Probabilistic Guarded P Systems, A Formal Definition

Manuel García-Quismondo, Miguel A. Martínez-del-Amor,
Mario J. Pérez-Jiménez

Research Group on Natural Computing
Department of Computer Science and Artificial Intelligence
University of Seville
Avda. Reina Mercedes s/n, 41012 Sevilla, Spain
mgarciaquismondo@us.es, mdelamor@us.es, marper@us.es

Summary. In this paper, we extend the general framework of Multienvironment P systems, which is a formal framework for modelling the dynamics of population biology. The extension is made by a new variant within the probabilistic approach, called Probabilistic Guarded P systems (in short, PGP systems). We provide a formal definition, a simulation algorithm to capture the dynamics, and a survey of the associated software.

1 Introduction

Since P systems were introduced in 1998 [18], they have been utilised as a high level computational modelling framework [9, 19]. Their main advantage is the integration of the structural and dynamical aspects of complex systems in a comprehensive and relevant way, while providing the required formalisation to perform mathematical and computational analysis [2].

In this respect, multienvironment P systems are a general formal framework for population dynamics modelling in Biology [6]. This framework has two approaches: stochastic and probabilistic. Stochastic approach is usually applied to model *micro*-level systems (such as bacteria colonies), whereas the probabilistic approach is normally used for *macro*-level modelling (real ecosystems, for example). Population Dynamics P systems [2, 15, 16, 3] (PDP systems, in short) are a variant of multienvironment P systems, in the probabilistic approach. PDP systems have been successfully applied to ecological modelling, specially with real ecosystems of some endanger [5, 3] and exotic species [3]. PDP systems have shown to comply with four desirable properties of a computational model [2]: *relevance* (capture the essential features of the modelled system), *computability* (inherent by P systems), *understandability* (objects and rules capture the dynamics in a simple way), and *extensibility* (rule design is module-oriented).

In this paper, we introduce a brand new variant inside the probabilistic approach of multienvironment P systems: Probabilistic Guarded P systems (*PGP systems*, for short). They are specifically oriented for ecological processes. PGP systems are a computational probabilistic framework which takes inspiration from different Membrane Computing paradigms, mainly from Tissue–Like P systems [22], PDP systems [2] and Kernel P systems [11]. This framework aims for simplicity, considering these aspects:

Model designers: In PGP systems, model designers do not need to worry about context consistency. That is to say, they do not need to take into account that all rules simultaneously applied in a cell must define the same polarization in the right–hand side [15]. This is because the framework centralizes all context changes in a single rule per cycle, rather than distributing them across all rules. Therefore, there exist two types of rules: *context–changing* rules and *non context–changing* rules. Due to the nature of the model, only one of such rules can be applied at the same time on each cell, so context inconsistency is not possible. Moreover, the fact that the context is explicitly expressed in each cell and that cells do not contain internal cell structures simplifies transitions between contexts without loss of computational or modelling power.

Simulator developers: The fact that the framework implicitly takes care of context consistency simplifies the development of simulators for these models, as it is a non–functional requirement which does not need to be supported by simulators. In addition, the lack of internal structure in cells simplifies the simulation of object transmission; the model can be regarded as a set of memory regions with no hierarchical arrangement, thus enabling direct region fetching.

Probabilistic Guarded P Systems can be regarded as an evolution of Population Dynamic P systems. In this context, PGP systems propose a modelling framework for ecology in which inconsistency (that is to say, undefined context of membranes) is handled by the framework itself, rather than delegating to simulation algorithms. In addition, by replacing alien concepts to biology (such as electrical polarizations and internal compartment hierarchies) by state variables known as *flags* and defined by designers models are more natural to experts, thus simplifying communication between expert and designer.

This paper is structured as follows. Section 2 introduces some preliminaries. Section 3 shows the formal framework of multienvironment P systems, and the two main approaches. Section 4 describes the framework of PGP systems, providing a formal definition, some remarks about the semantics of the model, and a comparison with other similar frameworks of Membrane Computing. Section 5 provides a simulation algorithm, and a software environment based on P–Lingua and a C++ simulator. Section 6 summarizes an ecosystem under study with PGP systems. Finally, Section 7 ends the paper with conclusions and future work.

2 Preliminaries

An *alphabet* Γ is a non–empty set whose elements are called *symbols*. An ordered finite sequence of symbols of Γ is a *string* or *word* over Γ. As usual, the empty string (with length 0) will be denoted by λ. The set of all strings over an alphabet Γ is denoted by Γ^*. A *language* over Γ is a subset of Γ^*.

A *multiset* m over an alphabet Γ is a pair $m = (\Gamma, f)$ where $f : \Gamma \to \mathbb{N}$ is a mapping. For each $x \in \Gamma$ we say that $f(x)$ is the *multiplicity* of the symbol x in m. If $m = (\Gamma, f)$ is a multiset then its *support* is defined as $supp(m) = \{x \in \Gamma \mid f(x) > 0\}$. A multiset is finite if its support is a finite set. A *set* is a multiset such that the multiplicity of each element of its support, is equal to 1.

If $m = (\Gamma, f)$ is a finite multiset over Γ, and $supp(m) = \{a_1, \ldots, a_k\}$ then it will be denoted as $m = a_1^{f(a_1)} \ldots a_k^{f(a_k)}$ (here the order is irrelevant), and we say that $f(a_1) + \cdots + f(a_k)$ is the cardinal of m, denoted by $|m|$. The empty multiset is denoted by $\emptyset$. We also denote by $M_f(\Gamma)$ the set of all finite multisets over Γ.

Let $m_1 = (\Gamma, f_1)$ and $m_2 = (\Gamma, f_2)$ multisets over Γ. We define the following concepts:

- The union of m_1 and m_2, denoted by $m_1 + m_2$ is the multiset (Γ, g), where $g = f_1 + f_2$, that is, $g(x) = f_1(x) + f_2(x)$ for each $x \in \Gamma$.
- The relative complement of m_2 in m_1, denoted by $m_1 \setminus m_2$ is the multiset (Γ, g), where $g = f_1(x) - f_2(x)$ if $f_1(x) \geq f_2(x)$ and $g(x) = 0$ otherwise.

We also say that m_1 is a submultiset of m_2, denoted by $m_1 \subseteq m_2$, if $f_1(x) \leq f_2(x)$ for each $x \in \Gamma$.

Let $m = (\Gamma, f)$ a multiset over Γ and A a set. We define the intersection $m \cap A$ as the multiset (Γ, g), where $g(x) = f(x)$ for each $x \in \Gamma \cap A$, and $g(x) = 0$ otherwise.

3 Multienvironment P systems

Definition 1. *A multienvironment P system of degree (q, m, n) with $q \geq 1$, $m \geq 1$, taking T time units, $T \geq 1$, is a tuple*

$$\Pi = (G, \Gamma, \Sigma, \Phi, T, n, \{\Pi_{k,j} \mid 1 \leq k \leq n,\ 1 \leq j \leq m\}, \{(f_j, E_j) \mid 1 \leq j \leq m\}, \mathcal{R}_E)$$

where:

- *$G = (V, S)$ is a directed graph. Let $V = \{e_1, \ldots, e_m\}$ whose elements are called environments;*
- *Γ, Σ and Φ are finite alphabets such that $\Sigma \subsetneq \Gamma$ and $\Gamma \cap \Phi = \emptyset$.*
- *T, n are natural numbers*
- *For each k, j ($1 \leq k \leq n$, $1 \leq j \leq m$), $\Pi_{k,j}$ is a tuple $(\Gamma, \mu, \mathcal{M}_{1,j}^k, \ldots, \mathcal{M}_{q,j}^k, \mathcal{R}_j, i_{in})$, where:*

  - *μ is a rooted tree with $q \geq 1$ nodes labelled by elements from $\{1, \ldots, q\} \times \{0, +, -\}$.*
  - *For each i, $1 \leq i \leq q$, $\mathcal{M}_{i,j}^k \in M_f(\Gamma)$.*
  - *$\mathcal{R}_j$ is a finite set of rules of the type: $u[v]_i^\alpha \xrightarrow{p} u'[v']_i^{\alpha'}$, being $u, v, u', v' \in M_f(\Gamma)$, $1 \leq i \leq q$, $\alpha, \alpha' \in \{0, +, -\}$ and p is a computable function whose domain is $\{0, \ldots, T\}$.*
  - *i_{in} is a node from μ.*
- *For each j, $1 \leq j \leq m$, $f_j \in \Phi$ and $E_j \in M_f(\Sigma)$.*
- *$\mathcal{R}_E$ is a finite set of rules among environments of the types:*

$$(x)_{e_j} \xrightarrow{p_1} (y_1)_{e_{j_1}} \cdots (y_h)_{e_{j_h}} \qquad (\Pi_{k,j})_{e_j} \xrightarrow{p_2} (\Pi_{k,j})_{e_{j_1}}$$
$$\{f\}\,(u)_{e_j} \xrightarrow{p_3} (v)_{e_{j_1}} \qquad \{f\}\,(u, f)_{e_j} \xrightarrow{p_4} (v, g)_{e_j}$$

being $x, y_1, \ldots y_h \in \Sigma$, $(e_j, e_{j_i}) \in S$, $1 \leq j \leq m, 1 \leq i \leq h$, $1 \leq k \leq n, f, g \in \Phi, u, v \in M_f(\Gamma)$ and p_1, p_2, p_3, p_4 are computable functions whose domain is $\{0, \ldots, T\}$.

In other words, 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 has a flag from Φ at any instant and also it can contains objects from Σ and P systems of the type $\Pi_{k,j} = (\Gamma, \mu, \mathcal{M}_{1,j}^k, \ldots, \mathcal{M}_{q,j}^k, \mathcal{R}_j^k, i_{in})$. Multisets $\mathcal{M}_{1j}^k, \ldots, \mathcal{M}_{q,j}^k$ describe the initial multisets of $\Pi_{k,j}$ corresponding to this environment. Every rule $r \in \mathcal{R}_j^k$ has a computable function $f_{r,j}$ (specific for environment j) associated with it.

In total, there are n systems $\Pi_{k,j}$, all of them with the same skeleton (identical working alphabets, objects and **flags**, the same membrane structure and the same rules $u[v]_i^\alpha \longrightarrow u'[v']_i^{\alpha'}$, specified in each environment (independently of k) through the computable function $f_{r,j}$ associated with them).

A *configuration* of the system at any instant t is a tuple whose components are the following: (a) the flags associated with each environment at instant t (initially $f_1, \ldots, f_m$); (b) the multisets of objects present in the m environments at instant t (initially $E_1, \ldots, E_m$); and (c) the multisets of objects associated with each of the regions of each P system $\Pi_{k,j}$ (initially $\mathcal{M}_{1,j}^k, \ldots, \mathcal{M}_{q,j}^k$), together with the polarizations of their membranes (initially 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 $\mathcal{R}_\mathcal{E}$ and $\mathcal{R}$) are synchronized in all environments.

The P system can pass from one configuration to another by using the rules from $\mathcal{R} = \mathcal{R}_E \cup \bigcup_{j=1}^m \mathcal{R}_j^k$ 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.

A rule of the type $u[v]_i^\alpha \xrightarrow{p} u'[v']_i^{\alpha'}$ is applicable to a configuration at any instant t if the following is satisfied: in that configuration membrane i

has polarization α, contains multiset v and its parent (the environment if the membrane is the skin membrane) contains multiset u. When that rule is applied, multisets u, v produce u', v', respectively, and the new polarization is α' (the value of function p in that moment provide the affinity of the application of that rule). 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 $\mathcal{R}_j^k$ simultaneously to the same membrane, all the rules must have the same electrical charge on their right-hand side.

A rule of the environment of the type $(x)_{e_j} \xrightarrow{p_1} (y_1)_{e_{j_1}} \cdots (y_h)_{e_{j_h}}$ is applicable to a configuration at any instant t if the following is satisfied: in that configuration environment e_j contains object x. When that rule is applied, object x passes from e_j to $e_{j_1}, \ldots, e_{j_h}$ possibly transformed into objects $y_1, \ldots, y_h$, respectively (the value of function p_1 in that moment provide the affinity of the application of that rule).

A rule of the environment of the type $(\Pi_{k,j})_{e_j} \xrightarrow{p_2} (\Pi_{k,j})_{e_{j'}}$: is applicable to a configuration at any instant t if the following is satisfied: in that configuration environment e_j contains the P system $\Pi_{k,j}$. When that rule is applied, the system $\Pi_{k,j}$ passes from environment e_j to environment $e_{j'}$ (the value of function p_2 in that moment provide the affinity of the application of that rule).

A rule of the environment of the type $\{f\}(u)_{e_j} \xrightarrow{p_3} (v)_{e_{j_1}}$ is applicable to a configuration at any instant t if the following is satisfied: in that configuration environment e_j has flag f and contains the multiset u. When that rule is applied multiset u produces multiset v and environment e_j keep the same flag. This kind of rule can be applied many times in a computation step. The value of function p_3 in that moment provide the affinity of the application of that rule.

A rule of the environment of the type $\{f\}(u, f)_{e_j} \xrightarrow{p_4} (v, g)_{e_j}$ is applicable to a configuration at any instant t if the following is satisfied: in that configuration environment e_j has flag f and contains the multiset u. When that rule is applied multiset u produces multiset v and flag f of environment e_j is replaced by flag g. Bearing in mind that each environment only has a flag in any instant, this kind of rules can only be applied once in any moment. Hence, the value of the function p_4 in any instant is equal to 1.

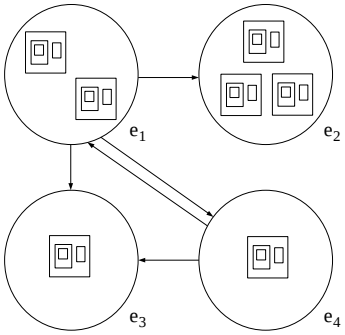
Next, we depict the two approaches (stochastic and probabilistic) for multienvironment P systems.

3.1 Stochastic approach

We say that a multienvironment P system has a stochastic approach if the following holds:

(a) The alphabet of flags, Φ, is an empty set.
(b) The computable functions associated with the rules of the P systems are **propensities** (obtained from the kinetic constants): These rules is function of the time but they do not depend on the environment.

(c) The P systems $\Pi_{k,j}$ do not depend on index j, this index is irrelevant in this approach.
(d) Initially, the P systems $\Pi_{k,j}$ are randomly distributed among the m en environments of the system.



Multicompartmental P systems

Multicompartmental P systems are multienvironment P systems with a stochastic approach which can be formally expressed as follows:

$$\Pi = (G, \Gamma, \Sigma, T, n, \{\Pi_{k,j} \,|\, 1 \leq k \leq n,\, 1 \leq j \leq m\}, \{E_j \,|\, 1 \leq j \leq m\}, \mathcal{R}_E)$$

These systems can be viewed as a set of m environment connected by the arcs of a directed graph G. Each environment e_j only can contains P systems of the type $\Pi_{k,j}$. The total number of P systems is n, all of them with the same skeleton. The functions associated with the rules of the system are propensities which are computed as follows: stochastic constants are computed from kinetic constants by applying the mass action law, and the propensities are obtained from the stochastic constants by using the concentration of the objects in the LHS at any instant. In these systems there are rules of the following types:

1. $u[v]_i^\alpha \xrightarrow{p} u'[v']_i^{\alpha'}$
2. $(x)_{e_j} \xrightarrow{p_1} (y_1)_{e_{j_1}} \cdots (y_h)_{e_{j_h}}$
3. $(\Pi_{k,j})_{e_j} \xrightarrow{p_2} (\Pi_{k,j})_{e_{j'}}$

The dynamics of these systems is captured by the multicompartmental Gillespie's algorithm [21] or the deterministic waiting time [4]. A software environment supporting this model is Infobiotics Workbench [1], which provides (in version 0.0.1): a modelling language, a multi-compartmental stochastic simulator based on Gillespies Stochastic Simulation Algorithm, a formal model analysis, and a structural and parameter model optimisation.

3.2 Probabilistic approach

We say that a multienvironment P system has a stochastic approach if the following holds:

(a) The total number of P systems $\Pi_{k,j}$ is, at most, the number m of environment, that is, $n \leq m$.

(b) Functions p_r associated with rule $r \equiv u[v]_i^\alpha \xrightarrow{p_r} u'[v']_i^{\alpha'}$ from $\Pi_{k,j}$ are **probability functions** such that for each $u, v \in M_f(\Gamma)$, $i \in \{1, \ldots, q\}$, $\alpha \in \{0, +, -\}$, if $r_1, \ldots, r_z$ are the rules in R_j^k whose LHS is $u\ [\ v\ ]_i^\alpha$, then $\sum_{j=1}^{z} p_{r_j}(t) = 1$, for each t $(1 \leq t \leq T)$.

(c) Functions p_1 associated with the rules of the environment $(x)_{e_j} \xrightarrow{p_1} (y_1)_{e_{j_1}} \cdots (y_h)_{e_{j_h}}$ are **probability functions** such that for each $x \in \Sigma$ and each environment e_j, the sum of all functions associated with the rules whose LHS is $(x)_{e_j}$, is equal to 1.

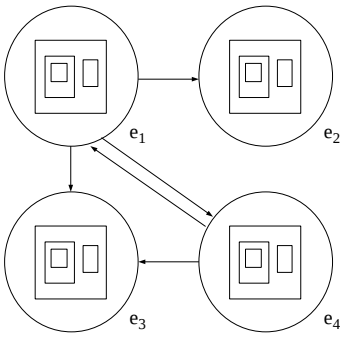
(d) Functions p_2 associated with the rules of the environment $(\Pi_{k,j})_{e_j} \xrightarrow{p_2} (\Pi_{k,j})_{e_{j'}}$ are constant functions equal to 0; that is, these rules will never be applied.

(e) Functions p_3 associated with the rules of the environment $\{f\}\,(u)_{e_j} \xrightarrow{p_3} (v)_{e_{j_1}}$ are **probability functions**.

(f) Functions p_4 associated with the rules of the environment $\{f\}\,(u, f)_{e_j} \xrightarrow{p_4} (v, g)_{e_j}$ are constant functions equal to 1.

(g) There is no rules $u[v]_i^\alpha \xrightarrow{p} u'[v']_i^{\alpha'}$ in the skin membranes of $\Pi_{k,j}$ and rules of the environment $(x)_{e_j} \xrightarrow{p_1} (y_1)_{e_{j_1}} \cdots (y_h)_{e_{j_h}}$ such that $x \in u$.

(h) Initially, each environment e_j contains at most one P system $\Pi_{k,j}$.



Population Dynamics P systems (PDP)

Population Dynamics P systems are multienvironment P systems with a probabilistic approach such that the alphabet Φ of the flags is an empty set and $n = m$, that is, the environment have not any flag and the total number n of P systems are equal to the number m of environments. Then in a PDP system each environment e_j contains exactly one P system $\Pi_{k,j}$ which will be denoted by Π_j

$$\Pi = (G, \Gamma, \Sigma, T, n, \{\Pi_j \mid 1 \leq j \leq m\}, \{E_j \mid 1 \leq j \leq m\}, \mathcal{R}_E)$$

In these systems there are rules of the following types:

1. $u[v]_i^\alpha \xrightarrow{p} u'[v']_i^{\alpha'}$
2. $(x)_{e_j} \xrightarrow{p_1} (y_1)_{e_{j_1}} \cdots (y_h)_{e_{j_h}}$

Let us recall that in theses kind of systems each rule has associated a probability function that depends on the time and on the environment where the rule is applied.

The dynamics of these systems is captured by the **D**irect **N**on-deterministic **D**istribution algorithm with **P**robabilities (**DNDP**) algorithm [16], or the **D**irect distribution based on **C**onsistent **B**locks **A**lgorithm (**DCBA**) [15]. DNDP aims to perform a random distribution of rule applications without using the concept of rule block, but this selection process is biased towards those rules with the highest probabilities. DCBA was first conceived to overcome the accuracy problem of DNDP, by performing an object distribution along the rule blocks, before applying the random distribution process. Although the accuracy achieved by the DCBA is better than the DNDP algorithm, the latter is much faster. In order to improve the performance of simulators implementing DCBA, parallel architectures has been used [14]. For example, a GPU-based simulator, using CUDA, reaches the acceleration of up to 7x, running on a NVIDIA Tesla C1060 GPU (240 processing cores). However, these accelerated simulators are still to be connected to those general environments to run virtual experiments. Therefore, P–Lingua and pLinguaCore are being utilised to simulate PDP systems [2, 10]. The provided virtual experimentation environment is called MeCoSim [20], and it is based on P–Lingua.

4 Probabilistic Guarded P systems (PGP)

Probabilistic Guarded P systems are multienvironment P systems with a probabilistic approach such that $n = 0$, that is, there is no P systems $\Pi_{k,j}$ (so the alphabet Γ can be considered as an emptyset), and the alphabet of the environment, Σ, and the alphabet of the flags, Φ are disjoint.

Definition 2. *A Probabilistic Guarded P system* (PGP system*, for short*) *of degree* $m \geq 1$ *is a tuple* $\Pi = (G, \Sigma, \Phi, T, \{(f_j, E_j) \mid 1 \leq j \leq m\}, \mathcal{R}_E)$*, where:*

- $G = (V, S)$ *is a directed graph whose set of nodes is* $V = \{e_1, \ldots, e_m\}$.
- Σ *and* Φ *are finite alphabets such that* $\Sigma \cap \Phi = \emptyset$. *Elements in* Σ *are called* ***objects*** *and elements in* Φ *are called* ***flags***.
- $\mathcal{R}_E$ *is a finite set of rules of the following types:*
  - $\{f\}(u)_{e_j} \to (v)_{e_{j_1}}$ *with* $u, v \in M_f(\Sigma)$, $f \in \Phi$ *and* $1 \leq j, j_1 \leq m$.
  - $\{f\}(u, f)_{e_j} \to (v, g)_{e_j}$ *with* $u, v \in M_f(\Sigma)$, $f, g \in \Phi$ *and* $1 \leq j \leq m$.

  There is no rules of types $\{f\}(u, f)_{e_j} \to (v, g)_{e_j}$ *and* $\{f\}(u)_{e_j} \xrightarrow{p_3} (v)_{e_{j_1}}$, *for* $f \in \Phi$, $1 \leq j, j_1 \leq m$ *and* $u \in M_f(\Sigma)$.

  For each $f \in \Phi$ *and* $j, 1 \leq j \leq m$, *there exists only one rule of type* $\{f\}(u, f)_{e_j} \to (v, g)_{e_j}$.
- *The arcs of graph* $G = (V, S)$ *is defined from* $\mathcal{R}_E$ *as follows:* $(e_j, e_{j_1}) \in S$ *if and only if there exists a rule of the type* $\{f\}(u)_{e_j} \to (v)_{e_{j_1}}$, *or* $j = j_1$ *and there exists a rule of the type* $\{f\}(u, f)_{e_j} \to (v, g)_{e_j}$.
- *Each rule from* $\mathcal{R}_E$ *has associated a probability, that is, there exists a function* $p_{\mathcal{R}_E}$ *from* $\mathcal{R}_E$ *into* $[0, 1]$, *such that:*
  - *For each* $f \in \Phi, u \in M(\Sigma), 1 \leq j \leq m$, *if* $r_1, \ldots, r_t$ *are rules of the type* $\{f\}(u)_{e_j} \to (v)_{e_{j_1}}$, *then* $\sum_{s=1}^{t} p_{\mathcal{R}_E}(r_s) = 1$.
  - *If* $r \equiv \{f\}(u, f)_{e_j} \to (u, g)_{e_j}$, *then* $p_{\mathcal{R}_E}(r) = 1$.
- *For each* $j, 1 \leq j \leq m$, *we have* $f_j \in \Phi$ *and* $E_j \in M_f(\Sigma)$.

A Probabilistic Guarded P system can be viewed as a set of m environments, called *cells*, labelled by $1, \ldots, m$ such that: (a) $E_1, \ldots, E_m$ are finite multisets over Σ representing the objects initially placed in the cells of the system; (b) $f_1, \ldots, f_m$ are flags that initially mark the cells; (c) G is a directed graph whose arcs specify connections among cells; (d) $\mathcal{R}_E$ is the set of rules that allow the evolution of the system and each rule r is associated with a real number $p_{\mathcal{R}_E}(r)$ in $[0, 1]$ describing the probability of that rule to be applied in the case that it is applicable.

In PGP systems, two types of symbols are used: *objects* (elements in Σ) and *flags* (elements in Φ). It can be considered that objects are **in** cells and flags are **on** (the borderline of) cells.

A *configuration* of a PGP system at any instant t is a tuple whose components are the following: (a) the flags associated with each cell at instant t (initially $f_1, \ldots, f_m$), and (b) the multisets of objects present in the m cells at instant t (initially $E_1, \ldots, E_m$).

Finally, in order to ease the understandability of the whole framework, Figure 1 shows a graphical summary of multienvironment P systems and the two approaches (stochastic and probabilistic).

4.1 Semantics of PGP systems

Definition 3. *A* configuration *at any instant* $t \geq 0$ *of a PGP system* Π *is a tuple* $\mathcal{C}_t = (x_1, u_1, \ldots, x_m, u_m)$ *where, for each* i, $1 \leq i \leq m$, $x_i \in \Phi$ *and* $u_i \in M(\Sigma)$. *That is to say, a configuration of* Π *at any instant* $t \geq 0$ *is described by all multisets of objects over* Σ *associated with all the cells present in the system and the flags*

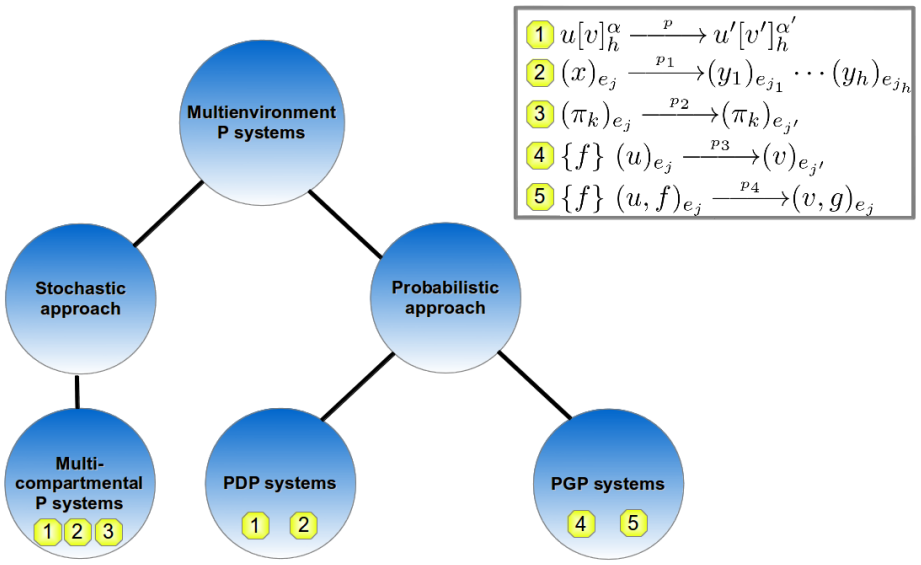


Fig. 1: The formal framework of Multienvironment P systems

marking these cells. $(f_1, E_1, \ldots, f_m, E_m)$ *is said to be the* initial configuration *of* Π*. At any instant, each cell has one and only one flag, in a similar manner to polarizations in cell–like P systems.*

Definition 4. *A rule* r *of the type* $\{f\}\,(u)_i \to (v)_j$ *is* applicable *to a configuration* $\mathcal{C}_t = (x_1, u_1, \ldots x_m, u_m)$ *if and only if* $x_i = f$ *and* $u \subseteq u_i$*, for all* $1 \leq i \leq m$*.*

When applying r *to* C_t*, objects in* u *are removed from cell* i *and objects in* v *are produced in cell* j*. Flag* f *is not changed; it plays the role of a catalyst assisting the evolution of objects in* u*.*

Definition 5. *A rule* r *of the type* $\{f\}\,(u,f)_i \to (v,g)_i$ *is* applicable *to a configuration* $\mathcal{C}_t = (x_1, u_1, \ldots x_m, u_m)$ *if and only if* $x_i = f$ *and* $u \subseteq u_i$*, for all* $1 \leq i \leq m$*.*

When applying r *to* C_t*, in cell* i *objects in* u *are replaced by those in* v *and* f *is replaced by* g*. In this case, Flag* f *is consumed, so* r *can be applied only once in instant* t *in cell* i*.*

Remark 1. After applying a rule r of the type $\{f\}\,(u,f)_i \to (v,g)_i$, other rules r' of the type $\{f\}\,(u)_i \to (v)_j$ can still be applied (the flag remains in vigour). However, f has been consumed, so no more rules of the type $\{f\}\,(u,f)_i \to (v,g)_i$ can be applied.

Definition 6. *A configuration is a* halting configuration *if no rule is applicable to it.*

Definition 7. *We say that configuration* $\mathcal{C}_1$ yields *configuration* $\mathcal{C}_2$ *in a transition step if we can pass from* $\mathcal{C}_1$ *to* $\mathcal{C}_2$ *by applying rules from* $\mathcal{R}_\mathcal{E}$ *in a non–deterministic, maximally parallel manner, according to their associated probabilities denoted by map* $p_{\mathcal{R}_\mathcal{E}}$*. That is to say, a maximal multiset of rules from* $\mathcal{R}_\mathcal{E}$ *is applied, no further rule can be added.*

Definition 8. *A* computation *of a PGP system* Π *is a sequence of configurations such that: (a) the first term of the sequence is the initial configuration of* Π*, (b) each remaining term in the sequence is obtained from the previous one by applying the rules of the system following Definition 7, (c) if the sequence is finite (called halting computation) then the last term of the system is a halting configuration.*

4.2 Comparison between PGP systems and other frameworks in Membrane Computing

Probabilistic Guarded P systems (*PGP systems*) display similarities with other frameworks in Membrane Computing. As a sample, in *P systems with proteins on membranes* are a type of cell-like systems in which membranes might have attached a set of proteins which regulate the application of rules, whilst in PGP systems each cell has only one flag. Therefore, some rules are applicable if and only if the corresponding protein is present. More information about this kind of P systems can be found in [17].

When comparing PGP systems and *Population Dynamics P systems* [2], it is important to remark the semantic similarity between flags and polarizations, as they both define at some point the context of each compartment. Nevertheless, as described at the beginning of this chapter, upon the application of a rule $r \equiv \{f\}\,(u, f)_i \rightarrow (v, g)_i$ flag f is consumed, thus ensuring that r can be applied at most once to any configuration. This property keeps PGP transitions from yielding inconsistent flags; at any instant, only one rule at most can change the flag in each membrane, so scenarios in which inconsistent flags produced by multiple rules are impossible. Moreover, in PDP systems the number of polarizations is limited to three (+, - and 0), whereas in their PGP counterpart depends on the system itself. Finally, each compartment in PDP systems contains a hierarchical structure of membranes, which is absent in PGP systems. Figure 2 summarizes this comparison.

5 Simulation of PGP systems

When simulating PGP systems, there exist two cases, according to if there exists object competition or not. In this work, only algorithms for the second case are

	PGP systems	**P systems with proteins**	**PDP systems**
Structure	Tissue–like (given by a directed graph)	Cell–like (given by a rooted tree)	Tissue–like (given by a directed graph of environments containing a rooted tree each)
Rule	Each left–hand side contains one flag and a multiset of objects	Each left–hand side contains one protein and one object	Each left–hand side contains one polarization and a multiset of objects
Affected compartments	The application of a rule might affect, at most, two cells in the system	The application of a rule affects one and only one cell in the system	The application of a rule might affect, at most, two cells in the system
Number of applications	Each rule of type $r \equiv \{f\}\,(u, f)_i \to (v, g)_i$ can be applied, at most, only once to any configuration	Every rule is possible to be applied multiple times to any configuration	Every rule is possible to be applied multiple times to any configuration
Number of flags	For each configuration, there exists only one flag per cell	For each configuration, there might exist multiple proteins per cell	For each configuration, there exists only one polarization per cell

Fig. 2: Comparison of PGP systems, PDP systems and P systems with proteins

introduced, but some ideas are given to handle object competition among rules in the model, and kept for future developments.

5.1 Some definitions on the model

As it is the case in Population Dynamic P systems, in PGP systems some definitions are introduced prior to describing simulation algorithms. It must be noted that these concepts are analogous to those described in [15], but obviously adapted to the syntax of PGP systems.

Remark 2. For the sake of simplicity, henceforth the following notation will be used. For every cell i , $1 \leq i \leq m$, and time t , $0 \leq t \leq T$, the flag and multiset of cell i in step t are denoted as $x_{i,t} \in \Phi$ and $u_{i,t} \in M(\Sigma)$, respectively. Similarly, $u(y)$, where $u \in M(\Sigma),\ y \in \Sigma$ denote the number of objects y in multiset u.

Definition 9 shows the notation regarding the left-hand and right-hand sides of rules.

Definition 9. *For each rule $r \in \mathcal{R}_{\mathcal{E}}$:*

Type 1: If r is of the form $r \equiv \{f\}\,(u)_i \rightarrow (v)_j$, we denote the left–hand side as $LHS(r) = (i, f, u)$ and the right–hand side as $RHS(r) = (j, f, v)$.
Type 2: If r is of the form $r \equiv \{f\}\,(u, f)_i \rightarrow (v, g)_i$, we denote the left–hand side as $LHS(r) = (i, f, u, f)$ and the right–hand side as $RHS(r) = (i, g, v)$.

Let us recall that for each i , $1 \leq i \leq m$ and $f \in \Phi$, there exists an *unique* rule of type 2: $r \equiv \{f\}\,(u, f)_i \rightarrow (v, g)_i$.

Next, Definition 10 introduces the concept of rule blocks in PGP systems, which is inspired by the one used in PDP systems [15].

Definition 10. *For each $1 \leq i \leq q$, $f \in \Phi$, and $u \in M(\Sigma)$, we will denote:*

- *The* block of communication rules $B^1_{i,f,u} = \{r \in \mathcal{R} : LHS(r) = (i, f, u)\}$*; that is, the set of rules of type 1 having the same left–hand side.*
- *The* block of context–changing rules $B^2_{i,f,u} = \{r \in \mathcal{R} : LHS(r) = (i, f, u, f)\}$*; that is, the set of rules of type 2 having the same left–hand side.*

Obviously, $B^1_{i,f,u} \cap B^2_{i,f,u} = \emptyset$. It is important to recall that, as it is the case in PDP systems, the sum of probabilities of all the rules belonging to the same block is always equal to 1 – in particular, rules with probability equal to 1 form individual blocks. Consequently, blocks of context–changing rules (type 2) are composed solely of a rule. In addition, rules with overlapping (but different) left–hand sides are classified into different blocks.

Definition 11. *For each $i, 1 \leq i \leq m$, we will consider the set of all rule blocks associated with cell i as $B_i = \{B^1_{i,f,u}, B^2_{i,f,u} : f \in \Phi \wedge u \in M(\Sigma)\}$.*

We will also consider a total order in B_i, for $1 \leq i \leq m$, $B_i = \{B_{i,1}, B_{i,2}, \ldots, B_{i,\alpha_i}\}$. Therefore, there are α_i blocks associated to cell i.

Furthermore, let $B_{i,j}$, $1 \leq i \leq m$, $1 \leq j \leq \alpha_i$ be a block associated to cell i. We define the following notations:

- $Type(B_{i,j})$ is equal to:
  - 1, if $\exists f \in \Phi, u \in M(\Sigma)$ such that $B_{i,j} = B^1_{i,f,u}$
  - 2, if $\exists f \in \Phi, u \in M(\Sigma)$ such that $B_{i,j} = B^2_{i,f,u}$
- $Flag(B_{i,j}) = f$, if $\exists k(1 \leq k \leq 2) \wedge \exists u \in M(\Sigma)$ such that $B_{i,j} = B^k_{i,f,u}$
- $Mult(B_{i,j}) = u$, if $\exists k(1 \leq k \leq 2) \wedge \exists f \in \Phi$ such that $B_{i,j} = B^k_{i,f,u}$

In addition, for each block $B_{i,j}$, $1 \leq i \leq m$ and $1 \leq j \leq \alpha_i$, associated to cell i, we consider a total order in the set of integrated rules: $B_{i,j} = \{r_{i,j,1}, \ldots, r_{i,j,h_{i,j}}\}$, where $h_{i,j}(1 \leq i \leq m,\ 1 \leq j \leq \alpha_i)$ denotes the number of rules in block $B_{i,j}$. Obviously, all the rules of a block are of the same type.

Definition 12. *A PGP system is said to feature* object competition*, if there exists at least two different blocks $B_{i,j}$ and $B_{i,j'}$ (possibly of different type), such that $Flag(B_{i,j}) = Flag(B_{i,j'})$, and $Mult(B_{i,j}) \cap Mult(B_{i,j'}) \neq \emptyset$. That is, their rules have overlapping (but not equal) left-hand sides.*

Remark 3. It is worth noting that all rules in the model can be consistently applied. This is because there can only exists one flag $f \in \Phi$ at every membrane at the same time, and, consequently, at most one context–changing rule $r \equiv \{f\}\,(u, f)_i \to (v, g)_i$ can consume f and replace it (where possibly $f = g$).

Definition 13. *Given a block $B^1_{i,f,u}$ or $B^2_{i,f,u}$, where $u \in M(\Sigma), f \in \Phi$, $1 \leq i \leq m$ and a configuration $C_t = \{x_1, u_1, \ldots, x_m, u_m\}$, $0 \leq t \leq T$, the* maximum number of applications *of such a block in C_t is the maximum applications of any of its rule in C_t.*

5.2 Simulation Algorithm

Next, we define some auxiliary data structures to be used in the simulation algorithms.

NBA (Number of Block Applications): a matrix of integer numbers of dimension $m \times N_{BM}$, where $N_{BM} = max(\alpha_i),\ 1 \leq i \leq m$ (maximum number of blocks for all cells). Each element $NBA_{i,j},\ 1 \leq i \leq m,\ 1 \leq j \leq N_{BM}$ stores the number of applications of block $B_{i,j}$.

NRA (Number of Rule Applications): a matrix of integer numbers of dimension $m \times N_{BM} \times N_{RM}$, where $N_{RM} = max(h_{i,j}),\ 1 \leq i \leq m,\ 1 \leq j \leq \alpha_i$ (maximum number of rules for all blocks in all membranes). Each element $NRA_{i,j,k},\ 1 \leq i \leq m,\ 1 \leq j \leq \alpha_i,\ 1 \leq k \leq h_{i,j}$, stores the number of applications of rule $r_{i,j,k}$, identified by its cell, block and local identifier inside its block, according to the established total order.

The algorithm for simulation of PGP systems receives three parameters:

- The PGP system Π of degree m.
- The integer number $T > 0$ (number of time steps).
- An integer number $K > 0$ (random accuracy). It indicates for how many cycles block applications are assigned among their rules in random fashion. That is, the algorithm distributes the applications of each block among its rules for K cycles, and after that, block applications are maximally assigned among rules in a single cycle. It is used as an accuracy parameter for the probabilistic method. Algorithm 5.4 performs this function.

When simulating PGP systems without object competition, it is not necessary to randomly assign objects among blocks; as they do not compete for objects, then the number of times that each block is applied is always equal to its maximum number of applications. As it is the case of DCBA for PDP systems [15], the simulation algorithm heavily relies on the concept of block, being rule applications secondary. However, DCBA handles object competition among blocks, penalizing more those blocks which require a larger number of copies of the same object which is inspired by the amount of energy required to join individuals from the same species. On the other hand, object competition is not supported on the proposed algorithm. Algorithm 5.1 describes a simulation algorithm for PGP systems without object competition.

Algorithm 5.1 Algorithm for simulation of PGP systems

Input:

- T: an integer number $T \geq 1$ representing the iterations of the simulation.
- K: an integer number $K \geq 1$ representing non–maximal rule iterations (i.e., iterations in which the applications selected for each rule do not necessarily need to be maximal).
- $\Pi = (G, \Sigma, \Phi, T, \{(f_j, E_j) \mid 1 \leq j \leq m\}, \mathcal{R}_E)$: a PGP system of degree $m \geq 1$.

```
Initialization (Π)
for t ← 1 to T do                                          ▷ See Algorithm 5.2
    C'_t ← C_{t−1}
    SELECTION of rules:
        PHASE 1: Objects distribution (C'_t)               ▷ See Algorithm 5.3
        PHASE 2: Rule application distribution (C'_t)      ▷ See Algorithm 5.4
    EXECUTION of rules:
        PHASE 3: Object production (C'_t)                  ▷ See Algorithm 5.5
    C_t ← C'_t
end for
```

On each simulation step t, $1 \leq t \leq T$ and cell i, $1 \leq i \leq m$, the following stages are applied: *Object distribution* (selection), *Rule application distribution* (selection) and *Object generation* (execution).

However, before starting the simulation process, we must initialize some data structures. In *Initialization* (Algorithm 5.2), the initial configuration C_0 is constructed with the input PGP system Π. Moreover, the information about blocks are created; that is, the blocks of rules are computed, and ordered for each cell. Moreover, the rules inside each block are also ordered. Finally, the data structures NBA and NRA are initialized with zeros.

In the *Object distribution* stage (Algorithm 5.3), objects are distributed among blocks. As the system to simulate does not feature object competition, the number of applications of each block is its maximum. Then, objects are consumed accordingly. It is in this stage that the flag checking for each block is performed.

Algorithm 5.2 Initialization

Input: $\Pi = (G, \Sigma, \Phi, T, \{(f_j, E_j) \mid 1 \leq j \leq m\}, \mathcal{R}_E)$

$C_0 \leftarrow \{f_1, E_1, \ldots, f_m, E_m\}$ ▷ Initial configuration
for $i \leftarrow 1$ **to** m **do** ▷ For each cell
  $B_i \leftarrow$ ordered set of blocks formed by rules of $\mathcal{R}$ associated with cell i
  $\alpha_i \leftarrow |B_i|$ ▷ Number of rule blocks
  for $j \leftarrow 1$ **to** α_i **do** ▷ For each block associated with the cell
    $B_{i,j} \leftarrow$ ordered set of rules from j^{th} block in B_i.
    $h_{i,j} \leftarrow |B_{i,j}|$ ▷ Number of rules within the block
    $NBA_{i,j} \leftarrow 0$ ▷ Initially, all blocks applications are 0
    for $k \leftarrow 1$ **to** $h_{i,j}$ **do** ▷ Initially, all rule applications are 0
      $NRA_{i,j,k} \leftarrow 0$
    end for
  end for
end for

Moreover, blocks of type 2 (context–changing rules) consume and generate the new flag.

Algorithm 5.3 Phase 1: Object distribution among blocks

Input: $C'_t = \{x_{1,t}, u_{1,t}, \ldots, x_{m,t}, u_{m,t}\}$

for $i \leftarrow 1$ **to** m **do** ▷ For each cell
  for $j \leftarrow 1$ **to** α_i **do** ▷ For each block associated with the cell
    if $Flag(B_{i,j}) = x_{i,t}$ **then**
      if $Type(B_{i,j}) = 1 \wedge Mult(B_{i,j}) \subseteq u_{i,t}$ **then**
        $NBA_{i,j} \leftarrow \min(\lfloor \frac{u_{i,t}(z)}{Mult(B_{i,j})(z)} \rfloor : z \in \Sigma)$ ▷ Maximal application
        $u_{i,t} \leftarrow u_{i,t} - NBA_{i,j} \cdot Mult(B_{i,j})$ ▷ Update the configuration
      end if
      if $Type(B_{i,j}) = 2 \wedge Mult(B_{i,j}) \subseteq u_{i,t}$ **then**
        $NBA_{i,j} \leftarrow 1$ ▷ Just one application
        $x_{i,t} \leftarrow g$, being $RHS(r_{i,j,1}) = (i, g, v)$ with $B_{i,j} = \{r_{i,j,1}\}$ ▷ Update cell flag
        $u_{i,t} \leftarrow u_{i,t} - NBA_{i,j} \cdot Mult(B_{i,j})$ ▷ Update the configuration
      end if
    end if
  end for
end for

Next, objects are distributed among rules according to a binomial distribution with rule probabilities and maximum number of block applications as parameters. This algorithm is composed of two stages *non–maximal* and *maximal* repartition. In the non–maximal repartition stage, a rule in the block is randomly selected according to a uniform distribution, so each rule has the same probability to be chosen. Then, its number of applications is calculated according to an *ad–hoc*

procedure based on a binomially distributed variable $Binomial(n, p)$, where n is the remaining number of block applications to be assigned among its rules and p is the corresponding rule probability. This process is repeated a number K of iterations for each block $B_{i,j}, 1 \leq i \leq m, 1 \leq j \leq \alpha_i$. Algorithm 5.4 describes this procedure. If, after this process, there are still applications to assign among rules, a rule per applicable block is chosen at random and as many applications as possible are assigned to it in the maximal repartition stage. An alternative approach would be to implement a multinomial distribution of applications for the rules inside each block, such as the way that it is implemented on the DCBA algorithm [15]. A method to implement a multinomial distribution would be the conditional distribution method, which emulates a multinomial distribution based on a sequence of binomial distributions [8]. This would require to normalize rule probabilities for each rule application distribution iteration. This approach has also been tested on the simulation algorithm, but was discarded because it tends to distribute too few applications in the non–maximal repartition stage, thus leaving too many applications for the rule selected in the maximal repartition one.

Algorithm 5.4 Phase 2: Rule application distribution

Input: $C'_t = \{x_{1,t}, u_{1,t}, \ldots, x_{m,t}, u_{m,t}\}$
  for $k \leftarrow 1$ **to** K **do** ▷ Non-maximal repartition stage
    for $i \leftarrow 1$ **to** m **do**
      for $j \leftarrow 1$ **to** α_i **do**
        $l \leftarrow Uniform\{1, \ldots, h_{i,j}\}$ ▷ Select a random rule $r_{i,j,l}$ in Block $B_{i,j}$
        $lnrap \leftarrow Binomial(NBA_{i,j}, p_{\mathcal{R}}(r_{i,j,l}))$
        $NRA_{i,j,l} \leftarrow NRA_{i,j,l} + lnrap$ ▷ Update rule applications
        $NBA_{i,j} \leftarrow NBA_{i,j} - lnrap$
      end for
    end for
  end for
  for $i \leftarrow 1$ **to** m **do** ▷ Maximal repartition stage
    for $j \leftarrow 1$ **to** α_i **do**
      $l \leftarrow Uniform\{1, \ldots, h_{i,j}\}$
      $NRA_{i,j,l} \leftarrow NRA_{i,j,l} + NBA_{i,j}$
      $NBA_{i,j} \leftarrow 0$
    end for
  end for

Lastly, rules produce objects as indicated by their right–hand side. Each rule produces objects according to its previously assigned number of applications. Algorithm 5.5 describes this procedure.

The algorithm proposed in this paper works only for models without object competition. This is because the models studied so far did not have object competition, so this feature was not required. However, it might be interesting to develop new algorithms supporting it. They would be identical to their counterpart

Algorithm 5.5 Phase 3: Object production

for $i \leftarrow 1$ **to** m **do** ▷ For each cell
  for $j \leftarrow 1$ **to** α_i **do** ▷ For each block associated with the cell
    for $k \leftarrow 1$ **to** $h_{i,j}$ **do** ▷ For each rule belonging to the block
      $u_{i,t} \leftarrow u_{i,t} + NRA_{i,j,k} \cdot v$, where $RHS(r_{i,j,k}) = (i', f', v)$
      $NRA_{i,j,k} \leftarrow 0$
    end for
  end for
end for

without object competition, solely differing in the protocol by which objects are distributed among blocks. As an example, it would be possible to adapt the way in which objects are distributed in the DCBA algorithm [15].

5.3 Software environment

Next, the developed simulators, a P–Lingua extension, and a GUI for PGP systems are going to be summarized.

Simulators

A simulator for PGP systems without object competition has been incorporated on P–Lingua [10]. In addition, a C++ simulator for PGP systems (namely PGPC++) has also been implemented. The libraries used for random number generation are COLT [23] in the P–Lingua simulator, and standard std::rand [24] for PGPC++. In the latter, the facilities provided by std::rand are directly used. These libraries provide a wide range of functionality to generate and handle random numbers, and are publicly available under open source licenses.

P–Lingua extension

In order to define PGP systems, P–Lingua has been extended to support PGP rules. Specifically, given $f, g \in \Phi,\ u, v \in M(\Sigma),\ 1 \leq i, j \leq m,\ p = p_{\mathcal{R}}(r)$, rules are represented as follows:

$$\{f\}\,(u)_i \xrightarrow{p} (v)_j,\ \equiv \texttt{@guard } f \texttt{ ?[}u\texttt{]'}i \texttt{ --> [}v\texttt{]'}j \texttt{ :: } p\texttt{;}$$
$$\{f\}\,(u, f)_i \to (v, g)_i \equiv \texttt{@guard } f \texttt{ ?[}u\texttt{,}f\texttt{]'}i \texttt{ --> [}v\texttt{,}g\texttt{]'}i \texttt{ :: 1.0;}$$

In both cases, if $p = 1.0$, then `:: p` can be omitted. If $i = j$, then $\{f\}\,(u)_i \xrightarrow{p} (v)_j$ can be written as `@guard f ?[u --> v]'i :: p;`. Likewise, $\{f\}\,(u, f)_i \to (v, g)_i$ can always be written as `@guard f ?[u,f --> v,g]'i;`. Moreover, some additional constructs have been included to ease parametrization of P systems. The idea is to enable completely parametric designs, so as experiments can be tuned by simply adjusting parameters, leaving modifications of P–Lingua files for cases in which changes in semantics are in order.

`&{`*multiset*`}:{`*iterators*`}` In this sentence, *multiset* is an ordinary multiset, whose indexes depend on the iterators defined in *iterators*. *iterators* is a standard list of iterators in P–Lingua separated by commas. It is worth noting that this sentence has some limitations. For instance, variables defined in these iterators cannot be used again in the same P–Lingua specification. In addition, those variables used in *multiset* which are defined in *iterators* can only be used as such, that is, they cannot be used as subindexes or arithmetical expressions. The reasons for these constraints correspond to technical implementation details which will not be discussed here.

`@mu(`*label*`)*=`*cell_structure*`;` In this sentence, *label* is a cell label defined at some point in the P–Lingua specification. *cell_structure* is a standard P–Lingua, tissue–like membrane structure, such as the ones which can be defined after the *@mu* sentence. This sentence adds the skin of *membrane_structure* as a child cell of *label*. As cells in tissue–like structures have no parent, $label = 0$ for all tissue–like models. In cell–like models, the behaviour is the same, with the exception that *cell_structure* is a cell–like structure, *label* can be any label in the system and the symbol `*=` is replaced by `+=`.

`@property(`*label*`)=`*set*`;` This sentence allows designers to define specific properties for objects. *set* is a set of symbols, which can be extended by external, standard iterators or internal ones as defined at the first point of this list. In the case of PGP systems, @property(flag)=*set* defines flags $f \in \Phi$.

In addition, two new formats have been integrated into P-Lingua. These formats (XML–based and binary) encode P systems representing labels and objects as numbers instead of strings, so they are easily parsed and simulated by third–part simulators such as PGPC++.

A graphical environment for PGP systems

MeCoGUI is a new GUI developed for the simulation of PGP systems. MeCoSim [20] could have been used instead. However, in the environment in which the simulators were developed there exist some pros and cons on this approach versus and ad–hoc simulator.

MeCoSim is an integrated development environment (IDE). That is to say, it provides all functionality required for the simulation and computational analysis of P systems. To define the desired input and output displays, it is necessary to configure a spreadsheet by using an *ad–hoc* programming language. However, it would entail teaching this language to prospective users, which are proficient in R programming language instead. In this sense, a more natural approach for them is to develop a GUI in which users can define input parameters and results analysis on R.

To do so, the developed GUI takes as input a P system file on P–Lingua format and a CSV file encoding its parameters, and outputs a CSV file which contains simulation results. This way, users can define inputs and analyse outputs on their programming language of choice. CSV is a widespread, simple and free

format with plenty of libraries for different languages. This flexibility comes at the cost concerning that the developed GUI is not an IDE, as input parameters and simulation analysis cannot be directly input and viewed on the GUI. Rather, it is necessary to develop applications to generate and process these CSV files which depend on the domain of use. In some simulators (such as PGPC++), the output CSV files represent labels and objects as integers, but this application includes a button to translate output files from PGPC++ into string–representative file formats. Figure 3 displays the main screen of this application.

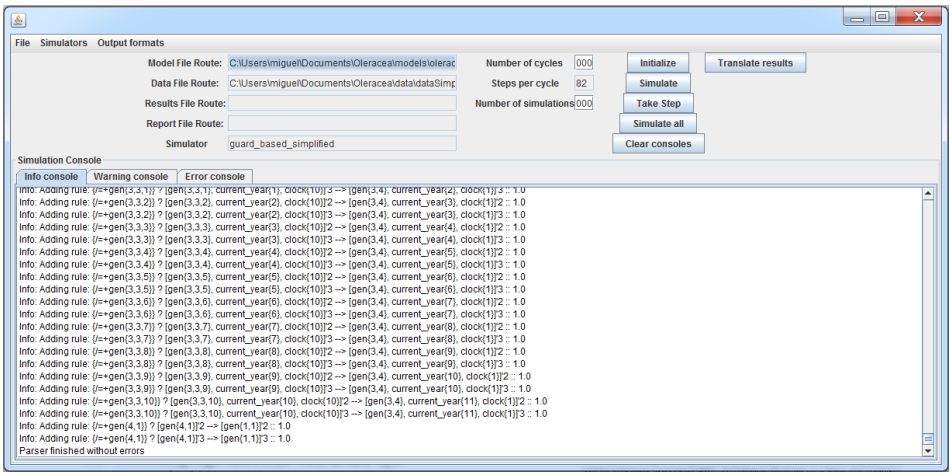


Fig. 3: Main screen of MeCoGUI

MeCoGUI can also translate P systems into machine–readable formats, such as those read by PGPC++. Finally, it is important to remark that these applications play the role of domain–specific spreadsheets on MeCoSim, so MeCoGUI can simulate any type of P system supported by P–Lingua. This is because only external applications for input data and simulation processing depend on the domain, not MeCoGUI itself, which is general for any type of P system. Figure 4 graphically describes the workflow for P–Lingua and for PGPC++.

6 Applications of PGP systems

A model of the ecosystem of the white cabbage butterfly (*Pieris oleracea*) [7], based on PGP systems, is a currently ongoing project. Such a species is suffering the invasion of the garlic mustard (*Alliaria petiolata*), which is replacing native host broadleaf toothwort (*Cardamine diphylla*) and ravaging the butterfly's natural habitat. Specifically, *A. petiolata* contains a deterrent agent for larvae of *P. oleracea*. Moreover, such a plant is toxic for these larvae, although it contains

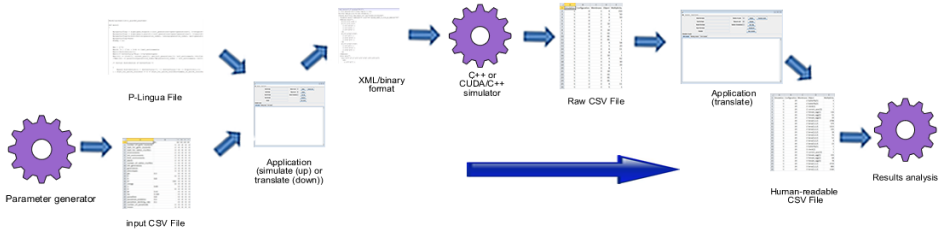


Fig. 4: Workflow for P-Lingua simulator (upper branch) and PGPC++ (lower branch) for MeCoGUI

a chemical compound which lures mature butterflies and frames them into laying eggs. Nevertheless, a minority of individuals tolerates such a deterrent, metabolize the toxin and reach the pupa stage [12, 13].

The distribution of phylogenetic profiles across the species consists of a majority of homozygous individuals unable to thrive on *A. petiolata* patches, a minority of homozygous individuals which do well on *A. petolata* rosettes and, in the midterm, an slightly larger population of heterozygous individuals with both alleles. The allele which enables butterflies to overcome the dietary restrictions imposed by *A. petiolata* is dominant, but individuals carrying this allele undergo a detoxification mechanism which entails an energetic cost and hampers their arrival at adulthood [12].

The model under development aims to identify if there has been any evolutionary adaptation of the butterfly species significant enough so as to ensure its survival in the new scenario. Specifically, the idea is to assess if the detoxification cost associated with individuals tolerating *A. petiolata* pays off in the new scenario or, on the other hand, the phylogenetic distribution will stay the same and other mechanism will come into effect, such as hybridization with other butterfly species such as *Pieris rapae* [7].

The approach taken in this project aims to validate the model *qualitatively*. A *qualitative validation* is defined as follows: a model is qualitatively validated if it can reproduce some properties verified by the ecosystem under different scenarios (according to the experts).

7 Conclusions and Future Work

Multienvironment P systems are a general, formal framework for modelling population dynamics in Biology. The framework has two main approaches: stochastic (micro–level oriented) and probabilistic (macro–level oriented). The framework has been extended in the probabilistic approach, with the inclusion of a new modelling framework called Probabilistic Guarded P (PGP) systems. PGP systems are inspired by Population Dynamics P systems, and aim to simplify the

design and simulation of models of ecological phenomena. The model has been formalized in this paper, and a simulation algorithm is introduced. This algorithm is restricted for models which do not feature object competition. Moreover, an extension of the P–Lingua language is provided to enable PGP systems in P–Lingua, as well as a Graphical User Interface (GUI) to simulate PGP systems (MeCoGUI).

The framework of PGP systems is being utilised for modelling the ecosystem of *Pieris napi oleracea*, a butterfly native to Northeaster U.S.A. The aim is to validate the model qualitatively; that is, checking that if the ecosystem verifies some properties under different scenarios (experts), our model reproduces those properties as well.

Although PGP systems provide a simplified alternative to PDP systems, some constraints to the supported models are imposed: only models without object competition are allowed. Therefore, future research lines will be focused on overcoming this constraint, providing new simulation algorithms permitting object competition. Moreover, new case studies will be considered, what can help to extend the framework. Finally, PGP simulation will be accelerated by using parallel architectures, such as GPU computing with CUDA.

Acknowledgements

The authors acknowledge the support of the project TIN2012-37434 of the "Ministerio de Economía y Competitividad" of Spain, co-financed by FEDER funds. Manuel García–Quismondo also acknowledges the support from the National FPU Grant Programme from the Spanish Ministry of Education. Miguel A. Martínez-del-Amor also acknowledges the support of the 3rd Postdoctoral phase of the PIF program associated with "Proyecto de Excelencia con Investigador de Reconocida Valía" of the "Junta de Andalucía" under grant P08-TIC04200.

References

1. J. Blakes, J. Twycross, F.J. Romero-Campero, N. Krasnogor. The Infobiotics Workbench: an integrated in silico modelling platform for Systems and Synthetic Biology, *Bioinformatics*, **27**, 23 (2011), 3323-3324.
2. M.A. Colomer-Cugat, M. García-Quismondo, L.F. Macías-Ramos, M.A. Martínez-del-Amor, I. Pérez-Hurtado, M.J. PérezJiménez, A. Riscos-Núñez, L. Valencia-Cabrera. Membrane system-based models for specifying Dynamical Population systems. In P. Frisco, M. Gheorghe, M.J. Prez-Jimnez (eds.), *Applications of Membrane Computing in Systems and Synthetic Biology. Emergence, Complexity and Computation series*, Volume **7**. Chapter 4, pp. 97–132, 2014, Springer Int. Publishing.
3. M. Cardona, M.A. Colomer, A. Margalida, A. Palau, I. Pérez-Hurtado, M.J. Pérez-Jiménez, D. Sanuy. A computational modeling for real ecosystems based on P systems, *Natural Computing*, **10**, 1 (2011), 39–53.

4. S. Cheruku, A. Păun, F.J. Romero-Campero, M.J. Pérez-Jiménez, O.H. Ibarra. Simulating FAS-induced apoptosis by using P systems, *Progress in Natural Science*, **17**, 4 (2007), 424–431.
5. M.A. Colomer, A. Margalida, D. Sanuy, M.J. Pérez-Jiménez. A bio-inspired computing model as a new tool for modeling ecosystems: The avian scavengers as a case study, *Ecological modelling*, **222**, 1 (2011), 33–47.
6. M.A. Colomer, M.A. Martínez-del-Amor, I. Pérez-Hurtado, M.J. Pérez-Jiménez, A. Riscos-Núñez. A uniform framework for modeling based on P Systems. *Proceedings IEEE Fifth International Conference on Bio-inpired Computing: Theories and Applications (BIC-TA 2010)*, **Volume 1**, pp. 616–621.
7. F.S. Chew. Coexistence and local extinction in two pierid butterflies, *The American Naturalist*, **118**, 5 (1981), 655–672.
8. C.S. Davis. The computer generation of multinomial random variates. *Computational Statistics and Data Analysis*, **16**, 2 (1993), 205–217.
9. P. Frisco, M. Gheorghe, M. J. Pérez-Jiménez (eds.) *Applications of Membrane Computing in Systems and Synthetic Biology*, Springer, 2014.
10. M. García-Quismondo, R. Gutiérrez-Escudero, I. Pérez-Hurtado, M.J. Pérez-Jiménez, Agustín Riscos-Núñez. An overview of P-Lingua 2.0, *LNCS*, **5957** (2010), 264–288.
11. M. Gheorghe, F. Ipate, C. Dragomir, L. Mierla, L. Valencia-Cabrera, M. García-Quismondo, M.J. Pérez-Jiménez. Kernel P systems - Version I, *Proceedings of the Eleventh Brainstorming Week on Membrane Computing (BWMC2013)*, 2013, pp. 97–124.
12. M.S. Keeler, F.S. Chew. Escaping an evolutionary trap: preference and performance of a native insect on an exotic invasive host, *Oecologia*, **156**, 3 (2008), 559–568.
13. M.S. Keeler, F.S. Chew, B.C. Goodale, J.M. Reed. Modelling the impacts of two exotic invasive species on a native butterfly: top-down vs. bottom-up effects, *Journal of Animal Ecology*, **75**, 3 (2006), 777–788.
14. M.A. Martínez-del-Amor, I. Pérez-Hurtado, A. Gastalver-Rubio, A.C. Elster, M.J. Pérez-Jiménez. Population Dynamics P systems on CUDA. *LNBI*, **7605** (2012), 247–266.
15. M.A. Martínez-del-Amor, I. Pérez-Hurtado, M. García-Quismondo, L.F. Macías-Ramos, L. Valencia-Cabrera, A. Romero-Jiménez, C. Graciani, A. Riscos-Núñez, M.A. Colomer, M.J. Pérez-Jiménez. DCBA: Simulating Population Dynamics P Systems with Proportional Object Distribution, *LNCS*, **7762** (2012), 27–56.
16. M.A. Martínez-del-Amor, I. Pérez-Hurtado, M.J. Pérez-Jiménez, A. Riscos-Núñez, F. Sancho-Caparrini. A simulation algorithm for multienvironment probabilistic P systems: A formal verification, *International Journal of Foundations of Computer Science*, **22**, 1 (2011), 107–118.
17. A. Păun, B. Popa. P systems with proteins on membranes, *Fundamenta Informaticae*, **72**, 4 (2006), 467–483.
18. G. Păun. Computing with membranes. *Journal of Computer and System Sciences*, **61**, 1 (2000), 108–143, and TUCS Report No 208.
19. G. Păun, G. Rozenberg, A. Salomaa (eds.). *The Oxford Handbook of Membrane Computing*, Oxford University Press, 2010.
20. I. Pérez-Hurtado, L. Valencia-Cabrera, M.J. Pérez-Jiménez, M.A. Colomer, A. Riscos-Núñez. MeCoSim: A general purpose software tool for simulating biological phenomena by means of P Systems, *Proceedings IEEE Fifth International*

Conference on Bio-inpired Computing: Theories and Applications (BIC-TA 2010), **volume I** (2010), pp. 637–643.

21. M.J. Pérez-Jiménez, F.J. Romero. P systems. A new computational modelling tool for Systems Biology. *Transactions on Computational Systems Biology VI. Lecture Notes in Bioinformatics*, **4220** (2006), 176–197.
22. L. Pan, M.J. Pérez-Jiménez. Computational complexity of tissue-like P systems, *Journal of Complexity*, **26**, 3 (2010), 296–315.
23. *COLT library*. `http://acs.lbl.gov/software/colt/index.html`
24. *RAND function in C++/C Standard General Utilities Library (cstdlib)*. `http://www.cplusplus.com/reference/cstdlib/rand`