A Modeling Approach Based on P Systems with Bounded Parallelism

Francesco Bernardini[1], Francisco J. Romero-Campero[2], Marian Gheorghe[3], and Mario J. Pérez-Jiménez[2]

[1] Leiden Institute of Advanced Computer Science
University of Leiden
Niels Bohrweg 1, 2333 CA Leiden, The Netherlands
bernardi@liacs.nl
[2] Research Group on Natural Computing
Department of Computer Science and Artificial Intelligence
University of Seville, Avda. Reina Mercedes, 41012 Sevilla, Spain
{fran,marper}@cs.us.es
[3] Department of Computer Science, The University of Sheffield
Regent Court, Portobello Street, Sheffield S1 4DP, UK
M.Gheorghe@dcs.shef.ac.uk

Abstract. This paper presents a general framework for modelling with membrane systems that is based on a computational paradigm where rules have associated a finite set of attributes and a corresponding function. Attributes and functions are meant to provide those extra features that allow to define different strategies to run a P system. Such a strategy relying on a bounded parallelism is presented using an operational approach and applying it for a case study presenting the basic model of quorum sensing for *Vibrio fischeri* bacteria.

1 Introduction

In 1998, Gheorghe Păun initiated the field of research called membrane computing with a paper firstly available on the web and later published in [19]. Membrane computing aims at defining computational models which abstract from the functioning and structure of the cell. In particular, membrane computing starts from the observation that compartmentalization through membranes is one of the essential features of (eucaryotic) cells. Unlike bacterium, which generally consists of a single intracellular compartment, a(n) (eucaryotic) cell is sub-divided into functionally distinct compartments. Thus, a class of computing devices called membrane systems, or P systems, are defined [19], which have three essential features: a membrane structure consisting of a hierarchical arrangement of several compartments defined as regions delimited by membranes; objects assigned to regions; and rules assigned to the regions of the membrane structure, acting upon the objects inside. In particular, each region is supposed to contain a finite set of rules and a finite multiset (or set) of objects. Rules encode generic processes for producing/consuming objects and for moving objects from one region to the other. Objects are described either as symbols from a

given alphabet or as strings over a given alphabet. The application of the rules is performed in a non-deterministic maximally parallel manner: all the applicable rules that should be used to modify existing objects must be applied, and this is done in parallel for all membranes.

Since this model was introduced for the first time in 1998, many variants of membrane systems have been proposed and studied – a comprehensive bibliography of P systems can be found at the P systems web page [29]. The most investigated membrane system topics are related to the computational power of different variants, their capabilities to solve hard problems, like NP-complete problems, decidability, complexity aspects and hierarchies of classes of languages produced by these devices.

Membrane computing represents nowadays a research area of a larger interdisciplinary field called natural computing, that involves scientists studying the emergence of new computational paradigms inspired from the behavior of various natural phenomena. In the same time there is a growing interest in applying mathematical and computational paradigms to model real natural systems. Computational biology is such a field, where mathematical and computational models of biological systems are designed for the analysis and simulation of the behavior of these systems. Biological modeling has involved standard continuous and stochastic mathematical approaches, as well as discrete models. Standard mathematical models with their simulation techniques have proved to be powerful tools for understanding the dynamics of biological systems (e.g., see [25], [26]). Discrete modeling instead advocates the use of different formalisms taken from various areas of computer science (e.g., formal grammars [6], Petri nets [16], X machines [9], [27], process algebra [24], [3], statecharts [15], etc.) to develop computational models of biological systems.

This paper presents a general framework for modeling with membrane systems that is based on a model of P systems where rules have associated a finite set of attributes and a corresponding function. Attributes and functions are meant to provide those extra features which are necessary to close the "gap" between the abstractness of more standard P system models and the "reality" of the phenomenon to be modeled (Section 2). The behavior of such P systems is defined in Section 3 in terms of bounded parallelism, which precisely formalizes the idea of a membrane system as a system where a certain number of components evolve in parallel at the same time by means of a certain number of rules applied inside each one of these components. Then, in Section 4, as a particular instance of the general model, we consider P systems where rules have associated a real constant as an attribute and the corresponding function is used to compute a value of a probability depending on this constant and on certain multisets of objects defining the context where the rule is applied; for this particular variant of P systems, we also define a strategy for the application of the rules which, in each step, selects the next rule to be applied depending on the aforementioned values of probabilities. Finally, in Section 5, as a case study, we present a P system model for the quorum sensing mechanism of bacterial cell-to-cell communication.

2 The Model

Usually a P system is defined as a hierarchical arrangement of a number of membranes identifying a corresponding number of regions inside the system, and with these regions having associated a finite multiset of objects and a finite set of rules. Moreover, one can also consider membrane systems where the underlying structure is defined as an arbitrary graph like in tissue P systems [20], and in population P systems [1]. In this paper, we only focus on membrane systems of the former type where the underlying structure is defined as being a tree of nested membranes.

Rules of many different forms have been considered for membrane systems in order to encode the operation of modifying the objects inside the membranes and the operation of moving objects from one place to the other. In particular, for communicating objects, one can use either the targets $here, in, out$, or symport/antiport rules, or boundary rules [20].

Here, in order to capture the features of most of these rules, we consider rules of the form:

$$u\,[\,v\,] \rightarrow u'\,[\,v'\,] \tag{1}$$

with u, v, u', v' some finite multisets. These rules are generalized boundary rules operating as multiset rewriting rules which simultaneously replace a multiset of objects placed outside the membrane and a multiset of objects placed inside the membrane with two new multisets placed in the same places. In this way, we are able to capture in a concise way the essential features of transformation and communication of objects usually considered in the area of membrane computing. Moreover, from a modeling point of view, rules like (1) allow us to express any sort of interactions occurring at the membrane level, like, for instance, the binding of a signal molecule to a specific receptor which occurs at cell-surface level (e.g., see [21]).

We associate to each rule a finite set of attributes in order to be able to capture specific quantitative aspects of the phenomenon to be modeled. Specifically, these attributes are used by rules as different reaction rates and/or different probabilities which overall affects the strategy of the application of the rules.

Definition 1 (program). *Let V, K be alphabets, let D be a set (possibly infinite), and let A be a finite subset of D. The set D is called the* set of values *and the set A is called the* set of attributes. *Let $\mathcal{F} = \{f_1, f_2, \ldots, f_p\}$ be a finite set of functions such that, for all $1 \leq t \leq n$, $f_t : \mathcal{P}(D) \times V^* \times V^* \times V^* \times V^* \longrightarrow D$ with $\mathcal{P}(D)$ the family of all subsets of D. A* program (over $V, K, D, A, \mathcal{F}$) *is a construct $\langle u\,[\,v\,] \rightarrow u'\,[\,v'\,], \sigma, f_i\rangle_l$ where: $u, v, u', v' \in O^*$, $\sigma \subseteq A$, $f_i \in \mathcal{F}$, for some $1 \leq i \leq p$, and $l \in K$.*

Thus, a program consists of a rule, a finite set of attributes and a function which, given this set of attributes and four multisets, returns a value from a certain chosen set. In particular, the first two multisets are usually considered as being the two multisets placed on the left side of the rule; the two remaining multisets are instead supposed to be the multisets placed respectively inside

and outside a given membrane. That is, each function is expected to compute a particular value depending on the rule and on the contents of the outside and the inside regions that define the "context" where the rule is applied.

Here our main focus is in modeling systems consisting of many different biochemical reactions distributed across different compartments. Therefore, the following interpretation for programs will be predominantly used throughout the paper. Objects represent chemicals and multisets of objects are interpreted as "bags" or "soup" of chemicals. Rules model transformations involving these chemicals. Each rule has associated a finite set of attributes and a corresponding function defining particular properties that affect the behavior of the rule itself; these properties are usually said to define the *reactivity of the rule* (e.g., see [2], [21]). Moreover, since we want to have systems consisting of many different compartments, we assign to each program a label to differentiate the set of programs from one compartment to the other.

Remark 1. Although the aforementioned interpretation will be mainly used in this paper, it is not the unique possible and that is why we introduce the notions of set of values and set of attributes in a more generic fashion. For instance, one may use attributes to model properties inherent to the membranes and may use the corresponding function to update these attributes every time a rule is applied. Alternatively, one may consider hybrid models where the attributes represent continuous variables which are updated through the use of certain functions (e.g., see [9] for hybrid X machines, and [16] for hybrid Petri nets).

Next, we introduce the notion of P specification which makes explicit the basic components necessary to define a particular P system model.

Definition 2 (P specification). *A* P specification *is a construct*

$$\mathcal{S} = (V, K, D, A, \mathcal{F}, P)$$

where:

1. V *is an alphabet; its elements are called* objects*;*
2. K *is an alphabet; its elements are called* labels*;*
3. D *is a* set of values*;*
4. $A \subseteq D$ *is a finite* set of attributes*;*
5. $\mathcal{F} = \{f_1, f_2, \ldots, f_p\}$ *is a finite set of functions as in Definition 1.*
6. P *is a finite set of programs over* $V, K, D, A, \mathcal{F}$ *of the form specified in Definition 1.*

Thus, a P specification provides a "scheme" for the definition of P systems with programs defined over the sets V, K, D, A, and F. In particular, from a modeling point of view, we can say that the alphabet V of objects specifies different "sorts" for the chemicals present inside a certain system, the alphabet K of labels specifies different "types" for the membranes possibly present inside a certain system, and the set D specifies the domain of interpretations for the attributes and the corresponding functions.

A P system is then obtained by augmenting a certain P specification with an initial configuration. In particular, we recall that the structure of a P system is given by a hierarchical arrangement of some $n \geq 1$ membranes labeled in an one-to-one manner with values in $\{1, 2, \ldots, n\}$ [20]. However, since here we use labels from a given alphabet to identify the "type" of a membrane, the value from $\{1, 2, \ldots, n\}$ assigned to a membrane is called the *index of the membrane*; the membrane structure is then said to be *indexed by the values* $1, 2, \ldots, n$.

Definition 3 (P system). *A* P system of degree $n \geq 1$ *is a construct:*

$$\Pi = (\mathcal{S}, \mu, M_1, M_2, \ldots, M_n)$$

where $\mathcal{S} = (V, K, D, A, \mathcal{F}, P)$ *is a P specification with* $V, K, D, A, \mathcal{F}, P$ *as in Definition 2,* μ *a membrane structure containing* n *membranes indexed by the values* $1, 2, \ldots, n$*, and, for all* $1 \leq i \leq n$*,* $M_i = (w_i, l_i)$*, with* $w_i \in V^*$ *the* content *of membrane* i *and* $l_i \in K$ *the* label *of membrane* i*.*

As usual, a P system of degree $n \geq 1$ is defined as consisting of a membrane structure containing n membranes. Each membrane contains a multiset of objects and gets assigned a label from the set K. This latter symbol is particularly useful for retrieving from the given specification the set of programs which can be used inside each membrane in the system. In other words, this label precisely identifies the "type" of the membrane in terms of the rules which can be applied inside.

Most of the P system variants utilize a maximal parallel rewriting manner [20]. This means, in each step, in each membrane, all the objects that can evolve by means of some rules must evolve in parallel, with the only restriction that the same occurrence of the same object cannot be used by more than one rule at a time. That is, in each step, for each membrane, a maximal set of rules to be applied is non-deterministically selected by making sure that no further rules can be applied to the objects left inside the membranes. In this paper we will introduce a mechanism to bound the number of applications of the rules and the number of membranes that will evolve in a step. In this respect, the key issues that need to be addressed in order to define a strategy for the application of the rules in a P system are: a) how to select the next rule to be applied inside a given membrane, b) how many different rules can be applied in parallel at the same time inside the membrane, and c) how many different membranes can evolve in parallel at the same time.

3 Parallelism of Type (k, q)

We formalize here the notion of a transition step in P systems evolving in a $(k, q)-$parallel manner: in each step, at most k membranes evolve in parallel at the same time and, inside each membrane, at most q rules are applied in parallel at the same time. Moreover, in a given step, if k membranes can evolve by means of some rules, then exactly k membranes must evolve in parallel in that step; inside each membrane, if q rules can be applied, then exactly q rules are applied

in parallel inside that membrane. In other words, parallelism of type (k, q) is assumed to be maximal and exhaustive with respect to k and q.

Our formalization makes use of some concepts of the operational semantics for P systems introduced in [5] and it is based on the explicit assignment of the rules contained to the programs to the respective membranes; this is obtained by assigning the index of a membrane to each object possibly present inside the system.

Let $\Pi = (\mathcal{S}, \mu, M_1, M_2, \ldots, M_n)$ with $\mathcal{S} = (V, K, D, A, \mathcal{F}, P)$ and $M_i = (w_i, l_i)$, for all $1 \leq i \leq n$, be a P system as specified in Definition 3. The following extra notions are associated to the P system Π:

- the *indexed alphabet (of Π)* denoted by $\bar{V}$ is the set $\bar{V} = \{\, a_i \mid a \in V, 1 \leq i \leq n \,\}$;
- for all $1 \leq i \leq n$, and for all $u \in V^*$, the *i-version of u* is the multiset denoted by $u_i \in \bar{V}$ such that, for all $a \in V$, for all $1 \leq j \neq i \leq n$, $|u_i|_{a_i} = |u|_a$ and $|u_{a_j}| = 0$;
- the *set of membrane rules (of Π)* is the set of programs denoted by MR and such that: $MR = \{\, u_j\, v_i \rightarrow u'_j\, v'_i \mid \langle u\,[\,v\,] \rightarrow u'\,[\,v'\,], \sigma, f\rangle_{l_i} \in P, j = \mathit{upper}(\mu, i) \,\}$, where $\mathit{upper}(\mu, i)$ is a function returning for a given membrane structure μ, the membrane containing the region i.

Thus, the indexed alphabet of Π is the alphabet of symbols from V with attached indexes of the membranes in the system. The i-version of a multiset u, with $1 \leq i \leq n$ and $u \in V^*$, is the multiset obtained by assigning the index i to all the objects in u. The set MR explicitly identifies, for each membrane, the set of rules which can be used inside that membrane by replacing the multisets in the rules with the corresponding indexed versions. In particular, for all $1 \leq i \leq n$, the set of rules which can be used inside membrane i are the rules contained in programs labeled by l_i.

Moreover, for all $1 \leq i \leq n$, a *multiset of rules for membrane i*, is a collection of membrane rules $r_1, r_2, \ldots, r_{k_i}$, with $k_i \geq 0$ and with these rules not necessarily distinct, such that, for all $1 \leq h \leq k_i$, r_h is a membrane rule in MR of the form $u_j\, v_i \rightarrow u'_j\, v'_i$ with $v_i \neq \lambda$. The *size* of R_i, denoted by $|R_i|$, is the number of rules in R_i.

A *multiset of rules in Π* is a collection of membranes rules of the form R_1, $R_2, \ldots, R_n$ such that, for all $1 \leq i \leq n$, R_i is a multiset of rules for membrane i.

Then, a configuration of a P system is defined as being a multiset over the indexed alphabet.

Definition 4 (Configuration). *Let $\Pi = (\mathcal{S}, \mu, M_1, M_2, \ldots, M_n)$ be a P system as in Definition 3 where $\mathcal{S} = (V, K, D, A, \mathcal{F}, P)$ and $M_i = (w_i, l_i)$, for all $1 \leq i \leq n$. A* configuration (of Π) *is a multiset $C \in \bar{V}$. The* initial configuration (of Π)*, denoted by C_0, is the multiset $u_1\, u_2 \ldots u_n$ such that, for all $1 \leq i \leq n$, u_i is the i-version of w_i.*

Notice that, with the notions introduced in this section so far, we have essentially reduced a P system of degree $n \geq 1$ to an equivalent one of degree 1 where the objects have an index specifying the membrane which they are assigned to in

the original system. Thus, the behavior of such a system can be defined as being a multiset rewriting system where the rules are selected according to a certain strategy depending on the indexes assigned to the objects. In fact, according to Definition 4, a configuration of a P system is just a multiset of objects and the membrane rules are usual multiset rewriting rules.

To this aim, we need first to introduce the concepts of i-irreducibility and the concept of (C, k, q)-consistency.

Definition 5 (i-irreducibility). *Let $\Pi = (\mathcal{S}, \mu, M_1, M_2, \ldots, M_n)$ be a P system as in Definition 3 where $\mathcal{S} = (V, K, D, A, \mathcal{F}, P)$, and let β be a finite multiset over $\bar{V}$. Let MR be the set of membrane rules in Π. Given $1 \leq i \leq n$, we say that β is* i-irreducible *if, for all $u_j\, v_i \rightarrow u'_j\, v'_i \in MP$ with $v_i \neq \lambda$, we have $\beta \not\supseteq u_j\, v_i$.*

Thus, given a P system of degree $n \geq 1$, for all $1 \leq i \leq n$, a multiset over the indexed alphabet is i-irreducible if there are no more rules which can be applied to the objects with index i. In other words, if we interpret these objects as being the content of membrane i, this means that there are no more rules that can be applied to the objects placed inside membrane i.

Definition 6 ((C, k, q)-consistency). *Let $\Pi = (\mathcal{S}, \mu, M_1, M_2, \ldots, M_n)$ be a P system as in Definition 3 where $\mathcal{S} = (V, K, D, A, \mathcal{F}, P)$, and let R be a multiset of rules in Π. Let $C \in \bar{V}^*$ be a configuration of Π, and let $k, q \neq 0$ be positive integers with $k \leq n$. We say that R is* (C, k, q)-consistent *if there exists $\{i_1, i_2, \ldots, i_g\} \subseteq \{1, 2, \ldots, n\}$ with $g \leq k$ such that R can be written as a collection of rules $R_{i_1}, R_{i_2}, \ldots, R_{i_g}$ with R_{i_h} a multiset of rules for membrane i_h, and*

1. *if $g < k$, then, for all $i \in (\{1, 2, \ldots, n\} \setminus \{i_1, i_2, \ldots, i_g\})$, C is i-irreducible;*
2. *for all $i \in \{i_1, i_2, \ldots, i_k\}$, if $R_i = u_j^1\, v_i^1 \rightarrow z_j^1\, w_i^1, \ldots, u_j^p\, v_i^p \rightarrow z_j^p\, w_i^p$, then we have $C = x\, u_j^1\, v_i^1 \ldots u_j^p\, v_i^p$, for some $x \in \bar{V}^*$;*
3. *for all $i \in \{i_1, i_2, \ldots, i_k\}$, $|R_i| \leq q$, and if $R_i = u_j^1\, v_i^1 \rightarrow z_j^1\, w_i^1, \ldots, u_j^p\, v_i^p \rightarrow z_j^p\, w_i^p$ with $p < q$, then $C = x\, u_j^1\, v_i^1 \ldots u_j^p\, v_i^p$, for some i-irreducible $x \in \bar{V}^*$.*

The notion of (C, k, q)-consistency precisely characterizes the multisets of rules which can be applied to a given configuration C in accordance to the parallelism of type (k, q). In fact, such a multiset of rules must contain a multiset of rules for at most k distinct membranes; if there are not k membranes that can evolve by means of some rules, then a smaller but maximal number of membranes must be selected (Condition 1 of Definition 6). The rules contained in the selected multiset must be applicable to the objects currently contained inside each membrane (Condition 2 of Definition 6). Moreover, for each membrane, at most q rules must be selected; if inside some membrane there are less than q rules that can be applied, then all of them must be applied (Condition 3 of Definition 6).

Therefore, in order to perform a (k, q)-parallel step in a given P system, it is necessary to first select a multiset of rules R to be applied to the current configuration C such that R is (C, k, q)-consistent. In this respect, we assume to have defined an *algorithm to select programs and membranes* $\mathcal{A}_{k,q}$ such that,

given a configuration of a P system Π and its set of programs, returns a multiset of rules which is (C, k, q) consistent. In all the previous sections, this selection has been defined as being non-deterministic but, in general, one may identify other strategies which, in particular, should take into account the attributes associated with the rules. Approaches in this direction are considered in [2], [21], [22] where strategies for the selection of the rules are defined which depend on a notion of rate of application of the rules, or on certain probabilities associated with the rules. In the next section, we present one such strategy where the rules to be applied in the next step are selected depending on a particular distribution of probabilities computed step by step.

Here, we define the notion of a (k, q)-parallel step of computation by assuming a generic algorithm for the selection of the rules.

Definition 7 ((k, q)-parallel step). *Let $\Pi = (\mathcal{S}, \mu, M_1, M_2, \ldots, M_n)$ be a P system as in Definition 3 where $\mathcal{S} = (V, K, D, A, \mathcal{F}, P)$, and let C_1, C_2 be configurations of Π. Let $\mathcal{A}_{k,q}$ be an algorithm for the selection of the rules which is able to return a (C_1, k, q)-consistent multiset of rules in Π. We say that C_2 can be obtained from C_1 in a* (k, q)-parallel step*, denoted by $C_1 \Rightarrow_{\Pi}^{(k,q)} C_2$, if there exists a multiset R of rules in Π such that:*

1. $\mathcal{A}_{k,q}(C_1, P) = R$;
2. $R = u_{j_1} v_{i_1} \rightarrow z_{j_1} w_{i_1}, \ldots, u_{j_p} v_{i_p} \rightarrow z_{j_p} w_{i_p}$, *for some* $p > 0$;
3. $C_1 = x\, u_{j_1} v_{i_1} \ldots u_{j_p} v_{i_p}$ *and* $C_2 = x\, z_{j_1} w_{i_1} \ldots z_{j_p} w_{i_p}$, *for some* $x \in \bar{V}^*$.

If that is the case, then we write $C_1 \Rightarrow_{\Pi}^{(k,q)} C_2$.

Thus, a (k, q)-parallel step in a P system consists in the parallel application of a (C, k, q)-consistent multiset of rules to a certain configuration C. The multiset of rules to be applied is supposed to be returned by a particular algorithm to select membranes and programs, and this has to be done before every step depending on the current configuration of the system.

Then, we introduce the notion of sequence of (k, q)-parallel steps and (k, q)-parallel execution of a P system.

Definition 8 (sequence of (k, q)-parallel steps). *Let $\Pi = (\mathcal{S}, \mu, M_1, M_2, \ldots, M_n)$ be a P system as in Definition 3. A* sequence of (k, q)-parallel steps in Π *is a sequence σ such that*

$$\sigma = C_1, C_2, \ldots, C_h$$

where, for all $1 \leq i \leq h$, C_i is a configuration of Π, and, if $i \neq h$, then $C_i \Rightarrow_{\Pi}^{(k,q)} C_{i+1}$. If that is the case, we say that σ is a sequence of (k, q)-parallel steps in Π that starts from C_1 *and that C_h is* obtained from C_1 in $h - 1$ steps*; we also write $C_1 \Rightarrow_{\Pi}^{(k,q),h} C_h$.*

Definition 9 ((k, q)-parallel execution). *Let $\Pi = (\mathcal{S}, \mu, M_1, M_2, \ldots, M_n)$ be a P system as in Definition 3. A* (k, q)-parallel execution of Π *is a sequence of (k, q)-parallel steps in Π which starts from the initial configuration of Π.*

Thus, we have characterized the behavior of P systems operating according to a bounded parallelism where the number of membranes and the number of rules which can be used in every step are overall bounded by some given constants.

Remark 2. From a computational point of view, the introduction of bounded parallelism in membrane systems does not affect the fundamental universality results concerning the computational power of different variants of P systems, such as P systems with catalysts, with symport/antiport, with boundary rules, etc. In fact, it is easy to see that, in all those cases, the simulation of counter machines is achieved by means of P systems where the number of rules applied in parallel in each step is actually overall bounded (e.g., see [7]). On the other hand, it is shown in [8], [10] that P systems with catalysts operating in sequential mode and P systems with symport/antiport operating in sequential mode (i.e., with parallelism of type $(1, 1)$) are strictly less powerful than their corresponding parallel versions. Moreover, one can also notice that, whenever k is equal to the number of membranes in the system, our notion of parallelism of type (k, q) coincides with the notion of q-**Max-Parallelism** introduced in [7].

4 An Algorithm to Select Membranes and Programs

We present an algorithm to select membranes and programs for P systems operating with parallelism of type $(1, 1)$ (i.e., in sequential mode) where the next membrane to evolve and the next rule to be applied inside this membrane is randomly selected according to a certain distribution of probabilities. However, with respect to Definition 3, the algorithm is here defined only for a restricted model of P systems where rules are all of the forms:

$$u\,[\,] \to [\,v\,], [\,v\,] \to u[\,], [\,v\,] \to [\,v'\,] \tag{2}$$

that is, there is a distinction between transformation rules and communication rules, communication is only unidirectional, and there is no interaction between the inside and the outside of a membrane.

Our strategy for selecting membranes and programs is based on Gillespie's algorithm [12]. This algorithm [12] 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 [17]. As well as this, Gillespie's algorithm is used in the implementation of stochastic π-calculus [4] and in its application to the modeling of biological systems [23].

We follow a similar approach to associate a stochastic behavior to membrane systems by considering P systems where each rule has associated a real constant which defines its rate of application and which is used to compute the probability of the rule to be applied in the next step in the same way as in Gillespie's algorithm. More precisely, we consider a class of P systems where, with respect to Definition 3, the set of values is the set of non-negative real numbers denoted by $\mathbb{R}_0^+$, each programs contains a rule like (2), a real constant as an attribute,

and a function to compute a probability depending on the value of this constant. For short, such a P system is called PPR (i.e., a P systems with Probabilities associated with the Rules).

In order to compute the probability values, we use, for all the programs, the function ϕ such that $\phi : \mathbb{R}_0^+ \times V^* \times V^* \longrightarrow \mathbb{R}_0^+$ with:

$$\phi(k, u, \alpha) = k \cdot \prod_{a \in alph(u)} \frac{|\alpha|_a!}{|u|_a! \cdot (|\alpha|_a - |u|_a)!} \tag{3}$$

for all $k \geq 0$, $u, \alpha \in V^*$ and $u \sqsubseteq \alpha$; $\phi(k, u, \alpha) = 0$ for all $k \geq 0$, $u, \alpha \in V^*$ and $u \not\sqsubseteq \alpha$.

That is, given a rule $[\, v\,] \rightarrow u[\,]$, or a rule $[\, v\,] \rightarrow [\, v'\,]$ with an associated attribute k, and given a multiset $\alpha \sqsupseteq v$, expression (3) returns the number of different ways of choosing $|v|_a$ occurrences of a from the multiset α, for all a such that $|a| \geq 0$. In particular, the multiset α is supposed to be the multiset of objects placed inside the membrane where the rule is going to be used. In a similar way, given a rule $u\,[\,] \rightarrow [\, v\,]$ with attribute k, expression (3) is used to compute a probability value for this rule by considering the multiset u and the multiset of objects placed in the outside region.

Remark 3. The function given by expression (3) is already used in [22] to compute probability values for rules. However, this is done in the context of a different algorithm to select rules and programs which is not directly related to Gillespie's algorithm.

Next, we provide a formal definition for the notion of a PPR.

Definition 10 (PPR). *A PPR of degree $n \geq 1$ is a construct*

$$\Pi = (V, K, \mathbb{R}_0^+, A, \phi, P, \mu, M_1, M_2, \ldots, M_n)$$

where:

- $(V, K, \mathbb{R}_0^+, A, \phi, P)$ *is a P specification with ϕ the function given by expression (3), and all the programs in P having the form $\langle r, k, \phi \rangle_l$ for r a rule like (2), and $k \in A$;*
- *μ, and $M_1, M_2, \ldots, M_n$ are as in Definition 3.*

Remark 4. The function ϕ is used to compute the probabilities associated with the rules in a slightly different form with respect to the type of functions considered in Definition 1: only two multisets instead of four are used by the function ϕ. This is because we are restricted to rules of the forms $u\,[\,] \rightarrow [\, v\,]$, $[\, v\,] \rightarrow u[\,]$, $[\, v\,] \rightarrow [\, v'\,]$ containing only one multiset on the left side. However, our approach could be easily generalized to the case of rules of the form $u\,[\, v\,] \rightarrow u'\,[\, v'\,]$ with $u, v \neq \lambda$ by considering a function ϕ' such that

$$\phi'(k, u, v, w_{out}, w_{in}) = \phi(k, u, w_{out}) \cdot \phi(1, v, w_{in})$$

where w_{out}, w_{in} denote the multisets of objects placed respectively inside and outside the membrane where the rule is going to be applied.

Let Π be a PPR, and let C be a configuration of Π. For all $1 \leq i \leq n$, we define the multiset $O_i \in V^*$ as being the multiset of objects such that $|O_i|_a = |C|_{a_i}$, for all $a \in V$ (i.e., O_i is the multiset of objects contained inside membrane i in the configuration C).

We associate to membrane i, with $1 \leq i \leq n$, a set TR_i containing all the triples:

- $(t, v_i \rightarrow u_j, p_t)$, with $\langle [\, v \,] \rightarrow u[\,], k, \phi \rangle_{l_i}$, l_i the label of membrane i, $j = upper(\mu, i)$, and $p_t = \phi(k, v, O_i)$;
- $(t', v_i \rightarrow v'_i, p_{t'})$, with $\langle [\, v \,] \rightarrow [\, v' \,], k, \phi \rangle_{l_i}$, l_i the label of membrane i, and $p_{t'} = \phi(k, v, O_i)$;
- $(t'', u_i \rightarrow v_j, p_{t''})$, with $\langle u[\,] \rightarrow [\, v \,], k_h, \phi \rangle_{l_j}$, $i = upper(\mu, j)$, l_j the label of membrane j, and $p_{t''} = \phi(k, u, O_i)$.

Thus, for each membrane i, the set TR_i is supposed to contain all the rules that can be used inside membrane i with these rules having associated a corresponding value of probability. In particular, if a certain rule is not applicable inside membrane i, then the probability of this rule to be applied turns to be equal to 0. Moreover, notice that rules which send a multiset inside a certain membrane are considered as rules to be used inside the surrounding region.

The following algorithm is then defined to select membranes and programs for P systems with parallelism of type $(1, 1)$.

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

1. calculate $a_0 = \sum p_j$, for all $(j, r, p_j) \in TR_i$;
2. generate two random numbers r_1 and r_2 uniformly distributed over the unit interval $(0, 1)$;
3. calculate the waiting time for the next reaction as $\tau_i = \dfrac{1}{a_0}\, ln(\dfrac{1}{r_1})$;
4. take the index j, of the program such that $\displaystyle\sum_{k=1}^{j-1} p_k < r_2 a_0 \leq \sum_{k=1}^{j} p_k$;
5. return the triple (τ_i, j, i).

Notice that the larger the real constant associated with a rule and the number of occurrences of the objects placed on the left-side of the rule inside a membrane are, the greater the chance that the 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:

- **Initialization**
  - set time of the simulation $t = 0$;
  - for each membrane i in μ compute a triple (τ_i, j, i) by using the procedure described above; construct a list containing all such triples;
  - sort the list of triple (τ_i, j, i) according to τ_i;

- **Iteration**
  - extract the first triple, (τ_m, j, m) from the list;
  - set time of the simulation $t = t + \tau_m$;
  - update the waiting time for the rest of the triples in the list by subtracting τ_m;
  - apply the rule contained in the program j only once changing the number of objects in the membranes affected by the application of the rule;
  - for each membrane m' affected by the application of the rule remove the corresponding triple $(\tau'_{m'}, j', m')$ from the list;
  - for each membrane m' affected by the application of the rule j re-run the Gillespie algorithm for the new context in m' to obtain $(\tau''_{m'}, j'', m')$, the next program j'', to be used inside membrane m' and its waiting time $\tau''_{m'}$;
  - add the new triples $(\tau''_{m'}, r'', m')$ to the list and sort this list according to each waiting time and iterate the process.
- **Termination**
  - Terminate simulation when time of the simulation t reaches or exceeds a preset maximal time of simulation, or no more rules can be applied to the objects left inside the membranes.

Therefore, in this approach, it is the waiting time computed by the Gillespie's algorithm to be used to select the membrane which is allowed to evolve in the next step of computation. Specifically, in each step, the membrane associated to the rule with the same minimal waiting time is selected to evolve by means of this rule. If there are more than one rule with the same waiting time, then we assume one of them to be randomly selected to be used in the next step.

Moreover, since the application of a rule can affect more than one membrane at the same time (e.g., some objects may be moved from one place to another), we need to reconsider a new rule for each one of these membranes by taking into account the new distribution of objects inside them.

Remark 5. The use of a variable time-unit for each step does not affect the semantics of our model; in each step, a single rule at a time is applied inside a specific membrane. This means the behavior of the systems is still synchronous although each application of a rule has associated a different time-unit. In fact, the waiting time is mainly used as a parameter necessary to determine the rule to be applied in the next step of computation.

Remark 6. The current algorithm brings some improvements with respect to the notion of step introduced in Definition 7. In fact, in the iteration phase, we need not to recompute all the probabilities associated with each program applicable inside each membrane, but we can do that only for those membranes which are actually affected by the last application of a program. That is so because the value of the probabilities associated with the other rules remain unchanged.

Remark 7. The use of the waiting time parameters leads to selecting a membrane using the minimum waiting time principle. Getting rid of this parameter will lead

to a variant of this algorithm that is associated to an $(n, 1)-$parallel behavior of the system, where n is the total number of membranes. Indeed, in this case there is no way to distinguish between membranes and all of them will be selected.

5 A Case-Study: Bacterial Quorum Sensing

We present an application of membrane systems to the modeling of quorum sensing in bacteria (QS, for short).

The QS mechanism is a communication strategy based on diffusible signals which kick-in under high cellular density. Bacteria use this mechanism to obtain a population-wide coordination of infection, invasion, and evasion of a host's defence. We refer to [13], [14], [28] for further details about the biology of QS. Moreover, a comprehensive bibliography of QS-related research can be found at the web page [30] maintained by the Nottingham Quorum Sensing Group.

QS bacteria produce and release chemical signal molecules, called *autoinducers*, whose external concentration increases as a function of increasing cell-population density. Bacteria detect the accumulation of a minimal threshold stimulatory concentration of these autoinducers and alter their gene expression, and therefore their behavior in response to the variation of the concentration of autoinducers. Using these signal-response systems, bacteria synchronize particular behaviors on a population-wide scale and thus function as multicellular organisms.

The first described quorum-sensing system is that of the bioluminescent marine bacterium *Vibrio fischeri*, and it is considered the basic paradigm for quorum sensing in most (gram-negative) bacteria [18]. *Vibrio fischeri* colonize the light organ of the Hawaiian squid *Euprymna scolopes*. In this organ, the bacteria grow to high cell density and induce the expression of genes for bioluminescence. The squid uses the light provided by the bacteria for counter-illumination to mask its shadow and avoid predation. The bacteria benefit because the light organ is rich in nutrients and allow proliferation in numbers unachievable in seawater. Two proteins, named LuxI and LuxR, control the expression of the luciferase operon (luxICDABE) required for light production. LuxI is the autoinducer synthase, which produces the autoinducer 3OC6-homoserine lactone (OHHL, for short), and LuxR acts as a receptor for these autoinducers. OHHL freely diffuses in and out of the cell and increases in concentration in correspondence of the increasing of the cell density. When this concentration reaches a critical threshold, OHHL binds to LuxR and this complex activates the transcription of the operon encoding luciferase. As well as this, the LuxR-OHHL complex also induces the expression of luxI because it is encoded in the luciferase operon. This regulatory configuration floods the environment with the signal. This creates a positive feedback loop that causes the entire population to switch into "quorum-sensing mode", and produce light; in this case, it is also said that the population is *quorated*.

QS systems have then been identified in other bacterial populations, for instance, *Pseudomonas aeruginosa*, *Vibrio harveyi*, and *Bacillus subtillis*, where

the existence of quorum-sensing networks relying on multiple signalling circuits acting synergistically has also been observed.

5.1 A P System Model of QS

A P system model for the QS system of *Vibrio fischeri* is here defined where a colony of such bacteria is represented by means of a membrane structure consisting of a number of elementary membranes, each one of them representing a bacterium, included in an unique membrane (the skin) representing a common shared environment. In particular, each membrane will contain a set of programs modeling the QS regulatory circuits responsible for the production of light.

To this aim, we use: the symbol $OHHL$ to denote the *autoinducer*, the symbol $LuxR$ to denote the receptor for the autoinducer $OHHL$, the symbol $LuxR\text{-}OHHL$ to denote the complex formed by the binding of the autoinducer $OHHL$ to the receptor $LuxR$, the symbol $LuxBox$ to denote the luciferase operon in its down-regulated state (i.e., when it is not active for the production of light), and the symbol $LuxBox\text{-}LuxR\text{-}OHHL$ to denote the luciferase operon in its up-regulated state (i.e., when it is active for the production of light). Then, we define the following P signature for QS in *Vibrio fischeri*.

$$BS(A) = (V, K, \mathbb{R}_0^+, \phi, A, P)$$

where $A = \{k_1, k_2, k_4, k_3, k_5, k_6, k_7, k_8, k_9, k_{10}, k_{11}, k_{12}, k_{13}, k_{14}\}$ is a set of real constants, $V = \{OHHL, LuxR, LuxR\text{-}OHHL, LuxBox, LuxBox\text{-}LuxR\text{-}OHHL\}$, $K = \{e, b\}$, ϕ is the function given by expression 2, and P is finite set containing all the following programs:

- $\langle [\, LuxBox \,] \to [\, LuxBox\, LuxR \,], k_1, \phi \rangle_b$,
  $\langle [\, LuxBox \,] \to [\, LuxBox\, OHHL \,], k_2, \phi \rangle_b$
  (at low cell density the autoinducer $OHHL$ and the receptor $LuxR$ are produced at a basal rate);
- $\langle [\, OHHL\, LuxR \,] \to [\, LuxR\text{-}OHHL \,], k_3, \phi \rangle_b$,
  $\langle [\, LuxR\text{-}OHHL \,] \to [\, OHHL\, LuxR \,], k_4, \phi \rangle_b$
  (the autoinducer $OHHL$ and the receptor $LuxR$ bind together to form the complex $LuxR\text{-}OHHL$ which, in turn, dissociates in its components);
- $\langle [\, LuxR\text{-}OHHL\, LuxBox \,] \to [\, LuxBox\text{-}LuxR\text{-}OHHL \,], k_5, \phi \rangle_b$,
  $\langle [\, LuxBox\text{-}LuxR\text{-}OHHL \,] \to [\, LuxR\text{-}OHHL\, LuxBox \,], k_6, \phi \rangle_b$
  (the complex $LuxR\text{-}OHHL$ binds to the region of DNA responsible for the production of light; such a complex can also dissociate from that region by returning the luciferase operon to a down-regulated state);
- $\langle [\, LuxBox\text{-}LuxR\text{-}OHHL \,] \to [\, LuxBox\text{-}LuxR\text{-}OHHL\, OHHL \,], k_7, \phi \rangle_b$,
  $\langle [\, LuxBox\text{-}LuxR\text{-}OHHL \,] \to [\, LuxBox\text{-}LuxR\text{-}OHHL\, LuxR \,], k_8, \phi \rangle_b$
  (the binding of the complex to the corresponding region of DNA produces an increase in the production of the autoinducer $OHHL$ and in the production of the receptor $LuxR$);

– $\langle [\, OHHL \,] \rightarrow OHHL\, [\,], k_9, \phi \rangle_b$
(the autoinducer $OHHL$ freely diffuses outside the bacterium and accumulates in the environment);
– $\langle [\, OHHL \,] \rightarrow [\,], k_{10}, \phi \rangle_b$,
$\langle [\, LuxR \,] \rightarrow [\,], k_{11}, \phi \rangle_b$,
$\langle [\, LuxR\text{-}OHHL \,] \rightarrow [\,], k_{11}, \phi \rangle_b$
(the autoinducer $OHHL$, the receptor $LuxR$ and the complex $LuxR$-$OHHL$ undergo a process of degradation inside the bacterium);
– $\langle OHHL\, [\,] \rightarrow [\, OHHL \,], k_{12}, \phi \rangle_b$
(the autoinducer $OHHL$ diffuse back from the environment into the bacterium);
– $\langle [\, OHHL \,] \rightarrow [\,], k_{13}, \phi \rangle_e$
(the autoinducer $OHHL$ is degraded in the environment).

Thus, we have identified 14 rules which model the main transformations involved in the QS system of *Vibrio fischeri*. Notice that the signature BS is parametric with respect to the particular constants associated with the rules.

Next, we define a parametric PPR system $\Pi(n, A)$ to represent a colony of $n \geq 1$ bacteria interacting by means of the QS system described by the aforementioned rules. Specifically, we have

$$\Pi(n, A) = (BS(A), \mu(n), M_1, \ldots, M_n, M_{n+1})$$

where:

– $A = \{k_1, k_2, k_4, k_3, k_5, k_6, k_7, k_8, k_9, k_{10}, k_{11}, k_{12}, k_{13}, k_{14}\}$;
– $\mu(n) = [_{n+1}\, [_1\,]_1 \ldots [_n\,]_n\,]_{n+1}$;
– $M_i = (LuxBox, b)$, for all $1 \leq i \leq n$;
– $M_{n+1} = (\lambda, e)$.

Thus, in the initial configuration, we assume all the bacteria in the colony to contain only one occurrence of the object $LuxBox$ representing the portion of DNA responsible for the production of the autoinducer $OHHL$ and the receptor $LuxR$; the environment is instead supposed to be initially empty. Moreover, notice that, by having the notion of P specification, we can represent an arbitrary large colony in a very compact way by avoiding repeating the same set of programs for every membrane in the system.

Simulation results have been presented under different formalisms and show the same behavior of the colony.

6 Discussion

As we have seen, there is a growing interest in using P systems for modeling biological systems. This often requires the introduction into the membrane system model of some extra features especially when the quantitative aspects characterizing the "reality" of the biological phenomenon to be modeled are considered.

Here we have addressed these issues by specifically introducing the notion of a program consisting of a rule with a finite set of attributes and a function from a given set (Definition 1). We have shown how attributes and functions can be used to define P system models for bio-chemical systems consisting of a number of bio-chemical reactions distributed across various compartments of the system. A precise strategy for the application of the rules has also been defined for this class of P systems which makes possible to associate a stochastic behavior to such P systems. Our approach is based on the well-known Gillespie's algorithm and it is developed alongside the work done in [2], [21], [22] where alternative strategies for the application of the rules are defined.

Acknowledgements

The first author's research is supported by NWO, Organisation for Scientific Research of The Netherlands, project 635.100.006 "VIEWS".

The second and fourth authors are supported by Ministerio de Ciencia y Tecnología of Spain, by *Plan Nacional de I+D+I* (TIN2005-09345-C04-01), cofinanced by FEDER funds, by Junta de Andalucía, by project of Excellence TIC 581, and by a FPU fellowship from the Ministerio de Ciencia y Tecnología of Spain.

References

1. Bernardini, F., Gheorghe, M. (2004). Population P Systems. *J. UCS* **10**,(5), 509–539.
2. Bianco, L., Fontana, F., Manca, V. (2006). P Systems with Reaction Maps. *International Journal of Foundations of Computer Science*, **17**, (1), 27–48.
3. Calder, M., Vyshemirsky, V., Gilbert, D, Orton, R (2006). Analysis of Signalling Pathways using Continuous Time Markov Chains. *Transactions on Computational Systems Biology*, to appear.
4. Cardelli, L., Philips, A.(2004). A Correct Abstract Machine for the Stochastic Pi-calculus. *Electronical Notes in Theoretical Computer Science*, to appear.
5. Ciobanu G., Andrei, O., Lucanu D. (2006). Structural Operational Semantics of P Systems, *WMC6, LNCS* **3850**, 31–48.
6. Collado-Vides, J. (1992). Grammatical Models of the Regulation of Gene Expression, *Proc. of National Academy of Science*, **89**, 9405–9409.
7. Dang, Z., Ibarra, O.H., Li, C., Gaoyan, X. (2006). Decidability of Model-Checking P Systems. *Journal of Automata, Languages and Combinatorics,*, to appear.
8. Dang, Z., Ibarra, O.H. (2005). On One-membrane P systems Operating in Sequential Mode. *Int. J. Found. Comput. Sci.* **16**, (5), 867–881.
9. Duan, Z., Holcombe, M., Bell, A. (2000). A Logic for Biological System, *Biosystems*, **53**, 93–155.
10. Freund, R. (2004). Asynchronous P systems and P systems working in Sequential Mode, *WMC5, LNCS* **3365**, 36–62.
11. Gillespie, D.T. (1976). A General Method for Numerically Simulating the Stochastic Time Evolution of Coupled Chemical Reactions. *J Comput Physics*, **22**, 403–434.

12. Gillespie, D.T. (1977). Exact Stochastic Simulation of Coupled Chemical Reactions. *The Journal of Physical Chemistry*, **81**, (25), 2340–2361.
13. Hardie, K.R., Williams, P., Winzer, K.(2002). Bacterial cell-to-cell communication: sorry, can't talk now, gone to lunch. *Current Opinion in Microbiology*, **5**, 216–222.
14. 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. *J. Mol. Biol.* **296**, 1127–1137.
15. Kam, N., Cohen, I.R., Harel, D. (2001). The Immune System as a Reactive Systems: Modelling T Cell Activation with Statecharts, *The Weizmann Institute of Science*, Israel.
16. Matsuno, H., Doi, A., Nagasaki, M., Miyano, S. Hybrid (2000). Petri Net Representation of Gene Regulatory Network, *Pacific Symposium on Biocompting*, World Scientific, 338–349.
17. Meng, T.C., Somani S., Dhar, P. (2004). Modelling and Simulation of Biological Systems with Stochasticity. *In Silico Biology*, **4**, (0024), 137–158.
18. Nealson, K.H., Hastings, J.W. (1979). Bacterial Bioluminescence: Its Control and Ecological Significance, *Microbiology Review*, **43**, 496–518.
19. Păun, Gh. (2000). Computing with Membranes, *Journal of Computer and System Sciences*, **61**, (1), 108 – 143.
20. Păun, Gh. (2002). *Membrane Computing. An Introduction*, Springer, Berlin.
21. Pérez-Jiménez, M.J., Romero-Campero, F.J. (2006). P Systems, a New Computationl Modelling Tool for Systems Biology, *Transactions on Computational Systems Biology VI, LNBI*, **4220**, 176-197.
22. Pescini, D., Besozzi, D., Mauri, G., Zandron, C. (2006). Dynamical probabilistic P systems, *International Journal of Foundations of Computer Science*, **17**, (1), 183–195.
23. 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.
24. Regev, A., Shapiro, E. (2004) The π-calculus as an abstraction for biomolecular systems. In Gabriel Ciobanu and Grzegorz Rozenberg, editors, *Modelling in Molecular Biology*, Springer, 219–266.
25. Segel, I.H. (1976). *Biochemical Calculations: How to Solve Mathematical Problems in General Biochemistry*, John Wiley and Sons, 2nd edition.
26. Till, J.E., McCulloch, F. Siminovitch, L. (1964). A Stochastic Model of Stem Cell Proliferation based on the Growth of Spleen Colony-Forming Cells, *Proc. National Academy of Science USA*, **51**, 117–128.
27. Walker, D., Holcombe, M., Southgate, J., McNeil, S., Smalwood, R. (2004). The Epitheliome: Agent-Based Modelling of The Social Behaviour of Cells, *Biosystems*, **76**, (1–3), 89–100.
28. Waters, C.M., Bassler, B.L. (2005). Quorum Sensing: Cell-to-Cell Communication in Bacteria. *Annu. Rev. Cell. Dev. Biol.* **21**, 319–346.
29. The P Systems Web Site: `http://psystems.disco.unimib.it`
30. Nottingham Quorum Sensing Web Site `http://www.nottingham.ac.uk/quorum/`