Chapter 4
Membrane System-Based Models for Specifying Dynamical Population Systems

M. A. Colomer-Cugat, M. García-Quismondo, L. F. Macías-Ramos, M. A. Martínez-del-Amor, I. Pérez-Hurtado, M. J. Pérez–Jiménez, A. Riscos-Núñez and L. Valencia-Cabrera

Abstract Population Dynamics P systems (PDP systems, in short) provide a new formal bio-inspired modelling framework, which has been successfully used for modelling population dynamics on real ecosystems. The semantics of these systems is captured by the Direct distribution based on Consistent Blocks Algorithm (DCBA), which has been engineered into software simulation tools. In particular, *MeCoSim* (Membrane Computing Simulator) is a GUI developed in the framework of *P-Lingua* that can be used as a simulation environment for running virtual experiments. The parameters of each scenario to be simulated can be easily adjusted in a visual way, as well as the settings for the desired output format, thus facilitating the validation

M. A. Colomer-Cugat (✉)
Department of Mathematics, University of Lleida, Avinguda Alcalde Rovira Roure, 191, 25198 Lleida, Spain
e-mail: colomer@matematica.udl.cat

M. García-Quismondo · L. F. Macías-Ramos · M. A. Martínez-del-Amor · I. Pérez-Hurtado
M. J. Pérez–Jiménez · A. Riscos-Núñez · L. Valencia-Cabrera
Research Group on Natural Computing Department of Computer Science and Artificial Intelligence, University of Sevilla, Avenida Reina Mercedes s/n, 41012 Sevilla, Spain
e-mail: mgarciaquismondo@us.es

L. F. Macías-Ramos
e-mail: lfmaciasr@us.es

M. A. Martínez-del-Amor
e-mail: mdelamor@us.es

I. Pérez-Hurtado
e-mail: perezh@us.es

M. J. Pérez–Jiménez
e-mail: marper@us.es

A. Riscos-Núñez
e-mail: ariscosn@us.es

L. Valencia-Cabrera
e-mail: lvalencia@us.es

P. Frisco et al. (eds.), *Applications of Membrane Computing in Systems and Synthetic Biology*, Emergence, Complexity and Computation 7, DOI: 10.1007/978-3-319-03191-0_4,

of the designed models against real data. The simulation of PDP systems is data intensive for large models. Therefore, the development of efficient simulators for this field is needed. In fact, the computational power of GPUs is currently being used to accelerate simulations of PDP systems. We illustrate the modelling framework presented with a case study concerning pandemics.

4.1 Introduction

Mathematical models are abstract representations of real–world complex systems onto a mathematical/computational domain. They use symbolic notations and formalisms rather than physical devices to represent and analyse the relationships which describe the system under study. Moreover, never do they contain all features of their real–world counterpart, as they would be the real–world complex system itself. Models highlight the relevant features while ignoring the irrelevant ones. Therefore, a mathematical model should be regarded as a description of our current knowledge about a phenomenon of interest, rather than an exact representation of the truth.

The use of models is inherent to any scientific activity. When examining a phenomenon, scientists regularly use abstractions of reality such as diagrams, graphs, laws, etc. with the aim of describing and understanding it. Designing a model for a biological system is intrinsically a complicated task, since there is usually a large number of important factors that need to be considered. Therefore, it is advisable to make efforts to minimize the number of parameters, as well as the interactions between them.

Nowadays, ordinary differential equations (ODEs) constitute the most widely used approach for the study of complex systems, and in particular for population dynamics. However, this approach has some drawbacks, since it has been reported that the deterministic and continuous approach followed by ODEs is questionable in some complex systems. Consequently, in recent years new models based on the latest computational paradigms and technological advances have been adopted. Some examples are Petri nets [1], process algebra (π-calculus [2], bioambients [3], brane calculus [4], κ-calculus [5], etc.), state charts [6], agent based systems [7], and viability models [8, 9]. Although each of these computational frameworks captures some aspects regarding complex systems and their components, none of them fully integrates their dynamics and structural details.

Membrane Computing is an emergent branch of Natural Computing introduced by Păun in [10], inspired by the structure and functioning of living cells. This research field was introduced with the purpose of defining unconventional distributed and parallel computing devices, called P systems. The key differential feature of such systems is the so–called *membrane structure*, which represents the compartmentalization in the structural organization of cells. One of the main advantages of P systems, regarding their potential use as a modelling framework, is that their elements are more similar to those used in population dynamics than the abstractions of other formalisms. In this work we show how to capture, in a natural way, some features relevant to populations by using membrane structures.

We present P systems as a high level computational modelling framework which integrates 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. Rather than an alternative to more classical modelling frameworks, such as ODEs, P systems constitute a complementary approach to be used in those cases where the previous approaches fail. Among the most important properties of these models one can include their capacity to work in parallel, as well as capture randomness. P systems explicitly represent the discrete character of the quantity of components of a system by using rewriting rules on multisets. Objects on these multisets represent relevant parameters, as well as individuals. Although each complex system has its own important peculiarities, the vast majority of them share some common aspects. Some of these aspects are: (a) large number of individuals; (b) life cycle consisting of some basic processes; (c) periodic repetitions; (d) the evolution often depends on the environment; and (e) the natural dynamics suffers modifications due to human activities. These common features impose some computational requirements on the models. Among them, one can shortlist the following: many processes take place simultaneously, there exists cooperation between different individuals and elements, there exists partial synchronization among the dynamic evolution sub-problems, and some stages are cyclically repeated.

These considerations led to the definition of a special-purpose type of P systems. In particular, *Population Dynamics P system* models have been designed to study the evolution of the habitats corresponding to three real case studies: the bearded vulture (*Gypaetus barbatus*) in the Catalan Pyrenees (Spain) [11, 12], the zebra mussel (*Dreissena polymorpha*) at the reservoir of Ribarroja (Spain) [13] and, recently, the Pyrenean brook newt (*Calotriton asper*) in the river Segre (*Serra del Cadí*, Pyrenees) [14]. In the first case, the purpose of the obtained model is to study the evolution of the considered ecosystem under different scenarios in order to make the most appropriate management decisions for the conservation of an endangered species. The second case study corresponds to a completely different situation: *Dreissena polymorpha* is an exotic invasive species that has displayed an excellent adaptation following its introduction in the reservoir. Its uncontrolled reproduction causes significant economic and ecological losses. Hence, the goal in this case is to learn how to minimize the mussel population at the reservoir. In the latter case, the experts have documented repeatedly dramatic population losses of the studied species caused by severe floods. The goal is to evaluate whether there exists actual risk for the species to become extinct in this area as a consequence of extreme rainfall. In all three cases we have designed a simulator to validate the models. Actually, three different tools have been released, enabling the corresponding users to perform virtual experiments under different conditions.

This chapter is structured as follows: Sect. 4.2 is devoted to define concepts and notations that we will use throughout the text. A P systems based probabilistic modelling framework is considered in Sect. 4.3, and an inference engine capturing the semantics of the model is presented in Sect. 4.4. Section 4.5 is devoted to the software tools implementing the theoretical framework. Section 4.6 describes the modelling of a case study about an outbreak of a pandemic disease, in order to illustrate the

theoretical and practical framework presented. To conclude the chapter, in Sect. 4.7 we present some final considerations and outline some future work on the topic.

4.2 Preliminaries

In this section we introduce some concepts and notations which we will use throughout this chapter.

An *alphabet*, Γ, is a finite non–empty set whose elements are called *symbols*. A *multiset*, w, over an alphabet, Γ, is a pair (Γ, f) where $f : \Gamma \to \mathbb{N}$ is a mapping. If $w = (\Gamma, f)$ is a multiset then its *support* is defined as $supp(w) = \{x \in \Gamma \mid f(x) > 0\}$. We denote $x \in w$ when $x \in supp(m)$ and $x^n \in w$ when $x \in w$ and $f(x) = n$. A multiset is finite if its support is a finite set. If $w = (\Gamma, f)$ is a finite multiset over Γ, and $supp(w) = \{a_1, \ldots, a_k\}$ then it will be denoted as $w = a_1^{f(a_1)} \ldots a_k^{f(a_k)}$ (here the order is irrelevant), and we say that $f(a_1) + \cdots + f(a_k)$ is the cardinality of w, denoted by $|w|$. The empty multiset is denoted by $\emptyset$. We also denote by $M_f(\Gamma)$ the set of all finite multisets over Γ.

If $w_1 = (\Gamma, f_1)$, $w_2 = (\Gamma, f_2)$ are multisets over Γ, then we define the union of w_1 and w_2 as $w_1 + w_2 = (\Gamma, g)$, where $g = f_1 + f_2$.

In what follows, we assume the reader is already familiar with the basic notions and the terminology of P systems. For details, see [15].

4.3 A P Systems-Based Probabilistic Modelling Framework

Population Dynamics P systems are a variant of *multienvironment P systems with active membranes* [16], a model with a network of environments, each of them containing a P system with features such as electrical charges associated with membranes to describe specific properties in a better way. All P systems share the same *skeleton*, in the sense that they have the same working alphabet, the same membrane structure and the same set of rules. Nevertheless, the probability functions associated with the rules can vary among environments, and also the initial multisets are independent.

In what follows, the formal definition of PDP systems is introduced. To continue, some additional definitions are given regarding to the rules of the systems. These definitions will be specially useful within the scope of the Direct distribution based on Consistent Blocks Algorithm (DCBA), an inference engine for PDP systems that will be covered in Sect. 4.4. Finally, some desirable properties of the PDP systems model are reviewed at the end of this section.

4.3.1 PDP Systems

Definition 1 A Population Dynamics P system (PDP) of degree (q, m), $q, m \geq 1$, taking $T \geq 1$ time units, is a tuple

$$\begin{aligned}\Pi =&(G, \Gamma, \Sigma, T, \mathcal{R}_E, \mu, \mathcal{R}, \{f_{r,j} \mid r \in \mathcal{R}, 1 \leq j \leq m\},\\ &\{\mathcal{M}_{ij} \mid 1 \leq i \leq q, 1 \leq j \leq m\})\end{aligned}$$

where:

- $G = (V, S)$ is a directed graph with $V = \{e_1, \ldots, e_m\}$.
- Γ and Σ are alphabets such that $\Sigma \subsetneqq \Gamma$.
- T is a natural number.
- $\mathcal{R}_E$ is a finite set of communication rules of the form $(x)_{e_j} \xrightarrow{p_r} (y_1)_{e_{j_1}} \cdots (y_h)_{e_{j_h}}$, where $x, y_1, \ldots, y_h \in \Sigma$, $(e_j, e_{j_l}) \in S \, forall \, 1 \leq l \leq h$, and $p_r : \{1, \ldots, T\} \longrightarrow [0, 1]$ is a computable function such that for each $e_j \in V$ and $x \in \Sigma$, the sum of functions associated with the rules of the type $(x)_{e_j} \xrightarrow{p_r} (y_1)_{e_{j_1}} \cdots (y_h)_{e_{j_h}}$ is the constant function 1.
- μ is a rooted tree injectively labelled by $1 \leq i \leq q$, and by symbols from the set $\{0, +, -\}$.
- $\mathcal{R}$ is a finite set of of evolution rules of the form $u[v]_i^{\alpha} \rightarrow u'[v']_i^{\alpha'}$, where $u, v, u', v' \in M_f(\Gamma)$, $u + v \neq \emptyset$, $1 \leq i \leq q$ and $\alpha, \alpha' \in \{0, +, -\}$, such that there is no rules $(x)_{e_j} \xrightarrow{p_r} (y_1)_{e_{j_1}} \cdots (y_h)_{e_{j_h}}$ and $u[v]_i^{\alpha} \rightarrow u'[v']_i^{\alpha'}$ having $x \in u$.
- For each $r \in \mathcal{R}$ and $1 \leq j \leq m$, $f_{r,j} : \{1, \ldots, T\} \longrightarrow [0, 1]$ is a computable function such that for each $u, v \in M_f(\Gamma)$, $1 \leq i \leq q$, $\alpha, \alpha' \in \{0, +, -\}$ and $1 \leq j \leq m$, the sum of functions $f_{r,j}$ with $r \equiv u[v]_i^{\alpha} \rightarrow u'[v']_i^{\alpha'}$, is the constant function 1.
- For each i, j $(1 \leq i \leq q, 1 \leq j \leq m)$, $\mathcal{M}_{ij} \in M_f(\Gamma)$.

A Population Dynamics P system defined as above can be viewed as a set of m environments $e_1, \ldots, e_m$ interlinked by the edges from the directed graph G. Each environment e_j can only contain symbols from alphabet Σ and all of them also contain a P system skeleton, $\Pi_j = (\Gamma, \mu, \mathcal{M}_{1,j}, \ldots, \mathcal{M}_{q,j}, \mathcal{R})$, of degree q, where:

(a) Γ is the working alphabet whose symbols are also called objects.
(b) μ is a rooted tree which describes a membrane structure consisting of q membranes (nodes of the tree) injectively labelled by $1, \ldots, q$. The skin membrane (the root of the tree) is labelled by 1 and its parent is the environment e_j. We also associate one with each membrane.
(c) $\mathcal{M}_{1,j}, \ldots, \mathcal{M}_{q,j}$ are finite multisets over Γ, describing the objects initially associated to the q membranes of μ, within the environment e_j.
(d) $\mathcal{R}$ is the set of evolution rules of each P system. Every rule $r \in \mathcal{R}$ in Π_j has a computable function $f_{r,j}$ associated with it. For each environment e_j, we denote by $\mathcal{R}_{\Pi_j}$ the set of rules with probabilities obtained by coupling each $r \in \mathcal{R}$ with the corresponding function $f_{r,j}$.

Furthermore, there is a set $\mathcal{R}_E$ of communication rules between environments, and the natural number T represents the simulation time of the system. The set of rules of the whole system is $\bigcup_{j=1}^{m} \mathcal{R}_{\Pi_j} \cup \mathcal{R}_E$.

The *semantics* of Population Dynamics P systems is defined through a non deterministic and synchronous model, in the sense that a global clock is assumed. Next, we describe some semantic aspects of these systems.

A communication rule $r \in \mathcal{R}_E$, of the form $(x)_{e_j} \xrightarrow{pr} (y_1)_{e_{j_1}} \dots (y_h)_{e_{j_h}}$ is applicable to environment e_j if it contains object x. When such a rule is applied, object x passes from e_j to $e_{j_1}, \dots, e_{j_h}$ possibly transformed into objects $y_1, \dots, y_h$ respectively. At any moment t $(1 \leq t \leq T)$ for each object x in environment e_j, if there exist communication rules of type $(x)_{e_j} \xrightarrow{pr} (y_1)_{e_{j_1}} \dots (y_h)_{e_{j_h}}$, then one of these rules will be applied. If more than one such a rule can be applied to an object at a given instant, the system selects one randomly, according to their associated functions.

In each Π_j, an evolution rule $r \in \mathcal{R}$, of the form $u[\, v \,]_i^{\alpha} \rightarrow u'[\, v' \,]_i^{\alpha'}$, is applicable to membrane i, whose electrical charge is α, and that contains multiset v, and whose parent contains multiset u. When such a rule is applied, the objects of the multisets v and u are removed from membrane i and from its parent membrane, respectively. Simultaneously, objects in multiset u' are introduced into the parent of membrane i, and objects of multiset v' are introduced into membrane i. The application also replaces the charge of membrane i with α'. In each environment e_j, the rule r has associated a probability function $f_{r,j}$ that provides an index of the applicability when several rules compete for 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 $\mathcal{R}_{\Pi_j}$ simultaneously to the same membrane, all the rules must have the same electrical charge on their right–hand side.

An *instantaneous description* or *configuration* of the system at any instant t is a tuple of multisets specifying the objects associated to each environment and membrane, with its polarization, of every Π_j. We assume that all environments are initially empty and that all membranes have initially a neutral polarization. We assume a global clock, synchronizing all membranes and the application of all the rules (from $\mathcal{R}_E$ and from $\mathcal{R}_{\Pi_j}$ in all environments).

In each time unit a given configuration can be transformed into another by using rules from the whole system as follows: the rules to be applied are selected in a non–deterministic way according to the probabilities assigned to them, and all applicable rules are simultaneously applied in a maximal way. In this way, we get *transitions* from one configuration of the system to the next one.

A *computation* is a sequence of configurations such that the first term of the sequence is the initial configuration of the system, and each non-initial configuration is obtained from the previous one by applying rules of the system in a maximally parallel manner with the restrictions previously mentioned.

4.3.2 Additional Definitions

Next, we define some concepts associated with the rules from the system that will be used in the Sect. 4.3.3.

Definition 2 Given a rule $r \in \mathcal{R}$ of the form $u[v]_i^{\alpha} \rightarrow u'[v']_i^{\alpha'}$ where $1 \leq i \leq q$, $\alpha, \alpha' \in \{0, +, -\}$ and $u, v, u', v' \in M_f(\Gamma)$:

- The left-hand side of r is $LHS(r) = (i, \alpha, u, v)$. The charge of $LHS(r)$ is $charge(LHS(r)) = \alpha$.
- The right-hand side of r is $RHS(r) = (i, \alpha', u', v')$. The charge of $RHS(r)$ is $charge(RHS(r)) = \alpha'$.

Definition 3 Given a rule $r \in \mathcal{R}_E$ of the form $(x)_{e_j} \xrightarrow{p_r} (y_1)_{e_{j_1}} \cdots (y_h)_{e_{j_h}}$, the left-hand side of r is $LHS(r) = (e_j, x)$, and the right-hand side of r is $RHS(r) = (e_{j_1}, y_1) \cdots (e_{j_h}, y_h)$.

Definition 4 A block of rules is a set of rules with the same left–hand side. We say that a block of rules is consistent if any pair of its rules can be applied in a simultaneous manner provided that there are enough objects.

Rules from $\mathcal{R}_E$ can be classified into blocks in a natural way.

Definition 5 Given (e_j, x), $1 \leq j \leq m$, and $x \in \Sigma$, the block associated with (e_j, x) is the set:

$$B_{e_j,x} = \{r \in \mathcal{R}_E \mid LHS(r) = (e_j, x)\}$$

Extending left-hand side definition, $LHS(B_{e_j,x}) = (e_j, x)$.

Bearing in mind that the restriction concerning the consistency of charges only applies to membranes, but not to environments, any block of rules from $\mathcal{R}_E$ is obviously a consistent block.

Let us then focus on the case of evolution rules from $\mathcal{R}$. Such rules can be classified into *consistent blocks* according to the following definition.

Definition 6 Given $(i, \alpha, \alpha', u, v)$, $1 \leq i \leq q$, $\alpha, \alpha' \in \{0, +, -\}$, and $u, v \in M_f(\Gamma)$, the consistent block associated with $(i, \alpha, \alpha', u, v)$ is the set:

$$B_{i,\alpha,\alpha',u,v} = \{r \in \mathcal{R} \mid LHS(r) = (i, \alpha, u, v) \wedge charge(RHS(r)) = \alpha'\}$$

Extending left-hand side definition, $LHS(B_{i,\alpha,\alpha',u,v}) = (i, \alpha, u, v)$ and the charge is α.

That is, the consistent block associated with $(i, \alpha, \alpha', u, v)$ is the set of rules $r \in \mathcal{R}$ whose left-hand side is (i, α, u, v) and such that the electrical charge of the membrane after their execution is the same: α'. Therefore, any pair of rules from $B_{i,\alpha,\alpha',u,v}$ can be simultaneously applied provided that there are enough objects. In that case, membrane i will modify its polarization from α to α'.

We recall that, according to the semantics of our model, the sum of probabilities of all the rules belonging to the same block is always equal to 1; in particular, rules whose probability is equal to 1 form individual blocks. Note that rules with overlapping (but different) left–hand sides are classified into different blocks. The latter leads to object *competition*, what is a critical aspect to manage with the simulation algorithms. This suggests introducing the concept of mutual consistency among blocks.

Definition 7 Two blocks $B_{i_1,\alpha_1,\alpha'_1,u_1,v_1}$ and $B_{i_2,\alpha_2,\alpha'_2,u_2,v_2}$ are mutually consistent with each other, if and only if $(i_1 = i_2 \wedge \alpha_1 = \alpha_2) \Rightarrow (\alpha'_1 = \alpha'_2)$.

That is, for two blocks mutually consistent with each other and with the same charge any rule of the first block can be simultaneously applied together with any rule of the second block, provided that there are enough objects.

Since rules in $\mathcal{R}_E$ (communication between environments) do not affect the electrical charge of any membrane and they do not interfere at all with the applicability of the rest of the rules, a block B_{e_{j1},x_1} is always mutually consistent with any other B_{e_{j2},x_2}, as well as with any block $B_{i,\alpha,\alpha',u,v}$ of evolution rules.

Definition 8 Given $x \in \Gamma$, $l \in H$, and $r \in \mathcal{R}$ such that $LHS(r) = (i, \alpha, u, v)$, we say that (x, l) appears in $LHS(r)$ with multiplicity k in any of the following cases:

- $l = i$, and x appears in multiset v with multiplicity k
- l is the label of the parent of membrane i, and x appears in multiset u with multiplicity k

4.3.3 Some Properties of PDP Systems Models

The following four properties [2] are desirable for any computational model:

- *Relevance:* A computational model must capture some essential features of the system investigated.
  It should present a unifying specification of its physical structure and the different components that constitute the system, the interaction between them and their dynamical behaviour.
  PDP systems are able to successfully capture the relevance of the underlying system by associating objects to individuals or other significant elements, and the interaction between them by means of evolution rules.

- *Computability:* It should be possible to implement or simulate a model in a computer, so that one can run simulations to study the dynamics of the system by manipulating experimental conditions in the model. In this manner, the model can be experimentally validated, and moreover the behaviour of the system under different scenarios of interest can be studied. The computability of the model also allows us to perform model checking and similar techniques to infer and to study qualitative and quantitative properties of the system in an automatic way. In this respect, the model should be mathematically tractable. That is, it should be possible to perform mathematical analysis on it.
  Since P systems are computing devices, the computability of PDP systems is an inherent property. Moreover, the inference engine (as detailed in Sect. 4.4) captures the semantics of the studied dynamics models.
- *Understandability:* The abstract formalism used should correspond well to the informal concepts and ideas which are used by the experts in the population under study.
  As PDP systems objects and rules capture in a simple manner the behaviour and dynamical interaction of the relevant elements in the modelled system, they are easy to be understood by experts in the problem domain.
- *Extensibility:* It should be easy to identify the different components and characteristics of the systems that are essential in the context of the management or scientific problem to be solved or comprehended [17], so they can be rearranged, duplicated, composed, etc. in an easy way to produce other models. Models of complex systems should also be extensible to higher levels of organizations.
  PDP system rule design is module-oriented, favouring low coupling between them. This approach allows a mostly independent modules development while enabling the addition, removal and/or reuse of them to the system. As modules are designed in a separate way, different levels of complexity can be achieved in each one.

4.4 An Inference Engine: The DCBA Algorithm

Within the framework of Membrane Computing the goal of a simulation algorithm is to select and execute, for each time unit, an applicable multiset of rules. The Direct distribution based on Consistent Blocks Algorithm (DCBA) follows this approach, paying special attention to the proportional distribution of objects among competing blocks (with overlapping LHS), thus determining the number of times that each rule in $\bigcup_{j=1}^{m} \mathcal{R}_{\Pi_j} \cup \mathcal{R}_E$ is applied. See [18] for a more detailed explanation and examples of how this algorithm works.

Algorithm 1 describes the main loop of the DCBA. It follows the same general scheme as its predecessors, DNDP and BBB [19] where the simulation of a transition step is structured in two stages: selection and execution. The firststage (selection)

selects which rules are to be applied (and how many times) on each environment. The second stage (execution) implements the effects of applying the previously selected rules, yielding the next configuration of the PDP system. Note that, although every Π_j shares the same set of rules R, the probability functions may differ for each environment.

As shown in Algorithm 1, the selection stage consists of three phases: Phase 1 distributes objects to the blocks in a proportional way, as it will be explained later on; Phase 2 assures the *maximality* by assigning to some of the blocks the objects still unassigned after Phase 1; and finally, Phase 3 translates block applications into rule applications by computing random numbers following the multinomial distribution with the corresponding *probabilities*.

Algorithm 1 DCBA MAIN PROCEDURE

Require: A Population Dynamics P system of degree (q, m), $T \geq 1$, and $A \geq 1$
1: $C_0 \leftarrow$ Initial configuration
2: *INITIALIZATION* ▷ (Algorithm 2)
3: **for** $t \leftarrow 1$ to T **do**
4: Calculate probability functions $f_{r,j}(t)$ and $p_r(t)$
5: $C'_t \leftarrow C_{t-1}$
6: *SELECTION* of rules:

- *PHASE 1*: Distribution ▷ (Algorithm 3)
- *PHASE 2*: Maximality ▷ (Algorithm 4)
- *PHASE 3*: Probabilities ▷ (Algorithm 5)

7: *EXECUTION* of rules. ▷ (Algorithm 6)
8: $C_t \leftarrow C'_t$
9: **end for**

The INITIALIZATION procedure (Algorithm 2) constructs a static distribution table $\mathcal{T}_j$ for each environment. Two variables, B^j_{sel} and R^j_{sel}, are also initialized, in order to store the selected multisets of blocks and rules, respectively.

Observation 1 *Each column label of the tables $\mathcal{T}_j$ contains the information of the corresponding block left–hand side.*

Observation 2 *Each row of the tables $\mathcal{T}_j$ contains the information related to the object competitions: for a given object, its row indicates which blocks are competing for it (those columns having non–null values).*

Algorithm 2 INITIALIZATION

1: Construction of the *static distribution* table $\mathcal{T}$:

- Column labels: consistent blocks $B_{i,\alpha,\alpha',u,v}$ of rules from R.
- Row labels: pairs (o, i) and $(x, 0)$, for all objects $o \in \Gamma$, $x \in \Sigma$ and membrane label i, being 0 the identifier of the environment.
- For each cell of the table: place $\frac{1}{k}$ if its row label (o, i) appears with multiplicity $k > 0$ in the LHS of its column label $B_{i,\alpha,\alpha',u,v}$.

2: **for** $j = 1$ **to** m **do** ▷ (Construct the *static expanded* tables $\mathcal{T}_j$)
3: $\mathcal{T}_j \leftarrow \mathcal{T}$. ▷ (Initialize the table with the original $\mathcal{T}$)
4: For each rule block $B_{e_j,x}$ from $\mathcal{R}_E$, add a column labelled by $B_{e_j,x}$ to the table $\mathcal{T}_j$; place the value 1 at row $(x, 0)$ for that column
5: Initialize the multisets $B^j_{sel} \leftarrow \emptyset$ and $R^j_{sel} \leftarrow \emptyset$
6: **end for**

The distribution of objects among the blocks with overlapping LHS (competing blocks) is performed in selection Phase 1 (Algorithm 3). The expanded static tables $\mathcal{T}_j$ are used for this purpose in each environment, together with three different filter procedures. FILTER 1 discards the columns of the table corresponding to non-applicable blocks due to mismatch charges in the LHS and in the configuration C'_t. Then, FILTER 2 discards the columns with objects in the LHS not appearing in C'_t. Finally, in order to save space in the table, FILTER 3 discards empty rows. These three filters are applied at the beginning of Phase 1, and the result is a *dynamic table* $\mathcal{T}^t_j$ (for the environment j and time step t).

The semantics of the modelling framework requires a set of mutually consistent blocks before distributing objects to the blocks. For this reason, after applying FILTERS 1 and 2, the mutual consistency is checked. Note that this checking can be easily implemented by a loop over the blocks. If it fails, meaning that an inconsistency was encountered, the simulation process is halted, providing a warning message to the user. Nevertheless, it could be interesting to find a way to continue the execution by non-deterministically constructing a subset of mutually consistent blocks. Since this method can be exponentially expensive in time, it is optional for the user whether to activate it or not.

Once the columns of the *dynamic table* $\mathcal{T}^t_j$ represent a set of mutually consistent blocks, the distribution process starts. This is carried out by creating a temporal copy of $\mathcal{T}^t_j$, called $\mathcal{TV}^t_j$, which stores the following products:

- The normalized value with respect to the row: this is a way to *proportionally* distribute the corresponding object along the blocks. Since it depends on the multiplicities in the LHS of the blocks, the distribution, in fact, penalizes the blocks requiring more copies of the same object. This is inspired in the amount of energy required to gather individuals from the same species.
- The value in the dynamic table (i.e. $\frac{1}{k}$): this indicates the number of possible applications of the block with the corresponding object.
- The multiplicity of the object in the configuration C'_t: this performs the distribution of the number of copies of the object along the blocks.

Algorithm 3 SELECTION PHASE 1: DISTRIBUTION

for $j = 1$ **to** m **do** ▷ (For each environment e_j)
  Apply filters to table $\mathcal{T}_j$ using C'_t, obtaining $\mathcal{T}^t_j$, as follows:
    a. $\mathcal{T}^t_j \leftarrow \mathcal{T}_j$
    b. FILTER 1 ($\mathcal{T}^t_j$, C'_t)
    c. FILTER 2 ($\mathcal{T}^t_j$, C'_t)
    d. Check *mutual consistency* for the blocks remaining in $\mathcal{T}^t_j$. **If** there is at least one inconsistency **then** report the information about the error, and optionally halt the execution (in case of not activating step *3*)
    e. FILTER 3 ($\mathcal{T}^t_j$, C'_t)
  (*OPTIONAL*) Generate a set S^t_j of sub-tables from $\mathcal{T}^t_j$, formed by sets of *mutually consistent* blocks, in a maximal way in $\mathcal{T}^t_j$ (by the inclusion relationship). Replace $\mathcal{T}^t_j$ with a randomly selected table from S^t_j.
  $a \leftarrow 1$
  repeat
    for all rows X in $\mathcal{T}^t_j$ **do**
      $RowSum_{X,t,j} \leftarrow$ total sum of the non-null values in the row X
    end for
    $\mathcal{TV}^t_j \leftarrow \mathcal{T}^t_j$ ▷ (A temporal copy of the dynamic table)
    for all non-null positions (X, Y) in $\mathcal{T}^t_j$ **do**
      $mult_{X,t,j} \leftarrow$ multiplicity in C'_t at e_j of the object at row X
      $\mathcal{TV}^t_j(X, Y) \leftarrow \lfloor mult_{X,t,j} \cdot \frac{(\mathcal{T}^t_j(X,Y))^2}{RowSum_{X,t,j}} \rfloor$
    end for
    for all not filtered column, labelled by block B, in $\mathcal{T}^t_j$ **do**
      $N_B \leftarrow \min_{X \in rows(\mathcal{T}^t_j)}(\mathcal{TV}^t_j(X, B))$ ▷ (The minimum of the column)
      $B^j_{sel} \leftarrow B^j_{sel} + \{B^{N_B}\}$ ▷ (Accumulate the value to the total)
      $C'_t \leftarrow C'_t - LHS(B) \cdot N_B$ ▷ (Delete the LHS of the block)
    end for
    FILTER 2 ($\mathcal{T}^t_j$, C'_t)
    FILTER 3 ($\mathcal{T}^t_j$, C'_t)
    $a \leftarrow a + 1$
  until $(a > A) \vee$ (*all the selected minimums at step 15 are* 0)
end for

Algorithm 4 SELECTION PHASE 2: MAXIMALITY

for $j = 1$ **to** m **do** ▷ (For each environment e_j)
  Set a random order to the blocks remaining in the last updated table $\mathcal{T}^t_j$
  for all block B, following the previous random order **do**
    $N_B \leftarrow$ number of possible applications of B in C'_t
    $B^j_{sel} \leftarrow B^j_{sel} + \{B^{N_B}\}$ ▷ (Accumulate the value to the total)
    $C'_t \leftarrow C'_t - LHS(B) \cdot N_B$ ▷ (Delete the LHS of block B, N_B times)
  end for
end for

After the object distribution process, the number of applications for each block is calculated by selecting the minimum value in each column. This number is then used for consuming the LHS from the configuration. However, this application could be non–maximal. The distribution process can eventually deliver objects to blocks that are restricted by other objects. As this situation may occur frequently, the distribution and the configuration update process is performed A times, where A is an input parameter referring to *accuracy*. The more the process is repeated, the more accurate the distribution becomes at the expense of simulation performance. We have experimentally checked that $A = 2$ gives the best accuracy/performance ratio. In order to efficiently repeat the loop for A, and also before going to the next phase (maximality), it is interesting to apply FILTERS 2 and 3 again.

After Phase 1, it may be the case that some blocks are still applicable to the remaining objects. This may be caused by a low A value or by rounding artefacts in the distribution process. Due to the requirements of P systems semantics, a maximality phase is now applied (Algorithm 4). Following a random order, a maximal number of applications is calculated for each block which is still applicable.

After the application of Phases 1 and 2, a maximal multiset of selected (mutually consistent) blocks has been computed. The output of the selection stage has to be, however, a maximal multiset of selected rules. Hence, Phase 3 (Algorithm 5) passes from blocks to rules, by applying the corresponding probabilities (at the local level of blocks). The rules belonging to a block are selected according to a multinomial distribution $M(N, g_1, \ldots, g_l)$, where N is the number of applications of the block, and $g_1, \ldots, g_l$ are the probabilities associated with the rules $r_1, \ldots, r_l$ within the block, respectively.

Algorithm 5 SELECTION PHASE 3: PROBABILITY

1: **for** $j = 1$ **to** m **do** ▷ (For each environment e_j)
2: **for all** block $B^{N_B} \in B^j_{sel}$ **do**
3: Calculate $\{n_1, \ldots, n_l\}$, a random multinomial $M(N_B, g_1, \ldots, g_l)$ with respect to the probabilities of the rules $r_1, \ldots, r_l$ within the block.
4: **for** $k = 1$ **to** l **do**
5: $R^j_{sel} \leftarrow R^j_{sel} + \{r_k^{n_k}\}$.
6: **end for**
7: **end for**
8: Delete the multiset of selected blocks $B^j_{sel} \leftarrow \emptyset$. ▷ (Useful for the next step)
9: **end for**

Finally, the execution stage (Algorithm 6) is applied. This stage consists on adding the RHS of the previously selected multiset of rules, as the objects present on the LHS of these rules have already been consumed. Moreover, the indicated membrane charge is set.

Algorithm 6 EXECUTION

1: **for** $j = 1$ **to** m **do** ▷ (For each environment e_j)
2: **for all** rule $r^n \in R^j_{sel}$ **do** ▷ (Apply the RHS of selected rules)
3: $C'_t \leftarrow C'_t + n \cdot RHS(r)$
4: Update the electrical charges of C'_t from $RHS(r)$.
5: **end for**
6: Delete the multiset of selected rules $R^j_{sel} \leftarrow \emptyset$. ▷ (Useful for the next step)
7: **end for**

4.5 Simulation of PDP Systems

As already mentioned in Sect. 4.1, computational models (and, in particular, P system–based models) are assumed to rely on software simulators that carry out virtual experiments in order to evaluate the usefulness of the formal model defined. At the current stage, multiple software tools have been developed in the Membrane Computing field (see e.g. [20]), most of them specifically tailored for a single type of P systems.

This section describes a set of software tools providing the needed infrastructure to define, simulate and virtually experiment with PDP systems.

4.5.1 P-Lingua, and the pLinguaCore Library

Each P system model features characteristic semantic constraints that determine not only the type of rules allowed, but also the way in which rules are applied. This general information is embedded in the simulator engine, but in order to perform simulations, we need additional information regarding the P system to be simulated. The term *simulator input* will be used to refer to this initial data which provides the formal specification of the P system.

One possible approach to implement the simulator input could be to require a specific input file for each simulator, or directly insert the data into the source code. This approach imposes a specific format for the input, given by the data structures used in the design of each software simulator. Moreover, if we wanted to run many different experiments, a great redundant effort would be required. An alternative approach could be to standardize the simulator input by establishing a common format. These two approaches raise a trade–off. On the one hand, specific simulator inputs could be defined in a more straightforward way, as the used format is closer to the P system features to simulate. On the other hand, although the latter approach involves analysing different P system models to develop a standard format, there is no need to develop completely a new simulator every time a new P system should be simulated, as it is possible to use a common software library in order to parse the standard input format. Moreover, users would no longer be obliged to learn a

new input format every time they use a different simulator, and they would not need to rewrite the specification of P systems which are going to be simulated every time they move on to another model. This second approach is the one considered in P-Lingua [21, 22], a specification language to define P systems within several P system models.

The P–Lingua project also provides free software tools under GNU GPL license [23] for compilation, simulation and debugging tasks. The main tools are integrated in a Java library called *pLinguaCore*. These tools include a parser to handle P-Lingua input files and check possible programming errors (both lexical/syntax and semantics). They also include several built-in sequential simulators to generate P system computations for the supported models. Furthermore, these tools can export the P–Lingua definition file into other file formats in order to get interoperability between different software environments. The approach to define the simulator input by using the P-Lingua framework is illustrated in Fig. 4.1 where we can see how P system definitions in P-Lingua format can be translated into other file formats by using pLinguaCore, eventually becoming the input for different simulator environments and gaining interoperability. Moreover, such input is free of programming errors, since the parser inside pLinguaCore has already checked them.

Since the initial release version, each new update of P-Lingua and pLinguaCore includes new supported models and implements new simulation algorithms, while in parallel fixing some bugs found. The current release (pLinguaCore 3.0, available for download at the P-Lingua website [22]), covers the following P system models:

- (Cell-like) Transition P systems.
- (Cell-like) Symport/antiport P systems.
- (Cell-like) Active membranes with division rules.
- (Cell-like) Active membranes with creation rules.
- (Cell-like) Probabilistic P systems.
- Tissue-like P systems with division rules.
- Population Dynamics P systems (PDP systems)

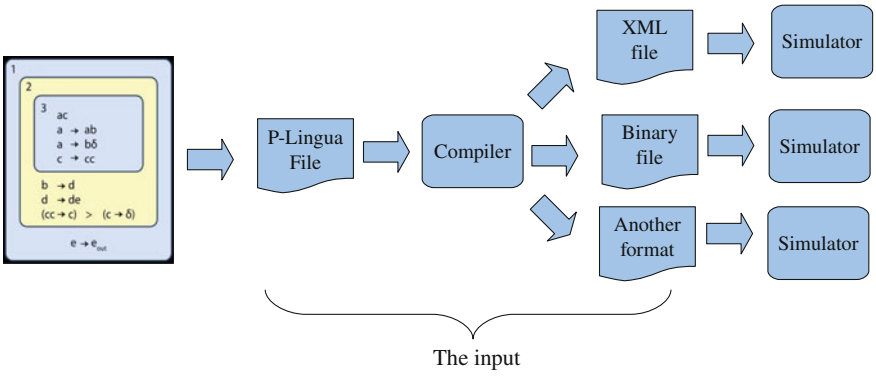

Fig. 4.1 The P-Lingua approach to define simulators input

As mentioned above, pLinguaCore includes at least one built–in simulator for each supported model. In particular, a sequential implementation of the algorithm discussed in this chapter for PDP systems (DCBA) is included, along with some alternatives (e.g. DNDP).

P-Lingua and pLinguaCore 3.0 can be used to assist in the design of PDP systems. It can also be used as simulation core for other software tools such as MeCoSim, as it will be explained below.

4.5.2 The Visual Environment MeCoSim

The availability of a general language, P-Lingua, to define P systems, along with a set of tools to parse, simulate and debug models based on this kind of systems, enables the designer to work with P systems in a high level of abstraction. In this context, an abstract problem is defined in P-Lingua format, but a number of scenarios (instances of the problem) can be analysed and simulated for that abstract problem. As the variety and size of the problems to model and simulate with this framework increase, the need for visual tools for modelling and simulation arises—in such a way that a P-Lingua model for a family of problems can be specified by a designer user, while the information about each specific scenario can be included by an end user in a visual way.

This need leads to software developments for providing Graphical User Interfaces (GUI) to introduce the input data for a specific scenario, both for the description of the initial configuration and for some variable parameters for the model. Moreover, such GUIs also take care of rendering some outputs showing the required information, possibly including tables and charts. Some examples of these applications were successfully used to model and simulate ecosystems [11, 13, 24].

While P-Lingua framework provided a general mechanism to define, simulate and debug P systems, each family of problems implied the design, development and maintenance of different ad-hoc GUIs. MeCoSim [25, 26], Membrane Computing Simulator, arises to solve the need of developing *ad-hoc* applications, by providing a general solution for defining custom GUIs adapted for each problem. The user can define a custom structure of tabs, input and output tables, charts, graphs, etc. for each given family of problems, so MeCoSim can be viewed as a new layer above P-Lingua framework, complementing its functionality.

The initial goal has been extended to provide a general integrated environment to work with P systems, including functionalities for modelling, simulation, debugging, analysis, properties extraction and verification of models based on P systems. MeCoSim platform provides a plugins architecture to extend the initial functionality with external programs.

4.5.3 Accelerating PDP Systems Simulations

Previous sections have introduced a simulation algorithm, DCBA, along with a sequential implementation inside pLinguaCore. However, the simulations were relatively slow, since pLinguaCore library is not performance-oriented.

In order to overcome this limitation, a more efficient implementation based on C++ and OpenMP was presented in [27], taking advantage of modern multicore architectures (e.g. 4-core Intel I5 and I7 microprocessors). These simulators save on memory by avoiding the creation of the static distribution table required in Phase 1. This feature (called *virtual table* solution) is carried out by translating the operations over the table to operations directly to rule blocks information. Only the row sums and the column minimums are stored in auxiliary data structure together with flags denoting that a column has been filtered. Concerning the parallelism implementation, simulations and environments were distributed along the cores. Runtime gains of up to 2.5× were achieved, so these preliminary results indicate that the simulation of PDP systems are memory bound.

Furthermore, since GPU computing [28] has been successfully used to implement other P systems simulators [29–31], the simulation of PDP systems on this technology was a natural step. NVIDIA's *CUDA* (Compute Unified Device Architecture) [32, 33] provides developers with a high-level programming model that allows them to take advantage of the NVIDIA's GPU parallel architecture. Most recent NVIDIA's GPUs (with Kepler architecture) provide thousands of cores, and a fast memory access. However, programs must fit data parallelism to achieve best performance.

The CUDA-based simulator for PDP systems [34] distributes simulations and environments along the multiprocessors of the GPU, and the rule blocks are parallelized within each multiprocessor. It has been benchmarked against a set of randomly generated PDP systems (without biological meaning), achieving speedups of 7x for large sizes (PDP systems with 50,000 rule blocks, 20 environments and running 50 simulations) on a NVIDIA Tesla C1060 GPU (240 cores and 4 GBytes of memory) in comparison to the multi–core CPU version. The source code is available under GPLv3 license, within the PMCGPU project [35], codenamed as *ABCD-GPU*.

4.6 A Case Study: Pandemics

A *pandemic* is an outbreak of a disease that occurs over a wide geographic area and affects an exceptionally high proportion of the population.

In this section, we present a SIR computational model based on Population Dynamics P systems. SIR is an acronyms: **S** stands for the *susceptible population*, those who are not yet infected, but may become infected; **I** stands for the *infected population*, those who are ill and can transmit the disease, and **R** stands for the dead or recovered individuals that are removed from the infected population and cannot transmit the disease. The SIR mathematical model for pandemics is an ODEs based

model that has been used to understand the spatial-temporal transmission dynamics of influenza. This model refers to systems where no human interaction is considered (e.g. vaccination campaigns, declarations of quarantine, prophylactic actions, etc.).

Our case study will be restricted to three physically separated communities (e.g. in different cities). Each community is formed by four neighbourhoods, where basic facilities for daily life are available: schools, work places, shops, etc. Individuals in a neighbourhood are organized in families (families may have different structure or number of members). In addition, seven groups will be considered, according to their age: Daycares/Playgroups, Elementary schools, Middle schools, High schools, and two groups for Adults (19–53 years old, and over 53 years old). A susceptible person can be infected either in the bosom of the family, at work, or in leisure time (at the neighbourhood or when travelling). Figure 4.2 displays the network corresponding to breeding grounds for each age group.

4.6.1 Design of a PDP Modelling Pandemics

In this section, the model for the exposed case study by using PDP systems is presented. The model is composed by seven modules of rules. The first one of them (initial infection) will only be executed when the model is initialized choosing randomly infected people within the population. The execution of the remaining six

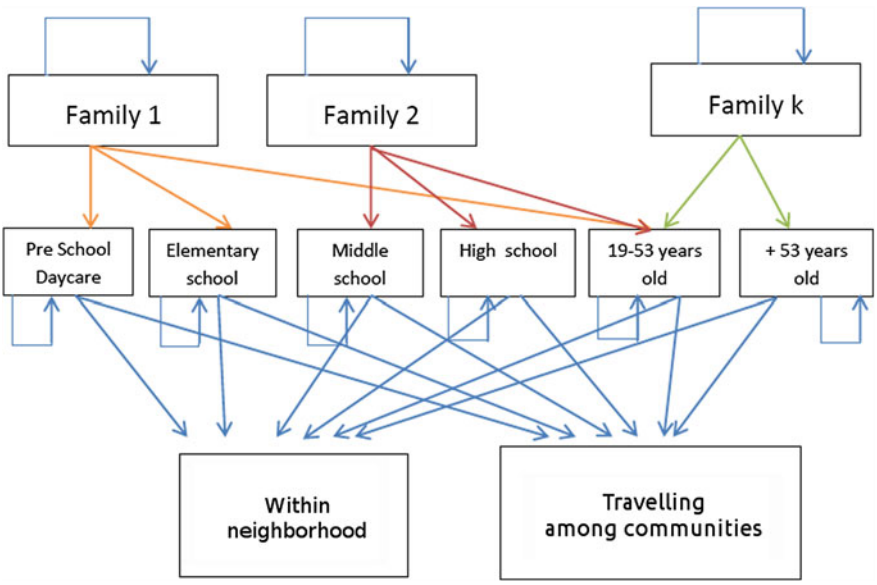

Fig. 4.2 A network of the infection flow

modules will be interpreted as the evolution of the pandemic scenario during one day.

The model consists of a PDP system of degree (2, 3) taking $T \geq 1$ time units,

$$\Pi = (G, \Gamma, \Sigma, T, \mathcal{R}_E, \mu, \mathcal{R}, \{f_{r,j} \mid r \in \mathcal{R}, 1 \leq j \leq 3\}, \{\mathcal{M}_{1,j}, \mathcal{M}_{2,j}, 1 \leq j \leq 3\})$$

where:

- $G = (V, S)$ is a complete directed graph, with $V = \{e_1, e_2, e_3\}$ and $S = \{(e_i, e_j) \mid 1 \leq i, j \leq 3\}$ (Fig. 4.3 shows the graph, including the P system inside each environment).

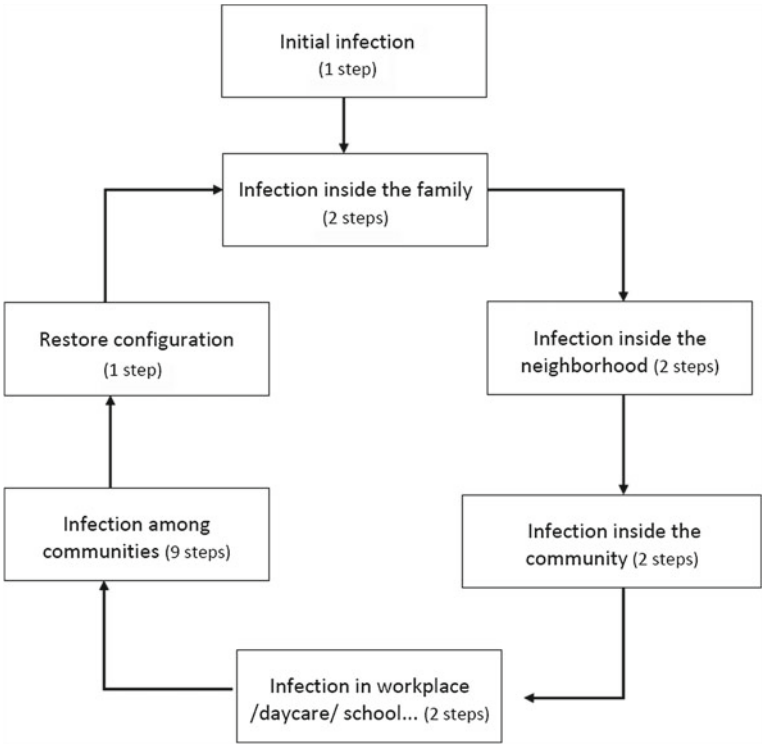

Fig. 4.3 Modules correspponding to the SIR model

- The working alphabet, Γ is the set
  $\{a_n \mid 1 \leq n \leq 4\} \cup \{C_i \mid 0 \leq i \leq 18\} \cup$
  $\{X_{f,g}, \overline{X}_{f,g}, Y_{f,g}, Z_{f,g}, V_{f,g}, A_{f,g}, S_{f,g},$
  $R_{f,g} \mid 1 \leq f \leq 4F,\ 1 \leq g \leq 7\} \cup$
  $\{\overline{A}_{f,g,i}, \overline{S}_{f,g,i} \mid$
  $1 \leq f \leq 4F,\ 1 \leq g \leq 7,\ 2 \leq i \leq 5\} \cup$
  $\{A'_{f,g,j}, \overline{X}'_{f,g,j}, X'_{f,g,j} V'_{f,g,j}, W_{f,g,j}, W'_{f,g,j} \mid$
  $1 \leq f \leq 4F,\ 1 \leq g \leq 7,\ 1 \leq j \leq 3\} \cup$
  $\{\widehat{A}, \widehat{X}, \widetilde{Y}, \widetilde{Z}, \widetilde{V}, \widetilde{W}\} \cup \{\widetilde{M}'_j \mid 1 \leq j \leq 3\}$
  Symbols a_n $(1 \leq n \leq 4)$ are used to represent in the initial configuration the individuals initially infected in each neighbourhood. The uninfected individuals are represented by symbols $X_{f,g}$, $X'_{f,g,j}$, $Y_{f,g}$, $Z_{f,g}$, $V_{f,g}$, $V'_{f,g,j}$, $W_{f,g,j}$, $W'_{f,g,j}$ (index f is associated with the families, index g is associated with the age groups, and index j is associated with the environments representing communities); the infected individuals are represented by symbols $\overline{X}_{f,g}$ and $\overline{X}'_{f,g,j}$; symbols $\overline{S}_{f,g,i}$ and $\overline{A}_{f,g,i}$ represent symptomatic and asymptomatic individuals, respectively (index i is associated with the stage of illness, indicating days since infection); symbols $\widetilde{X}$, $\widetilde{Y}$, $\widetilde{Z}$, $\widetilde{V}$, $\widetilde{W}$ are used to model the interactions of individuals infected by others represented by symbols $S_{f,g}$ and $A_{f,g}$ (symptomatic and asymptomatic, respectively); $\widetilde{V}'_j$, $A'_{f,g,j}$ and $\widehat{A}$ are auxiliary symbols used for travelling among communities; $R_{f,g}$ represent individuals who were infected but have been able to recover. A global clock, C_i, controls the evolution of the P system and the charge changes of membrane 2 along a cycle (that is, a day).

- $\Sigma = \{A'_{f,g,j}, V'_{f,g,j} \mid 1 \leq f \leq 4F, 1 \leq g \leq 7, 1 \leq j \leq 3\} \cup$
  $\{\widetilde{V}'_j \mid 1 \leq j \leq 3\} \cup$
  $\{X'_{f,g,i,j} \mid 1 \leq f \leq 4F, 1 \leq g \leq 7, 0 \leq i \leq 1, 1 \leq j \leq 3\}$

- $T = 18 \cdot Days$, where $Days$ is the number of days to simulate. Each day in the real scenario is simulated by 18 computational steps.

- $\mathcal{R}_E = \{r_{e_1,f,g,j,i}, r_{e_2,f,g,j,i} \mid 1 \leq f \leq 4F, 1 \leq g \leq 7, 1 \leq i, j \leq 3\} \cup$
  $\{r_{e_3,j,i} \mid 1 \leq i, j \leq 3\} \cup$
  $\{r_{e_4,f,g,i,j,p} \mid 1 \leq f \leq 4F, 1 \leq g \leq 7, 0 \leq i \leq 1, 1 \leq j, p \leq 3, j \neq p\} \cup$
  $\{r_{e_5,f,g,i,j} \mid 1 \leq f \leq 4F, 1 \leq g \leq 7, 0 \leq i \leq 1, 1 \leq j \leq 3\}$

- $\mu = [\,[\,]_2\,]_1$ is the membrane structure, and the corresponding initial multisets in the environment j are:

  – $\mathcal{M}_{1,j} = \emptyset$
  – $\mathcal{M}_{2,j} = \{\, X_{f,g}^{q_{f,g,j}} \mid 1 \leq f \leq 4F, 1 \leq g \leq 7, 1 \leq j \leq 3\} \cup$
  $\{a_n^{l_{n,j}} \mid 1 \leq n \leq 4, 1 \leq j \leq 3\} \cup \{C_0\}$

  Objects $X_{f,g}$ represent individuals in step 0, for families f from 1 to $4F$, with F the number of families per neighbourhood; they are divided by index g in 7 age groups as explained before. Objects a_n, as mentioned before, repre-

sent individuals initially infected at the beginning of the simulated period. The amount of objects depends, for each environment, on parameters $q_{f,g,j}$ and $l_{n,j}$, that should be specified by the user.

- In what follows we enumerate the rules in $\mathcal{R} \cup \mathcal{R}_E$ along with some comments on their functioning:

Initial infection

These rules take care of generating the initial scenario, randomly distributing symptomatic and asymptomatic individuals along the system, according to the parameters $l_{n,j}$. These parameters are provided by the user before starting the simulation, and they indicate the initial amount of infected individuals on each environment e_j (represented by the multiplicity of objects a_n).

These rules are only applied in the first step of the computation, they are not part of the loop that represents a day (see Fig. 4.4).

Each infected individual is estimated to interact with 20 other individuals during one day, and thus 20 copies of $S_{f,g}$ ($A_{f,g}$, respectively) are generated for each symptomatic (asymptomatic, respectively) individual.

Generate symptomatic individuals

$$r_{1,f,g,i,n} \equiv [a_n \; X_{f,g} \xrightarrow{ps/4} \overline{S}_{f,g,i} \; S_{f,g}^{20}]_2 \begin{cases} (n-1)F < f \leq nF, \\ 1 \leq g \leq 7, \\ 2 \leq i \leq 5, \\ 1 \leq n \leq 4 \end{cases}$$

Generate asymptomatic individuals

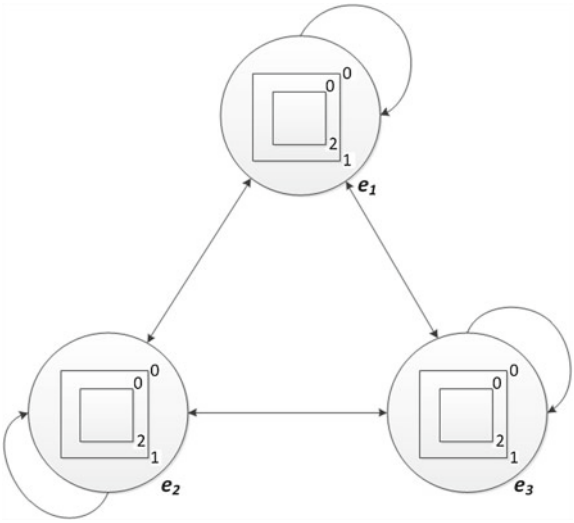

Fig. 4.4 Structure of the environment graph of the PDP system defined

$$r_{2,f,g,i,n} \equiv [a_n\ X_{f,g} \xrightarrow{(1-ps)/4} \overline{A}_{f,g,i}\ \widetilde{X}\ A^{20}_{f,g}]_2 \begin{cases} (n-1)F < f \leq nF, \\ 1 \leq g \leq 7, \\ 2 \leq i \leq 5, \\ 1 \leq n \leq 4 \end{cases}$$

Clock advance

$$r_3 \equiv [C_0 \longrightarrow C_1]_2$$

Infection inside the family (first step)

As we said before, any infected individual can spread the disease by interacting with other individuals. However, we distinguish two types of behaviour: symptomatic individuals are supposed to stay at home, hence only interacting with their relatives, while asymptomatic individuals are supposed to be unaware of their infection. In this module, rules $r_{4,f,g,g'}$ and $r_{5,f,g,g'}$ deal with the possible infection within the family. The rest of modules will cover different reasons for interaction between susceptible individuals and asymptomatic infected individuals.

The model also has special rules (see $r_{6,f,g}$ and $r_{7,f,g}$) to deal with the fact that two infected people, when interacting, cannot get infected again. Please note that these rules compete for objects $A_{f,g}$ and $S_{f,g}$ against rules $r_{4,f,g,g'}$ and $r_{5,f,g,g'}$.

Infection of susceptible individuals

$$r_{4,f,g,g'} \equiv [S_{f,g}\ X_{f,g'}]_2 \longrightarrow [S_{f,g}\ \overline{X}_{f,g'}\ \widetilde{X}]_2^- \begin{cases} 1 \leq f \leq 4F, \\ 1 \leq g, g' \leq 7 \end{cases}$$

$$r_{5,f,g,g'} \equiv [A_{f,g}\ X_{f,g'}]_2 \longrightarrow [A_{f,g}\ \overline{X}_{f,g'}\ \widetilde{X}]_2^- \begin{cases} 1 \leq f \leq 4F, \\ 1 \leq g, g' \leq 7 \end{cases}$$

Interaction between infected individuals

$$r_{6,f,g} \equiv [S_{f,g}\ \widetilde{X}]_2 \longrightarrow [S_{f,g}\ \widetilde{X}]_2^- \begin{cases} 1 \leq f \leq 4F, \\ 1 \leq g \leq 7 \end{cases}$$

$$r_{7,f,g} \equiv [A_{f,g}\ \widetilde{X}]_2 \longrightarrow [A_{f,g}\ \widetilde{X}]_2^- \begin{cases} 1 \leq f \leq 4F, \\ 1 \leq g \leq 7 \end{cases}$$

Clock advance with charge change

$$r_8 \equiv [C_1]_2 \longrightarrow [C_2]_2^-$$

Infection inside the family (last step)

We need to avoid that rules for infection inside the family are applicable in more steps. Note that by changing the charge we guarantee that all symbols are renamed by means of the following rules:

Renaming of susceptible (X) and infected ($\widetilde{X}$) individuals

$$r_{9,f,g} \equiv [X_{f,g}]_2^- \longrightarrow [Y_{f,g}]_2 \begin{cases} 1 \leq f \leq 4F, \\ 1 \leq g \leq 7 \end{cases}$$

$$r_{10} \equiv [\widetilde{X}]_2^- \longrightarrow [\widetilde{Y}]_2$$

Removal of the possibility of infection by symptomatic individuals

$$r_{11,f,g} \equiv [S_{f,g}]_2^- \longrightarrow [\#]_2 \left\{ \begin{array}{l} 1 \leq f \leq 4F, \\ 1 \leq g \leq 7 \end{array} \right.$$

Clock advance with charge change

$$r_{12} \equiv [C_2]_2^- \longrightarrow [C_3]_2$$

Infection inside the neighborhood (first step)

The following four blocks of rules represent the probability of infection by interacting with a "neighbour". The set of $4F$ families living in each community is divided into four intervals, representing four neighbourhoods. Thus, two individuals $A_{f,g}$ and $Y_{f',g'}$ live in the same neighbourhood if their family indexes (f and f') belong to the same interval.

This module includes, like discussed above, rules to deal with the interaction between two infected people (see $r_{17,f,g}$).

Asymptomatic individuals affecting susceptible ones

$$r_{13,f,f',g,g'} \equiv [A_{f,g}\ Y_{f',g'}]_2 \longrightarrow [A_{f,g}\ \overline{X}_{f',g'}\ \widetilde{Y}]_2^- \left\{ \begin{array}{l} 1 \leq f, f' \leq F, \\ 1 \leq g, g' \leq 7 \end{array} \right.$$

$$r_{14,f,f',g,g'} \equiv [A_{f,g}\ Y_{f',g'}]_2 \longrightarrow [A_{f,g}\ \overline{X}_{f',g'}\ \widetilde{Y}]_2^- \left\{ \begin{array}{l} F < f, f' \leq 2F, \\ 1 \leq g, g' \leq 7 \end{array} \right.$$

$$r_{15,f,f',g,g'} \equiv [A_{f,g}\ Y_{f',g'}]_2 \longrightarrow [A_{f,g}\ \overline{X}_{f',g'}\ \widetilde{Y}]_2^- \left\{ \begin{array}{l} 2F < f, f' \leq 3F, \\ 1 \leq g, g' \leq 7 \end{array} \right.$$

$$r_{16,f,f',g,g'} \equiv [A_{f,g}\ Y_{f',g'}]_2 \longrightarrow [A_{f,g}\ \overline{X}_{f',g'}\ \widetilde{Y}]_2^- \left\{ \begin{array}{l} 3F < f, f' \leq 4F, \\ 1 \leq g, g' \leq 7 \end{array} \right.$$

Interaction between infected individuals

$$r_{17,f,g} \equiv [A_{f,g}\ \widetilde{Y}]_2 \longrightarrow [A_{f,g}\ \widetilde{Y}]_2^- \left\{ \begin{array}{l} 3F < f, f' \leq 4F, \\ 1 \leq g, g' \leq 7 \end{array} \right.$$

Clock advance with charge change

$$r_{18} \equiv [C_3]_2 \longrightarrow [C_4]_2^-$$

Infection inside the neighborhood (last step)

Following the same strategy as in the previous module (infection inside the family), we include now the corresponding renaming rules.

Renaming of susceptible (Y) and infected ($\widetilde{Y}$) individuals

$$r_{19,f,g} \equiv [Y_{f,g}]_2^- \longrightarrow [Z_{f,g}]_2 \left\{ \begin{array}{l} 1 \leq f \leq 4F, \\ 1 \leq g \leq 7 \end{array} \right.$$

$$r_{20} \equiv [\widetilde{Y}]_2^- \longrightarrow [\widetilde{Z}]_2$$

Clock advance with charge change

$$r_{21} \equiv [C_4]_2^- \longrightarrow [C_5]_2$$

Infection inside the community (first step)

This module of rules represents the probability of infection by interacting with an individual living in the same community. No conditions are imposed on the indexes of the symbols, if they are in the same membrane (within the same environment) then they can interact.

Rules to deal with the interaction between two infected people are included here as well (see $r_{23,f,g}$).

Evolution of susceptible (Z) and previously affected ($\widetilde{Z}$) individuals

$$r_{22,f,f',g,g'} \equiv [A_{f,g}\ Z_{f',g'}]_2 \longrightarrow [A_{f,g}\ \overline{X}_{f',g'}\ \widetilde{Z}]_2^- \left\{ \begin{smallmatrix} 1 \le f, f' \le 4F, \\ 1 \le g, g' \le 7 \end{smallmatrix} \right.$$

$$r_{23,f,g} \equiv [A_{f,g}\ \widetilde{Z}]_2 \longrightarrow [A_{f,g}\ \widetilde{Z}]_2^- \left\{ \begin{smallmatrix} 1 \le f \le 4F, \\ 1 \le g \le 7 \end{smallmatrix} \right.$$

Clock advance with charge change

$$r_{24} \equiv [C_5]_2 \longrightarrow [C_6]_2^-$$

Infection inside the community (last step)

Following the same strategy as in the previous modules, we include now the corresponding renaming rules.

Renaming of susceptible (Z) and infected ($\widetilde{Z}$) individuals

$$r_{25,f,g} \equiv [Z_{f,g}]_2^- \longrightarrow [V_{f,g}]_2 \left\{ \begin{smallmatrix} 1 \le f \le 4F, \\ 1 \le g \le 7 \end{smallmatrix} \right.$$

$$r_{26} \equiv [\widetilde{Z}]_2^- \longrightarrow [\widetilde{V}]_2$$

Clock advance with charge change

$$r_{27} \equiv [C_6]_2^- \longrightarrow [C_7]_2$$

Infection in workplace/daycare/school...

In this part of the day cycle, we consider the interactions which take place at work. More precisely, the individuals can now be infected by interacting with other individuals of the same age group (since they will both be studying at school, or both working, etc.).

The renaming step is not needed in this module. By giving a positive charge to membrane 2 (in the previous modules only 0 and $-$ are used), the module for travelling among communities is initiated in the next step.

An asymptomatic individual may meet susceptible ones ($V_{k,g}$), or another infected individual ($\widetilde{V}$)

$$r_{28,f,f',g} \equiv [A_{f,g}\ V_{f',g}]_2 \longrightarrow [A_{f,g}\ \overline{X}_{f',g}\ \widetilde{V}]_2^+ \left\{ \begin{array}{l} 1 \le f, f' \le 4F, \\ 1 \le g \le 7 \end{array} \right.$$

$$r_{29,f,g} \equiv [A_{f,g}\ \widetilde{V}]_2 \longrightarrow [A_{f,g}\ \widetilde{V}]_2^+ \left\{ \begin{array}{l} 1 \le f \le 4F, \\ 1 \le g \le 7 \end{array} \right.$$

Clock advance with charge change

$$r_{30} \equiv [C_7]_2 \longrightarrow [C_8]_2^+$$

Infection among communities

This is the last possibility included in our model for spreading the disease. It considers the case of asymptomatic individuals traveling outside their communities.

In order to represent such "travels", objects are sent out of membrane 2, and then out of membrane 1 into their environment. Then, communication rules are applied, possibly moving to a different environment, and after that the objects representing the travellers move into membrane 1 and into membrane 2, and then they are ready for the infection rules (see $r_{52,f,g,j}$). Rule r_{53} has a double role: on one hand they are equivalent to rules for interaction between two infected people used in previous modules, but on the other hand, together with rules r_{55} and r_{56} they also take care of eliminating objects involved in infection rules, so that they will not interfere in the development of the next cycle.

Movement to skin membrane

$$r_{31,f,g} \equiv [V_{f,g}]_2^+ \longrightarrow V_{f,g}[\ \ ]_2 \left\{ \begin{array}{l} 1 \le f \le 4F, \\ 1 \le g \le 7 \end{array} \right.$$

$$r_{32} \equiv [\widetilde{V}]_2^+ \longrightarrow \widetilde{V}[\ \ ]_2$$

$$r_{33,f,g} \equiv [A_{f,g}]_2^+ \longrightarrow A_{f,g}[\ \ ]_2 \left\{ \begin{array}{l} 1 \le f \le 4F, \\ 1 \le g \le 7 \end{array} \right.$$

$$r_{34,f,g} \equiv [\overline{X}_{f,g}]_2^+ \longrightarrow \overline{X}_{f,g}[\ \ ]_2 \left\{ \begin{array}{l} 1 \le f \le 4F, \\ 1 \le g \le 7 \end{array} \right.$$

$$r_{35,f,g,i} \equiv [\overline{A}_{f,g,i}]_2^+ \longrightarrow \overline{A}_{f,g,i}[\ \ ]_2 \left\{ \begin{array}{l} 1 \le f \le 4F, \\ 1 \le g \le 7, \\ 2 \le i \le 5 \end{array} \right.$$

$$r_{36,f,g,i} \equiv [\overline{S}_{f,g,i}]_2^+ \longrightarrow \overline{S}_{f,g,i}[\ \ ]_2 \left\{ \begin{array}{l} 1 \le f \le 4F, \\ 1 \le g \le 7, \\ 2 \le i \le 5 \end{array} \right.$$

$$r_{37,f,g} \equiv [R_{f,g}]_2^+ \longrightarrow R_{f,g}[\ \ ]_2 \left\{ \begin{array}{l} 1 \le f \le 4F, \\ 1 \le g \le 7 \end{array} \right.$$

Clock advance with charge change

$$r_{38} \equiv [C_8]_2^+ \longrightarrow [C_9]_2$$

Movement to the environment

$$r_{39,f,g,j} \equiv [A_{f,g}]_1 \xrightarrow{1/3} A'_{f,g,j}[\quad]_1 \begin{cases} 1 \le f \le 4F, \\ 1 \le g \le 7, \\ 1 \le j \le 3 \end{cases}$$

$$r_{40,f,g,j} \equiv [V_{f,g}]_1 \xrightarrow{1/3} V'_{f,g,j}[\quad]_1 \begin{cases} 1 \le f \le 4F, \\ 1 \le g \le 7, \\ 1 \le j \le 3 \end{cases}$$

$$r_{41,j} \equiv [\widetilde{V}]_1 \xrightarrow{1/3} \widetilde{V}'_j[\quad]_1 \; 1 \le j \le 3$$

Clock advance

$$r_{42} \equiv [C_9 \longrightarrow C_{10}]_2$$

Communication rules among environments

$$r_{e_1,f,g,j,i} \equiv (A'_{f,g,i})_{e_j} \longrightarrow (\widehat{A})_{e_i} \begin{cases} 1 \le f \le 4F, \\ 1 \le g \le 7, \\ 1 \le i, j \le 3 \end{cases}$$

$$r_{e_2,f,g,j,i} \equiv (V'_{f,g,i})_{e_j} \longrightarrow (W_{f,g,j})_{e_i} \begin{cases} 1 \le f \le 4F, \\ 1 \le g \le 7, \\ 1 \le i, j \le 3 \end{cases}$$

$$r_{e_3,j,i} \equiv (\widetilde{V}'_i)_{e_j} \longrightarrow (\widetilde{W})_{e_i} \; 1 \le i, j \le 3$$

Clock advance

$$r_{43} \equiv [C_{10} \longrightarrow C_{11}]_2$$

Input into membrane 1

$$r_{44} \equiv \widehat{A}[\quad]_1 \longrightarrow [\widehat{A}]_1$$

$$r_{45,f,g,j} \equiv W_{f,g,j}[\quad]_1 \longrightarrow [W_{f,g,j}]_1 \begin{cases} 1 \le f \le 4F, \\ 1 \le g \le 7, \\ 1 \le j \le 3 \end{cases}$$

$$r_{46} \equiv \widetilde{W}[\quad]_1 \longrightarrow [\widetilde{W}]_1$$

Clock advance

$$r_{47} \equiv [C_{11} \longrightarrow C_{12}]_2$$

Input into membrane 2

$$r_{48} \equiv \widehat{A}[\quad]_2 \longrightarrow [\widehat{A}]_2$$

$$r_{49,f,g,j} \equiv W_{f,g,j}[\quad]_2 \longrightarrow [W_{f,g,j}]_2 \begin{cases} 1 \le f \le 4F, \\ 1 \le g \le 7, \\ 1 \le j \le 3 \end{cases}$$

$$r_{50} \equiv \widetilde{W}[\quad]_2 \longrightarrow [\widetilde{W}]_2$$

Clock advance

$$r_{51} \equiv [C_{12} \longrightarrow C_{13}]_2$$

An asymptomatic traveller may meet a susceptible individual ($W_{f,g,j}$), or another infected individual ($\widetilde{W}$)

$$r_{52,f,g,j} \equiv [\widehat{A}\ W_{f,g,j}]_2 \longrightarrow [\overline{X}'_{f,g,j}]_2^- \begin{cases} 1 \le f \le 4F, \\ 1 \le g \le 7, \\ 1 \le j \le 3 \end{cases}$$

$$r_{53} \equiv [\widehat{A}\ \widetilde{W}]_2 \longrightarrow [\#]_2^-$$

Clock advance with charge change

$$r_{54} \equiv [C_{13}]_2 \longrightarrow [C_{14}]_2^-$$

Start the reverse trip to return home (movement to skin membrane)

$$r_{55} \equiv [\widehat{A}]_2^- \longrightarrow [\#]_2$$

$$r_{56} \equiv [\widetilde{W}]_2^- \longrightarrow [\#]_2$$

$$r_{57,f,g,j} \equiv [W_{f,g,j}]_2^- \longrightarrow W'_{f,g,j}[\quad]_2 \begin{cases} 1 \le f \le 4F, \\ 1 \le g \le 7, \\ 1 \le j \le 3 \end{cases}$$

$$r_{58,f,g,j} \equiv [\overline{X}'_{f,g,j}]_2^- \longrightarrow \overline{X}'_{f,g,j}[\quad]_2 \begin{cases} 1 \le f \le 4F, \\ 1 \le g \le 7, \\ 1 \le j \le 3 \end{cases}$$

Clock advance with charge change

$$r_{59} \equiv [C_{14}]_2^- \longrightarrow [C_{15}]_2$$

Next step of the trip back (movement to the environment)

$$r_{60,f,g,j} \equiv [W'_{f,g,j}]_1 \longrightarrow X'_{f,g,j}[\quad]_1 \begin{cases} 1 \le f \le 4F, \\ 1 \le g \le 7, \\ 1 \le j \le 3 \end{cases}$$

$$r_{61,f,g,j} \equiv [\overline{X}'_{f,g,j}]_1 \longrightarrow \overline{X}'_{f,g,j}[\quad]_1 \begin{cases} 1 \le f \le 4F, \\ 1 \le g \le 7, \\ 1 \le j \le 3 \end{cases}$$

Clock advance

$$r_{62} \equiv [C_{15} \longrightarrow C_{16}]_2$$

Next step of the trip back (communication rules among environments)

$$r_{e_4,f,g,j,i} \equiv (X'_{f,g,j})_{e_i} \longrightarrow (X_{f,g})_{e_j} \begin{cases} 1 \le f \le 4F, \\ 1 \le g \le 7, \\ 1 \le j, i \le 3 \end{cases}$$

$$r_{e_5,f,g,j,i} \equiv (\overline{X}'_{f,g,j})_{e_i} \longrightarrow (\overline{X}_{f,g})_{e_j} \begin{cases} 1 \le f \le 4F, \\ 1 \le g \le 7, \\ 1 \le j, i \le 3 \end{cases}$$

Clock advance

$$r_{63} \equiv [C_{16} \longrightarrow C_{17}]_2$$

Next step of the trip back (going into membrane 1)

$$r_{64,f,g} \equiv X_{f,g}[\ \ ]_1 \longrightarrow [X_{f,g}]_1 \left\{ \begin{smallmatrix} 1 \le f \le 4F, \\ 1 \le g \le 7 \end{smallmatrix} \right.$$

$$r_{65,f,g} \equiv \overline{X}_{f,g}[\ \ ]_1 \longrightarrow [\overline{X}_{f,g}]_1 \left\{ \begin{smallmatrix} 1 \le f \le 4F, \\ 1 \le g \le 7 \end{smallmatrix} \right.$$

Clock advance with charge change

$$r_{66} \equiv [C_{17}]_2 \longrightarrow [C_{18}]_2^+$$

Restore configuration

The last step of the "trip back" is slightly modified to be used as a restoring mechanism, so that a new day cycle can start (if the maximum number of steps T has not yet been reached).

There are three possibilities for individuals who have been marked as infected during this cycle: either they are considered not actually infected, eventhough they have been exposed (see $r_{70,f,g}$); or they become infected and symptomatic (see $r_{68,f,g}$); or they become infected and asymptomatic (see $r_{69,f,g}$).

A third index is considered for symbols representing previously infected individuals. It represents the days since they became infected. We assume that after 5 days, asymptomatic individuals will change status to recovered. For symptomatic individuals, on the other hand, we have to take into account the probability of recovery, which is a parameter that may get different values for each age group.

Last step to complete the "one-day" cycle, renaming the symbols to start over again

$$r_{67,f,g} \equiv X_{f,g}[\ \ ]_2^+ \longrightarrow [X_{f,g}]_2 \left\{ \begin{smallmatrix} 1 \le f \le 4F, \\ 1 \le g \le 7 \end{smallmatrix} \right.$$

$$r_{68,f,g} \equiv \overline{X}_{f,g}[\ \ ]_2^+ \xrightarrow{ps \cdot pg} [\overline{S}_{f,g,2}\ S_{f,g}^{20}]_2 \left\{ \begin{smallmatrix} 1 \le f \le 4F, \\ 1 \le g \le 7 \end{smallmatrix} \right.$$

$$r_{69,f,g} \equiv \overline{X}_{f,g}[\ \ ]_2^+ \xrightarrow{pg \cdot (1-ps)} [\overline{A}_{f,g,2}\ A_{f,g}^{20}\ \widetilde{X}]_2 \left\{ \begin{smallmatrix} 1 \le f \le 4F, \\ 1 \le g \le 7 \end{smallmatrix} \right.$$

$$r_{70,f,g} \equiv \overline{X}_{f,g}[\ \ ]_2^+ \xrightarrow{1-pg} [X_{f,g}]_2 \left\{ \begin{smallmatrix} 1 \le f \le 4F, \\ 1 \le g \le 7 \end{smallmatrix} \right.$$

$$r_{71,f,g,i} \equiv \overline{S}_{f,g,i}[\ \ ]_2^+ \longrightarrow [\overline{S}_{f,g,i+1}\ S_{f,g}^{20}]_2 \left\{ \begin{smallmatrix} 1 \le f \le 4F, \\ 1 \le g \le 7, \\ 2 \le i \le 4 \end{smallmatrix} \right.$$

$$r_{72,f,g,i} \equiv \overline{A}_{f,g,i}[\ \ ]_2^+ \longrightarrow [\overline{A}_{f,g,i+1}\ A_{f,g}^{20}\ \widetilde{X}]_2 \left\{ \begin{smallmatrix} 1 \le f \le 4F, \\ 1 \le g \le 7, \\ 2 \le i \le 4 \end{smallmatrix} \right.$$

$$r_{73,f,g} \equiv \overline{A}_{f,g,5}[\ \ ]_2^+ \longrightarrow [R_{f,g}\ \widetilde{X}]_2 \left\{ \begin{smallmatrix} 1 \le f \le 4F, \\ 1 \le g \le 7 \end{smallmatrix} \right.$$

$$r_{74,f,g} \equiv \overline{S}_{f,g,5}[\ \ ]_2^+ \xrightarrow{prg} [R_{f,g}\ \widetilde{X}]_2 \left\{ \begin{smallmatrix} 1 \le f \le 4F, \\ 1 \le g \le 7 \end{smallmatrix} \right.$$

$$r_{75,f,g} \equiv \overline{S}_{f,g,5}[\ \ ]_2^+ \xrightarrow{1-prg} [\#]_2 \left\{ \begin{smallmatrix} 1 \le f \le 4F, \\ 1 \le g \le 7 \end{smallmatrix} \right.$$

$$r_{76,f,g} \equiv R_{f,g}[\ \ ]_2^+ \longrightarrow [R_{f,g}\ \widetilde{X}]_2 \left\{ \begin{smallmatrix} 1 \le f \le 4F, \\ 1 \le g \le 7 \end{smallmatrix} \right.$$

Clock reset with charge change

$$r_{77} \equiv [C_{18}]_2^+ \longrightarrow [C_1]_2$$

The constants associated with the rules have the following meaning:

- $q_{f,g,j}$: Number of individuals in family f, in age group g inside community j.
- $l_{n,j}$: Infected individuals in neighbourhood n of community j.
- p_g: Probability for a person in age group g in contact with the virus to become infected.
- pr_g: Probability for an infected person in age group g to recover.
- ps: Probability for an infected person to be symptomatic.

The proposed model consists of seven modules of rules. The first module (infection of people) will only be executed when the model is initialized. The six remaining modules will be executed in a loop. Each cycle of the loop is interpreted as one day in the scenario.

In this chapter, a SIR model has been presented. In this model, the population is structured in age groups. Moreover, different contact spaces have been defined. The basic SIR model groups all individuals who are in a common location in a single age group. The features of PDP models permits the introduction of features for a more accurate reflection of reality in a straightforward manner.

4.6.2 Results

In the previous section, a SIR computational model based on Population Dynamics P systems has been presented. In order to evaluate the accuracy of the model with respect to the phenomenon under study, a validation process has been performed. The inherent randomness in complex systems like the one presented makes it infeasible the formal validation of models that attempt to reproduce their behaviour. It is therefore necessary an experimental validation by comparison of results generated by simulation tools with experimental data obtained directly from the real system. For this purpose, several software simulations have been performed, making use of

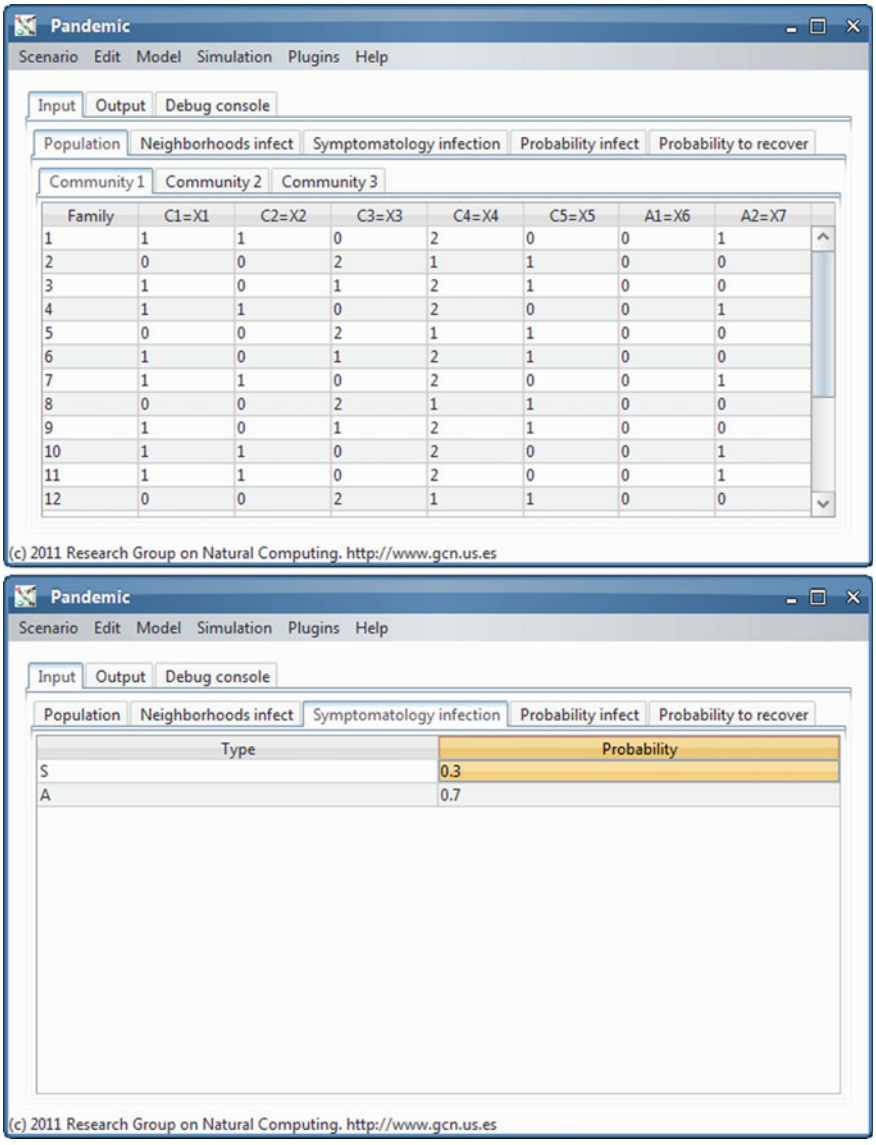

Fig. 4.5 MeCoSim Pandemic custom app—Input parameters tabs

the tools presented in Sect. 4.5; that is, P-Lingua and the pLinguaCore library [21], providing a standard language to define P systems and a Java library to manage P-Lingua files and simulate P system computations, and MeCoSim [25], providing a visual environment to perform the simulations.

An application for SIR (called Pandemic) has been supplied with MeCoSim by customization. Thus, by simply defining a configuration file, a visual GUI has been provided, adapted to the parameters required for the presented model. The interested reader can find in [36] the MeCoSim application files which define the model and instructions to reproduce the experiments.

A number of virtual experiments has been performed by providing the general model for SIR in P-Lingua format, and then introducing the appropriate values for the data corresponding to different scenarios in the input tables of the MeCoSim window (see Fig. 4.5). The process for each given scenario is as follows: the input data are introduced, the corresponding parameters $q_{f,g,j}$, $l_{n,j}$, p_g, pr_g and ps are generated, and then the computation is performed with Simulate! option, which calls DCBA-based simulation engine in pLinguaCore library. The simulation results have been obtained in the form of output tables and charts, as shown in Fig. 4.6.

A first scenario was simulated for the presented model. The detailed description of the scenario is what follows. The number of communities and families has been given as input parameters. These parameters have been obtained from [37]. The example depicted in Fig. 4.2 consists of 3 communities and 20 families for each community. At the initial stage, there are six infected people both in communities 1 and 3. In contrast, there are no infected people in community 2. The probability for a susceptible person

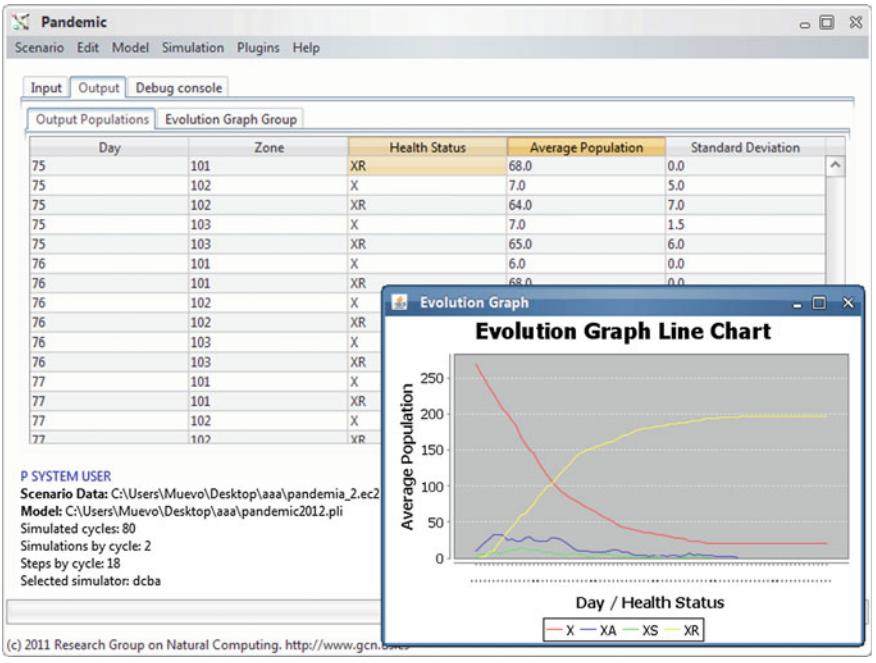

Fig. 4.6 Pandemic—Output table and chart. Number of healthy, asymptomatic, symptomatic and recovered individuals by zone

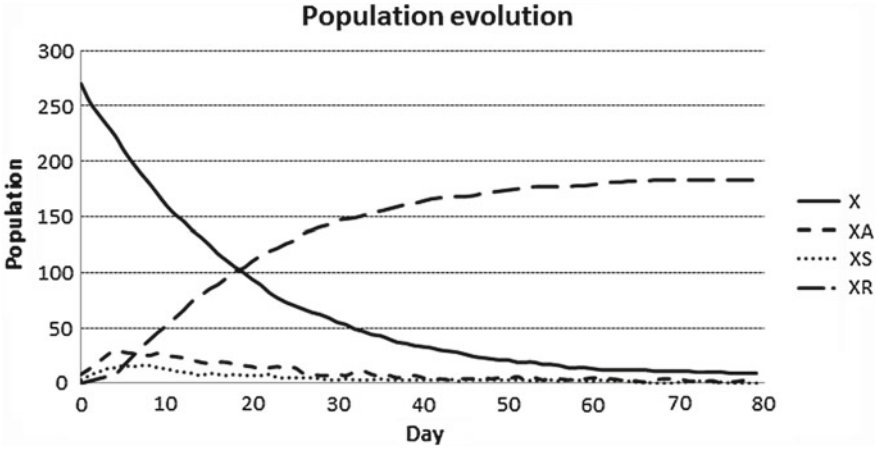

Fig. 4.7 Evolution of individuals—Original scenario

in contact with an infected one to become infected is 5 %. There is a 30 % probability for an infected person to manifest symptoms, whereas in the rest of the cases the symptoms are not manifested. Those who manifest symptoms are solely breeding grounds inside their family. On the other hand, those who do not manifest symptoms might infect people in other families and communities. There is a 5 % probability for a symptomatic person to recover. In contrast, asymptomatic people always recover. There exist no observable differences in the pandemic dynamics among the three communities due to the quick spreading of the disease. The designed simulator supports tuning of the parameters in the model. This feature provides a friendly way to study the behaviour of the disease under different scenarios. The presented model

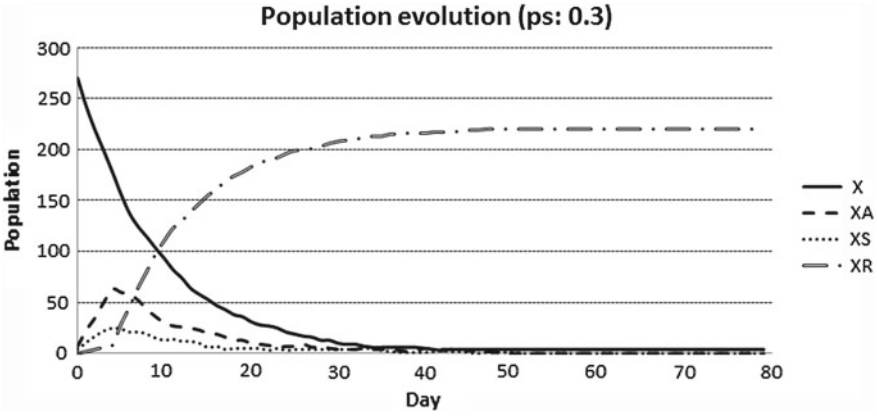

Fig. 4.8 Evolution of individuals—First alternative case

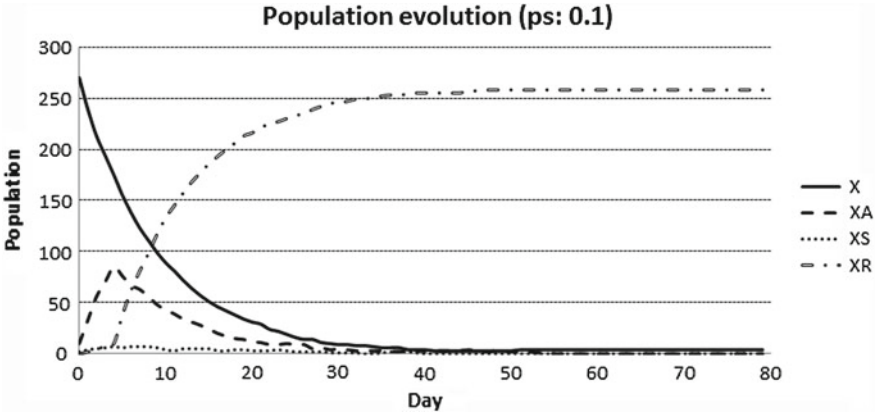

Fig. 4.9 Evolution of individuals—Second alternative case

does not take into account human interventions, such as vaccination campaigns, prophylactic actions, etc. These kind of measures are usually undertaken in the event of a pandemic outbreak. The results for this first scenario are shown in Fig. 4.7.

A number of other virtual experiments have been performed. Two of them are shown below. Both of them have similar input parameters; in particular, the probability for an individual to become infected is 10 %, and the probability of recovery is 30 % (the same for each age group). The only difference is the probability for an infected individual to be symptomatic. This probability is 30 % in the **first case** (see Fig. 4.8), whereas it is 10 % in the **second case** (see Fig. 4.9). As it can be seen, the second scenario presents a bigger number of recovered individuals, given the fact that asymptomatic individuals always recover.

Although the experiments carried out in this chapter refer to a virtual population, the results obtained by our simulations match the tendencies reported in the literature using classical SIR models [37–39].

4.7 Conclusions and Perspectives

Population Dynamics P systems (PDP systems) provide a new formal bio–inspired modelling framework. This is a novel and expressive approach that overcomes some limitations of classical mathematical models while keeping the most important features of the studied phenomena.

We illustrate the modelling/simulating workflow by means of a SIR computational model. First of all comes the design of the model, capturing complex social networks and interactions between individuals by means of the hierarchical structure of membranes (and the graph of connections between environments) along with their associated rules.

The model is then described in P–Lingua and given as input to the pLinguaCore library, which parses the model description and checks for errors. It is worth noting that such a description does not include data related to specific scenarios, only the initial structure and the rules. In order to input scenario–specific data related to a virtual experiment, MeCoSim generates a custom GUI. This GUI defines data fields specifically related to the problem at hand for the user to input the data.

In order to simulate the scenario, an algorithm capturing the semantics of PDP systems is required (e.g. MeCoSim calls pLinguaCore to perform this task relying on its built-in simulators). In this chapter we have explained the Direct distribution based on Consistent Blocks Algorithm (DCBA), which performs a proportional distribution of objects among rules in accordance to their associated probabilities. The algorithm grants a fair distribution in the case of competition among rules (i.e. rules having overlapping left-hand sides). Finally, simulation results are displayed by means of data tables and charts.

One of the drawbacks of sequential simulators is the time they spend to simulate considerably large instances. Nevertheless, the parallel structure of DCBA algorithm (and of PDP systems) appoints it suitable for its implementation on parallel hardware architectures, such as computer grids and graphic cards. In this sense, we have introduced an existing CUDA-based simulator which takes advantage of the computational parallel power of GPU computing in order to accelerate PDP systems simulations.

As regards to perspectives, the ultimate goal of these models is to serve as automatic assistants on management decision taking. That is, to give information about the effects of plausible measures by simulating presumed scenarios derived from undertaking these measures. In this sense, it is essential that future versions of the model consider and assess the effects of human measures. They will also need to analyse and foresee future trends on the studied populations. This goal calls for tight collaboration between experts and model designers, so as to define virtual experiments leading to feasible hypothesis to be verified by means of field work and live experiments.

When it comes to software development, the main work lines have to do with the improvement of the existing simulators, to reach higher performance and match even better the experimental results. As a short-term perspective, it is required to defined and implement efficient communication protocols to connect the mentioned parallel simulators from PMCGPU with the framework pLinguaCore. The addition of new functionalities to the interfaces is another important task which concerns software development. As a result, future versions of MeCoSim should permit a more exhaustive analysis on the results and augment the degree of automation of the design and simulation workflow from an end-user perspective.

Acknowledgments The authors acknowledge the support of "Proyecto de Excelencia con Investigador de Reconocida Valía" of the "Junta de Andalucía" under grant P08-TIC04200, and the support of the project TIN2012-37434 of the "Ministerio de Economía y Competitividad" of Spain, both co-financed by FEDER funds.

References

1. P.J.E. Goss, J. Peccoud, Quantitative modelling of stochastic system in molecular biology by using stochastic Petri nets. Proc. Nat. Acad. Sci. U.S.A. **95**, 6750–6755 (1998)
2. A. Regev, E. Shapiro, The π-calculus as an abstraction for biomolecular systems, in *Modelling in Molecular Biology*, ed. by G. Ciobanu, G. Rozenberg (Springer, Berlin, 2004), pp. 219–266
3. A. Regev, E.M. Panina, W. Silvermann, L. Cardelli, E. Shapiro, BioAmbients: an abstraction for biological compartments. Theoret. Comput. Sci. **325**, 141–167 (2004)
4. L. Cardelli, Brane calculi: interactions of biological membranes. Lect. Notes Bioinf. **3082**, 257–278 (2005)
5. V. Danos, C. Laneve, Formal molecular biology. Theoret. Comput. Sci. **325**(1), 69–110 (2004)
6. D. Harel, Statecharts: a visual formalism for complex systems. Sci. Comput. Program. **8**(3), 231–274 (1987)
7. M. Holcombe, M. Gheorghe, N. Talbot, A hybrid machine model of rice blast fungus, magnaphorte grisea. BioSystems **68**(2–3), 223–228 (2003)
8. M.L. Shaffer, Determining minimum viable population sizes for the grizzly bear, in *Proceedings International Conference on Bear Research and Management* vol. 5, pp. 133–139 (1983)
9. M.E. Soulé (ed.), *Viable Populations for Conservation* (Cambridge University Press, Cambridge, 1987)
10. Gh. Păun, Computing with membranes, J. Comput. Syst. Sci. **61**(1), 108–143 (2000) and Turku Center for Computer Science-TUCS, Report No 208
11. M. Cardona, M.A. Colomer, M.J. Pérez-Jiménez, D. Sanuy, A. Margalida, Modeling ecosystem using P systems: the bearded vulture, a case study. Lect. Notes Comput. Sci. **5391**, 137–156 (2009)
12. 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. Ecol. Model. **222**(1), 33–47 (2011)
13. 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. Nat. Comput. **10**(1), 39–53 (2011)
14. M.A. Colomer, A. Montori, I. Gaspa, C. Fondevilla. A computational model to study the dynamics of Pyrenean Newt (Calotriton asper), in *Twelfth International Conference on Membrane, Computing (CMC12)*, ed. by M. Gheorghe, Gh. Păun, S. Verlan, pp. 485–496 (2011)
15. Gh. Păun, *Membrane Computing: An Introduction* (Springer-Verlag, Berlin, 2002)
16. 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, in *IEEE Fifth International Conference on Bio-inpired Computing: Theories and Applications (BIC-TA 2010)*, vol. 1, ed. by K. Li, Z. Tang, R. Li, A.K. Nagar, R. Thamburaj, pp. 616–621 (2010)
17. S.E. Jørgensen, *Ecological Modelling: An introduction* (WIT press, Boston, 2009)
18. 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 objects distribution. Lect. Notes Comput. Sci. **7762**, 257–276 (2013)
19. 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. Int. J. Found. Comput. Sci. **22**(1), 107–118 (2011)
20. D. Díaz-Pernil, C. Graciani, M.A. Gutiérrez-Naranjo, I. Pérez-Hurtado, M.J. Pérez-Jiménez. Software for P systems, in *The Oxford Handbook of Membrane Computing*, Chapter 17, ed. by Gh. Păun, G. Rozenberg, A. Salomaa, (Oxford University Press, Oxford, 2009), pp. 437–454
21. I. Pérez-Hurtado. Desarrollo y aplicaciones de un entorno de programación para computación celular: P-Lingua. Ph.D. Thesis, University of Seville, 2010
22. The P-Lingua website. http://www.p-lingua.org
23. GNU Public License. http://www.gnu.org/licenses/gpl.html

24. M. Cardona, M.A. Colomer, A. Margalida, I. Pérez-Hurtado, M.J. Pérez-Jiménez, D. Sanuy, A P system based model of an ecosystem of some scavenger birds. Lect. Notes Comput. Sci. **5957**, 182–195 (2010)
25. 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, in *IEEE Fifth International Conference on Bio-inspired Computing: Theories and Applications (BIC-TA 2010)*, vol. I, ed. by K. Li, Z. Tang, R. Li, A.K. Nagar, R. Thamburaj, pp. 637–643 (2010)
26. MeCoSim website. http://www.p-lingua.org/mecosim
27. M.A. Martínez-del-Amor, I. Karlin, R.E. Jensen, M.J. Pérez-Jiménez, A.C. Elster. Parallel simulation of probabilistic P systems on multicore platforms, in *Tenth Brainstorming Week on Membrane Computing (BWMC 2012)*, vol. II, ed. by M. García, L.F. Macías, Gh. Păun, L. Valencia, pp. 17–26 (2012)
28. M. Harris, *Mapping Computational Concepts to GPUs* (ACM SIGGRAPH 2005 Courses, NY, 2005)
29. F. Cabarle, H. Adorna, M.A. Martínez-del-Amor, M.J. Pérez-Jiménez, Improving GPU simulations of Spiking Neural P systems. Rom. J. Inform. Sci. Technol. **15**(1), 5–20 (2012)
30. J.M. Cecilia, J.M. García, G.D. Guerrero, M.A. Martínez-del-Amor, I. Pérez-Hurtado, M.J. Pérez-Jiménez, Simulation of P systems with active membranes on CUDA. Briefings Bioinf. **11**(3), 313–322 (2010)
31. J.M. Cecilia, J.M. García, G.D. Guerrero, M.A. Martínez-del-Amor, I. Pérez-Hurtado, M.J. Pérez-Jiménez, Simulating a P system based efficient solution to SAT by using GPUs. J. Logic Algebraic Program. **79**(6), 317–325 (2010)
32. D. Kirk, W. Hwu, *Programming Massively Parallel Processors: A Hands On Approach* (Morgan Kauffman, Boston, 2010)
33. Ø. Krog, A.C. Elster, Fast GPU-based fluid simulations using SPH. Lect. Notes Comput. Sci. **7134**, 98–108 (2012)
34. M.A. Martínez-del-Amor, I. Pérez-Hurtado, A. Gastalver-Rubio, A.C. Elster, M.J. Pérez-Jiménez, population dynamics P systems on CUDA. Lect. Notes Bioinf. **7605**, 247–266 (2012)
35. The PMCGPU project website. http://sourceforge.net/p/pmcgpu
36. The Pandemic model in MeCoSim. http://www.p-lingua.org/mecosim/doc/case_studies/pandemics.html
37. H. Yasuda, K. Suzuki, Measures against transmission of pandemic H1N1 influenza in Japan in 2009: simulation model. Euro Surveill **14**, 44 (2009), pii=19385
38. H.W. Hethcote, Three basic epidemiological models. Appl. Math. Ecol. **18**, 119–144 (1989)
39. M. Qiao, A. Liu, U. Foryś, Qualitative analysis of the SICR epidemic model with impulsive vaccinations. Math. Methods Appl. Sci. **36**(6), 695–706 (2013)