

Modular Assembly of Cell Systems Biology Models Using P Systems

Francisco José Romero-Campero¹, Jamie Twycross²,
Malcolm Bennett², Miguel Cámara³, and Natalio Krasnogor⁴

¹ Automated Scheduling, Optimisation and Planning Research Group
School of Computer Science, Jubilee Campus, University of Nottingham

² Centre for Plant Integrative Biology
Sutton Bonington Campus, University of Nottingham

³ Centre for Biomolecular Sciences, University Park, University of Nottingham

⁴ Automated Scheduling, Optimisation and Planning Research Group
School of Computer Science, Jubilee Campus, University of Nottingham

Contact: Natalio.Krasnogor@Nottingham.ac.uk

Abstract. In this paper we propose an extension of the modelling framework based on P systems which explicitly includes modularity. Modularisation in cellular systems can be produced by chemical specificity and spatial localisation in different compartments. Both these features can be easily specified and analysed in P systems using sets of rewriting rules to describe chemical specificity and membranes to represent spatial localisation. Our methodology enables the assembly of cell systems biology models by combining modules which represents functional subsystems. A case study consisting of a bacterial colony system is presented to illustrate our approach.

1 Introduction

Systems biology seeks to achieve, for a variety of biological systems - particularly cellular ones-, a deeper understanding of what is currently available. Molecular (micro)biology has focused mainly on understanding the component features of biological systems but has not attempted to obtain an integrative view that covers all the levels of biological organisation simultaneously. Systems biology thus attempt to cover this gap and focuses on the nature of the interactions and links that connect cellular (sub)systems and the functional states of the networks that result from the assembly of such links [1, 17]. Due to the complexity of these connections and to the vast amount of -often very noisy- data produced by experimentalists, computational/mathematical modelling, simulation and analysis are essential techniques in this field. A deeper knowledge of the organisation and functioning of cellular processes will allow us not only to understand what is already available in nature but, perhaps more importantly, to engineer our own cellular systems exhibiting specific target behaviours or producing predetermined outputs. Thus better computational modelling techniques will also have an impact on the *synthetic biology* [2] arena.

The macroscopic continuous and deterministic approach based on ordinary differential equations (ODEs) is the most widely used methodology in cellular modelling within systems and synthetic biology. Although ODEs have been successfully applied in different systems there are two key assumptions in this approach, namely continuity and determinism, that are not always fulfilled. There are many systems where the number of particles of the reacting species are low and the reactions involved are slow. In these systems the previous assumptions are invalid and mesoscopic, discrete and stochastic approaches are more suitable [17]. These latter approaches have been implemented in different computational frameworks for example Petri nets [8] and process algebra [14]. Although

these computational frameworks capture some of the information regarding cellular systems and their components, none fully integrates the dynamics and structural details of the systems in an understandable and relevant way. One of the main points which is neglected in most computational frameworks so far is the key role played by membranes and compartmentalisation in the structure and functioning of living cells.

In this paper we use *P systems*, which unifies studies of compartment, metabolism and information processing. A *P system* [13] consists of a cell-like membrane structure, with compartments containing multisets of objects encoding molecules that evolve according to given rules, which in turn represent biochemical interactions. Rules are applied according to an extension of Gillespie's well known Stochastic Simulation Algorithm (SSA) [7] to the multicompartmental structure of *P system* models [10].

Biological functions are the results of the interactions between modules made up of many molecular species. By definition, a module is a discrete entity which performs a specific function that is to some extent separable from those of other modules. Modules can be isolated or connected to each other. Isolation allows a cell to carry out many diverse processes without interactions among these processes that would harm the cell, whereas connectivity allows higher level functions to be built by assembling different modules. Modules quasi-separation originates from the chemical specificity between the molecular species and/or from the spatial localisation in different compartments [9].

Chemical specificity and spatial localisation can be easily specified and analysed in *P systems* using rewriting rules to describe chemical specificity and membranes to represent spatial localisation. In this paper we present a computational framework based on *P systems* which will help elucidate how the modularisation and pattern of connections among functional modules results in specific behaviours such as amplification, adaption, robustness, insulation, error correction, coincidence detection, and so on.

The paper is organised as follows. The next section discusses stochastic *P systems* in cellular modelling. Section 3 presents a brief description of the simulation algorithm used in our approach. Modularisation in *P systems* is introduced in Section 4. A bacterial colony system is studied in section 5 to illustrate our methodology. Finally some conclusions and future work are discussed.

2 Stochastic *P Systems* in Cellular Modelling

Among the various executable biology methodologies [18], *P systems* [12] is the only one that recognises explicitly the role of biological compartments while implicitly representing metabolism and information processing through rules and (multi)sets/strings rewriting. The cell wall demarcates the boundaries of the cell's "self" while internal membranes represent organelles and other internal organisation. Far from being passive frontiers between the external environment and the living cell, the membrane is, in itself, a complex and dynamic object that actively participates in the internal dynamics of the cell and mediates its interaction with both the environment and other cells. Membrane computing formalises and abstracts these features of living cells by introducing the notion of *P systems*, also called *membrane systems*.

P systems, in their original motivation, were not intended to provide a comprehensive and accurate model of the living cell, but rather to explore the computational nature of various phenomena in cell biology (e.g. [19]). More recently, *P systems* have been used as *executable biology* tools. Examples of such models are: oscillatory systems [6], signal transduction [10], gene regulation control [16], quorum sensing [4, 15] and metapopulations [11].

In what follows we present the main definition used in this paper.

Definition 1 (Individual P system).

An individual P system is a construct $\Pi = (\Sigma, L, \mu, M_1, \dots, M_n, R_{l_1}, \dots, R_{l_m})$ where:

- Σ is a finite alphabet of symbols representing individual molecular entities;
- $L = \{l_1, \dots, l_m\}$ is a finite alphabet of symbols representing labels for the compartments and identifying compartment types⁵;
- μ is a membrane structure containing $n \geq 1$ membranes defining compartments arranged in a hierarchical manner. Each membrane is identified in a one to one manner with values in $\{1, \dots, n\}$ and is given a label from L which determines its type. The membrane structure is represented formally, as a rooted tree, where the nodes are called membranes, and the relationship of a membrane being inside another one is represented by the relationship of the node being the descendent of another one. The outermost membrane associated with the root is normally referred to as skin membrane. Alternatively, Venn diagram or pairs of matching square brackets representations can be used (see Figure 1);

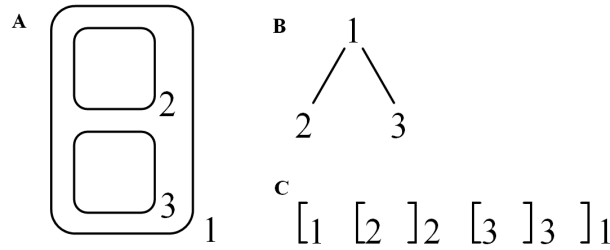


Fig. 1. Three possible representations of the same membrane structure consisting of three membranes.

- $M_i = (l_i, w_i)$, for each $1 \leq i \leq n$, is the initial configuration of membrane i with $l_i \in L$, the label of this membrane, $w_i \in \Sigma^*$ a finite multiset of objects from Σ . A multiset of objects, obj is represented as $obj = o_1 + o_2 + \dots + o_m$ with $o_1, \dots, o_m \in \Sigma$.
- $R_{l_t} = \{r_1^{l_t}, \dots, r_{k_{l_t}}^{l_t}\}$, for each $1 \leq t \leq m$, is a finite set of multiset rewriting rules associated with compartments of the type represented by the label $l_t \in L$ and of the following form:

$$r_j^{l_t} : obj_1 [obj_2]_l \xrightarrow{c_j^{l_t}} obj'_1 [obj'_2]_l \quad (1)$$

where $obj_1, obj_2, obj'_1, obj'_2$ are finite multisets of objects with alphabet Σ and l a label from L . Rules, operating on both sides of the membrane, rewrite the multiset. These multisets could be concurrently replaced with a multiset obj'_1 and a multiset obj'_2 , respectively. Note that a constant $c_j^{l_t}$ is associated specifically with each rule. This stochastic constant is used to compute the propensity of the rule, that is, the probability that the rule will be applied in the next infinitesimal time step dt .

3 Multicompartment Gillespie algorithm

Non-determinism and maximal parallelism in the original P system formulation [12] give rise to an incorrect rate of rule application that do not represents properly the time evolution of cellular

⁵ Compartments with the same label will be considered of the same type and thus the same set of rules will be associated with them.

systems. These two problems are interdependent and, as with other computational approaches [8, 14], must be addressed when devising a relevant modelling framework for cellular systems.

In the field of membrane computing, the discrete aspect of the different components, as well as the distributed and compartmentalised character of the structure where computation takes place, are fundamental. This is not the case with the non-deterministic and maximal parallel semantics as have been studied in different variants [5]. In this section the original approach will be replaced with a strategy, termed the Multicompartment Gillespie Algorithm (MGA), based on Gillespie's theory of stochastic kinetics [7].

To provide P systems with a stochastic extension a constant c , used to determine the probability and hence time-lag before applying the rule, is associated with each rule. This constant depends only on the properties of the molecules involved in the reaction encoded in the rule and on other physical parameters of the system such as temperature, pressure, etc. The starting point consists of treating each compartment, delimited by a membrane, as a well mixed and fixed volume where the classical *Gillespie algorithm* is applied. Then the compartments defined by membranes are notionally ordered in a priority queue according to the time lapse required until the rules are scheduled to be applied by the Gillespie algorithm (termed the waiting time). The first rule to be applied in the entire system occurs in the compartment on top of the priority queue. Depending on the type of rule that has been applied, the state of a single compartment or of two compartments is changed. Therefore the waiting time and rule to be applied in these compartments must be recalculated. The algorithm stops when a prefixed simulation time is reached or no more rules can be applied.

The pseudocode for the Optimised Multicompartment Gillespie Algorithm (oMGA) is given in Algorithm 1. The algorithm employs the minimal stopping condition, halting when no more rules can be applied (line 9). In practice, additional stopping conditions, such as execution for a certain length of time, or until a certain number of rules have been applied, are also desirable. It is trivial to incorporate such stopping conditions into line 9 of the algorithm. In order to efficiently implement this stopping condition, the waiting time is initially set to a large value (indicated as ∞ in lines 2, 9 and 11). This value needs to be greater than any realistic waiting time, and can in practice be set to a large number such as the `DBL_MAX` constant in C/C++.

Two functions, `ExecuteRule` and `GillespieDirectMethod` and called within the oMGA. The `ExecuteRule` function takes the index of a rule and the index of the compartment which contains this rule and applies the rule in the specified compartment in the standard way [7]. That is, the state vector \mathbf{x} giving the current number of molecules for each species in the compartment is updated by adding the state change vector \mathbf{v} for the specified rule. The state change vector defines the changes in molecular populations caused by one application of a given rule.

The second function, `GillespieDirectMethod`, performs the waiting time and rules selection steps of the Gillespie Direct Method [7] for the specified compartment. Pseudocode for this function is given in Procedure 2. Within this function, the `rand()` function returns a uniformly distributed random variable.

Formally, the oMGA is equivalent to the MGA, but employs a more computationally efficient strategy to determine the next rule to apply. The next rule to apply is the rule with the smallest waiting time τ , the time to the next application. In order to determine this rule, at the end of every iteration of the main loop the original MGA sorts the list of waiting times in ascending numerical order. The oMGA eliminates this sorting procedure, and instead keeps track of the smallest application time as execution progresses through the main loop (lines 14-16, 20-22, 26-28). This procedure makes less comparisons of waiting times (a maximum of $n + 2$, where n is the

Algorithm 1: Optimised Multicompartment Gillespie Algorithm

```
begin
  // preprocess
  1  set simulation time  $t \leftarrow 0.0$ 
  2  set time to next rule application  $\tau_0 \leftarrow \infty$ 
  3  for  $i \leftarrow 1$  to number of compartments do
  4    // perform Gillespie Direct Method for each compartment
     $(\tau_i, \mu_i) \leftarrow \text{GillespieDirectMethod}(i)$ 
    // keep smallest waiting time
  5    if  $\tau_i > 0.0$  and  $\tau_i < \tau_0$  then
  6      |  $(\tau_0, \mu_0, i_0) \leftarrow (\tau_i, \mu_i, i)$ 
  7    end
  8  end
  // main loop
  9  while  $\tau_0 \neq \infty$  do
  10   // advance simulation time
     $t \leftarrow t + \tau_0$ 
  11   // reduce time to next rule application
    set time to next rule application  $\tau_n \leftarrow \infty$ 
  12   for  $i \leftarrow 1$  to number of compartments do
  13     |  $\tau_i \leftarrow \tau_i - \tau_0$ 
    // keep smallest waiting time
  14     if  $\tau_i > 0.0$  and  $\tau_i < \tau_n$  then
  15       |  $(\tau_n, \mu_n, i_n) \leftarrow (\tau_i, \mu_i, i)$ 
  16     end
  17   end
  18   // apply rule  $\mu_0$  in compartment  $i_0$ 
    ExecuteRule( $\mu_0, i_0$ )
  19   // perform Gillespie Direct Method for compartment  $i_0$ 
     $(\tau_{i_0}, \mu_{i_0}) \leftarrow \text{GillespieDirectMethod}(i_0)$ 
  20   // keep smallest waiting time
    if  $\tau_{i_0} > 0.0$  and  $\tau_{i_0} < \tau_n$  then
  21     |  $(\tau_n, \mu_n, i_n) \leftarrow (\tau_{i_0}, \mu_{i_0}, i_0)$ 
  22   end
  23   // update target compartment of rule
    if rule  $\mu_0$  in compartment  $i_0$  is a communication rule then
  24     | set  $i$  to index of target compartment of rule  $\mu_0$ 
    // perform Gillespie Direct Method for compartment  $i$ 
  25     |  $(\tau_i, \mu_i) \leftarrow \text{GillespieDirectMethod}(i)$ 
    // keep smallest waiting time
  26     | if  $\tau_i > 0.0$  and  $\tau_i < \tau_n$  then
  27       | |  $(\tau_n, \mu_n, i_n) \leftarrow (\tau_i, \mu_i, i)$ 
  28     | end
  29   end
  30   // set next rule
     $(\tau_0, \mu_0, i_0) \leftarrow (\tau_n, \mu_n, i_n)$ 
  31 end
end
```

number of compartments) than sorting the waiting times list (a maximum of $2n$ comparisons), and so is more efficient, especially for large numbers of compartments.

4 Modularisation in P Systems Models

Modularisation in cellular systems can be produced by chemical specificity and spatial location in different compartments. In this section we present a methodology based on P systems which allows us to assemble cell systems biology models in a modular way. On the one hand, modularisation due to chemical specificity will be described using sets of rewriting rules which specify the interactions

Procedure GillespieDirectMethod(j)

Input: j = compartment index**Result:** (τ_0, μ_0) = (time to next rule application, index of next rule)

```
1 begin
  // calculate rule propensities and total propensity
2  set total propensity  $a_0 \leftarrow 0$ 
3  for  $i \leftarrow 1$  to number of rules in compartment  $j$  do
4     $a_i \leftarrow c_i \cdot x_i$  // calculate propensity  $a_i$  of rule  $i$ 
5     $a_0 \leftarrow a_0 + a_i$  // add rule propensity to total propensity
6  end
  // return if no rule can be applied
7  if  $a_0 = 0.0$  then
8    return (0.0, 0)
9  end
  // calculate time to next rule application
10 set time to next rule application  $\tau_0 = \frac{-\log(\text{rand}())}{a_0}$ 
  // calculate index of next rule
11 set target propensity  $a_m \leftarrow \text{rand}() \cdot a_0$ 
12 for  $i \leftarrow 1$  to number of rules in compartment  $j$  do
13    $a_m \leftarrow a_m - a_i$  // reduce target propensity by rule  $i$  propensity
14   if  $a_m \leq 0.0$  then
15     exit for loop
16   end
17 end
18 set index of next rule  $\mu_0 \leftarrow i$ 
19 return  $(\tau_0, \mu_0)$ 
20 end
```

between certain molecular species represented by individual objects. On the other hand, modularisation due to spatial localisation will be described using membranes delimiting compartments which can be arranged in a hierarchical tree-like structure as in definition 1, or in a lattice where each point is associated with a P system as discussed latter in this section.

Modularisation Due to Chemical Specificity:

Molecular interactions in P systems are specified as rewriting rules. As already stated, a cellular module resulting from chemical specificity consists of certain interactions between specific molecular species, and performs a particular function. Therefore, the following definition of a P system module is introduced.

Definition 2 (P system Module).

A P system module is a set of rules of the form in (1) representing molecular interactions which occur repetitively in many cell systems. A module is identified with a name and three sets of variables, V , C and L . V represents variables that can be instantiated using objects describing molecular species. C represents the stochastic constants associated with each rule. L specifies the labels of the compartments involved in the rules. A module, M , with variables V , constants C and labels L will be written as $M(V, C, L)$.

More complex modules can be constructed from simple modules by applying set unions. In this respect, the set of rules associated with a membrane in a P system model can be specified in a modular way as the set union of several P system modules.

In order to illustrate this definition, we introduce some P system modules describing the most common molecular interactions. These modules will be used in the section below describing our case study.

- Complexation module: This module describes the process by which two molecules X and Y collide and stick together forming a complex Z inside a compartment with label l . This interaction is reversible and Z can dissociate into its components X and Y . The propensities of these rules are computed using the stochastic constants c_1 and c_2 .

$$Com(\{X, Y, Z\}, \{c_1, c_2\}, \{l\}) = \left\{ [X + Y]_l \xrightarrow{c_1} [Z]_l, [Z]_l \xrightarrow{c_2} [X + Y]_l \right\} \quad (2)$$

- Enzymatic module: Some chemical reactions are assisted by certain proteins called enzymes located in specific compartments l . In an enzymatic module the enzyme X binds reversibly to the substrate Y . Once the enzyme is bound to the substrate it can produce a change in it and then the enzyme-substrate complex dissociates releasing the unchanged enzyme X and the product W . The propensities of the rules in this module are computed using the stochastic constants c_1, c_2 and c_3 .

$$Enz(\{X, Y, Z, W\}, \{c_1, c_2, c_3\}, \{l\}) = \left\{ \begin{array}{l} [X + Y]_l \xrightarrow{c_1} [Z]_l \\ [Z]_l \xrightarrow{c_2} [X + Y]_l \\ [Z]_l \xrightarrow{c_3} [X + W]_l \end{array} \right\} \quad (3)$$

When the substrate is available in such quantities that it does not affect the dynamics of the reactions this module can be simplified by considering the enzyme X as producing a product Y :

$$EnzS(\{X, Y\}, \{c\}, \{l\}) = \{[X]_l \xrightarrow{c} [X + Y]_l\} \quad (4)$$

- Degradation module: The cell machinery degrades molecules in order to control their activity. Furthermore, the number of molecules is also reduced by dilution due to cell growth and division. These processes are specified in the following module where X represents the molecular specie being degraded, c is the stochastic constant and l is the label of the compartment.

$$Deg(\{X\}, \{c\}, \{l\}) = \{[X]_l \xrightarrow{c} []_l\} \quad (5)$$

- Passive diffusion module: Certain molecules, X can readily move in and out of specific compartments l . In the diffusion module, in and out are described by rewriting rules that move the object X from the outside of compartment l to the inside or vice versa. The stochastic constants c_1 and c_2 are used to compute the propensities of the corresponding rules.

$$Diff(\{X\}, \{c_1, c_2\}, \{l\}) = \left\{ X []_l \xrightarrow{c_1} [X]_l, [X]_l \xrightarrow{c_2} X []_l \right\} \quad (6)$$

- Unregulated gene expression module: Some genes are expressed constitutively and independently of transcription factors. Such a gene, G , is transcribed continuously at the same rate into its corresponding mRNA, R , which in turn is translated into a protein P . The mRNA and protein can be degraded by the cell machinery. The propensities of these processes are determined by the stochastic constants c_1, c_2, c_3 and c_4 .

$$UnReg(\{G, R, P\}, \{c_1, c_2, c_3, c_4\}, \{l\}) = \left\{ \begin{array}{l} [G]_l \xrightarrow{c_1} [G + R]_l, [R]_l \xrightarrow{c_2} [R + P]_l, \\ [R]_l \xrightarrow{c_3} []_l, [P]_l \xrightarrow{c_4} []_l \end{array} \right\} \quad (7)$$

This module can be simplified under the assumption that the mRNA reaches its steady state much faster than the protein. In this case one can overlook the mRNA and assume that from the gene G the protein P is produced in a single process. For simplicity this assumption is adopted in all the following modules describing gene expression processes.

$$UnRegS(\{G, P\}, \{c_1, c_2\}, \{l\}) = \left\{ [G]_l \xrightarrow{c_1} [G + P]_l, [P]_l \xrightarrow{c_2} []_l \right\} \quad (8)$$

- Positive gene expression module: Unlike the previous case, some genes are only expressed in the presence of specific transcription factors called *activators*. In this case an activator protein Act binds reversibly to the gene G to form the complex $Act.G$, which turns on the production of the

protein P . As in the previous case the protein can be degraded. The propensities of these processes are computed using the stochastic constants c_1, c_2, c_3 and c_4 .

$$Pos(\{Act, G, Act.G, P\}, \{c_1, c_2, c_3, c_4\}, \{l\}) = \left\{ \begin{array}{l} [Act + G]_l \xrightarrow{c_1} [Act.G]_l \\ [Act.G]_l \xrightarrow{c_2} [Act + G]_l \\ [Act.G]_l \xrightarrow{c_3} [Act.G + P]_l, \\ [P]_l \xrightarrow{c_4} []_l \end{array} \right\} \quad (9)$$

• Negative regulated expression: In contrast to the previous case, in negative regulation a repressor protein Rep binds reversibly to the gene G producing the complex $Rep.G$ which does not produce any protein. The propensities of the binding and debinding of the repressor to the gene are computed using the stochastic constants c_1 and c_2 .

$$Neg(\{Rep, G, Rep.G\}, \{c_1, c_2\}, \{l\}) = \left\{ [Rep + G]_l \xrightarrow{c_1} [Rep.G]_l, [Rep.G]_l \xrightarrow{c_2} [Rep + G]_l \right\} \quad (10)$$

Modularisation Due to Spatial Localisation:

The localisation of molecular species in different regions of cellular systems is represented in P systems by placing the objects describing these molecules in specific compartments defined by membranes. To date, there has been no explicit geometric distribution associated with any components of P systems. In this section, we propose the association of geometric information to population P systems [3] through the use of a finite lattice. Using this scheme, different individual P systems can be located at different points on the lattice. These P systems will communicate by sending and receiving objects.

Definition 3 (Lattice Population P system).

A lattice population P system is a construct $\mathcal{P} = (\Sigma, Lat, (\Pi_1, \dots, \Pi_p), Pos, (T_1, \dots, T_p))$ where:

- Σ is a finite alphabet of symbols representing molecular species;
- Lat is a finite lattice in \mathbb{R}^2 or \mathbb{R}^3 ;
- Π_1, \dots, Π_p are individual P systems as introduced in definition 1 sharing the same alphabet Σ ;
- $Pos : Lat \longrightarrow \{\Pi_1, \dots, \Pi_p\}$ is a function that associates a specific P system with each position in the lattice;
- $T_j = \{r_1^j, \dots, r_k^j\}$, for each $1 \leq j \leq p$, is a finite set of translocation rules to be added to the set of rules associated with the skin membrane of each individual P system Π_j . Translocation rules are of the following form:

$$r_i^j : [obj]_l \xrightarrow{\mathbf{v}, c_i^j} []_l \xrightarrow{\mathbf{v}} []_l \xrightarrow{\mathbf{v}} [obj] \quad (11)$$

where $obj \in \Sigma^*$ is a multiset of objects from Σ , \mathbf{v} a vector in \mathbb{R}^2 or \mathbb{R}^3 and c_i^j is the stochastic constant used to compute the propensity of the rule. This type of rules move objects from one P system in the lattice to another. The application of rule r_i^j in the P system Π_j located in position \mathbf{p} , $Pos(\mathbf{p}) = \Pi_j$, moves the objects obj from the skin membrane of Π_j to the P system $\Pi_{j'}$ located in position $\mathbf{p} + \mathbf{v}$, $Pos(\mathbf{p} + \mathbf{v}) = \Pi_{j'}$.

5 Case Study: Quorum Sensing a Bacterial Colony System

In this section our approach is illustrated using a Quorum Sensing (QS), a bacterial colony-level phenotype. QS is a gene regulatory system that depends on cell density. It allows a population of bacterial cells to communicate and thus regulate the coordinated expression of specific genes [21].

QS relies on the synthesis, accumulation and sensing of small diffusible signals molecules. In most QS systems the receptor protein A which senses the signal S is expressed constitutively from the gene $geneA$. The protein A interacts with the signal S to form the complex $A.S$. In contrast to A , the enzyme E responsible for the synthesis of the signal S , is first expressed constitutively from the gene $geneE$ but is then positively regulated by the complex $A.S$. These interactions are represented in the following module obtained by combining the elementary modules introduced before.

$$\begin{aligned}
& QS(\{geneA, geneE, A, E, S, A.S, A.S.geneE\}, \{c_1, \dots, c_{14}\}, \{b\}) = \\
& \left\{ \begin{array}{l}
UnRegS(\{geneA, A\}, \{c_1, c_2\}, \{b\}) \\
UnRegS(\{geneE, E\}, \{c_3, c_4\}, \{b\}) \\
EnzS(\{E, S\}, \{c_5\}, \{b\}) \\
Diff(\{S\}, \{c_6\}, \{b\}) \\
Deg(\{S\}, \{c_7\}, \{b\}) \\
Com(\{A, S, A.S\}, \{c_8, c_9\}, \{l\}) \\
Deg(\{A.S\}, \{c_{10}\}, \{b\}) \\
Pos(\{A.S, geneE, A.S.geneE, E\}, \{c_{11}, c_{12}, c_{13}, c_{14}\}, \{b\})
\end{array} \right. = \left\{ \begin{array}{l}
\begin{cases} [geneA]_l \xrightarrow{c_1} [geneA + A]_l \\ [A]_l \xrightarrow{c_2} []_l \end{cases} \\
\begin{cases} [geneE]_l \xrightarrow{c_3} [geneE + E]_l \\ [E]_l \xrightarrow{c_4} []_l \end{cases} \\
[E]_l \xrightarrow{c_5} [E + S]_l \\
[E]_l \xrightarrow{c_6} [E + S]_l \\
[S]_l \xrightarrow{c_7} []_l \\
\begin{cases} [A + S]_l \xrightarrow{c_8} [A.S]_l \\ [A.S]_l \xrightarrow{c_9} [A + S]_l \end{cases} \\
[A.S]_l \xrightarrow{c_{10}} []_l \\
\begin{cases} [A.S + geneE]_l \xrightarrow{c_{11}} [A.S.geneE]_l \\ [A.S.geneE]_l \xrightarrow{c_{12}} [A.S + geneE]_l \\ [A.S.geneE]_l \xrightarrow{c_{13}} [A.S.geneE + E]_l \\ [E]_l \xrightarrow{c_{14}} []_l \end{cases}
\end{array} \right.
\end{aligned}$$

QS allows the expression of specific genes in a coordinated fashion. One common regulatory mechanism which produces the expression of a gene at a very specific time is an incoherent feedforward loop (IFFL) [1]. In our case, this mechanism is composed of positive regulation of two genes, $geneB$ and $geneC$ by $A.S$, and negative regulation of $geneC$ by B . The following module represents our IFFL.

$$\begin{aligned}
& IFFL(\{A.S, geneB, geneC, A.S.geneB, A.S.geneC, B.geneC\}, \{c_{15}, \dots, c_{24}\}, \{b\}) = \\
& \left\{ \begin{array}{l}
Pos(\{A.S, geneB, A.S.geneB, B\}, \{c_{15}, c_{16}, c_{17}, c_{18}\}, \{b\}) \\
Pos(\{A.S, geneC, A.S.geneC, C\}, \{c_{19}, c_{20}, c_{21}, c_{22}\}, \{b\}) \\
Neg(\{B, geneC, B.geneC\}, \{c_{23}, c_{24}\}, \{b\})
\end{array} \right. = \left\{ \begin{array}{l}
\begin{cases} [A.S + geneB]_l \xrightarrow{c_{15}} [A.S.geneB]_l \\ [A.S.geneB]_l \xrightarrow{c_{16}} [A.S + geneB]_l \\ [A.S.geneB]_l \xrightarrow{c_{17}} [A.S.geneB + B]_l \\ [B]_l \xrightarrow{c_{18}} []_l \end{cases} \\
\begin{cases} [A.S + geneC]_l \xrightarrow{c_{19}} [A.S.geneC]_l \\ [A.S.geneC]_l \xrightarrow{c_{20}} [A.S + geneC]_l \\ [A.S.geneC]_l \xrightarrow{c_{21}} [A.S.geneC + C]_l \\ [C]_l \xrightarrow{c_{22}} []_l \end{cases} \\
\begin{cases} [B + geneC]_l \xrightarrow{c_{23}} [B.geneC]_l \\ [B.geneC]_l \xrightarrow{c_{24}} [B + geneC]_l \end{cases}
\end{array} \right.
\end{aligned}$$

We study how a colony of bacteria expresses $geneC$ in a coordinated manner by using two individual P systems Π_{bact} and Π_e to describe bacteria and environment. The rules associated with the bacteria are obtained by combining the two previously defined modules QS and IFFL. Only

one rule is associated with the environment describing the degradation of the signal S . These two P systems are distributed on a rectangular lattice following the scheme showed in Figure 2 right to obtain a lattice P system modelling a colony of 2500 bacteria. Translocation rules are associated with the bacteria and environments to move the signals S from one P systems to another one.

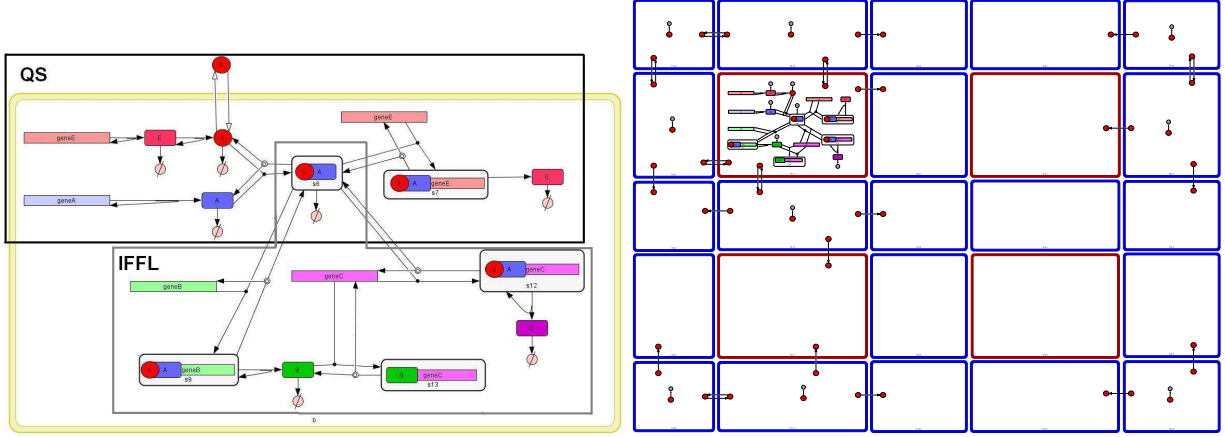


Fig. 2. Modular assembly of our bacterial colony system due to chemical specificity (left) and spatial localisation (right).

We have run simulations of this colony for a simulated time of 10 hours. The top left graph of Figure 3 shows the mean number of molecules of species A , $A.S$, B and C over all bacteria. The other three graphs in Figure 3 show the number of molecules of these species for individual bacteria. Initially, protein A is inactive and bacteria are producing signal S at a low rate. Over time, S accumulates and, once a threshold is reached ($t \sim 150$ min) interacts with A to produce the active transcription factor $A.S$, which in turn increases the production of S . At this point, the negative regulator B and the target protein C start to be expressed. Eventually ($t \sim 300$ min), the accumulation of B begins to repress the production of C , and ultimately ($t \sim 450$ min) stops expression of this target protein all together. The same mechanism is used by real bacteria to coordinate the expression of target proteins such as C .

6 Conclusions and Future Work

In this paper we have proposed a framework based on P systems which enables the design of cell systems biology models by combining modules representing functional subsystems. We defined modularity based on chemical specificity and spatial localisation. Based on this we have extended the P system modelling framework by introducing rule modules and spatial lattices. We have illustrated our approach through the modelling of a large colony of bacterial cells. In order to simulate this system in a computationally efficient manner we introduced an optimised version of the multicompartmental Gillespie algorithm. Results from these simulations show that bacteria which combine a QS and IFFL modules are able to express a specific gene in coordinated way at a specific time during their evolution. A future research line consists of the use of the modules in our framework to design artificial cell systems linking our research to synthetic biology and in coupling this work with the automated discovery of modules assembly as recently reported in [20].

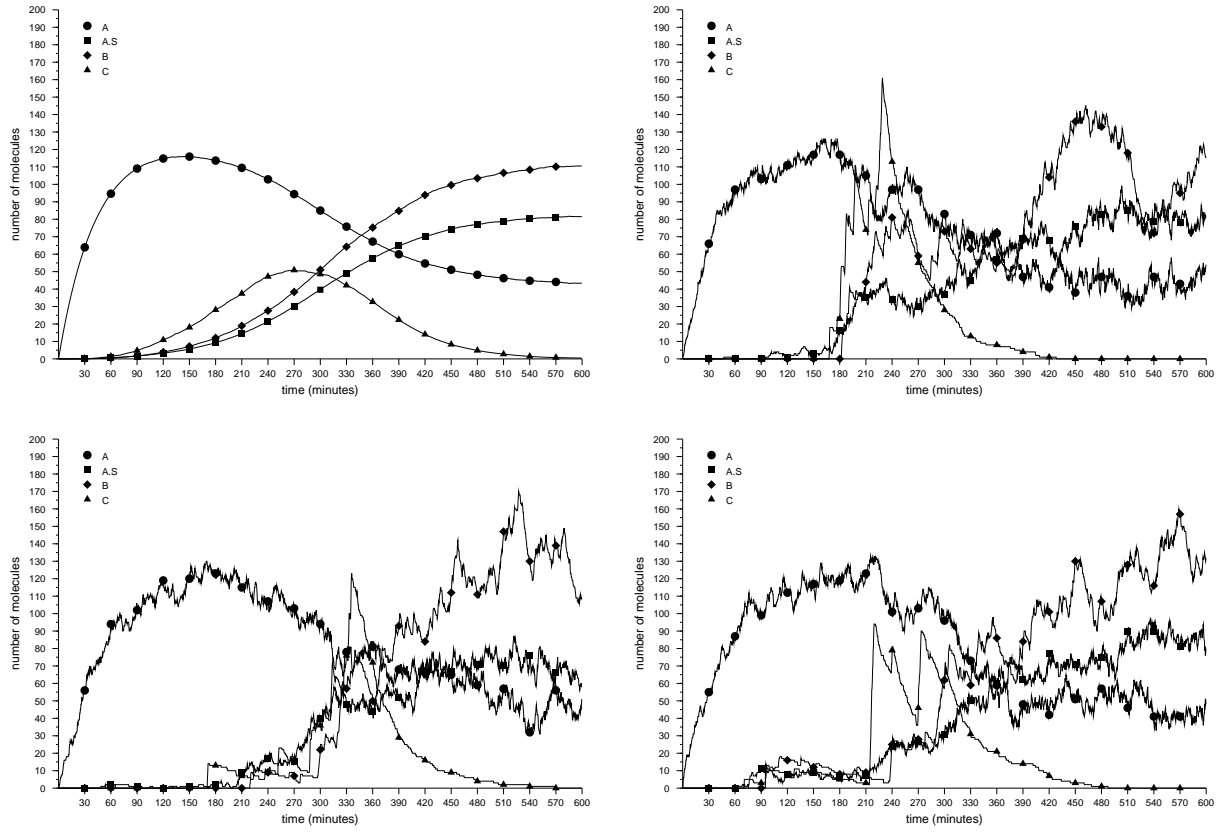


Fig. 3. Simulations results for a colony of 2500 bacteria.

7 Acknowledgements

We would like to acknowledge EPSRC grant EP/E017215/1 and BBSRC grants BB/F01855X/1 and BB/D019613/1.

Bibliography

- [1] U. Alon. *An Introduction to Systems Biology*. Mathematical and Computational Biology Series. Chapman & Hall/Crc., 2006.
- [2] S. A. Benner, A. M. Sismour. Synthetic biology. *Nature Reviews Genetics*, 6:533–543, 2005.
- [3] F. Bernardini, M. Gheorghe. Population P systems. *Journal of Universal Computer Science*, 10(5):509–539, 2004.
- [4] F. Bernardini, M. Gheorghe, N. Krasnogor. Quorum sensing P systems. *Theoretical Computer Science*, 371(1-2):20–33, 2007.
- [5] G. Ciobanu, L. Pan, G. Păun. P systems with minimal parallelism. *Theoretical Computer Science*, 378(1):117 – 130, 2007.
- [6] F. Fontana, L. Bianco, V. Manca. P systems and the modelling of biochemical oscillations. *Lecture Notes in Computer Science*, 3850:199 – 208, 2005.
- [7] D. T. Gillespie. Stochastic simulation of chemical kinetics. *Annual Review of Physical Chemistry*, 58:35–55, 2007.
- [8] P. J. Goss, J. Peccoud. Quantitative modelling of stochastic system in molecular biology by using stochastic Petri nets. *Proc. of the National Academy of Sciences, USA*, 95:6750–6755, 1998.
- [9] L. H. Hartwell, J. J. Hopfield, S. Leibler, A. W. Murray. From molecular to modular cell biology. *Nature, Impacts*, 402:c47–c52, 1999.
- [10] M. J. Pérez-Jiménez, F. J. Romero-Campero. P systems, a new computational modelling tool for systems biology. *Transactions on Computational Systems Biology VI*, pages 176–197, 2006.
- [11] D. Pescini, D. Besozzi, G. Mauri, C. Zandron. Dynamical probabilistic P systems. *International Journal of Foundations of Computer Science*, 17(1):183 – 195, 2006.
- [12] G. Păun. Computing with membranes. *Journal of Computer and System Sciences*, 61(1):108–143, 2000.
- [13] G. Păun. *Membrane Computing: An Introduction*. Springer-Verlag., Berlin, 2002.
- [14] A. Regev, E. Shapiro. The π -calculus as an abstraction for biomolecular systems. In *Modelling in Molecular Biology.*, pages 1–50. Springer Berlin., 2004.
- [15] F. J. Romero-Campero, M. J. Pérez-Jiménez. A model of the quorum sensing system in *Vibrio fischeri* using P systems. *Artificial Life*, 14(1):95 – 109, 2008.
- [16] F. J. Romero-Campero, M. J. Pérez-Jiménez. Modelling gene expression control using P systems: The *lac* operon, a case study. *BioSystems*, 91(3):438–457, 2008.
- [17] Z. Szallasi, J. Stelling, V. Periwal. *System Modeling in Cellular Biology*. MIT Press, 2006.
- [18] B. Di Ventura, C. Lemerle, K. Michalodimitrakis, L. Serrano. From *in vivo* to *in silico* biology and back. *Nature Reviews*, **443**, 527 – 533, 2007.
- [19] N. Krasnogor, M. Gheorghe, G. Terrazas, S. Diggle, P. Williams, M. Camara. An appealing computational mechanism drawn from bacterial quorum sensing. *Bulletin of the EATCS*, **85**, 135-148, 2005.
- [20] F. J. Romero-Campero, H. Cao, M. Camara, N. Krasnogor. Structure and parameter estimation for cell systems biology models. *Proc. of the Genetic and Evolutionary Computation Conference*, 2008.
- [21] S.P. Diggle , S.A. Crusz , M. Camara. Quorum sensing. *Current Biology* **17**:21, R907–10, 2007.