Modeling Logic Gene Networks by means of Probabilistic Dynamic P Systems

LUIS VALENCIA–CABRERA[1], MANUEL GARCÍA–QUISMONDO[1], MARIO J. PÉREZ–JIMÉNEZ[1], YANSEN SU[2], HUI YU[2], LINQIANG PAN[2]

[1] *Research Group on Natural Computing*

Dpt. of Computer Science and Artificial Intelligence, University of Sevilla

Avda. Reina Mercedes s/n. 41012 Sevilla, Spain

{lvalencia,mgarciaquismondo,marper}@us.es

[2] *Department of Control Science and Engineering, Huazhong University of Science and Technology*

1037 Luoyu Road, Wuhan, China

suyansen1985@163.com, yuhuihustac@gmail.com, lqpan@mail.hust.edu.cn

Gene Regulatory Networks (*GRNs*) are a useful tool for biologists to understand the interactions among genes in living organisms. A special kind of GRNs known as Logic Networks (*LNs*) has been recently introduced. These networks consider that the state of one or more genes can influence a third one. Therefore, genes influence each other by means of logic operations. The improved Logic Analysis of Phylogenetic Profiles (LAPP) method proposes an algorithm to capture the dynamics of this kind of networks. In this paper we provide a formalization of LNs and we also introduce a Membrane Computing model for LNs which reproduces their dynamics according to the improved LAPP method. Eventually, some conclusions and future work are outlined.

Keywords: Bioinformatics, Genetics, Gene networks, Membrane Computing, Modelling, LAPP, Logic networks

1 INTRODUCTION

Membrane Computing [10] has been an useful modelling framework for biochemical modelling since its beginning. As a natural evolution of this research field, genetic networks have also been modelled by means of P systems. In this work, we propose a Membrane Computing model for a specific type of gene networks. This type is known as Logic Networks (*LN*), which can be generated by Logical Analysis of Phylogenesis Profiles (LAPP) [1], a method for constructing logic networks out of statistical data. In these networks, gene states can be influenced by single gene or by combinations of multiple genes. We restrict these combinations to at most two genes, that is the most usual case in the real life. Our model intends to capture the behaviour of a specific algorithm, known as the Improved LAPP Method [17]. This algorithm provides mechanisms to combine different influences on the same gene. It is worth pointing out that we are interested in reproducing the behaviour of the Improved LAPP Method, rather than contrast its accuracy by using a P–system based approach. On the event of possible evolutions of the Improved LAPP Method to capture better the dynamics of LNs, our Membrane Computing framework is expected to be adapted to the resulting versions of the Improved LAPP Method, thus providing itself a more accurate tool to simulate the behaviour of LNs. To the best of our knowledge, our model is the first Membrane Computing model to reproduce this behaviour.
This paper is structured as follows. Section 2 formalizes the gene networks we consider in this work and describes the improved LAPP algorithm, whose semantics are intended to be captured in this work. Section 3 introduces the current state of the art about modelization of gene networks within the framework of Membrane Computing. Section 4 presents the model introduced and outlines the formal framework in which it is included. Section 5 describes a methodology for using MeCoSim interface to simulate the dynamics of logic networks, including a toy example on a simple logic network, in order to illustrate the interface and the model. Finally, Section 6 lists the conclusions obtained and proposes some open problems.

2 LOGIC NETWORKS

This section summarizes the concept of logic network and outlines a procedure to construct them out of statistical genetic data. In addition, the dynamics of logic networks and their dynamics, providing an algorithmic approach for this purpose is described.

2.1 A Formalization of Logic Networks

In this work, we focus on the study of genes regarding their activity or inactivity given a certain instant. This section gives a formal definition of Logic Networks (LN), including its syntax and semantics.

Syntax of Logic Networks

An alphabet is a nonempty set. Given a finite alphabet Γ we denote $\bar{\Gamma} = \{\bar{x} : x \in \Gamma\}$, where $\Gamma \cap \bar{\Gamma} = \emptyset$. We also denote $\bar{\bar{x}} = x$, for each $x \in \Gamma \cup \bar{\Gamma}$.

Definition 1 *A* gene g *over a finite alphabet* Γ *is an element in* Γ. *The* behaviour *of* g *is a mapping* φ_g *from* $\mathbb{N}$ *into* $\{0, 1\}$. *The* state *of* g *at any instant* $t \in \mathbb{N}$ *is* $\varphi_g(t)$. *If* $\varphi_g(t) = 1$ *(respectively,* $\varphi_g(t) = 0$*) we say that gene* g *is* active *(respectively, inactive) at instant* t.

Given a finite alphabet Γ and a gene g over Γ we consider the application $\varphi_{\bar{g}}$ as follows: $\varphi_{\bar{g}} = 1 - \varphi_g$, that is, for each $t \in \mathbb{N}$, $\varphi_{\bar{g}}(t) = 1 - \varphi_g(t)$.
For each alphabet $\Sigma \subseteq \Gamma$ we define the mapping l_Σ from $\Sigma \cup \bar{\Sigma}$ into $\{0, 1\}$ as follows: $l_\Sigma(x) = 1$, if $x \in \Sigma$, and $l_\Sigma(x) = 0$ otherwise.

Definition 2 *A logic network of size* n *over an alphabet* Γ *such that* $|\Gamma| \geq n$, *is a tuple* $(\Sigma, \{f_1^1, \ldots, f_1^{\alpha_1}\}, \{f_2^1, \ldots, f_2^{\alpha_2}\})$ *where:*

1. $\Sigma \subseteq \Gamma, |\Sigma| = n$ *(*Σ *is the set of genes of the network).*
2. *For each* $j, 1 \leq j \leq \alpha_1, f_1^j = (g_1^{j,1}, g_1^{j,2}, \omega_1^j, op_1^j)$, *where:*
   - $g_1^{j,1}, g_1^{j,2} \in \Sigma$.
   - ω_1^j *is a real number in* $[0, 1]$ *which represents the uncertainty of unary interaction* j *(see [17], noting that* ω_1^j *is equivalent to* $U(B|A)$*, with* $B = g_1^{j,2}$ *and* $A = g_1^{j,1}$*).*
   - op_1^j *is a mapping from* $\mathbb{N}$ *into* $\{-1, 0, 1\}$ *which can be of one of the following types:* sp^j, si^j, wp^j, wi^j. *These kinds of functions are defined as follows: for each* $t \in \mathbb{N}$*:*
     - $sp^j(t) = \varphi_{g_1^{j,1}}(t) - \varphi_{\bar{g}_1^{j,1}(t)}$ *(strong promotion)*
     - $si^j(t) = -sp_1^j(t)$ *(strong inhibition)*
     - $wp^j(t) = \varphi_{g_1^{j,1}}(t)$ *(weak promotion)*
     - $wi^j(t) = -wp_1^j(t)$ *(weak inhibition)*

     These operations provide the contribution from f_1^j *to gene* $g_1^{j,2}$ *in conjunction with* ω_1^j *in order to know its state at instant* $t + 1$.

3. *For each* $j, 1 \leq j \leq \alpha_2$, $f_2^j = (g_2^{j,1}, g_2^{j,2}, g_2^{j,3}, \omega_2^j, op_2^j)$, *where:*

   - $g_2^{j,1}, g_2^{j,2}, g_2^{j,3} \in \Sigma \cup \bar{\Sigma}$.
   - ω_2^j *is a real number in* $[0,1]$ *which represents the uncertainty of binary interaction* j *(see [17], noting that* ω_2^j *is equivalent to* $U(C|f(A,B))$*, with* $C = g_2^{j,3}$, $A = g_2^{j,1}$, $B = g_2^{j,2}$ *and* $f = f_2^j$*).*
   - op_2^j *is a mapping from* $\mathbb{N}$ *into* $\{-1,1\}$ *which can be of one of the following types:* and^j, or^j, xor^j. *These kind of functions are defined as follows: for each* $t \in \mathbb{N}$,

$$and^j(t) = [\varphi_{g_2^{j,1}}(t) \cdot \varphi_{g_2^{j,2}}(t) - \overline{\varphi_{g_2^{j,1}}(t) \cdot \varphi_{g_2^{j,2}}(t)}] \cdot (2 \cdot l_\Sigma(g_2^{j,3}) - 1)$$

$$or^j(t) = [\varphi_{g_2^{j,1}}(t) + \varphi_{g_2^{j,2}}(t) - \varphi_{g_2^{j,1}}(t) \cdot \varphi_{g_2^{j,2}}(t) - \overline{\varphi_{g_2^{j,1}}(t) + \varphi_{g_2^{j,2}}(t) - \varphi_{g_2^{j,1}}(t) \cdot \varphi_{g_2^{j,2}}(t)}] \cdot (2 \cdot l_\Sigma(g_2^{j,3}) - 1)$$

$$xor^j(t) = [(1 - \varphi_{g_2^{j,1}}(t)) \cdot \varphi_{g_2^{j,2}}(t) + (1 - \varphi_{g_2^{j,2}}(t)) \cdot \varphi_{g_2^{j,1}}(t) - \overline{(1 - \varphi_{g_2^{j,1}}(t)) \cdot \varphi_{g_2^{j,2}}(t) + (1 - \varphi_{g_2^{j,2}}(t)) \cdot \varphi_{g_2^{j,1}}(t)}] \cdot (2 \cdot l_\Sigma(g_2^{j,3}) - 1)$$

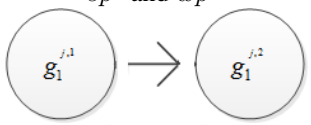
   where $\bar{b}$ *denotes* $1 - b$*, for each* $b \in \{0,1\}$.

   These operations provide the contribution from f_2^j *to gene* $g_2^{j,3}$ *in conjunction with* ω_2^j *in order to know its state at instant* $t+1$.

Next, operations f_1^j and f_2^j (Figures 1 and 2) are informally described.

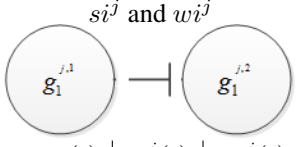
FIGURE 1: Behaviour of unary operations f_1^j

sp^j and wp^j



$\varphi_{g_1^{j,1}}(t)$	$sp^j(t)$	$wp^j(t)$
1	1	1
0	−1	0

si^j and wi^j

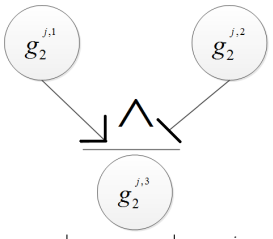


$\varphi_{g_1^{j,1}}(t)$	$si^j(t)$	$wi^j(t)$
1	−1	−1
0	1	0

Note 1 *Let us point out that, in graphic representations of operations* f_1^j *in the network, only genes in* Σ *appear.*

FIGURE 2: Behaviour of binary operations f_2^j

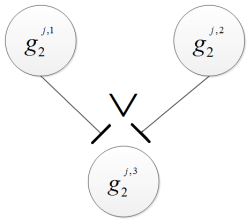
and^j



$\varphi_{g_2^{j,1}}(t)$	$\varphi_{g_2^{j,2}}(t)$	$and^j(t)$
1	1	1
1	0	-1
0	1	1
0	0	1

In this example $g_2^{j,1} \in \Sigma$ and $g_2^{j,2}, g_2^{j,3} \in \bar{\Sigma}$

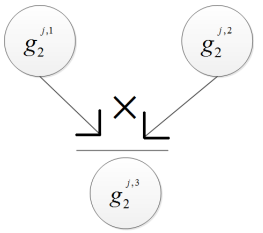
or^j



$\varphi_{g_2^{j,1}}(t)$	$\varphi_{g_2^{j,2}}(t)$	$or^j(t)$
1	1	-1
1	0	1
0	1	1
0	0	1

In this example $g_2^{j,1}, g_2^{j,2} \in \bar{\Sigma}$ and $g_2^{j,3} \in \Sigma$

xor^j



$\varphi_{g_2^{j,1}}(t)$	$\varphi_{g_2^{j,2}}(t)$	$xor^j(t)$
1	1	1
1	0	-1
0	1	-1
0	0	1

In this example $g_2^{j,1}, g_2^{j,2} \in \Sigma$ and $g_2^{j,3} \in \bar{\Sigma}$

Note 2 *In this example genes $g_2^{j,1}, g_2^{j,2}$, the membership of $g \in \Sigma$ (respectively, $g \in \bar{\Sigma}$) is translated into arrow–type operation $\rightarrow$ (respectively, $\dashv$). If $g_2^{j,3} \in \bar{\Sigma}$ then we denote — upon gene $g_2^{j,3}$.*

Semantics of logic networks

Next, we introduce a semantics for logic networks. Let $LN = (\Sigma, f_1, f_2)$ be a logic network with n nodes (genes) from $\Sigma = \{g_1, \ldots, g_n\}$ according to

Definition 2. A *configuration* of the logic network LN at instant t is a tuple $(\varphi_{g_1}(t), \ldots, \varphi_{g_n}(t))$ which describes the state of every gene g_i at that instant.

In order to define a transition step from t to $t+1$ in a logic network LN, we must compute $\varphi_{g_i}(t+1)$, for $1 \leq i \leq n$, from the configuration of LN at any instant t. For that, we introduce some previous concepts and notations.

- Let $f_1^j = (g_1^{j,1}, g_1^{j,2}, \omega_1^j, op_1^j)$ be a unary operation by which node $g_1^{j,1}$ acts on node $g_1^{j,2}$. We define the action of $g_1^{j,1}$ on $g_1^{j,2}$ at instant t, denoted by $action(g_1^{j,2}|g_1^{j,1})(t)$, as follows:

$$action(g_1^{j,2}|g_1^{j,1})(t) = op_1^j \cdot \omega_1^j$$

- Let $f_2^j = (g_2^{j,1}, g_2^{j,2}, g_2^{j,3}, \omega_2^j, op_2^j)$ be a binary operation by which nodes $g_2^{j,1}$ and $g_2^{j,2}$ act on node $g_2^{j,3}$. We define the action of $g_2^{j,1}$ and $g_2^{j,2}$ on $g_2^{j,3}$ at instant t, denoted by $action(g_2^{j,3}|g_2^{j,1}, g_2^{j,2})(t)$, as follows:

$$action(g_2^{j,3}|g_2^{j,1}, g_2^{j,2})(t) = op_2^j \cdot \omega_2^j$$

- Next we define the total effect of the action on gene i as follows:

$$\begin{aligned} Action(g_i, t) = &\quad Action_1(g_i, t) + Action_2(g_i, t), \text{being} \\ Action_1(g_i, t) = &\sum_{\substack{1 \leq j \leq \alpha_1 \\ g_1^{j,2} = g_i}} action(g_1^{j,2}|g_1^{j,1})(t) \\ Action_2(g_i, t) = &\sum_{\substack{1 \leq j \leq \alpha_2 \\ g_2^{j,3} = g_i}} action(g_2^{j,3}|g_2^{j,1}, g_2^{j,2})(t) \end{aligned}$$

Then, $g_i(t+1)$ is defined as follows:

$$\varphi_{g_i}(t+1) = \begin{cases} 1, & \text{if } \varphi_{g_i}(t) + Action_1(g_i, t) + Action_2(g_i, t) \geq 0.5, \\ 0, & \text{otherwise} \end{cases}$$

Taking into account this concept of logic networks and its associated dynamics, we introduce a model within the framework of Membrane Computing to reproduce the behaviour of these networks. In next section, this model and the formal framework which defines its semantics are described.

3 MODELLING GENE NETWORKS IN MEMBRANE COMPUTING

This section introduces some previous works on the modelling of biochemical systems by means of P systems, focusing on gene networks. Later on, the formal framework in which our model is based (that is, *PDP systems*) is defined, so as to outline the syntax and semantics of the model introduced.

3.1 Previous works

Since its introduction by Gheorghe Paun [12], Membrane Computing has been applied as a modelling framework for biological phenomena at a microscopical level. One of its main features is the capability of modelling different comparments by means of *membranes* interconnected by communication rules. The idea is that the reactions which take place may differ according to the compartment in which they occur. Some traditional approaches such as Ordinary Differential Equations (ODEs) already allowed this feature. For instance, Kawai [7] proposed a multidimensional stochastic ODE system. This system describes the evolution of the concentration of chemical drugs inside biological tissues such as liver, guts and muscles. Although ODEs are a well–known framework for biomolecular systems, they require some assumptions on the system to be modelled. Specifically, they require that the differential in the concentration of substances within each compartment is constant. In addition, their accuracy fails when the number of molecules taken into account is too small. This is due to their continuous nature, that is, the numbers of molecules of the modelled substances are approximated to a real number. This approximation works well when the number of molecules is high, but it does not reflect reality appropriately on scenarios which consider only a few molecules. A different approach from the field of Membrane Computing can help sort out these constraints from biomolecular phenomena. In contrast to ODEs, the computational devices in this field, P systems, do not need to make these assumptions. That is, they reproduce faithfully scenarios with few molecules and non-constant concentration differentials. There also exists another advantage of P systems over ODEs, the *modularity* of the system. A system is considered to be *modular* if small changes in the behaviour of the modelled system, usually entails a relatively small change in the model. ODE–based models have not this practical property. The model introduced in this paper aims to characterize the behaviour of a thoroughly studied kind of biomolecular systems known as Genetic Regulatory Networks (GRNs).

Informally speaking, GRNs are directed graphs in which vertices represent genes, whereas edges represent interactions between them. The dynamics of these networks are heavily influenced by the variations in the concentrations of the biochemical substances which interact with the genes, such as proteins. These variations of concentrations have been specially studied within the field of Membrane Computing, in order to understand the evolution of GRNs (see [6] for an example). To the best of our knowledge, the previously existing models of gene GRNs based on Membrane Computing do not reflect the case in which the combination of states of two genes influence the state of a third one. Bowers *et al.* [1, 2] claim that these scenarios are not rare on GRNs. In order to reflect them, they describe a statistical procedure to construct GRNs out of experimental data considering these interactions . In their procedure, they set thresholds to the frequency in which gene states are interrelated. The idea is to identify which combined gene states encode interactions and which ones are just coincidences. However, there is still open the question about how to simulate the influences of several interactions on a single gene. Shmulevich *et al.* [15] propose a solution by combining different influences on a gene by means of logic gates. In contrast to Bowers, they also maintain the probabilistic information within the constructed GRN, permitting a more thorough understanding of the interaction between genes and more data to simulate the evolution of their states. This problem has been addressed in a different manner by Wang *et al.* [17]. In their work, they propose an algorithm known as improved Logic Analysis of Philogenetic Profiles method (improved LAPP method). In their work, they assign a weight to each gene interaction. This weight is equal to its probability to occur. On each step, the sum of all weights on each influenced gene, plus an influence assigned to its own previous state, is computed in order to calculate a state weight. Eventually, they define the state of the gene in the next step by thresholding it against 0.5. That is, if the previous sum is greater or equal to 0.5, then the gene is active. Otherwise, it is inactive.

In our work, we formalize the concept of Logic Network. We also propose a Membrane Computing model to capture the behaviour of the improved LAPP method. For its simulation, the model has been specified on P–Lingua [4]. In addition, we propose a methodology for the simulation and analysis of Logic Networks, based on a custom–designed interface on MeCoSim.

3.2 A Formal framework: Population Dynamics P systems

Definition 3 *A Population Dynamics P system (PDP systems, for short) of degree (q, m) with $q, m \geq 1$, taking $T \geq 1$ time units, is a tuple*
$(G, \Gamma, \Sigma, T, R_E, \mu, R, \{f_{r,j} : r \in R, 1 \leq j \leq m\}, \{\mathcal{M}_{ij} : 1 \leq i \leq q,\ 1 \leq j \leq m\})$
where:

- *$G = (V, S)$ is a directed graph. Let $V = \{e_1, \ldots, e_m\}$ whose elements are called environments;*
- *Γ is the working alphabet and $\Sigma \subsetneqq \Gamma$ is an alphabet for the objects that can be present in the environments;*
- *T is a natural number that represents the simulation time of the system;*
- *R_E is a finite set of communication rules among environments of the form*
$$(x)_{e_j} \xrightarrow{p} (y_1)_{e_{j_1}} \ldots (y_h)_{e_{j_h}}$$
where $x, y_1, \ldots, y_h \in \Sigma$, $(e_j, e_{j_l}) \in S$ $(l = 1, \ldots, h)$ and $p(t) \in [0, 1]$, for each $t (1 \leq t \leq T)$. Function p depends on the variables $x, j, j_1, \ldots, j_h$. If $p(t) = 1$, for each t, then we omit the probabilistic function. These rules verify the following: for each environment e_j and for each object x, the sum of functions associated with the rules from R_E whose left-hand side is $(x)_{e_j}$ coincides with the constant function equal to 1.
- *μ is a rooted tree consisting of q membranes, with the membranes injectively labeled by $1, \ldots, q$. The skin membrane is labeled by 1. We also associate electrical charges from the set $\{0, +, -\}$ with membranes.*
- *R is a finite set of evolution rules of the form $r : u[\, v\,]_i^\alpha \to u'[\, v'\,]_i^{\alpha'}$ where u, v, u', v' are multisets over Γ, $i \in \{1, \ldots, q\}$, and $\alpha, \alpha' \in \{0, +, -\}$.*
- *For each $r \in R$ and for each j, $1 \leq j \leq m$, $f_{r,j}$ is a computable function whose domain is $\{1, \ldots, T\}$ and its range is $[0, 1]$, verifying the following:*
  - *$\star$ For each $u, v \in \Gamma^*$, $i \in \{1, \ldots, q\}$ and $\alpha, \alpha' \in \{0, +, -\}$, if $r_1, \ldots, r_z$ are the rules from R whose left-hand side is $u[v]_i^\alpha$ and the right-hand side have polarization α', then $\sum_{j=1}^{z} f_{r_j}(t) = 1$, for $1 \leq t \leq T$.*

- ⋆ *If $(x)_{e_j}$ is the left-hand side of a rule $r \in R_E$, then none of the rules of R has a left-hand side of the form $u[v]_0^\alpha$, for any $u, v \in \Gamma^*$ and $\alpha \in \{0, +, -\}$, having $x \in u$.*

- *For each j $(1 \leq j \leq m)$, $\mathcal{M}_{1j}, \ldots, \mathcal{M}_{qj}$ are multisets over Γ, describing the objects initially placed in the q regions of μ, within the environment e_j.*

A system as described in the previous definition can be viewed as a set of m environments $e_1, \ldots, e_m$ linked between them by the arcs from the directed graph G. Each environment e_j contains a P system, $\Pi_j = (\Gamma, \mu, R, \mathcal{M}_{1j}, \ldots, \mathcal{M}_{qj})$, of degree q, such that $\mathcal{M}_{1j}, \ldots, \mathcal{M}_{qj}$ represent the initial multisets for this environment, and every rule $r \in R$ has a computable function $f_{r,j}$ associated with it.

The tuple of multisets of objects present at any instant in the m environments and at each of the regions of each Π_j, together with the polarizations of the membranes in each P system, constitutes a *configuration* of the system at that instant. At the initial configuration we assume that all environments are empty and all membranes have a neutral polarization.

We assume that a global clock exists, marking the time for the whole system, that is, all membranes and the application of all rules (both from R_E and R) are synchronized in all environments.

The P system can pass from one configuration to another by using the rules from $R = R_E \cup \bigcup_{j=1}^{m} R_{\Pi_j}$ as follows: at each transition step, the rules to be applied are selected according to the probabilities assigned to them, and all applicable rules are simultaneously applied in a maximal way. After the application of the selected rules, no further rule can be applied to the remaining objects. For each j $(1 \leq j \leq m)$ there is just one further restriction, concerning the consistency of charges: in order to apply several rules of R_{Π_j} simultaneously to the same membrane, all the rules must have the same electrical charge on their right-hand side.

When a communication rule between environments

$$(x)_{e_j} \xrightarrow{p} (y_1)_{e_{j_1}} \ldots (y_h)_{e_{j_h}}$$

is applied, object x passes from e_j to $e_{j_1}, \ldots, e_{j_h}$ possibly modified into objects $y_1, \ldots, y_h$, respectively. At any instant t, $1 \leq t \leq T$, for each object x in environment e_j, if there exist communication rules whose left-hand side is $(x)_{e_j}$, then one of these rules will be applied. If more than one communication rule can be applied to an object, the system selects one randomly, according to their probability which is given by $p(t)$.

4 A FAMILY OF P SYSTEMS BASED ON LOGIC NETWORKS

In this work, we present a family of P systems, known as Logic Network Dynamic P systems (*LN DP systems*). These P systems aim to capture the behaviour of the improved LAPP method [17]. An LN DP system is described within an expansion of PDP systems.

An LN DP system Π_{LN} of degree (q, m) with $q, m \geq 1$, taking $T \geq 1$ time units, is a tuple

$$\begin{aligned} \Pi_{LN} \quad = \quad & (G, \Gamma, \Sigma, T, R_E, \mu, R, \{f_{r,j} : \ r \in R, 1 \leq j \leq m\}, \\ & \{\mathcal{M}_{ij} : \ 1 \leq i \leq q, \ 1 \leq j \leq m\}, \{\mathcal{M}_j : 1 \leq j \leq m\}) \end{aligned}$$

where:

- (G, Γ, Σ, T, R_E, μ, R, $\{f_{r,j} : \ r \in R, 1 \leq j \leq m\}$, $\{\mathcal{M}_{ij} : \ 1 \leq i \leq q, \ 1 \leq j \leq m\}$) is a PDP system.
- $\{f_{r,j} = 1 : \ r \in R, 1 \leq j \leq m\}$.
- For each j $(1 \leq j \leq m)$, $\mathcal{M}_j$ are multisets over Γ, describing the objects initially placed in environment e_j.

That is, an LN DP system can be viewed as a PDP system whose initial multisets are placed in the environments of the system. The improved LAPP method is a deterministic algorithm, so probabilities associated to communication rules are not necessary. Thus, they are not used in LN DP systems. In practical terms, LN DP System rules do not compete for objects. That is, the multisets associated to the left–hand sides of any two rules associated to any membrane in a system are disjoint. As the PDP system framework requires every rule to have an associated probability, the probability associated to every rule in LN DP Systems is equal to 1. The PDP System framework also requires that the sum all probabilities of these rules with the same left–hand side (that is, all rules within the same block) is equal to 1. As the multisets associated to the left–hand sides of any two rules associated to any membrane in a system are disjoint, there are not any two rules with the same left–hand side. Thus, that property is conserved in LN DP Systems.

4.1 The model

Here a model for a family of LN DP systems is presented. For a given network, each gene is represented by a P system with a single membrane inside an environment. The state of each gene in the network at every moment will be coded by the presence of a counter (1: active; 0: inactive) in its environment. This model covers every P system in this family, so the multisets, rules, etc. depend on the specific instance of a LN. The general model requires the use of parameters in our constructs, as explained at the end of this subsection. Let LN be a logic network. Let ng, nu, nb be the number of genes, unary and binary interactions, respectively. Let $m = ng + nu + nb$. The model consists of the following PDP system of degree $(1, n)$,

$$\Pi_{LN} = (G, \Gamma, \Sigma, T, R_E, \mu, R, \{\mathcal{M}_{ij} : 0 \leq i \leq q-1,\ 1 \leq j \leq m\}, \{\mathcal{M}_j : 1 \leq j \leq m\})$$

where:

- G is a directed graph containing a node (environment) for each gene, unary or binary interaction, following this order.
- In the alphabet Γ, we represent gene states, interaction types, contribution weights and targets, as outlined below.

$$\begin{aligned}
\Gamma = \quad & \{a_i, b_i, c_i : 0 \leq i \leq 1\} \cup \{go, d_0\} \cup \{unop_j, binop_j : 1 \leq j \leq 4\} \cup \\
& \{auxDest_{i,g_{j,1},k} : 0 \leq i \leq 1, 1 \leq j \leq ng, 1 \leq k \leq nb + nu\} \cup \\
& \{dest_{i,g_{j,1},t_{k,1}+ng} : 0 \leq i \leq 1, 1 \leq j \leq ng, 1 \leq k \leq nb\} \cup \\
& \{dest_{i,g_{j,1},unt_{k-nb,1}+ng+nb} : 0 \leq i \leq 1, 1 \leq j \leq ng, nb+1 \leq k \leq nb+nu\} \cup \\
& \{e_{t_{k,4}*i+(1-i)*(1-t_{k,4}),t_{k,1}+ng} : 0 \leq i \leq 1, 1 \leq k \leq nb\} \cup \\
& \{e_{t_{k,6}*i+(1-i)*(1-t_{k,6}),t_{k,1}+ng} : 0 \leq i \leq 1, 1 \leq k \leq nb\} \cup \\
& \{e_{unt_{k-nb,4}*i+(1-i)*(1-unt_{k-nb,4}),unt_{k-nb,1}+ng+nb} : \\
& \ 0 \leq i \leq 1, nb+1 \leq k \leq nb+nu\} \cup \\
& \{eF_{t_{k,8}*i+(1-i)*(1-t_{k,8}),t_{k,1}+ng} : 0 \leq i \leq 1, 1 \leq k \leq nb\} \cup \\
& \{eF_{i,(unt_{k,1}+ng+nb)} : 0 \leq i \leq 1, 1 \leq k \leq nu\} \cup \\
& \{clock_j : 0 \leq j \leq cc+3\}
\end{aligned}$$

  - Object go triggers the start of a new cycle in the evolution of the gene states. Objects $clock_i$ synchronize some steps of the cycle, such as the sum of the different contributions to each gene as a result of the interactions of each cycle.
  - Objects a_i represent the gene state: (a_0: $inactive$; a_1: $active$). Objects b_i represent the weight of the interactions (including self–influence interactions).
  - Objects $unop_j, 1 \leq j \leq 4$ participate in the unary interactions, representing *strong promotion*, *strong inhibition*, *weak promotion* and *weak inhibition*, respectively. Objects $binop_j, 1 \leq j \leq 3$ participate in the binary ones, representing *or*, *and* and *xor*.

- Objects $dest_{i,j,k}$, $auxDest_{i,j,k}$, $e_{i,k}$, c_i and $eF_{i,k}$ are auxiliary objects involved in the interactions.

- The environment alphabet is $\Sigma = \Gamma \setminus \{d_0\}$

- Each cycle evolution from a real network configuration to the next one involves 15 computational steps, so $T = 15 \cdot Cycles$, where $Cycles$ is the number of cycles to simulate.

- $\mu = [\quad]_1$ is the membrane structure.

- The initial multisets are:

  - $\mathcal{M}_{g_{k,1}} = \{\ {a_1}^{g_{k,3}}, {a_0}^{1-g_{k,3},go} : 1 \leq k \leq ng\}$. That is, inside each gene environment (labelled by $g_{k,1}$), we have its gene state (a_1:active or a_0:inactive), depending on the introduced value $g_{k,3}$, 0 or 1. Object go triggers the start of a cycle.
  - $\mathcal{M}_{ng+t_{i,1}} = \{\ binop_{t_{i,2}} : 1 \leq i \leq nb\}$. That is, inside each binary interaction environment (labelled by $ng + t_{i,1}$, we have an object ($binop_{t_{i,2}}$) representing the interaction (or, and, xor).
  - $\mathcal{M}_{ng+nb+unt_{i,1}} = \{\ unop_{unt_{i,2}} : 1 \leq i \leq nu\}$. That is, inside each unary interaction environment (labelled by $ng + nb + unt_{i,1}$, we have an object ($unop_{unt_{i,2}}$) representing the interaction (strong or weak promotion or inhibition).

- The rules of R and R_E to apply are showed below. They are put together to follow the sequential order of execution. Environment rules start with re and skeleton rules start with rs.

  - Cycle start, and contribution of each gene over its state: $rs_{1,i} \equiv go\, a_i[\,]_1 \longrightarrow c_i b_i {}^{max*i} b_0 {}^{threshold} clock_0[\,]_1 : 0 \leq i \leq 1$
  - For each source gene environment:
    - Auxiliary objects $auxDest$ for all possible interactions from the source gene are created:
      $re_{2,i,j,k} \equiv \quad (c_i \longrightarrow \{auxDest_{i,g_{j,1},k} : \{1 \leq k \leq nb + nu\}\})_{g_{j,1}}$
      $: 0 \leq i \leq 1, 1 \leq j \leq ng$
    - Destination objects are created for each possible binary interaction, including information about the target interaction environment $t_{k,1} + ng$:
      $re_{3,i,j,k} \equiv \quad (auxDest_{i,g_{j,1},k} \longrightarrow dest_{i,g_{j,1},t_{k,1}+ng})_{g_{j,1}}$
      $: 0 \leq i \leq 1, 1 \leq j \leq ng, 1 \leq k \leq nb$

* The same is done for each possible unary interaction, including information about the target interaction environment $untk - nb, 1 + ng + nb$:

$re_{4,i,j,k} \equiv \quad (auxDest_{i,g_{j,1},k} \longrightarrow dest_{i,g_{j,1},untk-nb,1+ng+nb})_{g_{j,1}}$
$: 0 \leq i \leq 1, 1 \leq j \leq ng, nb + 1 \leq k \leq nb + nu$

– For each actual interaction, in the gene environments, objects $e_{i,k}$ (value i and target k) are created for the contribution of each source gene involved in an interaction, from their source values $t_{k,4}$ and $t_{k,6}$ (binary interactions) and $unt_{k-nb,4}$ (unary interactions):

$re_{5,i,k} \equiv \quad (dest_{i,t_{k,3},t_{k,1}+ng} \longrightarrow e_{t_{k,4}*i+(1-i)*(1-t_{k,4}),t_{k,1}+ng})_{t_{k,3}}$
$: 0 \leq i \leq 1, 1 \leq k \leq nb$

$re_{6,i,k} \equiv \quad (dest_{i,t_{k,5},t_{k,1}+ng} \longrightarrow e_{t_{k,6}*i+(1-i)*(1-t_{k,6}),t_{k,1}+ng})_{t_{k,5}}$
$: 0 \leq i \leq 1, 1 \leq k \leq nb$

$re_{7,i,k} \equiv \quad (dest_{i,unt_{k-nb,3},unt_{k-nb,1}+ng+nb} \longrightarrow$
$e_{unt_{k-nb,4}*i+(1-i)*(1-unt_{k-nb,4}),unt_{k-nb,1}+ng+nb})_{unt_{k-nb,3}}$
$: 0 \leq i \leq 1, nb + 1 \leq k \leq nb + nu$

– Sending the values to the interaction environments:

$re_{8,i,k} \equiv \quad (\quad)_{t_{k,1}+ng}(e_{i,t_{k,1}+ng})_{t_{k,3}} \longrightarrow (a_i)_{t_{k,1}+ng}(\quad)_{t_{k,3}}$
$: 0 \leq i \leq 1, 1 \leq k \leq nb$

$re_{9,i,k} \equiv \quad (\quad)_{t_{k,1}+ng}(e_{i,t_{k,1}+ng})_{t_{k,5}} \longrightarrow (a_i)_{t_{k,1}+ng}(\quad)_{t_{k,5}}$
$: 0 \leq i \leq 1, 1 \leq k \leq nb$

$re_{8,i,k} \equiv \quad (\quad)_{unt_{k-nb,1}+ng+nb}(e_{i,unt_{k-nb,1}+ng+nb})_{unt_{k-nb,3}} \longrightarrow$
$(a_i)_{unt_{k-nb,1}+ng+nb}(\quad)_{unt_{k-nb,3}}$
$: 0 \leq i \leq 1, nb + 1 \leq k \leq nb + nu$

– Evaluating the result of the interactions (1/2).

* Binary interactions ot type **or**:

$rs_{11} \equiv \quad binop_1\ a_0{}^2[]_1 \longrightarrow binop_1\ c_0[]_1$
$rs_{12} \equiv \quad binop_1\ a_1{}^2[]_1 \longrightarrow binop_1\ c_1[]_1$
$rs_{13} \equiv \quad binop_1\ a_1\ a_0[]_1 \longrightarrow binop_1\ c_1[]_1$

* Binary interactions ot type **and**:

$rs_{14} \equiv \quad binop_2\ a_1{}^2[]_1 \longrightarrow binop_2\ c_1[]_1$
$rs_{15} \equiv \quad binop_2\ a_0{}^2[]_1 \longrightarrow binop_2\ c_0[]_1$
$rs_{16} \equiv \quad binop_2\ a_1\ a_0[]_1 \longrightarrow binop_2\ c_0[]_1$

* Binary interactions ot type **xor**:

$rs_{17} \equiv \quad binop_3\ a_1{}^2[]_1 \longrightarrow binop_3\ c_0[]_1$
$rs_{18} \equiv \quad binop_3\ a_0{}^2[]_1 \longrightarrow binop_3\ c_0[]_1$
$rs_{19} \equiv \quad binop_3\ a_1\ a_0[]_1 \longrightarrow binop_3\ c_1[]_1$

* Unary interactions of types **strong promotion**, **strong inhibition**, **weak promotion** and **weak inhibition**, respectively:

$rs_{23,i} \equiv \quad unop_1\ a_i[]_1 \longrightarrow unop_1\ c_i[]_1 : 0 \leq i \leq 1$
$rs_{24,i} \equiv \quad unop_2\ a_i[]_1 \longrightarrow unop_2\ c_{i-1}[]_1 : 0 \leq i \leq 1$
$rs_{25,i} \equiv \quad unop_3\ a_i[]_1 \longrightarrow unop_3\ c_i{}^i[]_1 : 0 \leq i \leq 1$
$rs_{26,i} \equiv \quad unop_4\ a_i[]_1 \longrightarrow unop_4\ c_{1-i}{}^i[]_1 : 0 \leq i \leq 1$

– Evaluating the result of the interactions (2/2).
For each interaction, objects of type eF are generated and sent to the target gene environment, depending on the previous result c_i and the type of the contribution (+ or -).

$re_{27,i,k} \equiv \quad (c_i)_{t_{k,1}+ng}(\quad)_{t_{k,7}} \longrightarrow$
$(\quad)_{t_{k,1}+ng}(eF_{tk,8*i+(1-i)*(1-t_{k,8}),t_{k,1}+ng})_{t_{k,7}}$
$: 0 \leq i \leq 1, 1 \leq k \leq nb$

$re_{28,i,k} \equiv \quad (c_i)_{unt_{k,1}+ng+nb}(\quad)_{unt_{k,5}} \longrightarrow$
$(\quad)_{unt_{k,1}+ng+nb}(eF_{i,(untk,1+ng+nb)})_{unt_{k,5}}$
$: 0 \leq i \leq 1, 1 \leq k \leq nu$

– The contribution of each interaction is calculated from the previously generated objects of type eF. These rules generate b_i objects whose multiplicity depends on the weight of the interaction.

$rs_{29,i,k} \equiv \quad eF_{i,(t_{k,1}+ng)}[]_1 \longrightarrow {b_i}^{t_{k,9}}[]_1 : 0 \leq i \leq 1, 1 \leq k \leq nb$

$rs_{30,i,k} \equiv \quad eF_{i,(unt_{k,1}+ng+nb)}[]_1 \longrightarrow {b_i}^{unt_{k,6}}[]_1 : 0 \leq i \leq 1, 1 \leq k \leq nu$

– Once the contribution of all the interactions over each gene has been received, the global influence over the gene is calculated. The next rule removes each pair of objects (b_1,b_0), whose contributions cancel each other.

$rs_{31} \equiv \quad b_1\, b_0[]_1 \longrightarrow []_1$

– The clock objects control the cycle flow, ensuring that all the contributions caused by the interactions and auto-influences have reached the target genes.

$rs_{32,i} \equiv \quad clock_{i-1}[]_1 \longrightarrow clock_i[]_1 : 1 \leq i \leq cc + 3$

– If objects b_0 are present, then the next state of the gene will be *inactive*. The object d_0 is created inside the membrane labelled by 1, and in a subsequent step will imply a new change of the charge of the membrane. Otherwise, any objects b_1 are removed, becoming the state of the gene *active*. The remaining objects (not used destination objects, for example) are removed from the configuration.

$rs_{33} \equiv \quad b_0[]_1^- \longrightarrow [d_0]_1^-$

$rs_{34} \equiv \quad b_1[]_1^- \longrightarrow []_1^-$

$rs_{35,i,j,k} \equiv \quad dest_{i,j,t_{k,1}+ng}[]_1^- \longrightarrow []_1^- : 0 \leq i \leq 1, 1 \leq j \leq ng, 1 \leq k \leq nb$

$rs_{36,i,j,k} \equiv \quad dest_{i,j,unt_{k-nb,1}+ng+nb}[]_1^- \longrightarrow []_1^-$
$: 0 \leq i \leq 1, 1 \leq j \leq ng, nb+1 \leq k \leq nb+nu$

$rs_{37} \equiv \quad [d_0]_1^- \longrightarrow []_1^+$

– Once the last step of the cycle is reached, the state of the gene is set to *active* (1) or *inactive* (0) depending of the charge of

membrane labelled by 1. Although electrical charges are no part of gene regulation, its use is required to set the state of the skin membrane of each environment, ensuring that all remaining objects d_0 are removed. In addition, the corresponding go objects are generated, the clock is removed and the charge of the membrane is reset to 0.

$$rs_{38} \equiv \quad clock_{cc+3}[]_1^+ \longrightarrow go\, a_0[]_1^0$$
$$rs_{39} \equiv \quad clock_{cc+3}[]_1^- \longrightarrow go\, a_1[]_1^0$$

TABLE 1: Parameters

Parameter	Description
	General parameters for the system
ng	Number of genes in the network
nb	Number of binary interactions
nu	Number of unary interactions
$threshold$	Maximum strength for an interaction
cc	Clock control
	Gene configuration parameters
$g_{i,1}$	Gene number (id)
$g_{i,3}$	Initial state of the gene
	Binary interactions parameters
$t_{i,1}$	Binary interaction number (id)
$t_{i,2}$	Interaction type (or: 1, and: 2, xor: 3)
$t_{i,3}$	1^{st} source gene number (id)
$t_{i,4}$	1^{st} source gene contribution (positive: 1, negative: 0)
$t_{i,5}$	2^{nd} source gene number (id)
$t_{i,6}$	2^{nd} source gene contribution (positive: 1, negative: 0)
$t_{i,7}$	Destination gene number (id)
$t_{i,8}$	Influence over destination gene (positive: 1, negative: 0)
$t_{i,9}$	Strength of the destination
	Unary interactions parameters
$unt_{i,1}$	Unary interaction number (id)
$unt_{i,2}$	Interaction type (strong promotion: 1, inhibition: 2; weak ones: 3, 4)
$unt_{i,3}$	Source gene number (id)
$unt_{i,4}$	Source gene contribution (positive, negative)
$unt_{i,5}$	Destination gene number (id)
$unt_{i,6}$	Influence over destination gene (positive, negative)

4.2 LN state interpretation

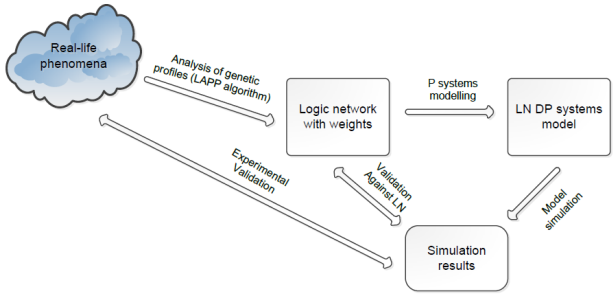
After the P system takes a predefined number of computation steps, the output information is analysed. This output information is encoded as the multiplicity of objects a_1 and a_0. Environments with an object a_1 represent active genes (a_0 represent inactive genes). Due to the nature of the system, membrane genes cannot have objects a_1 and a_0 simultaneously. If no object a_1 or a_0 is present within the membrane gene, then this membrane gene cannot be evaluated yet. That is, it will take some additional computation steps for the gene network to reach an evaluable state.

5 SIMULATION OF LOGIC NETWORKS IN MECOSIM

We have specified the model from section 4.1 on P–Lingua. This specification adheres to P–Lingua version 4 standard, available at [8]. Moreover, we have also developed a custom interface with MeCoSim [9, 11] to ease the introduction of specific data. MeCoSim defines a software interface with input and output tables and mappings from input data into model parameters. Then, it uses P–Lingua to simulate the model.

In this section, we define a methodology to simulate and analyse LNs. We start from a LN, possibly obtained by applying LAPP to a set of genetic profiles from real–life phenomena. Then, we instantiate its corresponding LN DP system. To do so, we load the model specification (available at [8]) and fill in the input tables. Following, we click on `Simulation > Simulate!` and visualize the results. The whole process is depicted in figure 3.

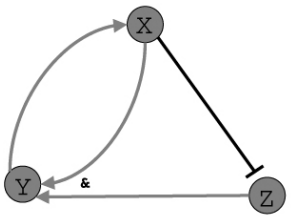
FIGURE 3: Methodology Overview



These results can be contrasted against real data, when available. These data are composed of initial and final network states. This assay is known as *experimental validation* [3], in contrast to formal validation.

As an example of application, figure 4 depicts a case study on a toy, 3–gene network extracted from [14]. In this example, all interactions have the same weight (say 100), so they are omitted from the picture. For a more detailed description of the whole process and a real–life case study, see [16].

FIGURE 4: Toy Example - MeCoSim Interface - Input Data



Network structure

t			t+1		
X	Y	Z	X	Y	Z
0	0	0	0	0	1
0	0	1	0	0	1
0	1	0	1	0	1
0	1	1	1	0	1
1	0	0	0	0	0
1	0	1	0	1	0
1	1	0	1	0	0
1	1	1	1	1	0

Network transitions

General parameters | Genes configuration | Unary interactions | Binary interactions

Number of genes	Number of binary int...	Number of unary int...	Maximum strength	Threshold	Clock control
3	1	2	0	50	10

General parameters | Genes configuration | Unary interactions | Binary interactions

Gene number (id)	Name	Value
1	X	1
2	Y	1
3	Z	1

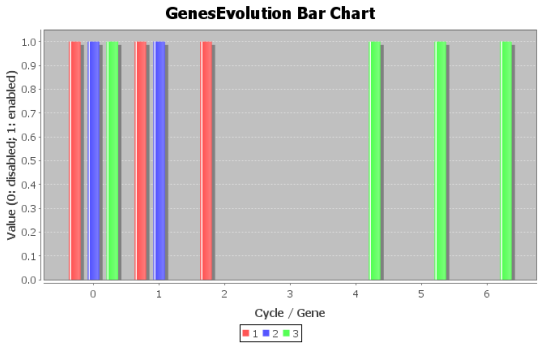
General parameters | Genes configuration | Unary interactions | Binary interactions

Interaction number	Type	1st source gene	Contribution	2nd source gene	Contribution	Destination gene	Influence	Strength
1	2	1	1	3	1	2	1	100

General parameters | Genes configuration | Unary interactions | Binary interactions

Interaction number	Type	Source gene	Contribution	Destination gene	Strength
1	1	2	1	1	100
2	2	1	1	3	100

Genes and interactions input tables



Results chart

6 CONCLUSIONS

In this work, we have presented a model for Genetic Networks based on P Systems. In contrast to ODEs, P systems do not require assumptions on the modelled phenomena, and display desirable features such as modularity [13]. The type of Gene Networks we have modelled are known as Logic Networks, in which one or more genes interact in order to influence another one. This model consists of a P system family (namely Logic Network Dynamic P Systems or *LN DP systems*) defined as an extension of Population Dynamic P Systems (*PDP Systems*), a Membrane Computing framework successfully applied on ecological modelling [3]. Our work proves the versatility of PDP systems by applying them to a completely different scenario from its original target phenomena. In addition, we provide a methodology to the simulation of LN DP systems on MeCoSim, illustrated with a toy example on a Gene Regulatory Network (GRN) taken from the literature [14].
As an additional complementary work, we propose the application of this model for large logic networks, such as *Arabidopsis thaliana*, a well–studied plant in systems biology. In this line, a first work has been published in [16]. We intend to keep this track by applying more well–grounded simulation methods, such as the Gillespie algorithm [5]. Another proposed line of work consists on a further enhancement by applying random mutations to the genes comprising the network. That is to say, we take into account dynamics in which gene states are not deterministically dictated by network interactions, but also subjected to random modifications. This enhancement could shed light into non–deterministic cell differentiation processes, so as to compare these new dynamics with the ones displayed by the deterministic model proposed here.

7 ACKNOWLEDGEMENTS

Luis Valencia–Cabrera, Manuel García-Quismondo and Mario J. Pérez-Jiménez are supported by project TIN2012-37434 from "Ministerio de Ciencia e Innovación" of Spain, co-financed by FEDER funds and "Proyecto de Excelencia con Investigador de Reconocida Valía P08-TIC-04200" from Junta de Andalucía. Manuel García-Quismondo is also supported by the National FPU Grant Programme from the Spanish Ministry of Education.

REFERENCES

[1] Peter M. Bowers, Shawn J. Cokus, Todd O. Yeates, and David Eisenberg. (2004). Use of logic relationships to decipher protein network organization. *Science*, 5705(306):2246–2249.

[2] Peter M. Bowers, Brian D. O'Connor, Shawn J. Cokus, Eniat Sprinzak, Todd O. Yeates, and David Eisenberg. (2005). Utilizing logical relationships in genomic data to decipher cellular processes. *the FEBS journal*, 272(1):5110–5118.

[3] M.Angels Colomer, Ignacio Pérez-Hurtado, Mario J. Pérez-Jiménez, and Agustín Riscos-Nez. (2012). Comparing simulation algorithms for multienvironment probabilistic p systems over a standard virtual ecosystem. *Natural Computing*, 11:369–379.

[4] Manuel García-Quismondo, Rosa Gutiérrez-Escudero, Miguel A. Martínez del Amor, Enrique Orejuela-Pinedo, and Ignacio Pérez-Hurtado. (September 2009). P-lingua 2.0: A software framework for cell-like P systems. *International Journal of Computers, Communications and Control*, IV:234–243.

[5] Daniel T. Gillespie. (1977). Exact stochastic simulation of coupled chemical reactions. *The Journal of Physical Chemistry*, 81(25):2340–2361.

[6] Thomas Hinze, Sikander Hayat, Thorsten Lenser, Naoki Matsumaru, and Peter Dittrich. (2007). Hill kinetics meets P systems: A case study on gene regulatory networks as computing agents *in silico* and *in vivo*. In George Eleftherakis, Petros Kefalas, Gheorghe Paun, Grzegorz Rozenberg, and Arto Salomaa, editors, *Workshop on Membrane Computing*, volume 4860 of *Lecture Notes in Computer Science*, pages 320–335. Springer.

[7] Reiichiro Kawai. (2012). Nonnegative compartment dynamical system modelling with stochastic differential equations. *Applied Mathematical Modelling*, page in press.

[8] P Lingua Web Page, (February 2009). http://www.p-lingua.org.

[9] MeCoSim Web Page, (July 2010). http://www.p-lingua.org/mecosim.

[10] Gheorghe Paun, Grzegorz Rozenberg, and Arto Salomaa. (2010). *The Oxford Handbook of Membrane Computing*. Oxford University Press, Inc., New York, NY, USA.

[11] Ignacio Pérez-Hurtado, Luis Valencia-Cabrera, Mario J. Pérez-Jiménez, M. A. Colomer, and Agustín Riscos-Núñez. (2010). MeCoSim: a general purpose software tool for simulating biological phenomena by means of P systems. *IEEE Fifth International Conference on Bio-inpired Computing: Theories and Applications (BIC-TA 2010)*, I:637–643.

[12] Gheorghe Păun. (2000). Computing with membranes. *Journal of Computer and System Sciences*, 61:108–143.

[13] Francisco José Romero-Campero and Mario J. Pérez-Jiménez. (2008). A model of the quorum sensing system in *vibrio fischeri* using P systems. *Artificial Life*, 14(1):95–109.

[14] Thomas Schlitt and Alvis Brazma. (2007). Current approaches to gene regulatory network modelling. *BMC Bioinformatics*, 8(Suppl 6):S9.

[15] Ilya Shmulevich and Edward R. Dougherty. (2010). *Probabilistic Boolean Networks - The Modeling and Control of Gene Regulatory Networks*. SIAM.

[16] Luis Valencia-Cabrera, Manuel García-Quismondo, Yansen Su, Mario J. Pérez-Jiménez, Hui Yu, and Linqiang Pan. (February 2013). Analysing gene networks with PDP systems. *Arabidopsis thaliana*, a case study. In *Proceedings of 11th Brainstorming Week on Membrane Computing (BWMC13)*, in press. Sevilla, Spain. Fénix Editora.

[17] Shudong Wang, Yan Chen, Qingyun Wang, Eryan Li, Yansen Su, and Dazhi Meng. (2010). Analysis for gene networks based on logic relationships. *Journal of Systems Science and Complexity*, 23:999–1011. 10.1007/s11424-010-0205-0.