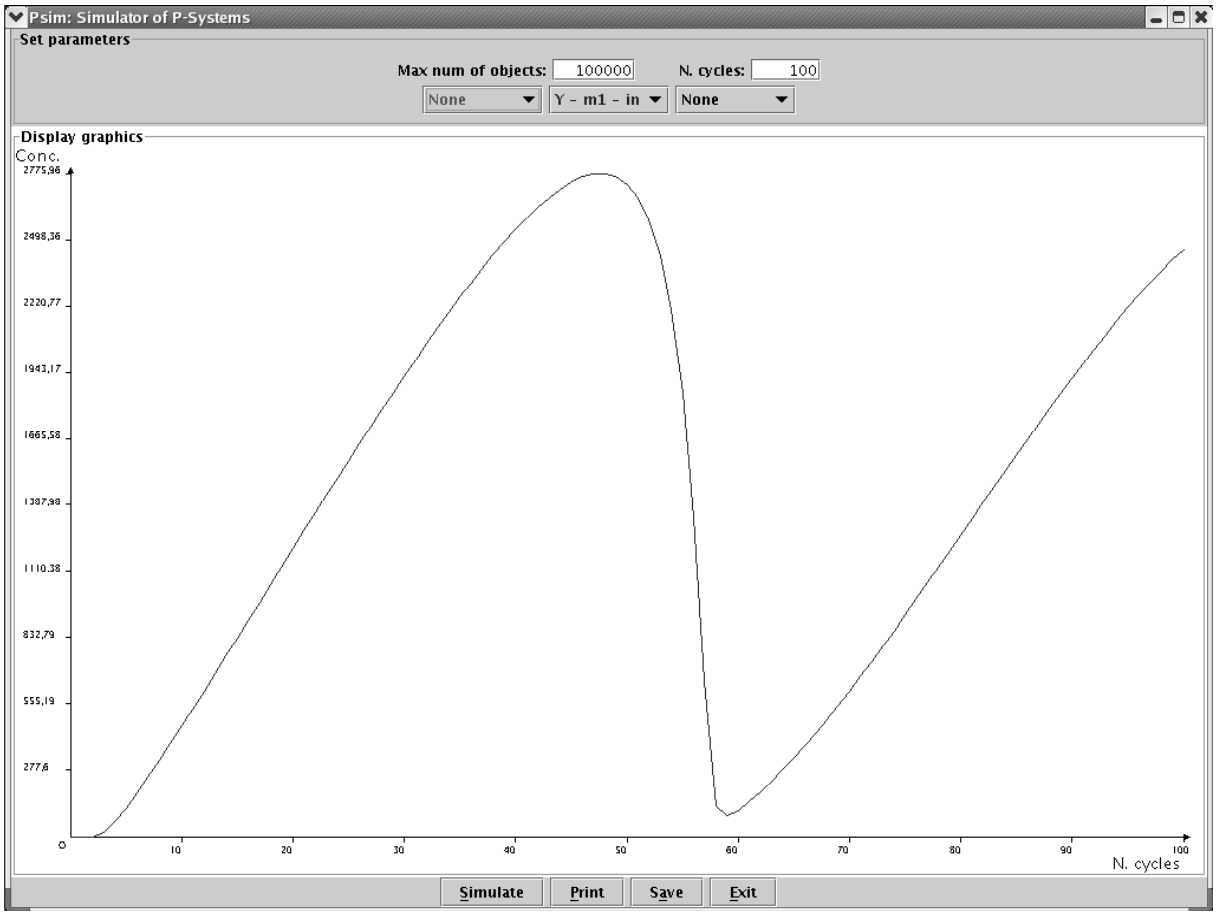

Figure 1: Screenshot of the simulator.

computation results in the form of graphs, according to the elements selected in the three drop down boxes. The frame at the bottom contains four command buttons, which tell *Psim* to start a simulation, to print results, to save the system state or to exit. The button *Simulate*, accessible through the shortkey "S", makes the simulator simulate the system specified at launch time, with parameters *Max* and *Cycle* explicited as said above. Once the simulation has been completed the user is allowed to select the output graphs in the drop down boxes. Every time Max or Cycle parameters are changed the simulator forces the user to resimulate the system with the new pair of parameters. The button *Print* is accessible through the shortkey "P"; it allows the user to send to a printer the graphs displayed in the middle part of the screen. The button *Save* has the shortkey "a"; it is useful to save the intermediate results of a simulation. When pressed it makes Psim to take a picture of the system, in terms of amount of objects, and to store it into an xml file, specifiable by the user. This file is syntactically identical to the input file, and thus could be used as input file of another session, but it represents the state of the system at the end of the last simulation performed. Finally the button *Exit*, with shortkey "E", forces Psim to return the control to the operating system.

Now we are ready to enter the details of the xml parameter file. Figure 2 contains a simple XML file that can be used as an input to the simulator. The choice of this markup language has the advantage that it ca be easily interfaced with many high level programming languages; moreover the nesting capabilities of xml make the definition of

```
<?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 reagent="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 reagent="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 reagent="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 reagent="E" r_mult="1" r_memb="m1" r_zone="in"
         product="A" p_mult="1" p_memb="m1" p_zone="out"/>
    </rule>
</simulator>
```

Figure 2: Example of 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.

the membrane structure of the P system extremely easy. By reading an input file from its top to the bottom we encounter some important elements that we would like to discuss shortly:

- the *standard header*, specifying the version of xml and the encoding of characters used. It also specifies the type of the xml document contained into the file. This is done by a reference to the Document Type Definition (DTD) `psim.dtd`, that has to be placed in the same folder of `paramfile.xml`. The file psim.dtd has the purpose to impose some syntactical constraints on the system topology and on the expression of rules in the paramfile.xml. This feature is not used yet, but we intend to take advantage of its capabilities in the near future;
- the `<simulator>` tag, which contains all other specifications. It doesn't contain any parameter now, but we intend to specify some global parameters of the system inside

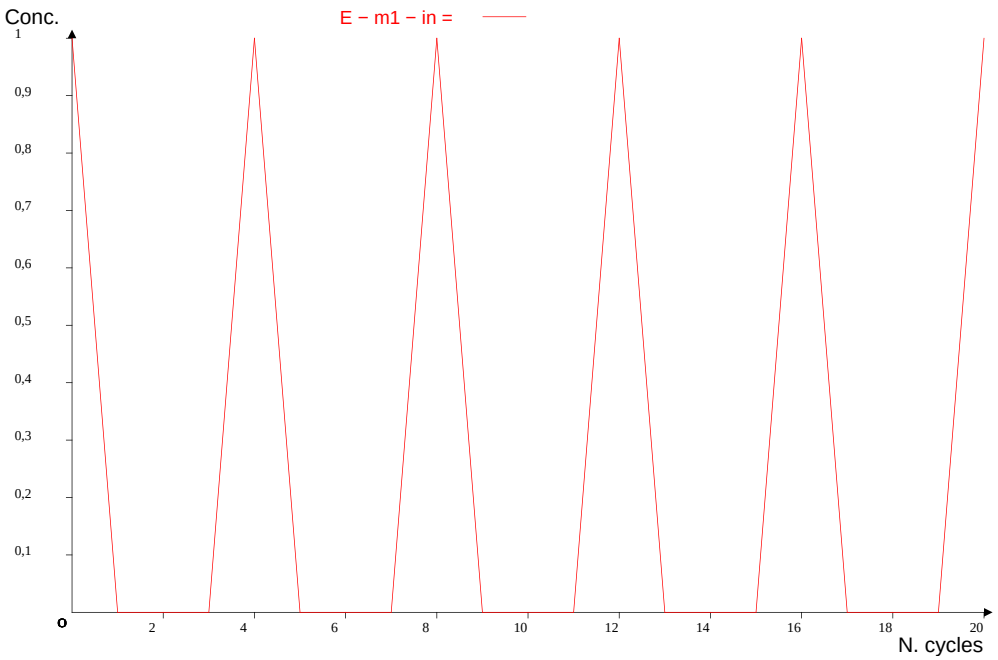

Figure 3: Simulation of the set of impulses described by the input file depicted in Figure 2. Values of *Max num. of objects.* and *N. cycles* are respectively set to 100 and 10.

it;

- the `<membrane>` tag that constitutes the specification of the topology of the membrane system. It has two parameters associated: its identifier (*id*) and multiplicity (*mult*). Every membrane could contain three nested tags: `<in>`,`<out>` and `<inter>`, each one corresponding to the respective region of a membrane (refer to Section 2 for more details on this aspect). For every membrane at least one of them must be specified but it's not necessary to specify all of them simultaneously. In the example shown in Figure 2, for instance, only the region IN is present. Note that this part describes only the starting configuration of the system. In fact the system described in the example uses all other regions of the membrane during its evolution[2], even if regions OUT and $INTER$ are initially empty and thus not reported in the starting configuration. Furthermore, note that none of these regions can be specified twice (or generally more than one single time) inside the same membrane. Every region of a membrane could contain other membranes as well as objects, provided that all of them are nested into one of the three regions discussed shortly before;

[2]In order to convince of this, please take a look at the definition of the rules where regions other than IN are considered.

- the tag `<object>`, that is used to specify objects within a membrane. It has, associated to it, its type (*name*) and multiplicity (*mult*). Please note that there's no need to specify objects that are not present into the system at start time;

- the tag `<rule>` is the entry tag for the specification of a rule. It has three parameters: the identifier (*name*), reactivity coefficient (*rho*) that actually is a constant value (for more details see Section 1) and the membrane that contains it (*membrane*). Every rule tag contains in its inside at least one `<elem>` tag, specifying the syntax of the rule;

- the `<elem>` tag consisting in two main parts: the *reagent* with the information of its position inside the membrane system (*r_memb* and *r_zone*) and the stoichiometric coefficient inside the rule (*r_mult*); and the *product* with the information of its position inside the membrane system (*p_memb* and *p_zone*) and the stoichiometric coefficient of the product inside the rule (*p_mult*). It's important to note that a rule could have more than one (but no less than one) `<elem>` tag. Furthermore we would like to stress the fact that the system assumes that every rule must not contain two `<elem>` tags sharing the same *reagent* part, but this does not limit the expressivity of the rules.

In the following we give a couple of examples of the syntax of some particular rewriting rules in order to let the reader acquire more familiarity with the definition of rules into Psim:

Example 3.1 *Suppose that we would like to model a rule named* r*, having the form* $\lambda \rightarrow A$*, whose reactivity coefficient equals* 0.1*, and placed inside membrane* $m1$ *into region* IN*. In Psim we describe such a rule by means of the following syntax:*

```
<rule name="r" rho="0.1"  membrane="m1">
    <elem reagent="A" r_mult="0" r_memb="m1" r_zone="in"
     product="A" p_mult="1" p_memb="m1" p_zone="in"/>
</rule>
```

Example 3.2 *If we would like to model a rule named* r*, having the form* $A \rightarrow \lambda$*, with reactivity coefficient equal to* 0.1*, placed inside membrane* $m1$ *into region* IN*, in Psim we have to use the following syntax:*

```
<rule name="r" rho="0.1"  membrane="m1">
    <elem reagent="A" r_mult="1" r_memb="m1" r_zone="in"
     product="A" p_mult="0" p_memb="m1" p_zone="in"/>
</rule>
```

that is equivalent to:

```
<rule name="r" rho="0.1"  membrane="m1">
    <elem reagent="A" r_mult="1" r_memb="m1" r_zone="in"
     product="" p_mult="0" p_memb="m1" p_zone="in"/>
</rule>
```

As we can argue from previous examples, the empty word λ, could be modelled in Psim either as an an empty object (say "") or as a certain object X having multiplicity equal to zero (p_mult= 0 or r_mult =0). Having this clear in mind the following examples would not result surprising:

Example 3.3 *Suppose now that our aim is to model a rule named r with reactivity coefficient 0.1, placed inside membrane $m1$, region IN, having the form $AB \rightarrow AA$; in Psim we have to use the following syntax:*

```
<rule name="r" rho="0.1"  membrane="m1">
    <elem reagent="A" r_mult="1" r_memb="m1" r_zone="in"
     product="A" p_mult="2" p_memb="m1" p_zone="in"/>
    <elem reagent="B" r_mult="1" r_memb="m1" r_zone="in"
     product="B" p_mult="0" p_memb="m1" p_zone="in"/>
</rule>
```

Example 3.4 *Let's now try modelling a rule named r with reactivity coefficient 0.1, placed inside membrane $m1$, region IN, having the form $AA \rightarrow AC$; in Psim we have to use the following syntax:*

```
<rule name="r" rho="0.1"  membrane="m1">
    <elem reagent="A" r_mult="2" r_memb="m1" r_zone="in"
     product="A" p_mult="1" p_memb="m1" p_zone="in"/>
    <elem reagent="C" r_mult="0" r_memb="m1" r_zone="in"
     product="C" p_mult="1" p_memb="m1" p_zone="in"/>
</rule>
```

Note that other rules could be modelled according to what said so far; the important thing to remember is that no rule can contain more than one `<elem>` tag with the same *reagent* parameter, but this does not limit the expressivity of the system.

4 Some application examples

In this Section we'll discuss some examples on how to use the simulator in order to get the dynamics of some well known systems.

Let's start by modelling one of the formulations of the BZ reaction, called *Brusselator* [3, 8], in the formulation of Nicolis and Prigogine [5]. The BZ reaction is a biochemical reaction whose resulting composite oscillates in color repeatedly from red to blue, due to the concentration of two kind of elements (for us, X and Y). This simplified model of the BZ reaction can be translated into the following rewriting rules:

$$\begin{array}{lrcl} r1: & \lambda & \rightarrow & X \\ r2: & X & \rightarrow & Y \\ r3: & XXY & \rightarrow & XXX \\ r4: & X & \rightarrow & \lambda \end{array} \tag{13}$$

that can be easily translated into rules suitable for Psim. Let's try simulate its behaviour. The example file `brus.xml` (contained into the folder "/psim/samples") is included into the distribution of the simulator. It implements the translation of the rewriting rules charachterizing the brusselator dynamics. Simulate it by typing at the command prompt:

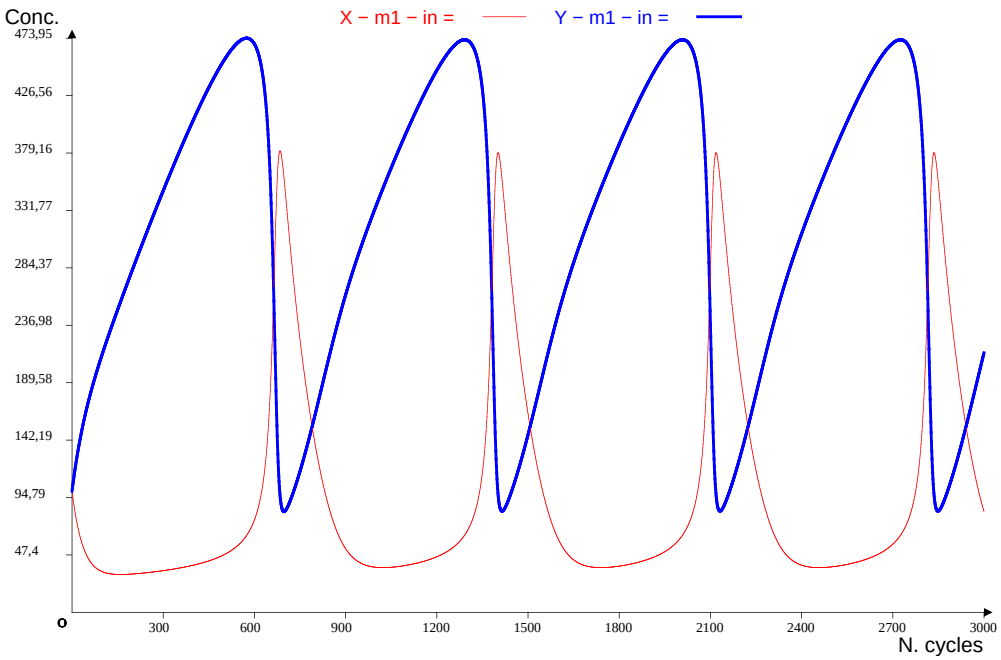

Figure 4: Oscillations of the simplified brusselator model simulated by *Psim* with reactivities scaled by a factor 100: $k_1 = 0.001$, $k_2 = 0.03$, $k_3 = 1$, $k_4 = 0.01$, and $\mu = 1000$ ($|M| = 100000$). The initial cardinality of X and Y is set to 100.

```
java psim/psim psim/samples/brus.xml
```

and wait until the simulator answers with the main screen. After this let's type the amount 100 000 in the "*Max num. of objects*" text box and 3000 in the "*N. cycles*" one. Then we click with the mouse button on the *Simulate* button, then after a few seconds of calculation the system is ready to provide the results of the computation. In order to display them we simply select "*X - m1 - in*" in the red combo box, while "*Y - m1 - in*" in the blue one. The system answers by drawing a graph like the one depicted in Figure 4. The reactivity coefficients of `brus.xml` input file are set as shown in the caption of Figure 4; note that for those coefficients the brusselator is known to start a stable oscillation in concentration of objects X and Y. Please remember to place the DTD file (psim.dtd) in the same folder of the input file in order to simulate it without errors.

The second example we want to investigate is the case of an infective disease spreading through a population [4]. This plague causes infected people's death or permanent immunity to the infection.

We study the system under the simplifying assumption that the population is closed (e.g it's made by a certain amount of people and where no births, immigration or emi-

gration are allowed). The population of this dynamical system is partitioned into three different categories (objects of our system): healthy people C, ill people G and immune people K. When an healthy person meets an ill one he becomes ill, with a probability depending on the reaction rate of the rule; an ill person has three possibilities: he may die, he otherwise could become immune forever to the infection or could indefinitely keep the state of illness. On the other hand an healthy individual could keep his state until he gets in contact with an ill one.

The behaviour just described can be expressed with the following set of rules:

$$\begin{array}{lcll} r_1 & : & CG & \rightarrow & GG \\ r_2 & : & G & \rightarrow & K \\ r_3 & : & G & \rightarrow & \lambda \end{array} \tag{14}$$

in which all the symbols have the meaning previously discussed.

According to literature the system described presents the existence of a threshold of activation for the epidemic: on the one hand, if the initial healthy population is below a certain amount the epidemic does not start and so ill people decrease in number until its complete vanishing. On the other hand, whenever the initial healthy population is beyond that threshold the epidemic activates and the number of ill people grows up until reaching its maximum and then drops back to zero, thus vanishing.

We have included the specification file of the epidemic model into the distribution of Psim. Try simulating it with the command:

```
java psim/psim psim/samples/epid.xml
```

as previously described. Let's now type in the *Max num. of objects*" text box the value 350 000 and 30 in the "*N. cycles*" space. Now we are ready to start the simulation; shortly after clicking on the Simulate button we can select the items we intend to plot as the result of the computation. If we choose the object C in the red combo box and G in the blue one we obtain the active epidemic model depicted in Figure 4. The value of the threshold of activation of the epidemic, according to the parameters chosen, is about 2570. In order to verify this different behaviour let's open the parameter file "/samples/epid.xml" with a text editor and change the initial amount of object C placed inside membrane $m1$ in the region IN, from the amount 7000 to the value 2000. If we now simulate the system again, as explained earlier, we can observe the behaviour of the not active epidemic case, illustrated in Figure 6. The number of ill people decreases to zero and the epidemic does not make its course.

We think that another useful exercise, in order to acquire more familiarity with the simulator, could be the implementation of the system described in Figure 2 with the purpose to reproduce the behaviour described in Figure 3, which also contains all parameters needed by the simulation.

5 Further work

The simulator described in those pages is a preliminary version of a set of tools that we intend to develop in order to study some concrete dynamical systems obtained from cellular metabolic pathways and signal transduction pathways. We know that our simulator Psim can be improved in several ways and for this reason we will appreciate all suggestions and comments that the reader would like to send us.

We report some guidelines that will lead us in the future developments of Psim:

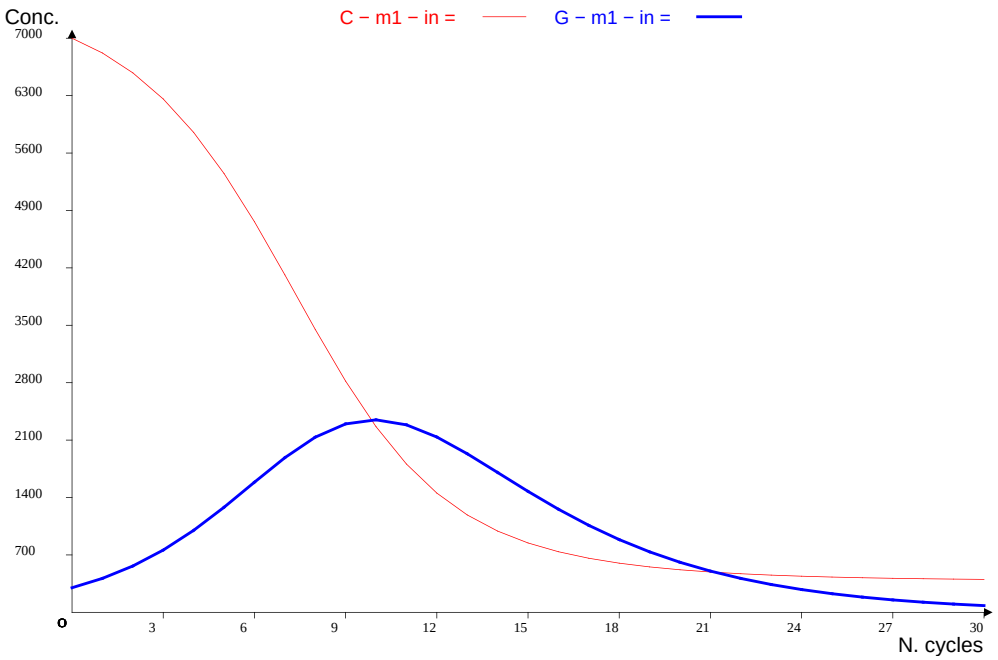

Figure 5: Active epidemic model simulated by *Psim* with $k_1 = 0.3$, $k_2 = 0.1$, $k_3 = 0.12$ and $\mu = 3500$.

- time-varying reactivity coefficients;
- introduction of rules having the ability to change the topological structure of the membrane system;
- plot of more complicated, and possibly more interesting, relations between the concentration of objects present into the system;
- chance to associate a different maximum dimension for every membrane;
- introduction of new types of elements like promoters or inhibitors;
- introduction of biological significative constraints on the input file;
- introduction of new mechanisms (for example binary guards) in order to make possible the expression of systems provided with signal transduction components.

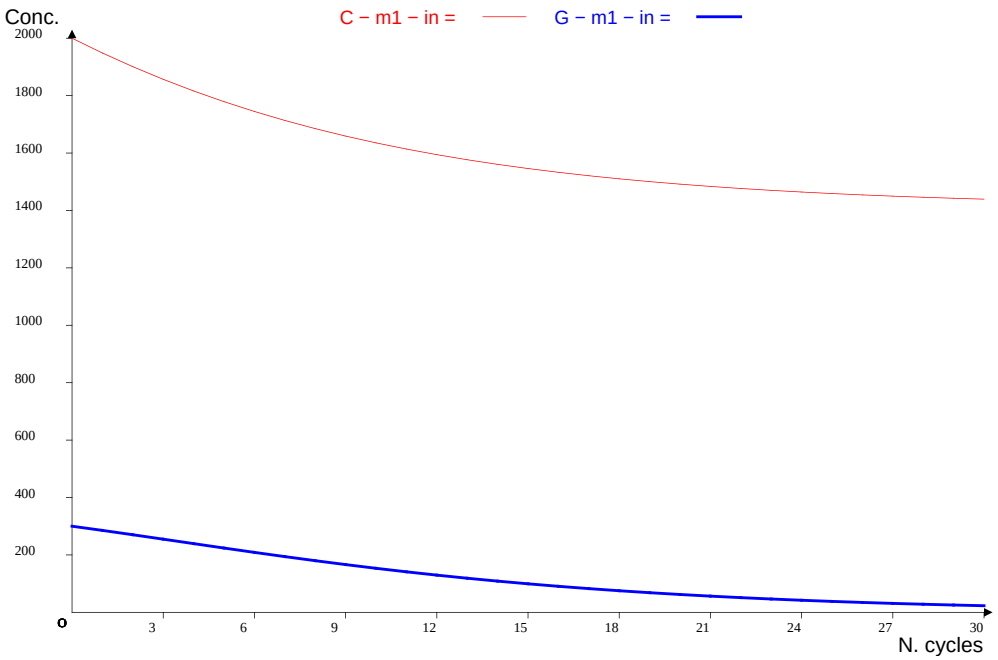

Figure 6: Not active epidemic model simulated by *Psim* with $k_1 = 0.3$, $k_2 = 0.1$, $k_3 = 0.12$ and $\mu = 3500$.

References

[1] F. Bernardini and V. Manca. Dynamical aspects of P systems. *BioSystems*, 70:85–93, 2002.

[2] L. Bianco, F. Fontana, G. Franco, and V. Manca. P systems for Bio Systems. In G. Păun, Application of P systems, to appear.

[3] C. R. Gray An Analysis of the Belousov-Zhabotinskii Reaction. Calhoun High School, Port Lavaca, TX. Mathjournal,2002. Accessible at the web site: http://www.rose-hulman.edu/mathjournal/2002/vol3-n1/paper1/v3n1-1pd.pdf

[4] D. S. Jones and B. D. Sleeman. *Differential equations and mathematical biology*. Chapman & Hall/CRC, London, 2003.

[5] G. Nicolis and I. Prigogine. *Exploring Complexity, An Introduction*. Freeman and Company, San Francisco, 1989.

[6] G. Păun. Computing with membranes. *J. Comput. System Sci.*, 61(1):108–143, 2000.

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

[8] Y. Suzuki, H. Tanaka Abstract Rewriting Systems on Multisets and Their Application for Modelling Complex Behaviours. Brainstorming Week on Membrane Computing, Tarragona, February 5-11 2003, p. 313–331. Matteo Cavaliere, Carlos Martín-Vide, Gheorghe Paun.

[9] W3C. *Extensible Markup Language (XML) 1.0 (Third Edition). Recommendation 04 February 2004*. Present at the url: http://www.w3.org/TR/REC-xml/