Noname manuscript No.
(will be inserted by the editor)

Rule-Based Programming for Integrative Biological Modeling

Application to the Modeling of the λ Phage Genetic Switch

Olivier Michel · Antoine Spicher · Jean-Louis Giavitto

Received: date / Accepted: date

Abstract Systems biology aims at integrating processes at various time and spatial scales into a single and coherent formal description to allow computer modeling. In this context, we focus on rule-based modeling and its integration in the domain-specific language MGS. Through the notions of *topological collections* and *transformations*, MGS allows the modeling of biological processes at various levels of description. We validate our approach through the description of various models of the genetic switch of the λ phage, from a very simple biochemical description of the process to an individual-based model on a Delaunay graph topology. This approach is a first step into providing the requirements for the emerging field of *spatial systems biology* which integrates spatial properties into systems biology.

Keywords Domain-Specific Languages - Rule-Based Modeling - Spatial Systems Biology

1 Introduction

As natural living systems exhibit complex and highly structured behaviors at various scales (in time and in space), they have inspired many *unconventional* computation models. Among them, we can name *cellular automata* designed by S. Ulam and J. Von Neumann (1; 2) for the study of crystal growth and self-replicating systems, the

This work is partially funded by the BQR of the University of Évry on "Synthetic Biology".

Olivier Michel *&* Antoine Spicher
LACL - EA 4213 - Université de Paris 12
Faculté des Sciences et Technologie
61 avenue du Général de Gaulle
94010 Créteil Cedex - France
E-mail: {olivier.michel,antoine.spicher}@univ-paris12.fr

Jean-Louis Giavitto
IBISC Lab. - FRE 3190 CNRS - Université d'Évry *&* Genopole
523 place des terrasses de l'agora
91000 Évry - France
E-mail: giavitto@ibisc.univ-evry.fr

CHAM (3) formalism and the Γ language (4) which take their origins in chemistry, *Păun's systems* (5) (or P-systems) corresponding to the metaphor of reactions taking place inside cellular compartments, or *Lindenmayer's systems* (6) (or L-systems) inspired by the growth process of plants development.

All these models share the common property that they allow the specification of systems in a *discrete* and *local* way:

– **Discrete.** In spite of their strength, continuous formalisms make the expression of the discrete nature of the biological systems very difficult (if not very inaccurate, when the number of entities in a given volume is small or heterogeneously distributed). Furthermore, a discrete approach allows to adopt an algebraic point of view for the description of the systems that fits well with the language theory domain used for the different models above.
– **Local.** The high complexity of biological systems makes it hard to describe their behavior at the level of the whole system. A common solution to that problem is to decompose it into subsystems exhibiting local interactions. The behavior of the system is then recovered as the *integration* of the local laws.

In all these models, computations are specified by a set of *local rules* that describes how *neighboring* data may *interact* to build some new data. Their specificities lie in how the neighborhood relationship is defined: abstract chemistry with (multi-)sets, P systems with a spatial organization formalized as Venn diagrams without intersection (corresponding to nested compartments), cellular automata with a regular neighborhood, L-systems with tree-like structures, etc. In other words, these models may be seen as specific forms of *local* application of *rewriting rules* on different kinds of *spatially organized data* (7).

In the following paragraphs, we detail how a rewriting system can be used to model some natural system, starting from their usual point of view in computer science to their use for natural systems.

1.1 Rewriting and Computation

A *rewriting system* (8) is a formal system used as a mean to compute by substituting a part of an *entity* by another. A rewriting system is defined by a set of *rules* R. Each rule $R_i \in R$:

$$\alpha \to \beta$$

is composed of a *left-hand side* α (lhs for short), specifying the part to be substituted, and a *right-hand side* β (rhs for short), specifying the new part replacing α.

Computing by local substitutions is common in computer science: it has been developed in a lot of contexts and applied on various kinds of *entities*. For instance in *multiset rewriting*, the lhs is a submultiset which is replaced by the multiset corresponding to the rhs; in *term rewriting*, the lhs is a subtree which is replaced by the tree corresponding to the rhs; in *graph rewriting*, the lhs is a subgraph which is replaced by the graph corresponding to the rhs; etc.

Let the variables e, e', ... denote the considered entities. A rewriting rule defines a binary relationship between these entities. We write

$$e \Rightarrow_{R_i} e'$$

to denote the *local transformation* of e into e' by applying the rule R_i: e' is defined as e where *one of its subpart* α is replaced by β. If no α exists in e, the relation is not defined on e.

We write

$$e \Rightarrow_R e'$$

to denote that e is transformed into e' by applying the rules in R. We need here to be more specific about *which* rules are applied (and in what order). For this purpose, we write

$$e \Rightarrow_{R,\mathrm{St}} e'$$

to characterize the *rule application strategy* where St represents the considered policy. For instance, St may mean that only one rule R_i, chosen in a non-deterministic way, is applied; or that rules are applied all together on different subparts of e in a concurrent mode.

Considering a set of rules R together with a rule application strategy St, we define the relation $\Rightarrow^*_{R,\mathrm{St}}$ as the reflexive and transitive closure of $\Rightarrow_{R,\mathrm{St}}$. In other words, if e and e' are two entities, $e \Rightarrow^*_{R,\mathrm{St}} e'$ means that there exists a finite sequence of entities $e_1, \ldots, e_n$ such that

$$e = e_1,\, e_1 \Rightarrow_{R,\mathrm{St}} e_2, \ldots, e_{n-1} \Rightarrow_{R,\mathrm{St}} e_n,\, e_n = e'$$

This sequence is called a *derivation* of e w.r.t. the set of rules R and the strategy St. This transformation of e into e' can be seen as the result of the computation specified by R where the derivation corresponds to the sequence of the intermediate results.

1.2 Rule-Based Modeling of Natural Systems

The framework proposed by rewriting systems suits well the formal description of natural phenomena. As a matter of fact, many authors (9; 10; 11; 12) have recognized the importance of term rewriting in that field. The state of a biological system is represented by a term t of the form

$$t_1 + t_2 + \cdots + t_n$$

where a subterm t_i represents a biological entity composing the system or a message (signal, information, process, etc.) sent to an entity. Following this point of view, the dynamics of the biological system is captured by the rewriting steps of the term. For example, rules of the form

$$e + m \to e' + m'$$

can be used to model the reaction of the entity e when it receives the message m: the lhs corresponds to the message m and to the entity e which m is sent to. The $+$ operator denotes that e and m are somehow neighbor in the system. In the rhs, e' specifies the new state of the entity and a message m' may (possibly) be sent. Other kinds of local evolutions may be specified in the same way. The direct interaction between entities can be expressed by rules of the form

$$e_1 + e_2 \to e_1' + e_2'$$

and the creation of a new entity by a rule of the form

$$e \to e' + e''$$

etc. Intuitively, we can deduce that, given the state of a system as a term t, a derivation of t is understood as the description of the evolution of the system along time; we speak of the *trajectory* of the system.

Table 1 Dictionary establishing a link between the notions required for the modeling and simulation of natural systems with their counterparts in the field of rewriting systems.

	Natural System	**Rewriting System**
Model	*Spatial organization*	A data structure T $T \in \{\text{term}, \text{set}, \text{graph}, \dots\}$
	State	An element $e \in T$
	Local evolution laws	Set of rules R
	Interaction $\Longrightarrow$ result	A rule $\alpha \to \beta \in R$
Simulation	Local and atomic evolutions	Relation $\Rightarrow_{R_i}$ with $R_i \in R$
	Time steps	Relation $\Rightarrow_{R,\text{St}}$
	Trajectories	Relation $\Rightarrow^{*}_{R,\text{St}}$

More generally, the modeling and the simulation of a natural system using a rewriting system may be summarized by the analogy given in Table 1. In fact, choosing rewriting systems as a formalism to describe dynamical systems implies to consider the handling of discrete time and local space.

Local Specification of Space. For a particular natural system to be modeled, according to Table 1, it is required to choose a particular type T of data structure that describes its *local spatial organization*. The state of the system is given by a value of type T. Once the state is defined, a rule R_i must be defined, for each interacting subpart, to specify the set R of *local evolution laws*. Such rules can be understood as following:

– the lhs represents a subsystem whose elements are in interaction,
– the rhs corresponds to the computation of the local result of this interaction.

In the previous example, the $+$ operator connects entities and signals as they are spatially and/or functionally organized. It denotes at the same time the interacting subparts of the system as well as it specifies the way that the subsystems are organized. Terms built using arbitrary operators can be used to capture hierarchical organizations. However the $+$ operator is (usually) associative and commutative. This implies that the tree structure of an arbitrary term can be reorganized to order arbitrarily the leaves of the tree structure (the leaves of the tree are the constants of the expression). Using associative-commutative operators we obtain multiset. In a multiset, all elements are considered as being all neighbors which corresponds to the metaphor of a *well mixed* chemical solution where Brownian motion ensures that each element will encounter all the others.

Discrete Time. The handling of time is a key point in the modeling of a natural system. The model of time that is naturally induced by the use of rewriting systems, is a *discrete* time: the application of a rule corresponds to an atomic modification of the state of the system and to the progression of time.

As shown in Table 1, the trajectories of the systems in its phase space are clearly captured by the definition of the relations $\Rightarrow_{R,\mathrm{St}}$ and $\Rightarrow^{*}_{R,\mathrm{St}}$. Intuitively, they correspond to event-based simulations of the model as they represent the iterative applications of the local evolution laws specified within R with respect to a rule application strategy St.

1.3 The MGS project

The MGS project[1] proposes to investigate the previously referred computation models by developing a domain-specific programming language, also called MGS, where they may be expressed in a uniform and generic setting. In MGS, the state of a dynamical system is specified using an original and generic data structure, called *topological collection* (7), based on the topological relations between the interacting subparts of the system. Furthermore, the specification of the evolution law is simplified by the definition of a case-based function, called *transformation*, based on the rewriting of topological collections.

Topological Collection. One of the key features of the MGS language is its ability to describe and manipulate entities structured by an *abstract topology.* Topological collections provide a unified view of the notion of data structure (7) seen as an *aggregate of elements organized with a neighborhood relationship.* The structure of the defined topological space allows to specify the organization of the data structure.

The formalization of topological collections has been thoroughly studied in previous work of the authors (13; 14). Without getting too much into details, we give a few technical details on topological collections. Its formalization relies on the notion of *abstract cell complex* (15) defined in algebraic topology allowing the description of the topology of the collection. A cellular complex makes it possible to have a discrete representation of a topological space through a set of *topological cells* (an abstraction of elementary spaces characterized by their dimensions) connected to each other through their *incidence relationship.* This relation is based upon the notion of boundary. This structure is then decorated with values leading to the notion of *topological chain* (15). This final notion of topological chain allows the association of elements of an abelian group to cells of a complex resulting in a rich algebraic structure.

In the context of the modeling of natural systems, topological collections allow an intuitive representation of the states of the system: the elements of the collection are the atomic components, and the interaction graph between components specializes the topological structure of the collection.

Many topologies are available in MGS. In this article, we focus on multiset collections (corresponding to a complete neighborhood) in sections 3 and 4, GBF collections (corresponding to a regular topology) in section 5.1, and Delaunay collections (whose topology relies on the computation of a Delaunay triangulation) in section 5.2.

[1] The Web site of the project is `http://mgs.ibisc.univ-evry.fr`

Transformation. Topological collections represent an adequate medium to extend rewriting systems as previously presented. Indeed, the neighborhood relationship gives a local point of view of the organization of the elements. The *transformations* extend the notion of rewriting systems to topological collections, and are used to specify the evolution function of the modeled system as show on Table 1.

In MGS, the specification of a transformation T:

```
trans T = {
  σ => f(σ,...);
  ...
}
```

corresponds to the definition of a set of rules, where σ is a pattern, matching for a subcollection, and $f(\sigma, \dots)$ is an expression that evaluates a new subcollection that will be inserted in place of the matched one. The application of transformation T on a topological collection e using a strategy St, written `T[strategy = St](e)`, consists in computing a collection e' such that $e \Rightarrow_{\mathrm{T,St}} e'$.

In the current implementation of MGS, all available strategies are built-in. In the following, we will use two of the many rule application strategies provided by MGS: the Gillespie strategy based on the stochastic simulation algorithm proposed by Gillespie to simulate chemical reactions (in sections 3 and 4), and the maximal-parallel strategy (in section 5) widely used in the context of L-systems and P systems.

1.4 Motivations and Organization of the Article

MGS provides features allowing the use of rewriting techniques on a large number of different data structures. This point of view is very effective in the modeling of natural systems as it allows to use a different type of organization in the lhs and in the rhs of a rule. Consequently, rules may specify modifications of the system's structure. Thus, this allows the modeling of a class of dynamical systems particularly complex to describe and simulate, *dynamical systems with a dynamical structure* (16; 17; 18).

In this article, we claim that the generic approach of MGS allows an easy way to express models of biological systems. Previous publications by the authors were devoted to the formalization of the theoretical foundations of MGS, and to the use of MGS for the modeling and simulation of biological processes at a *specific level of description*. Here, we focus on the suitability of MGS for the description of the *same biological process*, from the molecular level to the level of spatially distributed populations. The article is organized as follows: in section 2, we describe the biological process of the regulation of the genetic switch of the λ phage. This process will be our running example throughout this article.

We propose in section 3 a first model of the switch based on a biochemical description of the regulation. For that purpose, we show how Gillespie's stochastic simulation algorithm can be seen a rewriting strategy in MGS. This first model only consider a single cell.

A first model of a population of cells is proposed in section 4. In this section, the population of cell is considered as being homogeneously distributed in space and therefore represented by a multiset topological collection in MGS.

We still consider a population of cells and complexify our model in section 5 by considering a population in an homogeneous and heterogeneous spatial distribution. For that purpose, we propose two models, one relying on a cellular automaton with a Von Neumann neighborhood and one with an individual-based model on a neighborhood defined by a Delaunay triangulation of the cells in space. Both models are naturally implemented in MGS.

Finally, we conclude in section 6 by emphasizing the importance of rule-based languages and formalisms in systems biology and we stress the importance of spatial relationships in biological processes.

2 The Paradigmatic Example of the λ Phage Switch

In the forthcoming sections, we propose to illustrate the expressiveness brought by rule-based programming for the description of various models of the same biological process: the regulation of the switch of the λ phage. We first start in this section by briefly describing the behavior of the phage.

2.1 The Genetic Switch

The λ phage (19) is a virus that infects the cells of the bacterium *Escherichia coli.* It is a phage that has two possible outcomes:

1. replication and *lytic* phase where the virus dissolves and destroys the host cell, releasing about 100 virions;
2. integration of its DNA in the DNA of the bacterium, and start of a *lysogenic* phase.

In the *lysogenic* phase, the virus will silently replicate at each cell division. Moreover, a lysogeny produces an immunity towards further phage infection, by protecting the bacterium from the destruction during a possible new infection by a phage. Under certain conditions (exposure to U.V. for example) a lytic phase can be *induced*: the viral DNA is released from the bacterial DNA and starts a normal replication and a lysis.

Based on the local conditions, it is one of the two possible phases that is chosen, the decision being under the control of a small region of the phage genome (a hundred base pairs, comparatively to the 48502 base pairs of the genome of the bacterium) and of two genes (cI and cro) and two promoters This regulatory region is called the *genetic switch.*

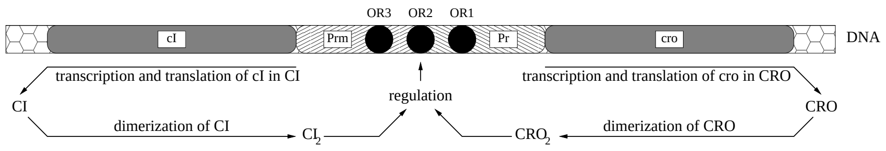

OR3
OR2
OR1
cI
Prm
Pr
cro
DNA
transcription and translation of cI in CI
regulation
transcription and translation of cro in CRO
CI
CRO
dimerization of CI
CI2
CRO2
dimerization of CRO

Fig. 1 Description of the regulatory region of the genetic switch.

2.2 Regulation of the Genetic Switch

The region of the DNA (see Figure 1) of the λ phage is composed of two genes cI and cro coding respectively for the proteins CI and CRO. During the transcription, the RNA polymerase binds to the promoters of these genes (Prm and Pr respectively) to synthesize the mRNA that is then translated into monomer proteins CI and CRO.

These monomers dimerize into CI_2 and CRO_2 which can bind to the operators. Operators OR1, OR2 and OR3 overlap the promoters (see Figure 1). The absence or presence of dimers bound to the promoters, eases or hinders the binding of the RNA polymerase, thus regulating the expression of genes cI and cro. Binding of the dimers to the promoters follows certain rules of affinity:

- CI_2 binds first to OR1, then to OR2 and finally to OR3;
- CRO_2 binds first to OR3, then to OR2 and finally to OR1;
- If CI_2 is bound to OR1, it facilitates the binding of another dimer CI_2 to OR2 and consequently the binding of the RNA polymerase on the Prm promoter. This property does not hold for CRO_2;
- The RNA polymerase can be bound to the promoter Pr but CI_2 has to be bound to OR2 so that it can be bound to Prm.

All these rules together lead to a complex behavior and the genetic regulation of the transcription.

Simply stated, each of the two proteins CI and CRO will inhibit each others and increase its own synthesis (they self-inhibit themselves when their expression becomes too high). According to the local conditions, one of the two proteins will take over the other one (in terms of concentration) and the system will enter either a lytic (CRO “wins”) or a lysogeny phase (CI “wins”).

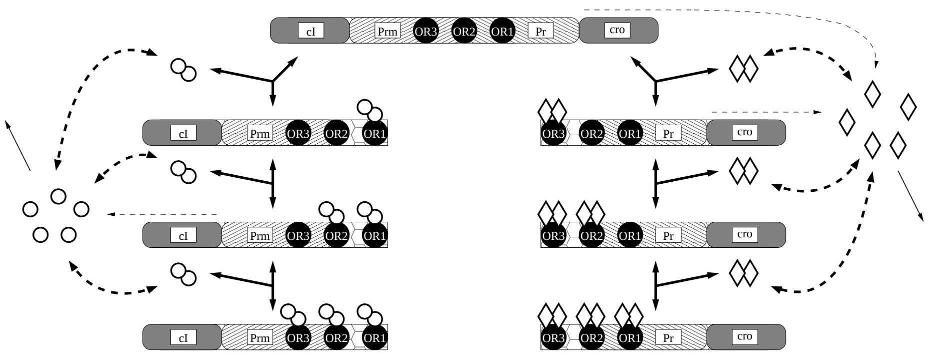


Fig. 2 Regulation network of the genetic switch of the λ phage. Monomers CI and CRO are described in circles and diamonds; dimers are composed by monomers. Thin dotted arrows represent transcription and translation; bold dotted arrows represent (de)polimerization; thin arrows represent degradation; bold arrows represent regulation. The drawing follows the description of the DNA given in Figure 1.

3 Biochemical Modeling of a Single Bacterium

Our first modeling of the biological process of the phage genetic switch, relies on a translation of the regulation principles into biochemical reactions. It is a straight translation of the previously described process where the spatial distribution of the biological entities is not taken into account.

3.1 Chemical Reactions Based Model

The stochastic simulation of biochemical systems becomes useful when the number of molecules and/or the time interval is very small. It is the case of the regulation of the λ phage where only around ten molecules CI_2 are enough to enter the lysogenic phase (20). Figure 2 describes the model used for the chemical modeling. It follows the description given above. For the sake of simplicity, only a subset of all the dimers-operator bond combinations is considered. It reduces from 27 possible states (3^3 corresponding to all the combinations of *no binding*, CI_2 and CRO_2 bound on each operator) to only 7 states of the DNA. Actually, we have focused on the subset of the bonds that are most-likely to happen. CI_2 binding on OR2, with OR1 free, could have been taken into account; but this state is rare. We also consider that cI is expressed when OR2 is occupied by CI_2 and OR3 is free. The cro gene is expressed when both OR1 and OR2 are free.

The set of interactions (dimerization, binding of proteins on the DNA and gene expression) are treated as the following biochemical equations:

$$\mathrm{CI} \xrightarrow{C_0} . \tag{1}$$

$$2\,\mathrm{CI} \underset{C_{21}}{\overset{C_{12}}{\rightleftharpoons}} \mathrm{CI}_2 \tag{2}$$

$$\mathrm{D} + \mathrm{CI}_2 \underset{C_{-1}}{\overset{C_1}{\rightleftharpoons}} \mathrm{D}_1 \tag{3}$$

$$\mathrm{D}_1 + \mathrm{CI}_2 \underset{C_{-2}}{\overset{C_2}{\rightleftharpoons}} \mathrm{D}_2 \tag{4}$$

$$\mathrm{D}_2 + \mathrm{CI}_2 \underset{C_{-3}}{\overset{C_3}{\rightleftharpoons}} \mathrm{D}_3 \tag{5}$$

$$\mathrm{D}_2 + \mathrm{P} \xrightarrow{C_t} \mathrm{D}_2 + \mathrm{P} + \mathrm{CI} \tag{6}$$

$$\mathrm{CRO} \xrightarrow{C'_0} . \tag{7}$$

$$2\,\mathrm{CRO} \underset{C'_{21}}{\overset{C'_{12}}{\rightleftharpoons}} \mathrm{CRO}_2 \tag{8}$$

$$\mathrm{D} + \mathrm{CRO}_2 \underset{C'_{-3}}{\overset{C'_3}{\rightleftharpoons}} \mathrm{D}'_3 \tag{9}$$

$$\mathrm{D}'_3 + \mathrm{CRO}_2 \underset{C'_{-2}}{\overset{C'_2}{\rightleftharpoons}} \mathrm{D}'_2 \tag{10}$$

$$\mathrm{D}'_2 + \mathrm{CRO}_2 \underset{C'_{-1}}{\overset{C'_1}{\rightleftharpoons}} \mathrm{D}'_1 \tag{11}$$

$$\mathrm{D} + \mathrm{P} \xrightarrow{C'_t} \mathrm{D}_2 + \mathrm{P} + \mathrm{CRO} \tag{12}$$

$$\mathrm{D}'_3 + \mathrm{P} \xrightarrow{C'_t} \mathrm{D}_2 + \mathrm{P} + \mathrm{CRO} \tag{13}$$

Equations (1) and (7) describe the natural degradation of the CI and CRO monomers. Equations (3)–(5) and (9)–(11) express the bindings of the dimers on the operators; the different states of the DNA are represented: constant D corresponds to the DNA with no bonds, D_1, D_2 and D_3 to the DNA with 1, 2 or 3 dimers CI_2 bound, and D'_3, D'_2 and D'_1 to DNA with 1, 2 or 3 dimers CRO_2 bound – see Figure 2. The gene expression is given by reactions (6), (12) and (13), where P stands for the RNA polymerase. Each reaction is parameterized with a *stochastic constant*, C_i for the reactions involving CI, and C'_i for CRO.

3.2 Rule-Based Programming of Chemical Reactions

A usual abstraction in the simulation of biochemical systems consists in considering the system, here the bacterium, as an homogeneous chemical solution where the reactions of the model are taking place. As the solution is considered well mixed, it can be represented by a multiset, that is a topological collection where any element may interact with all the other (10). This point of view is the starting of abstract chemical computation models like CHAM and Γ. However, the simulation of "real" chemical reactions requires more than only multiset rewriting to take into account the kinetics, corresponding to each reaction constant C_i and C'_i.

D.T. Gillespie has proposed in (21) an algorithm for producing the trajectories of a chemical system by choosing the *next reaction* and the *elapsed time* since last reaction occurred. Let μ be a chemical reaction, the probability that μ takes place during an infinitesimal time step is proportional to:

- c_μ, the *stochastic reaction constant*[2] of reaction μ;
- h_μ, the number of distinct molecular combinations that can activate reaction μ;
- $d\tau$, the length of the time interval.

Gillespie proved that the probability $P(\tau, \mu)d\tau$ that the next reaction will be of type μ and will occur in the time interval $(t + \tau, t + \tau + d\tau)$ is:

$$P(\tau, \mu)d\tau = a_\mu e^{-a_0 \tau} d\tau$$

where $a_\mu = c_\mu h_\mu$ is called the *propensity* of reaction μ, and $a_0 = \sum_\nu a_\nu$ is the combined propensity of all reactions.

This probability leads to the first straightforward Gillespie's algorithm called *first reaction method*. It consists in choosing an elapsed time τ for each reaction μ according to the probability $P(\tau, \mu)$. The reaction with the lowest elapsed time is selected and applied on the system making its state evolve. A new probability distribution is then computed for this new state and the process is iterated.

[2] Evaluating the stochastic constants is one of the hardest task in stochastic simulations of biochemical reactions. The interested reader should refer to (22; 23) that describe two experiences in that field.

3.3 Stochastic Simulation in MGS

We now present how the previous chemical model can be implemented in MGS. We need to represent the state of the dynamical system and its evolution function.

State of the System. As said above, the state of the bacterium is represented by a multiset of values. The considered molecules are abstractly represented using MGS *symbols* that are back-quoted identifiers. For example, the MGS symbol `‘CRO2` corresponds to a dimer CRO_2.

The initial state consists in a single molecule of the phage's DNA with some copies of RNA polymerase. As it has been noticed in vivo, the probability for CI to gain over CRO is low[3]. In order to have the simulation evolve in favor of CI, we put three copies of the CI protein to the initial state:

```
BactInit := ‘D :: #10 ‘P :: #3 ‘CI :: bag:()
```

This code defines a new variable, `BactInit`. The empty multiset is represented by `bag:()`, and operator `::` inserts an element in the given multiset. The expression `#`n `v` represents n copies of the value `v`.

Chemical Reactions Representation and Evolution. Each chemical reaction is translated into a transformation rule (or two if the reaction is reversible) that is characterized by an option `C` representing the stochastic constant of the reaction. For example, reaction (2) corresponds to the two following MGS rules

`#2 ‘CI ={ C = `C_{12}` }=> ‘CI2` and `‘CI2 ={ C = `C_{21}` }=> #2 ‘CI`

Consequently, the dynamics is captured by the following set of rules in the `Phage` transformation:

```
trans Phage = {
  (* Rules for CI *)                          (* Rules for CRO *)
  ‘CI       ={ C = $C_0$    }=> .;                  ‘CRO        ={ C = $C'_0$    }=> .;
  #2 ‘CI    ={ C = $C_{12}$ }=> ‘CI2;               #2 ‘CRO     ={ C = $C'_{12}$ }=> ‘CRO2;
  ‘CI2      ={ C = $C_{21}$ }=> #2 ‘CI;             ‘CRO2       ={ C = $C'_{21}$ }=> #2 ‘CRO;
  ‘D0, ‘CI2 ={ C = $C_1$    }=> ‘D1;                ‘D0, ‘CRO2  ={ C = $C'_1$    }=> ‘D’3;
  ‘D1       ={ C = $C_{-1}$ }=> ‘D0, ‘CI2;          ‘D’3        ={ C = $C'_{-1}$ }=> ‘D0, ‘CRO2;
  ‘D1, ‘CI2 ={ C = $C_2$    }=> ‘D2;                ‘D’3, ‘CRO2 ={ C = $C'_2$    }=> ‘D’2;
  ‘D2       ={ C = $C_{-2}$ }=> ‘D1, ‘CI2;          ‘D’2        ={ C = $C'_{-2}$ }=> ‘D’3, ‘CRO2;
  ‘D2, ‘CI2 ={ C = $C_3$    }=> ‘D3;                ‘D’2, ‘CRO2 ={ C = $C'_3$    }=> ‘D’1;
  ‘D3       ={ C = $C_{-4}$ }=> ‘D2, ‘CI2;          ‘D’1        ={ C = $C'_{-3}$ }=> ‘D’2, ‘CRO2;
  ‘D2, ‘P   ={ C = $C_t$    }=> ‘D2, ‘P, ‘CI;       ‘D0, ‘P     ={ C = $C'_t$    }=> ‘D0, ‘P, ‘CRO;
                                                    ‘D’3, ‘P    ={ C = $C'_t$    }=> ‘D’3, ‘P, ‘CRO
}
```

MGS integrates the *first reaction method* as a transformation rule application strategy. The transformation `Phage` is called and iterated using the Gillespie's strategy of MGS by the following expression:

```
Phage[iter = (tau <= $T$), strategy = ‘gillespie](BactInit)
```

An application of the transformation using this strategy consists in:

[3] An even more realistic behavior is obtained by considering models involving gene cII and cIII (24).

1. Computing the propensity a_μ of each rule r_μ by using the user-defined stochastic constant c_μ (taken from the `C` constant of the rule) and by evaluating the number of combinations h_μ (that is the number of subcollections matched by the lhs of the rule);
2. Computing for each rule r_μ, the value of the elapsed time τ_μ w.r.t. the definition of $P(\tau, \mu)d\tau$;
3. Applying once the rule with the lowest elapsed time.

Rule r_μ with the smallest τ_μ value is chosen and *fired one time* on the collection argument of the transformation, after instantiation of the lhs pattern.

A Simulation Example. During the iterations of the application of the `Phage` transformation using the Gillespie's strategy, MGS system variable `tau` is incremented using the elapsed times τ_μ of the chosen reactions. The value of `tau` is available anywhere in the transformation in case it is required. In the example given above, iterations stop as soon as `tau` reaches or goes over T arbitrary units of time. The use of the option `iter` allows to have a fine control over the number of applications of the transformation.

Figure 3 gives the results of three executions of the MGS program above (the instructions required for the output have not been included). Stochastic constants C_i and C'_i have been determined by biological experiments detailed in (25). The first plot shows the system in a state that has not yet evolved into a lytic or lysogenic phase; the next plot shows the system after a switch has occurred in a lytic phase (only molecules of CRO remain); the last plot shows the system in the lysogenic phase (only molecules of CI remain). Each simulation is run until 2000 arbitrary units of time are reached for the `Phage` transformation on an initial state consisting of three copies of the CI protein, one copy of the DNA and ten copies of the RNA polymerase. On 500 runs of the simulation, the lytic phase dominates in 51% of the cases and the lysogenic phase dominates in 39% of the cases.

4 Modeling of an Homogeneously Distributed Population of Bacteria

The previous chemical model only considers a single isolated bacterium and the reactions taking place in that bacterium. One of the interesting phenomena of the λ phage infection (24) is that two subpopulations (following either a lytic or a lysogenic phase)

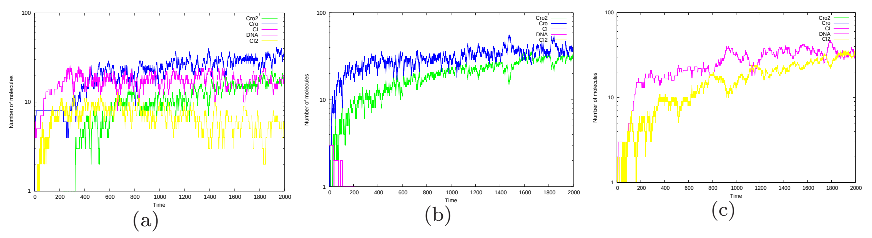


Fig. 3 Three different runs of the simulation, plotted with Gnuplot of the switch of the *lambda* phage. The light (resp. dark) curves show the number of CI (resp. Cro) molecules as a function of time (arbitrary units).

rapidly segregate after the introduction of the virus, that is, within a time interval of the same order than the cell division time which is about 20 minutes.

We propose to describe in this section the behavior of a population of bacteria through a quantitative and stochastic model relying on an extension of the Gillespie's first reaction method to nested compartments. As a matter of fact, we consider that all bacteria are homogeneously distributed in a “soup”.

4.1 A Simple Model of Infection Diffusion

We propose to study the segregation and the growth of the population during an infection process. Here, our main objective is to reuse the genetic switch model described and implemented in the previous section and coupling it with a model of infection diffusion taking place at the level of a population of bacteria. For the sake of simplicity, we keep our model as concise as possible.

This model involves a population of two kinds of entity that freely diffuse: bacteria and viruses. A bacterium can be in one of the following four different states:

1. *Safe* when it is not infected by a phage.
2. *Infected* when it has just integrated an external virus by endocytosis. We suppose that only safe bacteria can be infected.
3. *Lytic* when the infection has led to a lytic phase.
4. *Lysogenic* when the infection has led to a lysogenic phase.

To represent the growth of the population, we consider the two following behaviors of a bacterium:

– *Division*: all cells may reproduce. We assume that both daughter cells are sharing the same state of their mother cell.
– *Death*: there are two ways for a cell to die, either when it has reached a certain number of divisions (because of cell aging), or when it is a lytic bacterium that “explodes” releasing its virions in the population.

Viruses are considered as “passive” entities which are degradated.

4.2 Representation of a Population of Bacteria

The simulation of the population model requires to deal with chemical reactions that are not uniformly taking place in space: the cell membrane of each bacterium separates inner chemical reactions from their outside environment. A bacterium plays the role of a *compartment*, and compartments break the hypothesis of homogeneity of the system required by Gillespie's first reaction method. Nevertheless, Gillespie's method can be extended to compartmentalized systems under the following reasonable hypotheses:

– each compartment is an insulator that makes inner and outer reactions being independent;
– chemical reactions take place homogeneously within each compartment;
– chemical reactions take place homogeneously in the outside of the compartments, that is, at the level of the population.

This last hypothesis requires that the population of compartments is well mixed. In this model, we assume that segregation is space-independent as the infection is uniformly spread at the initial state.

Extension of Gillespie's First Reaction Method. The previous hypotheses ensure that Gillespie's approach can be independently considered at the level of the population and inside each compartment. Following the same reasoning as in section 3.2, let us consider a compartment σ and a reaction μ, and let us respectively denote c_μ^σ, h_μ^σ and a_μ^σ, the stochastic reaction constant of reaction μ in σ, the number of molecular combinations that can activate reaction μ in σ, and the propensity of reaction μ in σ. We can then define the probability $P(\tau,\mu,\sigma)d\tau$ that μ is the next reaction occuring in compartment σ in the time interval $(t+\tau, t+\tau+d\tau)$ as:

$$P(\tau,\mu,\sigma)d\tau = a_\mu^\sigma e^{-b_0\tau} d\tau$$

where the quantity $b_0 = \sum_\theta a_0^\theta = \sum_{\theta,\nu} a_\nu^\theta$ corresponds to the combined propensity of the whole system. Consequently, a new algorithm can be defined in the same way as the original Gillespie's first reaction method: an elapsed time τ is chosen for each compartment σ and each reaction μ w.r.t. probability $P(\tau,\mu,\sigma)d\tau$, and the couple (μ,σ) with the lowest elapsed time is considered.

Integration of the Extended First Reaction Method in MGS. Whereas nested compartments can be obviously handled in MGS as nested multisets, the extended algorithm cannot be easily implemented as a transformation rule application strategy. By contrast, it can be programmed as a function that uses the previously defined transformation `Phage`. This point of view has already been investigated in (26) where the authors propose to simulate biochemical processes with dynamic compartments in MGS using the formalism of P systems. Based on these results, we implement our model for the phage infection at the level of the population of bacteria.

We first have to define the transformation `Env` of reactions taking place at the population level:

```
trans Env = {
 'D          ={ C = $C_{degradation}$ }=> .
  b:bact     ={ C = $C_{death}$       }=> .
  b:bact     ={ C = $C_{division}$    }=> division(b)
  b:safe, 'D ={ C = $C_{infection}$   }=> 'D, #3 'CI, b
  b:lytic    ={ C = $C_{lytic}$       }=> #10 'D
}
```

Viruses are represented by their DNA in the environment. The rules of transformation `Env` respectively specify the natural degradation of viruses, the local evolutions of bacteria (death and division), the infection of "safe" bacteria by a virus, and the death of bacteria in lytic phase. The predicate `bact`, `safe` and `lytic` are used to distinguish the different states of a bacterium.

The implementation of the extended first reaction method consists in computing the local evolution of each compartment and then only considering the faster evolving compartment. We do not describe the MGS code of the `CompartmentsEvolve` that is straightforward: it consists in the distribution of the `Phage` transformation onto each cell and the selection of the fastest reaction.

Given the current state of the system, the final function `ExtendedFirstReaction` computes the new state of the system by choosing if the evolution has to take place at the level of a compartment or at the level of the population:

```
fun ExtendedFirstReaction(pop) = (
  let pop1, tau1 = CompartmentsEvolve(pop) in
  let pop2, tau2 = Env[strategy = 'gillespie,
                            postlude = (fun x -> x,tau)](b) in
        if (tau1 < tau2) then pop1 else pop2 fi
    )
```

The whole program (including instructions for producing the output) is about 150 lines of MGS code.

4.3 A Simulation Example

Figure 4 presents some simulations of this first model of infection diffusion. Parameters have been chosen in order to observe an exponential growth of a safe population (see left plot on Fig. 4(a)) and the same proportion of lysis and lysogeny in a population of infected bacteria (see on right plot of Fig. 4(a), that mean numbers of CI_2 and CRO_2 per bacteria are somehow the same).

The protocol of our study has consisted in running 500 simulations of the model, during 40 arbitrary units of time, and starting with a population composed of 9 safe bacteria and 1 lytic bacterium. When the lytic cell dies, it releases some viruses and makes the infection start. The simulations have underlined four different outcomes at the end of the simulation:

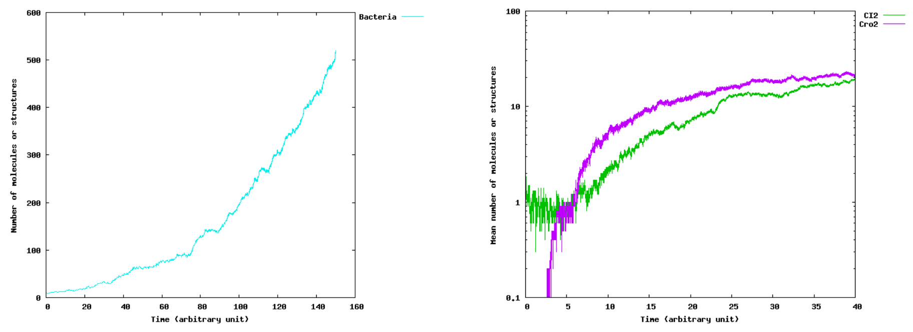
1. *Failure of the infection*: no virus remains in the environment and all bacteria are safe. This happens with frequency of about 10%.
2. *Death of the population*: as shown in Figure 4(b), majority of cells enter the lytic phase (CRO predominates when all the population is infected) and die. Then, remaining lysogenic cells are to few (about 3 on the plot) to maintain the population, and disappear due to the stochastic noise. This appears with frequency of about 12%.
3. *Homogeneous lysogenic population*: Figure 4(c) presents this case where all bacteria become lysogenic (CRO is not expressed anymore after time 25 and only CI remains). The population keep on growing exponentially; the phage DNA is silently replicated with each cell division. This outcome represents about 40% of the simulations.
4. *Heterogeneous lysogenic and safe population*: this interesting outcome is the more scarce (about 8% of the cases). The behavior is quite similar to the previous one, but the lysogenic phase appears while the whole population has not been infected yet (see Fig. 4(d)). It results a mixed population with safe and lysogenic bacteria.

Last 30% of the simulations corresponds to states that have not reach one of the previous 4 outcomes within the 40 units of time.

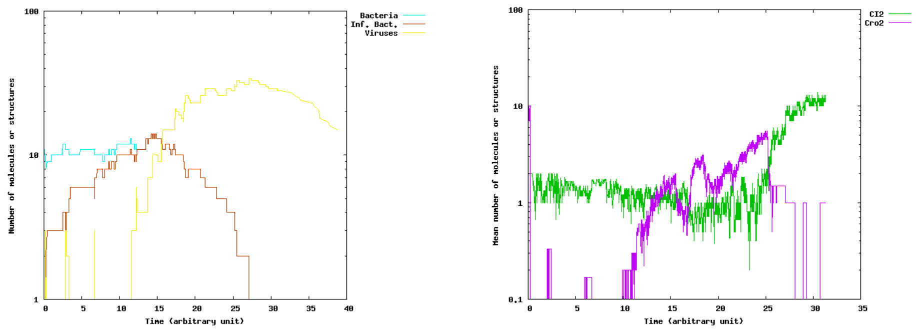
5 Modeling of an Heterogeneously Distributed Population of Bacteria

The previous description of the phage infection was a first step into the modeling at the level of the population of cells. However, the spatial distribution of cells and viruses was homogeneous. Because of that property, we have been able to extend and use Gillespie's algorithm to nested compartments. Nevertheless, this modeling method has two essential drawbacks.

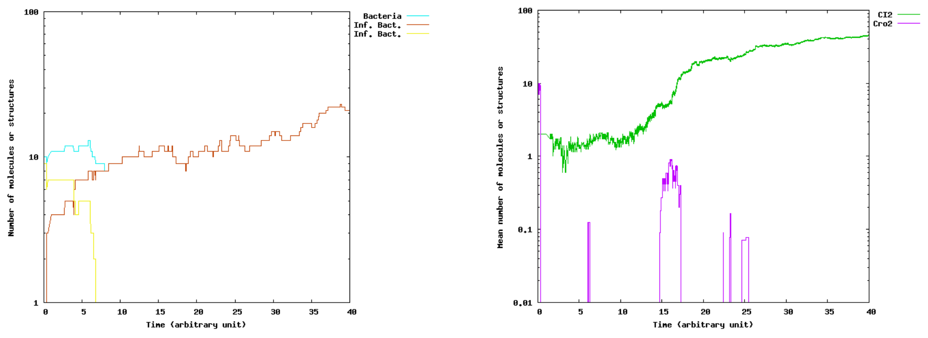
The first drawback is of practical importance: Gillespie's algorithm becomes slower as the number of molecular species increases. Consequently, the complexity of the



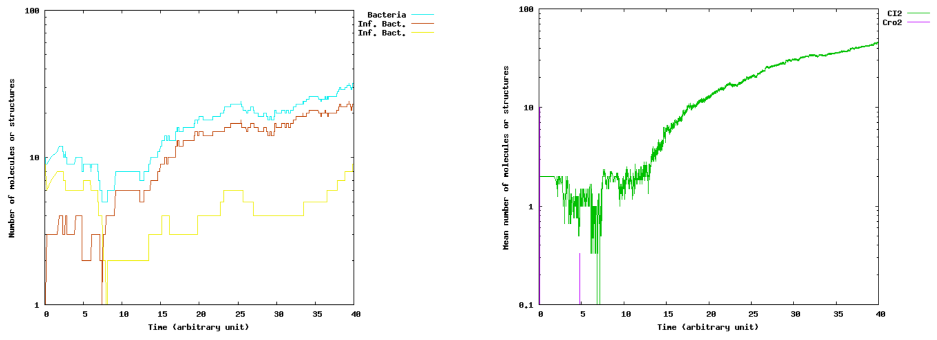
(a) Simulations illustrating the set of parameters.



(b) Simulation leading to the death of the whole population.



(c) Simulation leading to a purely lysogenic population.



(d) Simulation leading to a mixed lysogenic/safe population.

Fig. 4 Results of simulations of the nested compartments model.

extended first reaction method increases with the number of bacteria in the population. If the model of a bacterium involves N molecular species, simulating a population of n bacteria is equivalent to handling a solution involving $n \times N$ distinct molecular species. In this case, quantitative simulations require computation power that is rapidly out of reach for serious simulations (in size and duration).

The second drawback lies on the strong assumption of uniform and homogeneous organization of the system at all scales (from compartments to populations). The (multi-)set topology together with Gillespie's strategy capture the Brownian motion of well-mixed solutions. Nevertheless, this implicit representation of space cannot be easily extended to heterogeneous systems where space matters. Furthermore, it is not possible, with such models to capture spatial properties of biological systems like pattern formation, spatial segregation, ...

With the second kind of model presented in this section, we focus on this last issue. We consider a population of *spatially distributed* bacteria and propose a model allowing the study of the impact of cells organization on the evolution of the population. Such kind of model is of major importance since it may drastically change the interpretation of the considered biological system, as it has been shown in (27).

We propose in the next sections two models based on two different kinds of spatial representation: a model of population of cells on a static medium implemented as a cellular automaton and an individual-based model evolving on a dynamic graph topology.

5.1 Regular and Static Space Model: Cellular Automaton

Cellular automata provide a very simple way to handle spatial phenomena in biological simulations, as long as the topology of the underlying space do not evolve. We detail first how they can be considered as a rule-based computation model. Then, we present a model of the phage infection based on the use of a cellular automaton.

5.1.1 Cellular Automata in MGS

A cellular automaton (CA) is a *regular* lattice of *cells*, where each cell is characterized by a state taken from a finite set. The global evolution of the CA consists in applying *synchronously* on each cell, a *local* evolution function that computes the new state of the cell as a function of its current state and of the states of the cells in its *neighborhood.*

This computation model is naturally translated into a rule-based formalism:

– The regular lattice corresponds to a topological collection with a regular neighborhood. The MGS language provides a specific kind of topological collection called *Group Based Fields* (GBF for short), to deal with such a regular topology (see references (28; 29) for details on GBFs).
– The local evolution function can be encoded in rules of the form:

$$s => s' \quad \text{if some condition } c \text{ holds on the neighborhood}$$

Such a rule specifies the transition of a cell from the state s to the state s' where the condition c defines the context for the application of the rule. This rule can be straightforwardly translated in an MGS transformation rule:

s `as x / ` $\mathtt{P}_c$`(x) => ` s'

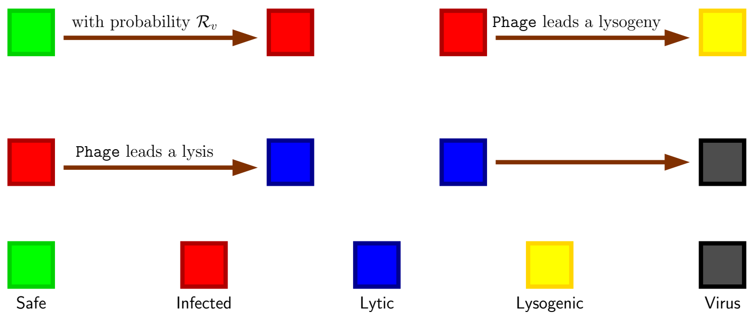


Fig. 5 States and evolution rules of the cellular automaton. The top of the figure describes the five different states of a cell; the middle and lower figures describe the evolution rules for a safe cell, an infected cell and a lytic cell.

where s and s' are constants of the language and $\mathtt{P}_c$ is a predicate that implements condition c.

– As all the cells have to be updated in a synchronous mode, the local evolution rules have to be applied all together and everywhere: this corresponds to the *maximal-parallel* rule application strategy.

5.1.2 Symbolic Model of an Infection Spreading

There are many ways to define a CA model of a biological system (30). We propose here an abstract and symbolic model of the infection spreading where the CA allows to underline the spatial organization.

To achieve this goal, we consider a CA based on a 2D square lattice. As summarized on Figure 5, the infection is characterized by the state of a cell: a cell is either safe or infected or lytic or lysogenic; the death of a lytic bacterium leads to the release of viruses. The local dynamics consists in updating this infection state using the rules given in Figure 5:

– a safe cell becomes infected with a probability $\mathcal{R}_v$ equal to the ratio of virus cells in its neighborhood;
– an infected cell becomes either lytic or lysogenic with respect to the chemical model given in section 3;
– a lytic cell dies and releases viruses.

The translation in the MGS formalism is straightforward. The following expressions define the different types of value used to represent the state of the CA:

```
    type States        = ‘Safe | ‘Infected | ‘Lytic | ‘Lysogenic | ‘Virus
and gbf Grid           = < North, South, East, West |
                           North = -South, East = -West >
and collection Lattice = [States]Grid
```

These declarations defines the type of the considered lattice as a GBF with values of type `States`. States are represented by symbols. The GBF topology is the Cayley graph of the finite presentation of a commutative group. The group is generated by the elementary displacements available to “move” from one node of the graph to one of its neighbors. The `Grid` GBF is defined by a finite presentation: a finite set of generators and equations between the generators. Here, four displacements (generators) are considered following the North, the South, the East, and the West direction. Knowing that the group is commutative and that North and South are opposite, as well as East and West, the Cayley graph of the group is isomorphic to a 2D grid with a von Neumann neighborhood.

The dynamics is implemented by the following transformation describing the three rules of evolution:

```
trans LocalEvolution = {
 ‘Safe / (random(1.0) < $\mathcal{R}_v$) => ‘Infected;
 ‘Infected                    => (
   let bact = Phage[strategy=‘gillespie](BactInit)
   in
     if lysogeny(bact)
     then ‘Lysogenic
     else ‘Lytic);
 ‘Lysic                       => ‘Virus;
}
```

The whole program (including instructions for producing the output) is about 100 lines of MGS code. In this transformation, we should emphasize that:

– The first rule is guarded by a predicate that depends on the value of a random number. This allows to apply the rule with probability $\mathcal{R}_v$ as required by the model.
– In the second rule, the choice between lytic and lysogenic phase depends on the call of the previously defined transformation Phage using Gillespie's strategy.
– Finally, if a cell is not updated using any of the 3 rules of the transformation, by the semantics of the transformation application, it remains unchanged.

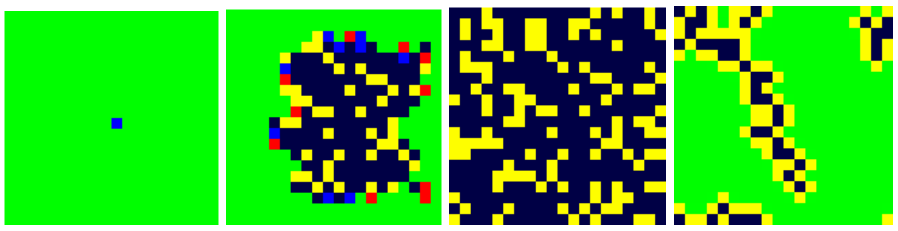

(a) Initial state: a lytic cell in the center of a population of safe bacteria.
(b) State of the system after 25 iterations.
(c) Fixed point after 75 iterations: no safe cell remains.
(d) Fixed point ater 150 iterations: formation of a cluster.

Fig. 6 Results of the simulation of the CA model on a toric 20x20 lattice. The color code of the four pictures is: safe bacteria are plotted in green, infected bacteria in red, lytic bacteria in blue, lysogenic bacteria in yellow, and viruses in dark blue.

5.1.3 A Simulation Example

Pictures in Figure 6 show some screen shots of runs of the CA model simulations. As the initial state is designed with a single infected cell (see Fig. 6(a)), the infection propagates as a concentric wave (see Fig. 6(b)) leading to a fixed point.

One may expect a fixed point where all CA cells are either in a lysogenic state or in a viral state (see Fig. 6(c)). Surprisingly, another kind of fixed point also occurs where lysogenic bacteria surround the *cluster* of viruses and stop the spread of the infection (see Fig. 6(d)). This outcome appears for sets of parameters C_i, C'_i leading to a majority of lysogeny.

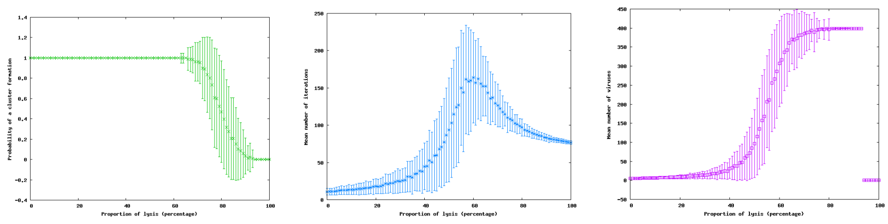
In order to study this phase transition, we have simulated the model for different *lytic rates*. The lytic rate represents the probability for an infected bacterium to switch to a lytic phase. The protocol of these simulations has consisted in 400 runs of the simulation for each lytic rate, varying from 0% to 100%.

Figure 7(a) shows that for lytic rates lower than 65%, there is always a formation of a cluster (the probability is equal to 1); for lytic rates higher that 95%, the infection is never stopped. Between these two thresholds, a phase transition occurs quite quickly (in a range of 30%). Figure 7(b) shows that even if a cluster appears for lytic rates lower than 60%, the activity in the CA (due to the lytic bacteria) is not constant and increases with the lytic rate to reach a maximum (at about 150 iterations). By contrast, when no cluster appears, the system stabilizes faster (at about 75 iterations) while the infection covers the whole lattice. It is interesting to see that the activity and the size of the clusters (given Fig. 7(c)) are not correlated showing that lytic rates of about 60% increase the stabilization time.

5.2 Amorphous and Dynamical Space Individual-Based Model

While the previous model exhibits the interesting effect of spatial aggregation, one could wonder whether it is meaningful to have a square lattice representing (and therefore *constraining*) the neighborhood of the cells. Furthermore, important features like cell replication and cell death (except for lytic cells) are not taken into account. Actually, considering such behaviors in a CA requires to deal with way more complex rules that leave the scope of standard CA description to more complex CA such as Margolus neighborhood or lattice gas automata.

In order to study how the propagation of the infection is related to the population growth, we propose to extend the previous model to a more phenomenological model with a dynamical topology where the spatial coordinates of cells matter. Such a topology is available in MGS using the *Delaunay* topological collections. Even if this kind of



(a) Probability of cluster formation according to the lysis rate.

(b) Mean number of iterations for stability according to the lysis rate.

(c) Mean size of the clusters according to the lysis rate.

Fig. 7 Results of the simulation of the CA model on a toric 20x20 lattice. On the three graphs, plots show some properties of the system according to the lytic rate of the model. The size of the error-bars clearly shows the effects of stochasticity on the results.

modeling falls in the field of *agent-based models* (ABM) or *multi-agent models* (MAS), we consider it as being part of what can be described and implemented in a rule-based language like MGS. We develop this point of view in the conclusion of this article.

5.2.1 Individual-Based Model on a Delaunay Graph

By contrast with the complete neighborhood of multisets, this collection (automatically) computes the neighborhood between the collection's elements according to their spatial coordinates. The neighborhood is computed using a *Delaunay triangulation* of the coordinates in $\mathbb{R}^n$. This kind of neighborhood together with a mechanical model has already been successfully used in biology for the modeling of growth of cells (31; 32). The mechanical model consists in a mass/spring system. It allows to keep a coherent global structure where cells are always close to each other and are able to "push" their neighbors if they lack space, or to fill holes in the structure.

5.2.2 A Phenomenological Model

The model is defined by extending the previous CA model. A cell of the CA lattice corresponds now to a punctual mass that is localized in the $\mathbb{R}^3$ space. A mass is characterized by its spatial position, its velocity, and its biological state (*i.e.*, a value of type `States` since it represents a biological entity). The neighborhood relation of the CA is replaced by springs between masses, and the presence of a spring between two masses depends on the neighborhood induced by the Delaunay triangulation of the masses. The biological evolution of the masses corresponds to the update of their biological states with respect to the previous transformation `LocalEvolution` extended with 3 rules to express cellular division, death, and virus degradation. The spatial evolution[4] of the masses corresponds to the definition of a new transformation `SpatialEvolution` that computes the movement of a mass due to its incident springs during a time interval Δt. Finally, the whole evolution step consists in composing the transformations:

$$\texttt{SpatialEvolution[*]} \circ \texttt{LocalEvolution}$$

Here, the mechanical model is iterated until an equilibrium is reached to consider a quasi-static approximation: the time-scale of the mechanical model is considered faster than the biological one. The whole program (including instructions for producing the output) is about 200 lines of MGS code.

5.2.3 A Simulation Example

Figure 8 shows four graphical interpretations of simulations of this model using the **Imoview** (33) visualization tool developed in the MGS project. In these simulations, viruses and bacteria do not have the same size: viruses are smaller than cells and thus allow a more efficient diffusion in the population. This prevents from the formation of clusters. Nevertheless, another kind of behavior appears as shown in Fig. 8(d). The population is split into two subgroups: a safe one and a lysogenic one. This outcome is quite similar to the behavior shown on Fig. 4(d) for the nested compartment model. It

[4] Because of the lack of space, we do not include here the MGS code for this model. Nevertheless, the code of all the examples given in this article is available upon request from the authors.

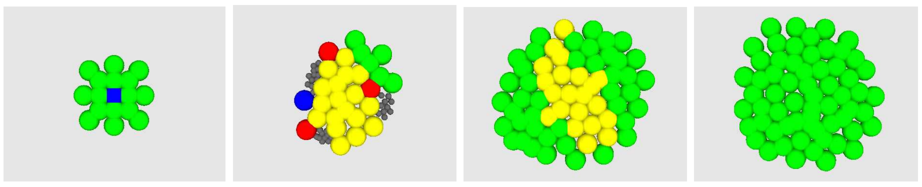

(a) Initial state, a lytic cell surrounded by 16 safe bacteria. (b) State of the system during the infection. (c) The evolution leads to a silent infection. (d) The infection failed.

Fig. 8 Results of the simulation of the dynamical space model. Safe bacteria are plotted in green, infected bacteria in red, lytic bacteria in blue, lysogenic bacteria in yellow, and viruses in dark gray.

should be of importance in real infections since lysogenic cells are immunized towards further infection. If a lysis is induced in the lysogenic population, the safe bacteria will prove to be a good source of hosts for the released viruses.

6 Conclusion: towards Spatial Systems Biology

Rule-based programming has been heavily investigated in the field of natural systems modeling. It was an essential contribution of (34) that has started an important effort in the π-calculus community to focus on problems raised by *systems biology* (35; 36; 37; 38; 39; 40; 41). This approach has been very efficient and successful, yet only limited to the consideration of biological processes that did not involve *space*: Brownian motion is supposed to allow each entity to encounter all other entities. Other formalisms have additional structure and spatial relationships: P systems consider nested compartments, L-systems allow through the interpretation of their symbols the modeling of complex spatial relationships, etc. but to our knowledge, besides *vertex-vertex systems* (42) which offer rewriting on graph topologies, no language nor formalism in the field of natural systems modeling has considered the *explicit* representation of space together with an *implicit* handling in evolution rules.

The MGS project is an attempt to combine rule-based programming with topological collections in a effort to offer a versatile, expressive and efficient mean for the modeling of natural systems. Rules allow to focus on the interactions between the elements of the systems while topological collections provide an explicit description of space. Here, we only have considered a small subset of the topologies available in MGS (18).

Nevertheless, we have shown that it is sufficient for the modeling of the biological process at various levels in *systems biology*: our paradigmatic example of the switch of the λ phage has been described from the transcription network at the level of the biochemical interactions to the level of population of cells embedded in a general three dimensional space. Furthermore, while our approach is clearly individual based, classical ABM/MAS (like NetLogo (43)) focus on the description of *agents behavior*, rule-based languages and formalisms promote the description of the *interactions* between the individuals. Among the many consequences of this point of view, it allows us to consider any kind of interactions between individuals (and not only interactions between two agents); since the rule application strategy is explicit, it makes possible to have a fine control over the scheduling of the selection of the individuals in interactions; by changing the type of topological collection, it is easy to change the nature of the underlying space where the individuals are embedded, etc.

Our running example is a very well studied biological process at the level of gene regulation. It is a classical case in *systems biology* which aims at integrating processes at various time and spatial scales into a single and coherent formal description. However, while classical molecular systems biology focuses on gene or protein interaction networks, cellular and supra-cellular organization levels are rarely considered. Since the fate of biological systems is not only determined by genes, the *spatial organization* of cells, tissues and organs plays a key role in most physiological processes (see for example (44)) and must therefore be a part of the models. Recognizing this importance by integrating spatial properties extends *systems biology* towards *spatial systems biology*.

We believe that languages and tools have to be further developed to take into account these spatial interactions, and that rule-based languages, like MGS, can play a key role. More specifically, the generalization of topological collections using notions from algebraic topology (14), allows us already to model complex spatial interactions. The relevance of these concepts is currently under validation through various large scale examples: simulation of morphogenesis processes (32), amorphous and autonomic computing examples and simulation at various levels of the behavior of genetically engineered bacteria in the french team of MIT's iGEM competition (45).

Acknowledgments. The authors gratefully acknowledge the reviewers for their valuable comments on a first version of this article.

References

1. Ulam, S.M.: On some mathematical problems connected with patterns of growth of figures. Proc. Symposia Appl. Math. **14**, 215–224 (1962)
2. Von Neumann, J.: Theory of Self-Reproducing Automata. Univ. of Illinois Press (1966)
3. Berry, G., Boudol, G.: The chemical abstract machine. In: Conf. Record 17th ACM Symp. on Principles of Programmming Languages, POPL'90, San Francisco, CA, USA, 17–19 Jan. 1990, pp. 81–94. ACM Press, New York (1990)
4. Banâtre, J.P., Fradet, P., Métayer, D.L.: Gamma and the chemical reaction model: Fifteen years after. Lecture Notes in Computer Science **2235**, 17–44 (2001)
5. Păun, G.: From cells to computers: Computing with membranes (P systems). Biosystems **59**(3), 139–158 (2001)
6. Rozenberg, G., Salomaa, A.: Lindenmayer Systems. Springer, Berlin (1992)
7. Giavitto, J.L., Michel, O.: Data structure as topological spaces. In: Proceedings of the 3nd International Conference on Unconventional Models of Computation UMC02, vol. 2509, pp. 137–150. Himeji, Japan (2002). `http://www.ibisc.fr/~michel/PUBLIS/2002/umc02.pdf`
8. Dershowitz, N., Jouannaud, J.P.: Handbook of Theoretical Computer Science, vol. B, chap. Rewrite systems, pp. 244–320. Elsevier Science (1990)
9. Manca, V.: Logical string rewriting. Theoretical Computer Science **264**, 25–51 (2001)
10. Fisher, M., Malcolm, G., Paton, R.: Spatio-logical processes in intracellular signalling. BioSystems **55**, 83–92 (2000)
11. Eker, S., Knapp, M., Laderoute, K., Lincoln, P., Meseguer, J., Sonmez, K.: Pathway logic: Symbolic analysis of biological signaling. In: Proceedings of the Pacific Symposium on Biocomputing, pp. 400–412 (2002)
12. Giavitto, J.L., Malcolm, G., Michel, O.: Rewriting systems and the modelling of biological systems. Comparative and Functional Genomics **5**, 95–99 (2004). `http://www.ibisc.fr/~michel/PUBLIS/2004/CFG04.pdf`

13. Giavitto, J.L., Michel, O.: The topological structures of membrane computing. Fundamenta Informaticae **49**, 107–129 (2002). `http://www.ibisc.fr/~michel/PUBLIS/2002/FI.pdf`
14. Giavitto, J.L., Spicher, A.: Topological rewriting and the geometrization of programming. Physica D **237**, 1302–1314 (2008). DOI 10.1016/j.physd.2008.03.039
15. Munkres, J.: Elements of Algebraic Topology. Addison-Wesley (1984)
16. Giavitto, J.L.: Topological collections, transformations and their application to the modeling and the simulation of dynamical systems. In: Rewriting Technics and Applications (RTA'03), *LNCS*, vol. LNCS 2706, pp. 208 – 233. Springer, Valencia (2003)
17. Giavitto, J.L., Godin, C., Michel, O., Prusinkiewicz, P.: Modelling and Simulation of biological processes in the context of genomics, chap. "Computational Models for Integrative and Developmental Biology". Hermes (2002). `http://www.ibisc.fr/~michel/PUBLIS/2002/autran02.ps.gz`
18. Michel, O.: There's plenty of room for unconventional programming languages or declarative simulations of dynamical systems (with a dynamical structure). Ph.D. thesis, Université d'Évry (2007). `http://www.ibisc.univ-evry.fr/~michel/Hdr/hdr.pdf`
19. Ptashne, M.: A genetic switch : phage lambda and highet organisms. Cold Spring Harbor Laboratory Press (1992)
20. Lou, C., Yang, X., Liu, X., He, B., Ouyang, Q.: A quantitative study of λ-phage switch and its components. Biophysical Journal **92**, 2685–2693 (2007)
21. Gillespie, D.T.: Exact stochastic simulation of coupled chemical reactions. J. Phys. Chem. **81**(25), 2340–2361 (1977)
22. De Cock Katrienand Zhang, X., Bugallo, M.F., Djuric, P.M.: Stochastic simulation and parameter estimation of first order chemical reactions. In: 12th European Signal Processing Conference (EUSIPCO-2004) (2003)
23. Zhang, X., De Cock, K., Bugallo, M.F., Djuric, P.M.: Stochastic simulation and parameter estimation of enzyme reaction models. In: IEEE Workshop on Statistical Signal Processing (2003)
24. Arkin, A., Ross, J., McAdams, H.H.: Stochastic kinetic analysis of developmental pathway bifurcation in phage λ-infected *Escherichia coli* cells. Genetics **149**, 1633–1648 (1998)
25. Kuttler, C., Niehren, J.: Gene regulation in the pi calculus : simulation coperativity at the lambda switch. Transactions on Computational Systems Biology (2005)
26. Spicher, A., Michel, O., Cieslak, M., Giavitto, J.L., Prusinkiewicz, P.: Stochastic p systems and the simulation of biochemical processes with dynamic compartments. BioSystems **91**(3), 458–472 (2008)
27. Shnerb, N.M., Louzoun, Y., Bettelheim, E., Solomon, S.: The importance of being discrete - life always wins on the surface (1999). Comment: 4 pages, 4 figures
28. Giavitto, J.L., Michel, O.: Declarative definition of group indexed data structures and approximation of their domains. In: PPDP '01: Proceedings of the 3rd ACM SIGPLAN international conference on Principles and practice of declarative programming, pp. 150–161. ACM Press, New York, NY, USA (2001). DOI http://doi.acm.org/10.1145/773184.773201
29. Giavitto, J.L., Michel, O., Cohen, J.: Pattern-matching and rewriting rules for group indexed data structures. In: ACM Sigplan Workshop RULE'02, pp. 55–66. ACM, Pittsburgh (2002). `http://www.ibisc.fr/~michel/PUBLIS/2002/rule02.pdf`
30. Ermentrout, G.B., Edelstein-Keshet, L.: Cellular automata approaches to biological modeling. J. Theor. Biol. **160**(1), 97–133 (1993)
31. Gibson, M.C., Patel, A.B., Nagpal, R., Perrimon, N.: The emergence of geometric order in proliferating metazoan epithelia. Nature **442**, 1038–1041 (2006)
32. Barbier de Reuille, P., Bohn-Courseau, I., Ljung, K., Morin, H., Carraro, N., Godin, C., Traas, J.: Computer simulations reveal novel properties of the cell-cell signaling network at the shoot apex in arabidopsis. PNAS **103**(5), 1627–1632 (2006)
33. Letierce, F., Giavitto, J.L., Michel, O., Spicher, A.: The imoview vizualisation tool (2005). Available at `http://mgs.ibisc.univ-evry.fr/Imoview/`
34. Regev, A., Silverman, W., Shapiro, E.: Representation and simulation of biochemical processes using the π-calculus process algebra. In: R.B. Altman, A.K. Dunker, L. Hunter, T.E. Klein (eds.) Pacific Symposium on Biocomputing, pp. 459–470 (2001)
35. Priami, C., Quaglia, P.: Beta binders for biological interactions. In: V. Danos, V. Schächter (eds.) CMSB, *Lecture Notes in Computer Science*, vol. 3082, pp. 20–33. Springer (2004)

36. Regev, A., Panina, E.M., Silverman, W., Cardelli, L., Shapiro, E.: Bioambients: An abstraction for biological compartments. TCS: Theoretical Computer Science **325** (2004)
37. Cardelli, L.: Brane calculi. In: V. Danos, V. Schächter (eds.) CMSB, *Lecture Notes in Computer Science*, vol. 3082, pp. 257–278. Springer (2004)
38. Calder, M., Gilmore, S., Hillston, J.: Modelling the influence of RKIP on the ERK signalling pathway using the stochastic process algebra PEPA. In: C. Priami, A. Ingólfsdóttir, B. Mishra, H.R. Nielson (eds.) Transactions on Computational Systems Biology VII, *Lecture Notes in Computer Science*, vol. 4230, pp. 1–23. Springer (2006)
39. Hlavacek, W.S., Faeder, J.R., Blinov, M.L., Posner, R.G., Hucka, M., Fontana, W.: Rules for modeling signal-transduction systems. Science STKE **344** (2006)
40. Danos, V., Feret, J., Fontana, W., Harmer, R., Krivine, J.: Rule-based modelling of cellular signalling. In: CONCUR'07, pp. 17–41. Springer Berlin (2007). DOI 10.1007/978-3-540-74407-8_3
41. Danos, V., Feret, J., Fontana, W., Harmer, R., Krivine, J.: Rule-based modelling, symmetries, refinements. In: Proceedings of FMSB 2008, Lecture Notes in Bio Informatics (2008). To appear.
42. Smith, C., Prusinkiewicz, P., Samavati, F.F.: Local specification of surface subdivision algorithms. In: J.L. Pfaltz, M. Nagl, B. Böhlen (eds.) AGTIVE, *Lecture Notes in Computer Science*, vol. 3062, pp. 313–327. Springer (2003)
43. Wilensky, U.: Netlogo. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL. (1999). `http://ccl.northwestern.edu/netlogo`
44. Farge, E.: Mechanical induction of Twist in the Drosophila foregut/stomodeal primordium. Current Biology **13**(16), 1365–1377 (2003)
45. iGEM: Modeling a synthetic multicellular bacterium. Modeling page of the Paris team wiki at iGEM'07 (2007). `http://parts.mit.edu/igem07/index.php/Paris/Modeling`