On P Systems as a Modelling Tool for Biological Systems

Francesco Bernardini[1], Marian Gheorghe[1], Natalio Krasnogor[2], Ravie C. Muniyandi[1], Mario J. Pérez-Jímenez[3], and Francisco José Romero-Campero[3]

[1] Department of Computer Science, The University of Sheffield, Regent Court, Portobello Street, Sheffield, S1 4DP, UK
{F.Bernardini, M.Gheorghe, R.Muniyandi}@dcs.shef.ac.uk

[2] Automated Scheduling, Optimisation and Planning Research Group, School of Computer Science and Information Technology, University of Nottingham, Jubilee Campus, Nottingham NG8 1BB, UK
Natalio.Krasnogor@nottingham.ac.uk

[3] Research Group on Natural Computing, Department of Computer Science and Artificial Intelligence, University of Seville, Avda. Reina Mercedes 41012, Sevilla, Spain
{marper, fran}@us.es

Abstract. We introduce a variant of P systems where rules have associated a real number providing a measure for the "intrinsic reactivity"of the rule and roughly corresponding to the kinetic coefficient which, in bio-chemistry, is usually associated to each molecular reaction. The behaviour of these P systems is then defined according to a strategy which, in each step, randomly selects the next rule to be applied depending upon a certain distribution of probabilities. As an application, we present a P system model of the quorum sensing regulatory networks of the bacterium *Vibrio Fischeri*. In this respect, a formalisation of the network in terms of P systems is provided and some simulation results concerning the behaviour of a colony of such bacteria are reported. We also briefly describe the implementation techniques adopted by pointing out the generality of our approach which appears to be fairly independent from the particular choice of P system variant and the language used to implement it.

1 Introduction

Membrane computing represents a new and rapidly growing research area which is part of the natural computing paradigm and which was initiated by Gheorghe Păun in 1998 with a seminal paper initially circulated on the web and later published in [13]. Already a monograph has been dedicated to this subject [14] and some fairly recent results can be found in [15],[9],[10]. Membrane computing aims at defining computational models which abstract from the functioning and structure of the cell. Specifically, membrane computing starts from the observation that membranes play a fundamental role in the functioning of a living cell. Membranes are essentially involved in many reactions taking place inside various compartments of a cell, and they act as selective channels of communication between different compartments as well as between the cell and its environment [1].

Membrane computing formalises these essential features of living cells by introducing the notion of *membrane systems*, which are usually called *P systems*. P systems are characterised by four fundamental features: a *membrane structure* where *objects* evolve according to some *evolution rules*, which also determine the *communication* of objects between membranes. Specifically, the *membrane structure* consists in a number of membranes arranged in a hierarchical structure, all of them but one included in an unique main membrane called *skin membrane*. This most external membrane defines the boundary between the inside of the system and its outside, which is called *environment*. A membrane without any membrane inside is called *elementary*. Each membrane identifies a corresponding *region* inside the system: the space between the membrane and the membranes (if any) directly contained in it. A graphical representation of such a membrane structure is reported in Figure 1.

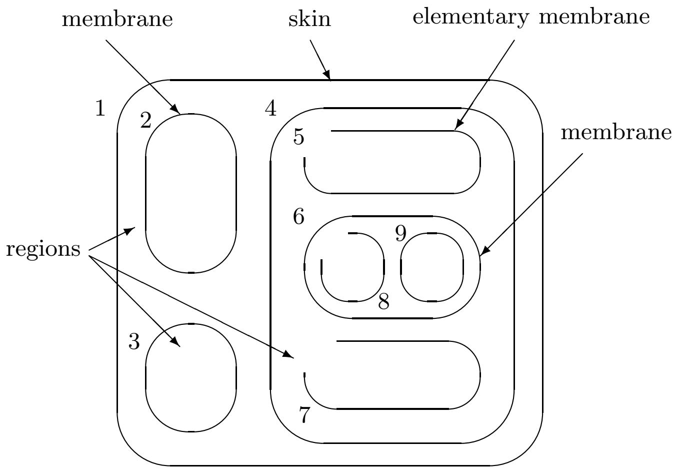

Fig. 1. A membrane structure containing 9 membranes and 9 corresponding regions; labels are used to uniquely identify each distinct membrane in the system.

Some *objects* are then assigned to the regions, each object appearing with a specific multiplicity. That is, each region, in general, contains a *multiset of objects* rather than a set. As well as this, finite sets of *evolution rules* are assigned to the region, one per each region, which are used to modify the objects associated with the regions and to move them across the membrane from one region to the other. Rules have a *local scope*: the rules assigned to a specific region inside the system can be applied only to the objects associated with that same region. P systems were originally introduced to investigate the computational nature of various features of biological membranes [13], with an approach typical to formal language theory and theory of computing, rather than to provide a comprehensive model of the living cell. Nevertheless, some recent researche trends [4], [5], [16] have been actually dedicated to the study of P systems as a modelling tool where P systems are used as a formalism for describing, and possibly simulating, the behaviour of biological systems. Therefore, there is a growing interest in developing implementations for the membrane computing paradigm in order to be able to execute P system models and run simulations of biological phenomena of various interest. In this respect, a number of tools have already been produced (some of them are available from `http://psystems.disco.unimib.it/`, the P systems web pages) but yet correct implementation techniques need to be identified, especially when the quantitative aspects featuring the "reality" of a biological phenomenon are considered in the model.

In this paper, we present a variant of P systems (Section 2) where rules are generalised boundary rules which allow us to express transformations affecting simultaneously the objects placed on both sides of a membrane, that is, both the objects placed inside that compartment and the objects placed into the surrounding region. As well as this, each rule has associated a real number providing a measure for the "intrinsic reactivity"of the rule and roughly corresponding to the kinetic coefficient which, in bio-chemistry, is usually associated to each molecular reaction [16]. Moreover, in this variant, rules are applied according to a strategy which, in each step, randomly selects the next rule to be applied depending upon a certain distribution of probabilities. The main difference with respect to the usual approach adopted in membrane computing is that, in our approach, there is no parallelism in the application of the rules as the system evolves only by means of a rule at a time. Next, in Section 3, we present, as a case-study, a P system model for the quorum sensing system of the marine bacterium *Vibrio fischeri* together with some simulation results obtained by implementing the model in Scilab (a free software package available at `http://scilabsoft.inria.fr/`). The novelty of our approach consists in the fact that we do not only provide a description for the reactions involved in the quorum sensing regulatory network but we are also able to provide a model for an arbitrarily large colony of bacteria. In this respect, we can say our simulations provides a snapshot of the behaviour of the colony as a whole complex system. Finally, the last section describes implementation techniques for our P system model by presenting the data structures and the code that are necessary to support its execution. Moreover, by following [12], we advocate the use of the mark-up language SBML as a "machine-interpretable" language for defining executable specifications of P systems and the corresponding code that can be automatically generated from. In this respect, we want to stress the generality of our approach which appears to be fairly independent of the particular choice of a P system variant, the language used to implement it, and flexible enough with respect to the strategy of applying the rules of the system.

2 Definitions

We start by recalling from [14] some basic notions of formal language theory which are commonly used in the area of membrane computing. An *alphabet* is a finite non-empty set of abstract symbols. Given an alphabet O, we denote by O^* the *set of all possible strings over* O, including the empty string λ. The *length of a string* $x \in O^*$ is denoted by $|x|$ and, for each $a \in O$, $|x|_a$ denotes the number of occurrences of the symbol a in x. A *multiset* over O is a mapping $M : O \longrightarrow N$ such that, $M(a)$ defines the multiplicity of a in the multiset M (N denotes the set of natural numbers). Such a multiset can be represented by a string $a_1^{M(a_1)}\, a_2^{M(a_2)} \dots a_n^{M(a_n)} \in O^*$ and by all its permutations with $a_j \in O$, $M(a_j) \neq 0$, $1 \leq j \leq n$. In other words, we can say that each string $x \in O^*$ identifies a finite multiset over O defined by $M_x = \{\, (a, |x|_a) \mid a \in O\,\}$. Moreover, given two strings $x, y \in O^*$, we denote by xy their catenation, which corresponds to the union of the multiset represented by string x and the multiset represented by string y.

Membrane structures are represented as usual by means of strings of matching pairs of square-brackets, with each pair of square-brackets representing a membrane and each one of them being labelled with a different value in $\{1, 2, \dots, n\}$, for n the number of membranes in the structure. For example, the membrane structure of 1 can be represented by using the following string of matching square brackets:

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

where:

- membrane 1 is the *skin membrane*,
- membrane 2, membrane 3, membrane 5, membrane 7, membrane 8 and membrane 9 are *elementary membranes*, Moreover,
- membrane 1 *directly contains* membrane 2, membrane 3 and membrane 4,
- membrane 4 *directly contains* membrane 5, membrane 6 and membrane 7,
- membrane 6 *directly contains* membrane 8 and membrane 9.

We refer to [14] for further details about this representation.

A P system is then defined in the following way.

Definition 1. *A P system is a construct*

$$\Pi = (O, L, \mu, C_1, C_2, \dots, C_n, R)$$

where:

- *O is a finite alphabet of symbols representing objects;*
- *L is a finite alphabet of symbols representing labels for the compartments;*
- *μ is a membrane structure consisting of $n \geq 1$ membranes;*
- *$C_i = (l_i, w_i)$, for each $1 \leq i \leq n$, is the initial configuration of the compartment i with $l_i \in L$ and $w_i \in O^*$ a finite multiset of objects;*

- *R is a finite set containing $m \geq 1$ rules that are labelled in one-to-one manner with values in $\{1, 2, \ldots, m\}$ and that are of the form*

$$j : u\ [\ v\ ]_l \xrightarrow{k_j} u'[\ v'\ ]_l$$

with $1 \leq j \leq m$, $u, v, u', v' \in V^*$ some finite multisets of objects, $l \in L$ a label for the compartment, and k_i a real number.

Thus, a P system is characterised by a finite alphabet O for the objects placed into the compartments, a finite alphabet L for labelling the compartments, a membrane structure μ, an initial configuration for each compartment in the system, and a finite set R containing rules describing transformations that can be applied to the objects placed inside the compartments. Specifically, the initial configuration of a compartment consists of a label from the alphabet L and a finite multiset of objects from O represented as a string in O^*; these objects are those which are initially placed inside that compartment. Compartments can interact each other by means of the rules in R which are of the form $u\ [\ v\ ]_l \xrightarrow{k_j} u'[\ v'\ ]_l$. Such a rule specifies that a multiset u, which is supposed to be contained in the outside part of a compartment labelled by l, and a multiset v, which is supposed to be contained inside a compartment labelled by l, can be simultaneously replaced by the multisets u', v' in the respective places. Moreover, each rule in R has associated a real constant which is meant to provide a measure of the "reactivity" of the rule in a similar way to what was done in [5], [16]. In other words, in our P systems, multisets of objects are used to model bags or soups of chemicals whereas rules are used to model generic biochemical processes which affect the number and distribution of these objects within the system. All these rules are supposed to consume certain chemicals in order to produce some new ones.

Then, in order to make the system transit from one configuration to the other, a strategy for the application of the rules is adopted that makes the system evolve only by means of a rule at a time. Moreover, in each step, only one rule to be applied inside a specific cell is randomly selected according to a given distribution of probabilities. To this aim, we developed an adaption of Gillespie's algorithm in order to associate a stochastic behaviour to population P systems. Gillespie's algorithm [8] (see also [7] for some recent improvements) provides an exact method for the stochastic simulation of systems of bio-chemical reactions; the validity of the method is rigorously proved and it has been already successfully used to simulate various biochemical processes [11]. As well as this, Gillespie's algorithm is used in the implementation of stochastic π-calculus [17] and in its application to the modelling of biological systems [18] (an implementation of the stochastic pi-machine is avaliable at `http://www.doc.ic.ac.uk/~anp/spim/`). Here, with respect to the original algorithm, we have to take into account the fact that in P systems we have different cells, each one with its own set of rules, and the fact that the application of a rule inside a cell can affect the content of environment too.

Specifically, let $\Pi = (O, L, \mu, (w_1, l_1), \ldots, (w_n, l_n), R)$ be a P system as specified in Definition 1. At any moment, a configuration of the system Π can be represented as a tuple

$$\Gamma = ((x_1, l_1), \ldots, (x_n, l_n), \mu)$$

where, for each $1 \leq i \leq n$, x_i is the multiset of objects currently contained in compartment i. Thus, given such a configuration, for each $1 \leq i \leq n$, we define the set $R(i)$ of pairs (j, p_j) such that:

- j is the index of a rule in R of the form $u\,[\,v\,]_{l_i} \xrightarrow{k_j} u'[\,v'\,]_{l_i}$, with $x_f = y\,u$ and $x_i = z\,v$, for f the index of the compartment that directly contains i and some $y, z \in O^*$ (i.e., the rule j is applicable inside compartment i because it is labelled by l_i, the surrounding region contains the multiset u, and compartment i contains the multiset v);
- p_j is the probability of the rule j to be applied in the next step of evolution; this probability is computed by multiplying the constant k_j by the number of possible combinations of the objects present on the left-side of the rules with respect to the multisets x_i and x_f (for example, if we have a rule $[\,ab\,]_{l_i} \rightarrow [\,w\,]_{l_i}$, with $a, b \in O$, $w \in O^*$, the probability p_j is given by $k_j * |x_i|_a * |x_i|_b$ (i.e., there are $|x_i|_a * |x_i|_b$ different possible ways of assigning objects to the rule $[\,ab\,]_{l_i} \rightarrow [\,w\,]_{l_i}$);

Then, given these probabilities, the strategy for the application of the rules is defined according to the following procedure.

First, for each compartment i, we compute the index of the next rule to be used inside cell i and its waiting time by using the classical Gillespie's algorithm:

1. construct the sets $R(i)$ containing pairs (j, p_j) where p_j is the probability associated to rule j currently applicable inside compartment i; let us denote by M_i, $M_i \geq 1$, the number of elements of $R(i)$; the pairs in $R(i)$ are supposed to be associated in an one-to-one manner with values in $\{1, \ldots, M_i\}$, i.e. $k : (j_k, p_{j_k})$, for $1 \leq k \leq M_i$;
2. calculate $a_0 = \sum p_j$, for all $(j, p_j) \in R(i)$;
3. generate two random numbers r_1 and r_2 uniformly distributed over the unit interval $(0, 1)$;
4. calculate the waiting time for the next reaction as $\tau_i = \dfrac{1}{a_0}\, ln(\dfrac{1}{r_1})$
5. take the index h, $1 \leq h \leq M_i$, such that $\displaystyle\sum_{k=1}^{h-1} p_k < r_2 a_0 \leq \sum_{k=1}^{h} p_k$, with $k : (j_k, p_{j_k}) \in R(i)$, and $p_k = p_{j_k}$, for all $1 \leq k \leq h$;
6. return the triple (τ_i, j_h, i), if $h : (j_h, p_{j_h}) \in R(i)$.

Notice that the larger the stochastic constant of a rule and the number of occurrences of the objects placed on the left-side of the rule inside a membrane are, the greater is the chance that a given rule will be applied in the next step of the simulation. There is no constant time-step in the simulation. The time-step is determined in every iteration and it takes different values depending on the configuration of the system.

Next, a step of application of the rules is simulated by using the following procedure:

- **Initialisation**
  - set time of the simulation $t = 0$;
  - for each compartment i in μ compute a triple (τ_i, j, i) by using the procedure described above;
  - sort the list according to each waiting time;
- **Iteration**
  1. extract the first triple, (τ_m, j, m) from the list;
  2. set time of the simulation $t = t + \tau_m$;
  3. update the waiting time for the rest of the triples in the list by subtracting τ_m;
  4. apply the rule j only once changing the number of objects in the compartment and in the surrounding region
  5. if the surrounding region has been affected by the application of the rule then remove the corresponding triple from the list;
  6. re-run the Gillespie algorithm for the compartment m and for the compartment associated with the surrounding region in order to obtain the new corresponding triples; add these new triples to the list;
  7. sort this list according to each waiting time and iterate the process.
- **Termination**
  1. Terminate simulation when time of the simulation t reaches or exceeds a preset maximal time of simulation.

Specifically, in each step, the compartment selected is the cell with the minimal waiting time.

3 Modelling Quorum Sensing in *Vibrio Fischeri*

Bacteria are generally considered to be independent organisms. However it has been observed that certain bacteria, like the marine bacterium *Vibrio Fischeri*, exhibit coordinated behaviour which allows an entire population of bacteria to regulate the expression of certain or specific genes in a coordinated way depending on the size of the population. This cell density dependent gene regulation system is referred to as *quorum sensing* [6], [19], QS for short. In this respect, a comprehensive literature about QS can be found at `http://www.nottingham.ac.uk/quorum/` – a web page maintained by the Nottingham Quorum Sensing Group.

This phenomenon was first investigated in the marine bacterium *Vibrio Fischeri.* This bacterium exists naturally either in a free-living planktonic state or as a symbiont of certain luminescent squid. The bacteria colonise specialised light organs in the squid, which cause it to luminesce. Luminescence in the squid is thought to be involved in the attraction of prey, camouflage and communication between different individuals. The source of the luminescence is the bacteria themselves. The bacteria only luminesce when colonising the light organs and do not emit light in the free-living state. The QS process in *Vibrio Fischeri* relies on the synthesis, accumulation and subsequent sensing of a signal molecule, 3-oxo-C6-HSL, an N-acyl homoserine lactone or AHL, we will call it OHHL. When only a small number of bacteria are present these proteins are produced at a low level. OHHL diffuses out of the bacterial cells and into the surrounding environment. At high cell density the signal accumulates in the area surrounding the bacteria and can also diffuse to the inside of the bacterial cells. The signal is able to interact with the LuxR protein to form the complex LuxR-OHHL. This complex binds to a region of DNA called the Lux Box causing the transcription of the luminescence genes, a small cluster of 5 genes, luxCDABE. As well as the transcription of LuxR and OHHL, which are therefore called autoinducers as they activate their own synthesis. In this way, bacteria can effectively communicate each other by responding to changes in the concentration of signal molecules inside and in the surrounding environment.

Next, a model for quorum sensing in *Vibrio fischeri* is obtained by considering a P system consisting of a number of distinct compartments placed inside an unique main membrane, which represents the environment, and where each one of these compartments represents a bacterium and contains rules describing the reactions involved in the regulation of the luminescence genes. Compartments representing bacteria interact each other by sending objects into the environment and receiving some others from it. Specifically, given a population of $m \geq 1$ bacteria, we define the P system $\Pi(m)$ such that

$$\Pi(m) = (O, \{e, b\}, \mu, C_1, C_2, \ldots, C_m, C_{m+1}, R)$$

and where:

- $O = \{OHHL, LuxR, LuxR\text{-}OHHL, LuxBox\} \cup$ $\cup \{LuxBox\text{-}LuxR\text{-}OHHL\}$,
- $\mu = [\, [\,]_1 \, [\,]_2 \cdots [\,]_m \,]_{m+1}$,
- $C_i = (b, LuxBox)$, for each $1 \leq i \leq m$,
- $C_{m+1} = (e, \lambda)$,
- $R = R_b \cup R_e$ with R_b the set of rules to be used inside compartments labelled by b and R_e the set of rules to be used inside the compartment labelled by e. Each compartment labelled by b represents a bacterium whereas the unique compartment labelled by e represents the environment.

Notice that the P system $\Pi(m)$ is a parametric one as its definition depends on the value m, the number of bacteria in the colony.

The set R_b contains the following rules:

An unstressed bacterium produces the signal OHHL and the protein LuxR at basal rates - very low rates:

$$\begin{aligned} 1 &: [\, LuxBox \,]_b \xrightarrow{k_1} [\, LuxBox,\ OHHL \,]_b, \\ 2 &: [\, LuxBox \,]_b \xrightarrow{k_2} [\, LuxBox,\ LuxR \,]_b. \end{aligned}$$

The protein LuxR acts as a receptor and OHHL as its ligand. Both together form the complex LuxR-OHHL which in turn can dissociate into OHHL and LuxR again:

$$3 : [\, LuxR,\ OHHL\,]_b \xrightarrow{k_3} [\, LuxR\text{-}OHHL\,]_b,$$
$$4 : [\, LuxR\text{-}OHHL\,]_b \xrightarrow{k_4} [\, LuxR,\ OHHL\,]_b.$$

The complex LuxR-OHHL acts as a transcription factor or as a promoter binding to a region of the bacterium DNA called LuxBox and starting the transcription of different proteins involved in the production of light. The complex LuxR-OHHL can also dissociate from the LuxBox:

$$5 : [\, LuxBox,\ LuxR\text{-}OHHL\,]_b \xrightarrow{k_5} [\, LuxBox\text{-}LuxR\text{-}OHHL\,]_b,$$
$$6 : [\, LuxBox\text{-}LuxR\text{-}OHHL\,]_b \xrightarrow{k_6} [\, LuxBox,\ LuxR\text{-}OHHL\,]_b.$$

The binding of the complex LuxR-OHHL to the LuxBox produces a massive increase of the production of the signal OHHL and of the protein LuxR. In this sense OHHL and LuxR are autoinducers:

$$7 : [\, LuxBox\text{-}LuxR\text{-}OHHL\,]_b \xrightarrow{k_7} [\, LuxBox\text{-}LuxR\text{-}OHHL,\ OHHL\,]_b,$$
$$8 : [\, LuxBox\text{-}LuxR\text{-}OHHL\,]_b \xrightarrow{k_8} [\, LuxBox\text{-}LuxR\text{-}OHHL,\ LuxR\,]_b.$$

OHHL is a small molecule that diffuses outside the bacterium and so it can accumulate in the environment:

$$9 : [\, OHHL\,]_b \xrightarrow{k_9} OHHL\ [\,]_b.$$

Due to the presence of proteases and other chemical substances OHHL, LuxR and the complex LuxR-OHHL undergo a process of degradation in the bacterium:

$$10 : [\, OHHL\,]_b \xrightarrow{k_{10}} [\,]_b,$$
$$11 : [\, LuxR\,]_b \xrightarrow{k_{11}} [\,]_b,$$
$$12 : [\, LuxR\text{-}OHHL\,]_b \xrightarrow{k_{12}} [\,]_b.$$

The set R_e contains the following rules:

When the signal OHHL accumulates in the environment it can diffuse inside the bacteria. OHHL also undergoes a process of degradation in the environment

$$13 : OHHL\ [\ ]_b \xrightarrow{k_{13}} [\, OHHL\,]_b,$$
$$14 : [\, OHHL\,]_e \xrightarrow{k_{14}} [\ ]_e.$$

4 Simulation Results and Discussion

In order to implement our model in the aforementioned simulator, we have chosen the following set of kinetic constants [5], $k_1 = 2, k_2 = 2, k_3 = 9, k_4 = 1, k_5 = 10, k_6 = 2, k_7 = 250, k_8 = 200, k_9 = 1, k_{10} = 50, k_{11} = 30, k_{12} = 15,$

$k_{13} = 20, k_{14} = 20$. These values have been set such that the degradation rates $(k_{11}, k_{12}, k_{13}, k_{14})$ compensate the basal production of the signal and the protein (k_1, k_2) and such that the production rates when the regulatory region is occupied (k_7, k_8) produce a massive increase in the transcription of the signal and the protein.

We have studied the behaviour of the system for populations of different sizes to examine how bacteria can sense the number of bacteria in the population and produce light only when the number of individuals is big enough. First we have considered a population of 300 bacteria. Next we show in Figure 2 the evolution over time of the number of quorated bacteria and the number of signal (OHHL) in the environment. We say a bacterium is quorated if and only if, the LuxBox is occupied by the complex LuxR-OHHL.

It may be observed that the signal, OHHL, accumulates in the environment until saturation and then, when this threshold is reached, bacteria are able to detect that the size of the population is big enough. At the beginning, a few bacteria get quorated and then they accelerate a process of recruitment that makes the whole population behave in a coordinated way. There exists a correlation between the number of signals in the environment and the number of quorated bacteria such that, when the number of signals in the environment drops, so does the number of quorated bacteria and when the signal goes up it produces a recruitment of more bacteria.

Now we show in Figure 3 the evolution over time of the average bacterium across the population of the number of signal (OHHL), protein (LuxR) and the complex (LuxR-OHHL).

Note that on average there is a correlation among the signal OHHL, the protein LuxR and the complex LuxR-OHHL. Moreover, the patterns in the evolution of the average number of complexes across the population and the number of quorated bacteria are similar.

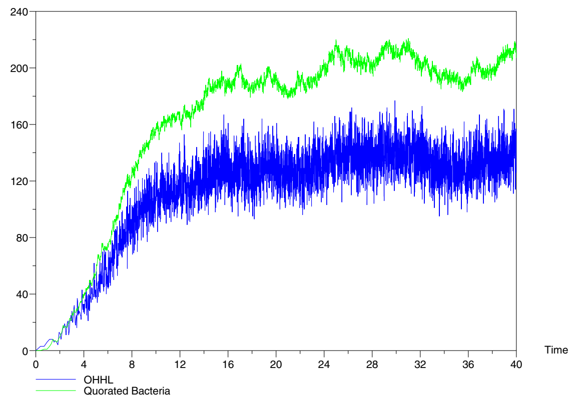

Fig. 2. The evolution of the quorated bacteria and the number of OHHL in the environment

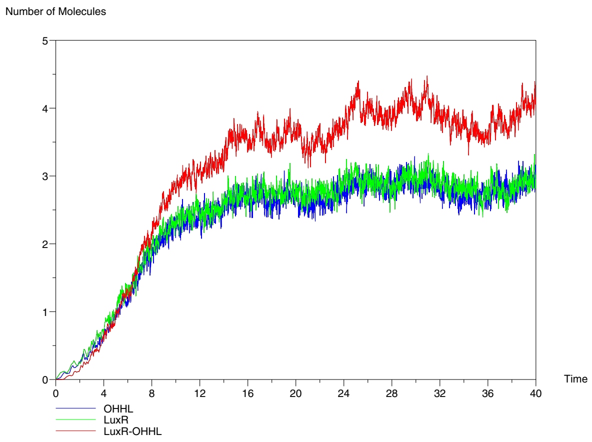

Fig. 3. The evolution of the number of signal molecule (OHHL), protein (LuxR) and the complex (LuxR-OHHL) for the average bacterium

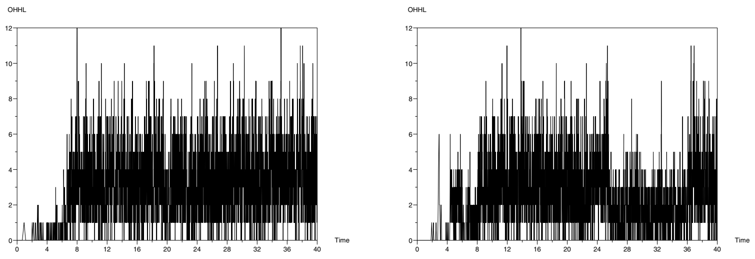

Fig. 4. The correlation between the amount of signal inside each bacterium (left) and the occupation of the LuxBox by the complex (right)

In our approach the behaviour of each individual in the colony can be tracked. We have taken a sample of two bacteria and have studied (see Figure 4) the correlation between the amount of signal inside each bacterium (left) and the occupation of the LuxBox by the complex (right) which represents that the bacterium has been quorated.

In Figure 5 it is shown that the number of signal molecules inside the bacterium has to exceed a threshold in order to recruit the bacterium. It may be observed that when the number of molecules is greater than the threshold the bacterium gets quorated or up-regulated (left), but when there are less signal molecules the bacterium switches off (right) the system and goes down-regulated.

We can also study how rules are applied across the evolution of the system. For instance, we can show the evolution of the number of applications of the rule representing the basal production of the signal OHHL (Figure 6) and the number of applications of the rules representing the production of the signal OHHL after the binding of the complex to the LuxBox (Figure 7).

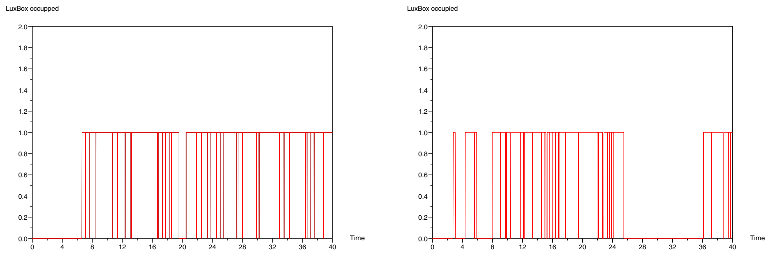

Fig. 5. The number of signal molecules inside bacterium when the level is greater than the threshold (left) and under the threshold (right)

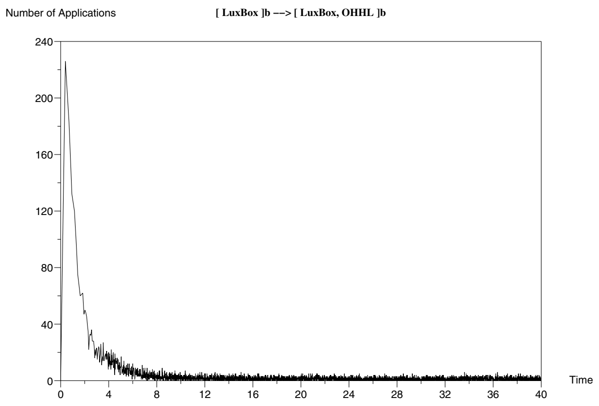

Fig. 6. The evolution of the number of applications of the rule representing the basal production of the signal OHHL

This can be compared with the number of applications of the rules representing the production of the signal OHHL after the binding of the complex to the LuxBox. In this way, we can show how at the beginning the basal production rule is the most applied rule while the other one is seldomly applied. Then, as a result of the recruitment process the bacteria sense the size of the population and they behave in a coordinate way by applying massively the third rule. Thus, the system moves from a down-regulated state to an up-regulated one where the bacteria collectively emit light. Specifically, this can be clearly seen if you compare the last graph above with the next one. Two similar graphs can be obtained for the rules producing the protein LuxR.

Finally, in order to study how bacteria can sense the number of individuals in the colony and get quorated only when the size of the colony is big enough, we have examined the behaviour of a population of only 10 bacteria. In this case, as shown in Figure 8, we observed that the recruitment process does not take place. Only one of the bacteria guessed wrong the size of the popu-

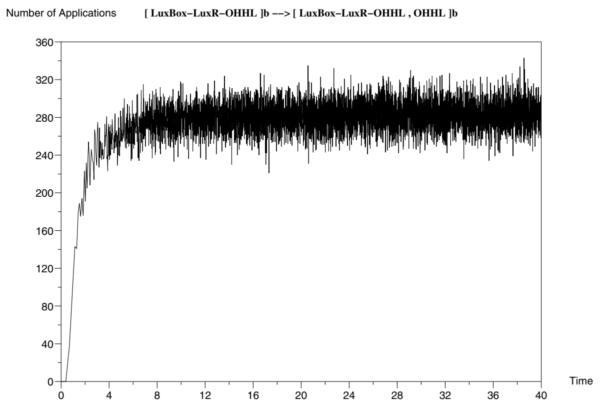

Fig. 7. The evolution of the number of applications of the rule representing the production of the signal OHHL after binding the complex to the LuxBox

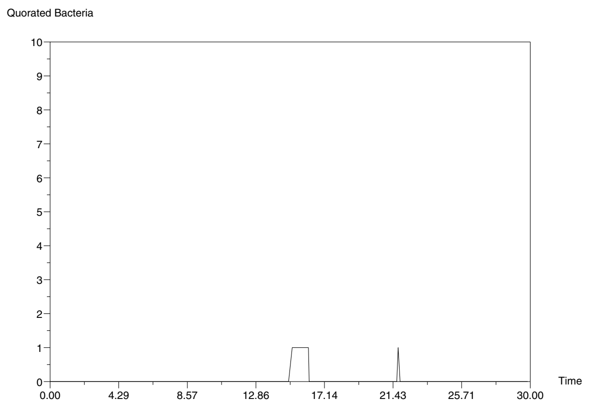

Fig. 8. No recruitment

lation and got up-regulated but then it switches off after sensing that the signal does not accumulate in the environment. The average number of molecules (see Figure 9) shows no pattern which means that the colony is not coordinating its behaviour.

Furthermore, we tracked the behaviour of two bacteria in the colony (see Figure 10) by obtaining that one never got quorated whereas the other one got quorated. Observe that this bacterium got quorated because the amount of signal inside exceeded the threshold.

Summing up, our simulations show that *Vibrio fischeri* has a quorum sensing system where a single bacterium can guess that the size of the population is big enough and starts to produce light. Then this bacterium starts to massively

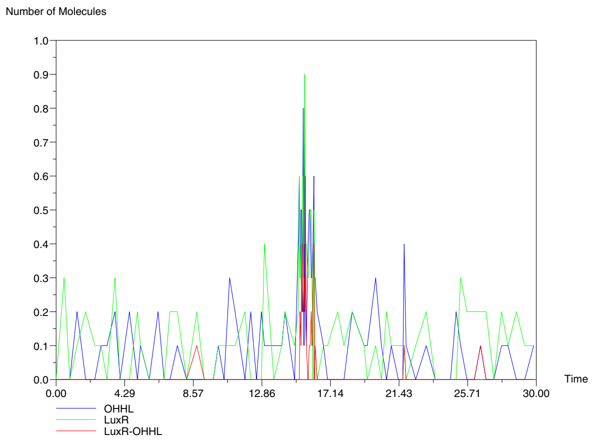

Fig. 9. No pattern of coordinating behaviour

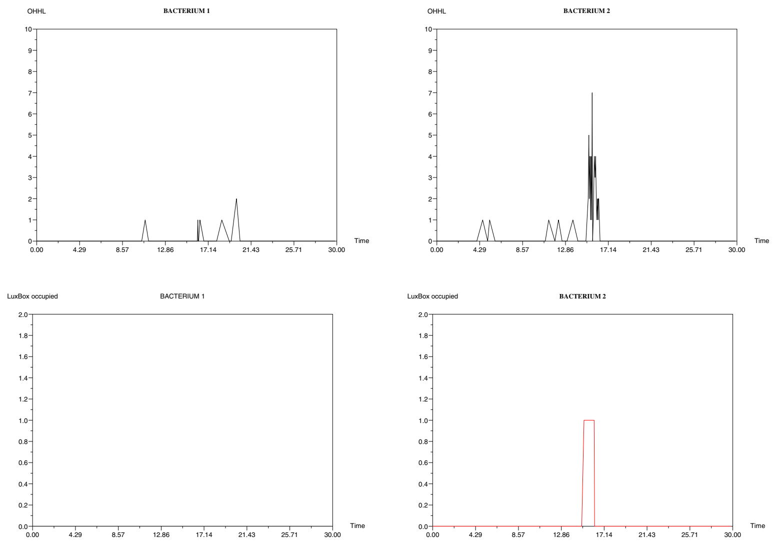

Fig. 10. The behaviour of two bacteria

produce signals, if the signal does not accumulate in the environment meaning that the guess was wrong it switches off. On the other hand if the signal does accumulate in the environment, meaning that the number of bacteria in the colony is big, then a recruitment process takes place that makes the whole population of bacteria to luminesce. These results agree well with in vitro experiments and with results obtained by using differential equations [6].

5 Implementation of the P System Model

We implemented the P system model of Definition 1 by following the approach proposed in [12] that is based on an initial specification in SBML of the model and a subsequent automatic generation of the executable code. In this section, we briefly describe the data structures necessary to support the execution of our variant of P systems. The language chosen is Scilab but similar considerations may apply to other commonly-used programming languages, such as C, Java, MatLab. Moreover, our approach appears to be fairly independent from the particular choice of P system variant. An SBML specification of the P system modelling quorum sensing in *Vibrio fischeri* is reported as an appendix.

The data structures used to represent the different components of P systems are the follows:

- **Rules:**

Recall that we are using rules of the form:

$$j : u\ [\ v\ ]_l \overset{k_j}{\rightarrow} u'[\ v'\ ]_l$$

Which will be represented as:

$$Comp\ father(l)\ l\ k_j\ multisets$$

with $multisets = length(u)\ u\ length(v)\ v\ length(u')\ u'\ length(v')\ v'$ and where $Comp$ represents the compartment where the rule j can be applied, $father(l)$ represents the father of the membrane with label l in the membrane structure, l is the label of the compartment involved in the rule and k_j is the kinetic constant. $length(u)$, $length(v)$, $length(u')$ and $length(v')$ tell us the size of u, v, u' and v', respectively. And u and v are the strings of objects representing the left-hand side (reactants) and u' and v' represent the right-hand side (products) of the rule j.

- **Compartments:**

Each compartment is represented by:

$$label\ n\text{-}copies\ multiplicity\text{-}of\text{-}o_1 \cdots multiplicity\text{-}of\text{-}o_n$$

The first component represents the label associated with the compartment, the second component is the number of instances of the compartment in the initial configuration; the other components describe for each object $o_i \in O$ its corresponding multiplicity inside that compartment.

- **Configurations of the system:**

A configuration of the system is made up of compartments; each compartment is represented:

$$identifier\ label\ multiplicity\text{-}of\text{-}o_1 \cdots multiplicity\text{-}of\text{-}o_n$$

identifier is an index associated in a one-to-one manner with each compartment, *label* is the label of the compartment and the last n components are the multiplicities of the objects in the compartment in the current configuration of the system.

Thus, the operation of a multiset u with a multiset v can be implemented by just subtracting and adding the corresponding vectors to the vectors representing the content of a certain compartment.

6 Conclusions

There is a growing interest in membrane computing in using P systems for modelling biological systems. This often requires the introduction into the model of quantitative aspects featuring the "reality" of the biological phenomenon to be modelled which are not usually considered in the abstract model of P systems. In this paper, these quantitative aspects have been considered for P systems by associating to each rule a real number (i.e., a kinetic constant), and by defining a Gillespie-like strategy for the application of the rules. This approach has been used to model the quorum sensing process in a colony of *Vibrio fischeri* bacteria by obtaining some simulation results which show the transition from a population of down-regulated cells to a population of up-regulated cells.

Our interest for the future is in developing a flexible software platform for running *in silico* experiments that integrates tools for the specification, execution and verification/validation of P system models. The details of the implementation provided in this paper can be viewed as a first step in this direction. A model checking approach is now being investigated that is based on Maude term rewriting tool [2]. In this framework, a central issue is the integration of the specification at individual level (e.g., a bacterium) with the specification at population level (e.g., the colony) such us to allow us to model more complex and larger biological systems. In this respect, a number of case studies need to be identified together with appropriate simulation/validation/verification techniques.

Acknowledgements

The research of F.B. and M.G. has been supported by the Engineering and Physical Sciences Research Council of United Kingdom (EPSRC), grant GR/R84221/01. N.K. acknowledges the support of the Biotechnological and Biological Sciences Research Council (BBSRC) for supporting his grant BB/C511764/1 and also to the EPSRC for the grant GR/T07534/01. The research of M.P. and F.R. has been supported by the Ministero de Ciencia y Tecnología of Spain, by the Plan Nacional de I+D+I (2002-2003) (TIC-2002-04220-C03-01), co-financed by FEDER funds, and by a FPI fellowship from the Universidad de Seville, Spain.

1. Alberts, B., Johnson, A., Lewis, J., Raff, M., Roberts, K., Walter, P., (2002). *The Molecular Biology of The Cell. Fourth Edition.* Garland Publ. Inc., London.
2. Andrei, O., Ciobanu, G., Lucanu, D., (2005). Executable Specifications of P Systems. In [10], 126–145
3. Bernardini, F., Gheorghe, M., (2004). Population P systems. *Journal of Universal Computer Science*, **10**, 509–539.
4. Besozzi, D., (2004). *Computational and Modelling Power of P systems.* PhD Thesis, Università degli Studi di Milano, Milan, Italy.
5. Bianco, L., Fontana, F., Franco, G., Manca, V. (2005). P Systems for Biological Dynamics. In: *Applications of Membrane Computing* (Ciobanu, G., Păun, Gh., Pérez-Jiménez, M.J., eds.), Springer-Verlag, Berlin, Heidelberg, New York, 81–126.
6. Fargerströn, T., James, G., James, S., Kjelleberg, S., Nilsson, P., (2000). Luminescence Control in the Marine Bacterium *Vibrio Fischeri*: An Analysis of the Dynamics of lux Regulation. *Journal of Molecular Biology*, **296**, 1127–1137.
7. Gibson, M.A., Bruck, J., (2000). Efficient Exact Stochastic Simulation of Chemical Systems with Many Species and Many Channels. *Journal of Physical Chemistry*, **104**, 25, 1876–1889.
8. Gillespie, D.T., (1977). Exact Stochastic Simulation of Coupled Chemical Reactions. *The Journal of Physical Chemistry*, **81**, 25, 2340–2361.
9. Martin-Vide, C., Mauri, G., Păun, Gh., Rozenberg, G., Salomaa, A., eds., (2004). Membrane Computing. International Workshop, WMC 2003, Tarragona, Spain, July 2003. Revised Papers. *Lecture Notes in Computer Science*, **2933**, Springer-Verlag, Berlin, Heidelberg, New York.
10. Mauri, G., Păun, Gh., Pérez-Jiménez, M., J., Rozenberg, G., Salomaa, A., eds., (2005). Membrane Computing. International Workshop, WMC 2004, Milan, Italy, June 2004. Revised and Invited Papers. *Lecture Notes in Computer Science*, **3365**, Springer-Verlag, Berlin, Heidelberg, New York.
11. Meng, T.C., Somani S., Dhar, P., (2004). Modelling and Simulation of Biological Systems with Stochasticity. *In Silico Biology*, **4**, 0024.
12. Nepomuceno, I, Nepomuceno, J.,A, Romero-Campero, F., (2005) . A Tool for Using the SBML Format to Represent P System which Model Biological Reaction Networks. In: *Proceeding of the Third Brainstorming Week in Membrane Computing, Seville, Spain, January 31st-February 4th, 2005*, University of Seville, Seville, Spain.
13. Păun, Gh., (2000). Computing with Membranes. *Journal of Computer and System Sciences*, **61**, 1, 108–143.
14. Păun, Gh. (2002). *Membrane Computing. An Introduction.* Springer-Verlag, Berlin, Heidelberg, New York.
15. Păun, Gh., Rozenberg, G., Salomaa, A., Zandron, C., eds., (2003). Membrane Computing. International Workshop, WMC-CdeA 02, Curtea de Arges, Romania, August 19-23, 2002. Revised Papers. *Lecture Notes in Computer Science*, **2597**, Springer-Verlag, Berlin, Heidelberg, New York.
16. Pérez-Jiménez, M.J.; Romero-Campero, F.J.,(2005). Modelling EGFR Signalling Cascade Using Continuous Membrane Systems. In: *Proceedings of the Third International Workshop on Computational Methods in Systems Biology 2005 (CMSB 2005)* (Plotkin, G., ed.), University of Edinburgh, Edinburgh, United Kingdom.
17. Philips, A., Cardelli. L., (2004). A Correct Abstract Machine for the Stochastic Pi-calculus. *Electronical Notes in Theoretical Computer Science*, to appear.

18. Priami, C., Regev, A., Shapiro, E., Silverman, W., (2001). Application of a Stochastic Name-Passing Calculus to Representation and Simulation of Molecular Processes. *Information Processing Letters*, **80**, 25–31.
19. Taga, M., E., Bassler, B., L., (2003). Chemical Communication among Bacteria. *Proceedings of the National Academy of Sciences of the United States of America* (*PNAS*), **100**, 2, 14549–14554.

A An SBML Specification

Consider the P system $\Pi(m)$, with $m = 100$, defined in Section 3. We start by specifying the structure of the system by listing the compartments present in the system and the relationships of inclusion between them.

```
<listOfCompartments>
  <compartment id="e" />
  <compartment id="b" outside="b"/>
</listOfCompartments>
```

There are two different "types" of compartments: compartments labelled by e and compartment labelled by b; all the compartments labelled by b, the bacteria, are included in a compartment with label e, the environment. Specifically, this is just a shorthand for a membrane structure consisting of a number of membranes, each one associated with a compartment labelled by b, contained inside an unique main membrane associated with a compartment labelled by e. The actual number of bacteria in the system is specified as a parameter of the system together with the constants k_i, $1 \leq i \leq 14$.

```
<listOfParameters>
   <parameter id="k1" value="2''constant="true"/>
   <parameter id="k2" value="2" constant="ture"/>
   <parameter id="k3" value="9" constant="true"/>
   <parameter id="k4" value="1" constant="true"/>
   <parameter id="k5" value="10" constant="true"/>
   <parameter id="k6" value="2" constant="true"/>
   <parameter id="k7" value="250" constant="true"/>
   <parameter id="k8" value="200" constant="true"/>
   <parameter id="k9" value="1"  constant="true"/>
   <parameter id="k10" value="50" constant="true"/>
   <parameter id="k11" value="30" constant="true"/>
   <parameter id="k12" value="15" constant="true"/>
   <parameter id="k13" value="20" constant="true"/>
   <parameter id="k14" value="20" constant="true"/>
    <parameter id="m" value="100" constant="true"/>
</listOfParameters>
```

Next, we specify the initial distribution of objects inside the system by listing out the species and their initial concentration inside each compartment.

```
<listOfSpecies>
  <specie id="OHHL_e"
  initialConcentration="0" compartment="e" />
  <specie id="OHHL_b"
  initialConcentration="0" compartment="b" />
  <specie id="LuxR_b"
  initialConcentration="0" compartment="b" />
  <specie id="LuxR_OHHL_b"
  initialConcentration="0" compartment="b" />
  <specie id="Lux_Box_b"
  initialConcentration="1" compartment="b" />
  <specie id="Lux_Box_LuxR_OHHL_b"
  initialConcentration="0" compartment="b" />
</listOfSpecies>
```

The objects that can be contained inside the environment are labelled by e whereas the objects that can appear inside a bacterium are labelled by b.

Finally we specify the rules as a list of SBML reactions. We just report here two of them as an example.

```
  <reaction name="Reaction1" reversible="false">
    <listOfReactants>
     <specieReference specie="Lux_Box_b" />
    </listOfReactants>
    <listOfProducts>
     <specieReference specie="Lux_Box_b" />
     <specieReference specie="OHHL_b" />
    </listOfProducts>
    <kineticLaw>
      <math xmlns="http://www.w3.org/1998/Math/MathML">
        <apply>
          <times/>
          <ci>k1</ci>
          <ci>Lux_Box_b</ci>
        </apply>
     </math>
    </kineticLaw>
  </reaction>

  <reaction name="Reaction9" reversible="false">
    <listOfReactants>
     <specieReference specie="OHHL_b" />
    </listOfReactants>
    <listOfProducts>
     <specieReference specie="OHHL_e" />
    </listOfProducts>
    <kineticLaw>
```

```
      <math xmlns="http://www.w3.org/1998/Math/MathML">
        <apply>
          <times/>
          <ci>k9</ci>
          <ci>OHHL_b</ci>
        </apply>
      </math>
    </kineticLaw>
  </reaction>
```

The movement of objects is specified by changing the labels of the products according to the labels of the reactants.