Rewriting and Simulation
Application to the Modeling of the Lambda Phage Switch

Antoine Spicher[1], Olivier Michel[1] & Jean-Louis Giavitto[1]

[1]IBISC – FRE 2873 du C.N.R.S.
Informatique, Biologie Intégrative et Systèmes Complexes,
Équipe Langages, Interactions et Simulations
523 place des terrasses de l'Agora
91000 ÉVRY CEDEX, France
{aspicher, michel, giavitto}@ibisc.univ-evry.fr

1 Introduction

MGS is a high-level programming language dedicated to the modeling and simulation of biological systems. Biological systems, due to their complex mechanisms and to their highly concurrent nature, are systems very difficult to model.

The local expression of evolution rules of biological systems coupled with the modification of the structure of the described systems (we call these systems *dynamical systems with a dynamic structure* or $(DS)^2$ [GGMP02], that is, those which structure evolves along time) leads to many problems in the description and manipulation of biological systems. To address these problems, we propose to use rewriting techniques on complex and dynamical data structures.

Rewriting systems (RS), initially developed for the formalization of equational reasoning, are also used to express changes of states in a process. The ability of RS to specify the changes of a sub-part of an object by another one tends to identify RS as a formalism that suits well to describe *local* evolution rules. The simulation of a biological system can be seen as the iterated application (following some given strategy) of rules from a set of rewriting rules on a data structure representing the current state of the system.

To add some expressivity to rewriting systems (that is, to go further than terms that only allow the representation of tree-like organizations), we have developed new rewriting techniques applied to more complex data structures that we call *topological collections*. The rule application *strategies* of the transformations allow to specify the interaction mode of the sub-parts of the biological system.

The topological point of view considers a data structure as a set of organized elements following a neighborhood relationship that defines which elements of the structure are accessible from a given element. The *transformations*, following a local point of view, are functions allowing the handling of topological collections through the definition of rewriting rules based on the neighborhood relationship of the collection.

The aim of the MGS project is to integrate the formalism of rewriting systems in a programming language for the modeling and simulation of $(DS)^2$.

2 Motivations and Objectives

In this paper, we are interested in the use of rewriting, a computer tool, as a starting point for the development of a set of tools for the modeling and simulation of biological phenomenons.

Modeling in biology raises many questions about the tools to use for the description of phenomenons. Indeed, the systems are complex and highly structured, with many sub-models interacting at various scales (in time and space). Furthermore, they often exhibit an additional feature: their behaviour depends on the structure (spatial or functional) itself and this structure is induced by the behavior.

From a computer science point of view, the phase space of these systems cannot be defined *a priori* but must be computed together with the state of the system. The phase space becomes a variable of the system itself (see [Gia03]).

Many computation models have been inspired by these problems raised by biological systems. We can name the CHAM formalism and the Gamma [BFM01] language which take their origins in chemistry, Păun's systems [Pău01] (or P systems) corresponding to the metaphor of cell's membrane, or Lyndenmayer's systems [RS92] (or L systems) inspired by the growth of plants. All these models share the common property that they allow the specification of systems in a *discrete, local* and *declarative* manner:

- **discrete**: In spite of their strength, continuous formalisms make the expression of the discrete nature of the biological systems very difficult. Furthermore, a discrete approach leads to adopt an algebraic point of view for the systems that fits well to the language theory domain, used for the different models above.
- **local**: The high complexity of the systems in biology usually leads to problems for their description. Nevertheless, the description of the evolution rules consists in the straightforward specification of the local interactions between the elements of the system. The global behavior can be simulated by *integrating* the local laws on the whole system.
- **declarative**: A declarative style leads to a specification of the models close to a mathematical formalism: the description of the model is small and expressive, theoretically well founded and close to the concepts used by the modelers. Furthermore, formal techniques can be used to prove the correctness or to check some properties of the biological model.

Computation models like L systems, P systems or the abstract chemistry can be seen as a specific form of *rewriting* to be applied on different spatial structures: Venn diagrams for P systems, tree-like structures for L systems, a multi-set for abstract chemistry. In the MGS project, we are interested in these different kind of rewriting, to their unification or generalization, and to their use for the modeling and simulation of biological processes.

In this paper, we start by presenting the rewriting systems and their use for the simulation of dynamical systems. The MGS formalism will then be introduced for the modeling of the λ phage genetic switch.

3 Rewriting and Simulation

In this section, we show how rewriting can be used for the simulation of *dynamical systems* (DS), that is, systems which states evolve in time.

3.1 A Medium for Computation

A *rewriting system* [DJ90] is a tool used for the substitution of a part of an entity by another. In computer science, the entities subject to this process are, in general, expressions represented by formal trees. A RS is defined by a set of rules, and each rule $\alpha \rightarrow \beta$ specifies how a sub-part of the expression that is matched by the pattern is replaced by a new part computed from expression β. We'll be using the terms of *left hand side* and *right hand side* of a rule for respectively α and β.

The writing $e \rightarrow^* e'$ means that expression e is transformed by a set of rewriting steps in e'. In other terms, there is a sequence of expressions $e_1, \ldots, e_n$ such that $e = e_1$, $e_1 \rightarrow e_2$, $e_2 \rightarrow e_3$, $\ldots$, $e_{n-1} \rightarrow e_n$ and $e_n = e'$. This sequence is called a *derivation* of e. The transformation of e in e' can be seen as the result for a computation defined by the rewriting rules; the derivation corresponds to the sequence of the intermediate results. The notion of derivation allows to define some properties for a set of rules:

- A set of rules *terminates* if all the possible derivations for any expression are of finite length. This means that for each expression e, there are some expressions e' such that $e \rightarrow^* e'$ and where no rule can be applied to e'.
- A set of rules is *confluent* if, for any expression e, there are two derivations leading to expressions e'_1 and e'_2 with $e \rightarrow^* e'_1$ and $e \rightarrow^* e'_2$, then, there exists an expression e' such that $e'_1 \rightarrow^* e'$ and $e'_2 \rightarrow^* e'$.
- If a rewriting system *terminates* and is *confluent*, then, for all expression e there is a *unique normal form* for that expression.

3.2 Rewriting and Simulation

In order to be able to go from an informal description of a biological phenomenon to a computable model, many authors [Man01, FMP00, EKL+02, GGMP02] have proposed to represent the state of a biological system by a term t of the form

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

where a sub-term t_i represents a biological entity or a message (signal, command, information, action, etc.) sent to an entity. The simulation of the evolution of the biological system is done by rewriting of the term. Rules of the form

$$e + m \rightarrow e' + m'$$

are used. The left hand side corresponds to a message m and to the entity e which m is sent to; the right hand side specifies the new state e' of the entity and (eventually) some new messages m'. Note that the direct interaction between entities can be specified by rules of the form

$$e_1 + e_2 \rightarrow e'_1 + e'_2$$

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

$$e \rightarrow e' + e''$$

etc.

To summarize, the notions developed for RS and DS can be related in the following way:

- a state is represented by a term and a sub-term corresponds to the state of a sub-system;
- the evolution rule is coded by the rules of the RS:
  - the left hand side of the rule matches a sub-system whose elements are in interaction,
  - the right hand side of the rule computes the result of this interaction.

Intuitively, we can deduce that, given the state of a DS through expression e, a derivation of e is understood as the description of the evolution of the system along time; we speak of the *trajectory* of the system.

Choosing RS as a formalism to describe DS implies to consider the handling of time and space from the discrete point of view, as well as local.

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

Locality of Space. The $+$ operator used above, connecting entities and signals expresses the spatial and/or functional organization of the modeled system. It is used to denote the sub-parts of the system that are in interaction, as well as to specify the way that the sub-systems are organized. Consequently, the rules represent the local evolutions laws of the DS. Furthermore, the organization of the structures specified in the left hand side and in the right hand side of the rule may differ, producing a modification of the system's structure. This allows the modeling of a class of DS particularly complex to describe and simulate, *dynamical systems with a dynamic structure*.

Nevertheless, even if the parallel between RS and $(DS)^2$ is easy to observe, it is not straightforward to set up in a real environment. As a matter of fact, standard rewriting systems are aimed at a specific data structure, namely terms. But the term structure does not allow to represent states of a system that are not always structured in a tree-like way but highly and arbitrarily structured. The terms are not expressive enough to take into account these organisations. One of the key point is to extend the RS to data structures arbitrarily complex. We'll detail in the next section how this extension has been made possible through the development of the MGS language.

4 A Brief Description of the MGS Formalism

The goal of the MGS project is to study and develop a programming language for the modeling and the simulation of phenomena which structures are dynamic. Domain specific languages are usually declarative and organized around a small kernel, define new abstractions and notations targeted towards new specific application domains. They consequently allow to take into account the modeled phenomena in a more concise way with a high ability to reuse developed code.

The fundamental point of MGS is to allow the specification of those states using a new data structure [GM02] based on the topological organization of the modeled systems. The evolution function of these systems corresponds to a function that transforms the state of the system. The specification of these evolution functions is simplified by the definition of a case-based function based on a powerful pattern language. MGS can be considered a standard functional language extended with a new data structure, the *topological collections*, and a new way to define functions, the *transformations*. An alternative point of view is to interpret the transformations as the application of rules specifying the local evolution of a state seen as a complex data structure.

4.1 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* are a unified view of the notion of data structure [GM02]. In this framework, a data structure is defined 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.

Topological Collection and Representation of the State of a $(DS)^2$. Topological collections suits well to the representation of the states of a complex dynamical system. The elements of a collection are the atomic elements of the system.

4.2 Transformations

Topological collections represent an adequate medium to extend RS. 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 structures different than trees, and are used to specify the evolution function of the modeled DS. The transformation of a topological collection S consists in *applying* a set of *local rewriting rules*. A rewrite rule r specifies the substitution of a sub-collection

by another sub-collection. The application of a rule $\sigma \Rightarrow f(\sigma, ...)$ on collection S : (1) selects by a matching algorithm a sub-collection S_i of S corresponding to an instantiation of *pattern* σ, (2) computes a new collection S'_i as the evaluation of the application of function f onto S_i and its neighborhood, and (3) specifies the insertion of S'_i in place of S_i in S.

The Matching Language. Pattern σ of the left hand side of the rule specifies a sub-collection; we represent here the occurrence of an interaction whose generic form is given by σ. This sub-collection can have an arbitrary shape, thus making its characterization delicate. The expressivity of the pattern language where σ is defined, corresponds to the class of sub-collections, and therefore of interactions, that we want to specify.

MGS allows the use of two different pattern languages: the first one, the *patches*, is very expressive but complex to use; the other one, the *paths* is less expressive but easier to use. This last language consists in the sequential enumeration of neighbor elements in the collection. Such an enumeration if called a *path*. In the rest of the paper, we'll only consider this language.

Substitution. The right hand side of the rule specifies a collection that replaces the sub-collection matched by the pattern specified in the left hand side of the rule. Once again, the specification of such a collection is not an easy task.

Nevertheless, there is an alternative point of view in the case of the path matching: the succession of the patterns describing a sequence of elements, the right hand side of a rule can be an expression that also evaluates to a sequence of values. The substitution is then done element-wise: element i in the matched pattern is replaced by the i^{th} element in the right hand side of the rule. This point of view allows a very concise expression of the rules.

Rule Application Strategy. We have just briefly described one rule application (matching of the left hand side and substitution by the right hand side). Nevertheless, to have the collection evolve, it is required to iterate the application of the rules on the whole system. Many problems of "collisions" arise; indeed, what should be selected if a rule allows to match more than sub-collection, or, if a sub-collection can be matched by more than a rule ?

The *rule application strategy* deals with those conflicts: it implies on one side the number of sub-collection to match (whatever the rule), emulating thus synchronous or asynchronous evolutions and on the other side the priority between the different rules.

MGS has many application strategies to allow an easy programming of most of the possible evolutions of the model.

The default strategy of the MGS language prototype is to make a *maximal parallel* matching with a priority on the rules, following the approach adopted in Lindenmayer's systems [LJ92] that allows the maximal parallel rewriting on strings.

Even if this strategy is very close to a biological metaphor that would expect all different interactions to happen in parallel, it does not allow the description of complex models relying on, for example, stochastics properties. The next section focuses on the description of an application strategy suits well to discrete, stochastic, chemical simulations, used in the specification of the λ phage genetic switch.

5 The Lambda Bacteriophage in MGS

5.1 Abstract Chemistry

Abstract chemistry is a computation procedure inspired by chemical reactions. The initial intuition of the computation paradigm is that the state of a chemical solution can be represented by a *multi-set*: a multi-set is a set where multiple occurrences of the same element may occur, in the same

way that many molecules of the same species are present in the solution, establishing therefore a concentration of the species.

Multi-sets can be represented by terms whose formal building operator $+$ (introduced in section 3.2) is *commutative* and *associative*. For example, if a, b and c represent chemical species, expression $(a+b)+(b+c)$ evaluates a multi-set containing one occurrence of a and c and two of b. Associativity and commutativity allow respectively to remove the parentheses and to permute the operands of the $+$ operator. Therefore

$$(a+b)+(b+c) = a+b+b+c = a+b+c+b = \ldots$$

Commutativity and associativity simulate a Brownian motion of the multi-set elements such that two elements may encounter at any time. So, computing using these multi-sets consists in applying *chemical reactions*, represented by rewriting rules, according to the elements collisions. For example, rules:

$$r_1 : a+a \rightarrow a+a+b \qquad r_2 : a+b \rightarrow a+b+b \qquad r_3 : b+b \rightarrow b+b+a$$

represent catalytic reactions of second order between molecules of type a and b (a collision of two molecules catalyse the formation of a third one, the two initial molecules being not consumed).

5.2 Abstract Chemistry in MGS

Abstract chemistry is straightforward using the notions provided by MGS.

A Multi-Set Seen as a Topological Collection. The chemical soup is represented with a multi-set. As for all standard data structures, a multi-set can be described through a local neighborhood between the elements. Here, commutativity and associativity allow all elements to collide. We then define a topological collection where each element is neighbor of all the others, the neighborhood being the possibility for two elements to react.

Reaction and Transformation. As stated before, a chemical reaction can be translated as a rewriting rule. The set of rules that is taken by a chemical solution, can therefore be specified as a MGS transformation. As an example, if we take rules r_1, r_2 and r_3 of the preceding paragraph, their translation in MGS is straightforward:

```
trans catalytic = {
  r1 :  'a, 'a => 'a, 'a, 'b ;
  r2 :  'a, 'b => 'a, 'b, 'b ;
  r3 :  'b, 'b => 'a, 'b, 'b
}
```

The `trans` keyword states that a new transformation whose name is `catalytic` is defined. The three rules of the transformation are identical to the preceding rewriting rules, at the difference that the operator of construction $+$ is replaced by MGS's neighborhood's operator that is the comma (introduced in the framework of the topological collections and the matching of paths). The first rule states that if an entity whose value is `'a` (it is a constant value) is in the neighborhood of a second entity of value `'a`, then this path can be rewritten in the sequence `'a, 'a, 'b`, where the two entities of type `'a` remain, and where a new kind of species, `'b`, is introduced.

5.3 Abstract Chemistry for the Simulation of a Chemistry Soup

The usual strategy used in artificial chemistry (in an algorithmic framework for example) is the maximal parallel strategy proposed as the default strategy in MGS. Nevertheless, this strategy is not adequate for the description and simulation of "real" chemical reactions because it does not take into account the different kinetics of the chemical reactions.

A standard stochastic strategy allows to parametrize each rule by an application probability. This is found in MGS as well as in other languages, like Elan [BK02] for example. We propose in the next section a specialization of this strategy to take into account the modeling and simulation of stochastic biochemical systems.

The presence of stochastic phenomenon in a large number of physical system is no longer to be proved. These last years, a growing number of researchers have studied randomness and noise present in biological systems. The interested reader should refer to [MSD04] where the detail is given of many stochastic phenomena happening in biology, and this from the molecular level to the cellular level. Before presenting the integration of the SSA (standing for *Stochastic Simulation Algorithm*) in MGS, we introduce some mathematical notions (in a form very close to the presentation made in [MSD04]) to help its understanding.

5.3.1 Chemical Reactions and Master Equation

The behavior, along time, of a spatially homogeneous soup of molecular species can be described by a chemical equation called the *master equation* [Gil77] (ME). This ME describes the transition of the system from a state to another in probabilistic terms. State X of the system is supposed to be given by a concentration of each of the chemical products in the soup. The evolution of $p(X,t)$, the probability of the system to be a state $X = (\dots, x_i, \dots)$ at time t by

$$p(X, t+\Delta t) = p(X,t)[1 - \sum_{j=1}^{N}(\alpha_j \Delta t)] + \sum_{j=1}^{N} \beta_j \Delta t$$

where x_i is the amount of chemical species i in reaction in the system; N is the number of reactions; $\alpha_j \Delta t$ the probability that, the system being in state X at time t, the j reaction takes place between time t and $t+\Delta t$; $\beta_j \Delta t$ the probability that the j reaction puts the system in state X at time $t+\Delta t$.

When Δt goes to 0, we get the classical form of the ME

$$\frac{\partial p(X,t)}{\partial t} = \sum_{j=1}^{N} \beta_j - p(X,t) \sum_{j=1}^{N} \alpha_j$$

One can notice that the transition of a state of the system is described through the change of probabilities of the system to be found in a given state. The ME approach consists in trying to describe a set of equations for each possible transition and to solve them *simultaneously*. It is quite easy to generate a single trajectory, but when the dimension of the problem gets higher, the possible trajectories face a combinatorial explosion and the problem becomes intractable. A solution to this problem is proposed by Gillespie [Gil77] who, rather than writing explicitly the ME, produces the trajectories in choosing the *reactions* and the *elapsed time* since last reaction occurred according to the probability distributions such that the probability of producing a given trajectory be exactly the same as the solution of the ME.

5.3.2 Gillespie's Two Algorithms

Two algorithms are proposed in [Gil77] to solve the ME described above, by making the hypothesis that all chemical species are spatially homogeneously distributed. These algorithms are called SSA

for *Stochastic Simulation Algorithm*. At each time-step, the chemical system is in a single and unique state. The algorithm simulates the temporal evolution of the system by computing when and which reaction should take place.

We suppose that the system is in state α at time t, that it is composed of M chemical species that can interact following N chemical reactions. We denote reactions R_μ, with μ ranging from 1 to N. We then define

- $P(\tau, \mu)$ the probability that the next reaction R_μ takes place in the infinitesimal time interval $(t + \tau, t + \tau + d\tau)$;
- c_μ the *stochastic reaction constant*[1] of reaction R_μ;
- h_μ the number of distinct molecular combinations that can activate reaction R_μ;
- a_μ the propensity function of reaction R_μ;
- $a_\mu dt = h_\mu c_\mu dt$ the probability that a reaction R_μ takes place in the interval $(t, t + dt)$.

Gillespie proved that

$$P(\tau, \mu)d\tau = a_\mu e^{-\tau \Sigma_j a_j} d\tau \tag{1}$$

This equation leads to a first straightforward Gillespie's algorithm called *first reaction method*. It consists in choosing an elapsed time τ for each reaction R_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.

Gillespie also proposed to separate (1) into two independent probability distributions given by the following equations

$$P(\mu) = \frac{a_\mu}{\sum_j a_j} \tag{2}$$

$$P(\tau)d\tau = \sum_j a_j e^{-\tau \Sigma_j a_j} d\tau \tag{3}$$

Equation (2) computes the probability for reaction R_μ to be the next reaction to take place; equation (3) takes into account the amount τ of time elapsed to compute the time when the reaction did occur. By using a random number generator and by deriving these two probability distributions, we get the second Gillespie's algorithm called the *direct method*. Difference between both methods stands in the selection of the reaction to be applied. For this second version, the elapsed time and the reaction are independently computed with respect to distributions (2) and (3). This procedure is more efficient because only two random numbers have to be computed against N (one for each reaction) for the first version.

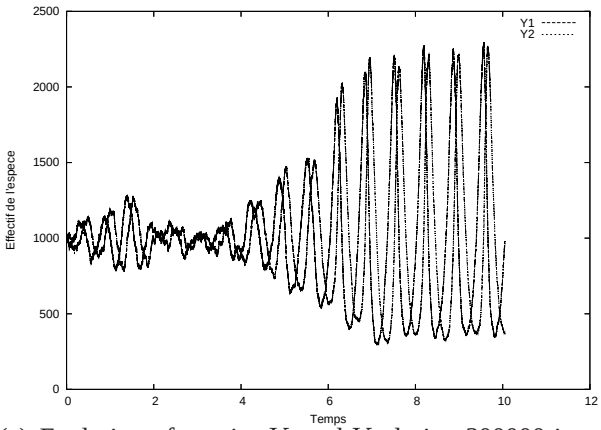
5.3.3 Integration of Gillespie's Algorithm in MGS

MGS integrates the *first reaction method* as an additional stochastic strategy. Let us have a look on how to use it in the example of coupled auto-catalytic equations of Lotka[2]. The reaction rules, together with their own *stochastic reaction constants* are the following:

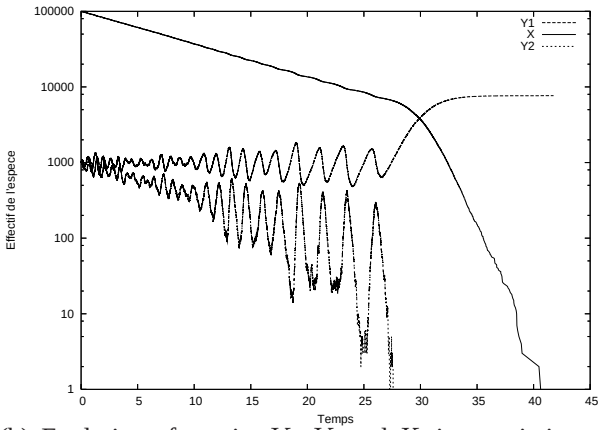
$$X + Y_1 \quad \rightarrow X + Y_1 + Y_1 \qquad \text{with } C = 0.001$$

[1] Computing the stochastic constant is one of the hardest task for who wishes to use SSA for the modeling and simulation of biochemical systems. The interested reader should refer to [DCBD03, ZDCBD03] that describes two experiences in that field.

[2] Studied later by Volterra, after the first world war as a model for the study of an eco-system of prey-predator.



(a) Evolution of species Y_1 and Y_2 during 300000 iterations with an initial state of 10000 `'X`, 1000 `'Y1` and 1000 `'Y2` and where the quantity of food remains constant.



(b) Evolution of species Y_1, Y_2 and X, in a variation of the initial model where the food is no longer an infinite resource, up to the point where the quantity of `'X` goes to zero, with an initial state made of 100000 `'X`, 1000 `'Y1` and 1000 `'Y2` and where the consumed food is not reintroduced.

Figure 1: Results of the simulation, with Gnuplot, of the Lotka equations for two sets of different parameters. The oscillations are coupled with the available food `'X`. When food is no longer there, `Y1` are rapidly overtaking.

$$\begin{array}{lll} Y_1 + Y_2 & \rightarrow Y_2 + Y_2 & \text{with } C = 0.01 \\ Y_2 & \rightarrow Z & \text{with } C = 10 \end{array}$$

First rules express that a given prey Y_1 reproduces by feeding a certain kind of food X that remains constant (X is supposed to be an infinite resource). Second rule states that a given species of predator Y_2 reproduces by feeding from species Y_1; the last rule expresses the disappearance of species Y_2 by natural causes.

A stochastic reaction constant has to be given to each rule, which corresponds to the average probability that a particular combination of molecules will interact in the next infinitesimal time interval dt. The expression of these equations in MGS is straightforward: molecules are represented by symbols and the constants by the `C` parameter of the rules:

```
trans lotka_volterra = {
  'X, 'Y1  ={ C = 0.001 }=> #2 'Y1, 'X;
  'Y1,'Y2  ={ C = 0.01  }=> #2 'Y2;
  'Y2      ={ C = 10    }=> 'Z
}
```

The application of the transformation to an initial state requires to use the `strategy` option whose value has to be set to `'gillespie`:

```
lotka_volterra[iter     = fun _ -> (tau == 5.0),
               strategy = 'gillespie
              ](#1000 'X, #100 'Y1, #100 'Y2, bag:()) ;;
```

Figure 1 gives the result of two simulations of the model.

The initial state is a multi-set because it is required to fulfill Gillespie's hypothesis of having an homogeneous soup; this is verified in the case of a multi-set topology [FMP00]. Each rule application increments MGS system variable `tau` by the computed τ value. During the application

of the transformation, the value of `tau` is available anywhere in the transformation to do any kind of computation. The iterations above stop as soon as `tau` reaches or goes over 5. The use of `iter` allows to have a fine control over the number of applied transformations.

The `C` options in the expression of the rules are used by MGS to compute the new value of τ after each rule application. The algorithm is:

1. for each rule R_μ, according to the path pattern of the rule, the value of the H_μ function is computed (for example, for the first rule $H_\mu = |\texttt{'X}| * |\texttt{'Y1}|$ where $|\texttt{'X}|$ represents the number of occurrences of `'X` in the collection),

2. for each rule, the value $A_\mu = H_\mu * C_\mu$ is computed (C_μ being given by the `C` option of the rule),

3. for each rule, the value $\tau_\mu = \frac{1}{A_\mu} * ln\left(\frac{1}{random()}\right)$ is computed.

Rule R_i with the smallest τ_i value is chosen and *fired one time* and on the collection argument of the transformation, after instantiation of the path pattern. The system variable `tau` is incremented by the τ_i value associated to the rule.

5.4 Application To the Modeling of the Bacteriophage Genetic Switch of the λ Phage

The stochastic simulation of biochemical systems becomes useful when the number of molecules and/or the time interval is very small. We describe in this section the use of MGS for the description and the simulation of a well-known biological process, the *regulation of the λ phage*.

5.4.1 Biological Context

The lambda phage [Pta92] 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 the *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[3], comparatively to the 48502 base pairs of the bacterium genomes) and of two genes[4] (cI and cro[5]) and two promoters[6]. This regulatory region is called the *genetic switch*.

The region of the DNA (see figure 2) of the λ phage is composed of two genes cI and cro coding respectiviley for the proteins CI and CRO. During the transcription, the RNA polymerase[7]

[3] A *base pair* is a couple A-T or G-C on the double helix of strands of DNA.

[4] A *gene* is a part of the DNA that codes for an information (a protein for example).

[5] CRO stands for ***C**ontrol of **R**epressor and **O**ther things*.

[6] A *promoter* is a specific site on the DNA, before the gene, whose role is to allow the initiation of the transcription process.

[7] The RNA polymerase is the enzyme responsible for the transcription.

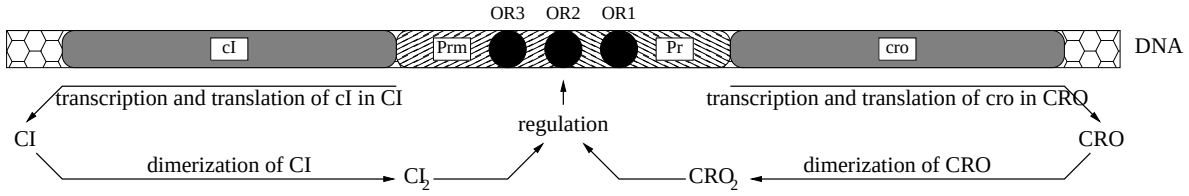


Figure 2: Description (taken from [Mes06]) of the regulatory region of the genetic switch. The DNA has two genes, cI and cro coding respectively for the proteins CI and CRO. These proteins are able to dimerize and bind on the operators OR1, OR2 and OR3. Operator OR1 overlaps Pr, OR3 overlaps Prm and OR2 overlaps both Prm and Pr. Binding of the RNA-polymerase on those promoters (starting the transcription of both genes) is activated or inhibited by the dimers, and is therefore regulated.

binds to the promoters of these genes (respectively Prm and Pr) 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.[8] Operators 0R1, OR2 and OR3 overlap the promoters (see figure 2). The absence or presence of dimers bound to the promoters, eases or hinders the binding of the RNA polymerase, thus regulating the expression of cI and cro. Binding of the dimers to the promoters follows certain rules of affinity:

- CI_2 binds first to OR1, then OR2 and finallly OR3;
- CRO_2 binds first to OR3, then OR2 and finally 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 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 behaviour 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).

5.4.2 The Bio-Chemical Equations

Figure 3 describes the model followed for our modeling. It follows exactly the four rules for the dimers CI_2 and CRO_2 presented above. Nevertheless, it is simplified since it does not consider the set of dimers-operator bonds.

Actually, we have focused on the subset of the bonds that are most-likely to happend. CI_2 binding on OR2, with OR1 free, could have been taken into account; but this state is rare [Mes06]. 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 biochemical equations:

$$\mathrm{CI} \qquad \longrightarrow_{C_0} \qquad . \tag{4}$$

[8] The *operators* are small regions of the DNA that are recognized by molecular complexes inducing a transcriptional regulation.

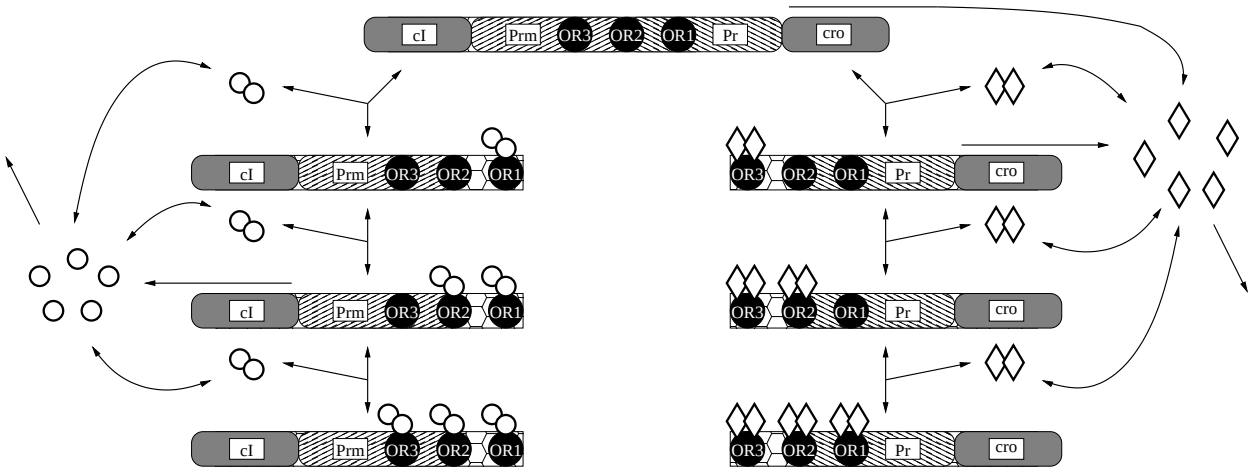


Figure 3: Regulation network of the genetic switch of the λ phage: when CI_2 and CRO_2 bind to the operators, they regulate the expression of genes cI and cro. Monomers CI and CRO are described in circles and diamonds; diamers are made out of monomers. The drawing follows the description of the DNA given in figure 2.

$$
\begin{array}{lll}
2\,\mathrm{CI} & \longleftrightarrow_{C_{12}\,C_{21}} & \mathrm{CI}_2 \quad (5)\\
\mathrm{D} + \mathrm{CI}_2 & \longleftrightarrow_{C_1\,C_{-1}} & \mathrm{D}_1 \quad (6)\\
\mathrm{D}_1 + \mathrm{CI}_2 & \longleftrightarrow_{C_2\,C_{-2}} & \mathrm{D}_2 \quad (7)\\
\mathrm{D}_2 + \mathrm{CI}_2 & \longleftrightarrow_{C_3\,C_{-3}} & \mathrm{D}_3 \quad (8)\\
\mathrm{D}_2 + \mathrm{P} & \longrightarrow_{C_t} & \mathrm{D}_2 + \mathrm{P} + CI \quad (9)\\
\mathrm{CRO} & \longrightarrow_{C'_0} & . \quad (10)\\
2\,\mathrm{CRO} & \longleftrightarrow_{C'_{12}\,C'_{21}} & \mathrm{CRO}_2 \quad (11)\\
\mathrm{D} + \mathrm{CRO}_2 & \longleftrightarrow_{C'_3\,C'_{-3}} & \mathrm{D}'_3 \quad (12)\\
\mathrm{D}'_3 + \mathrm{CRO}_2 & \longleftrightarrow_{C'_2\,C'_{-2}} & \mathrm{D}'_2 \quad (13)\\
\mathrm{D}'_2 + \mathrm{CRO}_2 & \longleftrightarrow_{C'_1\,C'_{-1}} & \mathrm{D}'_1 \quad (14)\\
\mathrm{D} + \mathrm{P} & \longrightarrow_{C'_t} & \mathrm{D}_2 + \mathrm{P} + CRO \quad (15)\\
\mathrm{D}'_3 + \mathrm{P} & \longrightarrow_{C'_t} & \mathrm{D}_2 + \mathrm{P} + CRO \quad (16)
\end{array}
$$

Equations 4 and 10 describe the natural degradation of the CI and CRO monomers. Equations 6–8 and 12–14 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 3). The gene expression is given by reactions 9, 15 and 16, 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.

From our biochemical point-of-view, the bacteria is seen as an homegeneous chemical solution where the reactions described above are taking place.

5.5 Translation in MGS

The translation into an MGS program is straightforward. Each equation is translated into a rewriting rule (or two if the equation corresponds to a reversible reaction) and the value of the C constants, corresponding to the values C_i and C'_i, are determined by biological experiments

(see [KN05]). The set of constants is given below:

$$
\begin{aligned}
lk_0 = k'_0 &= 0.005 \\
k_{12} = k'_{12} &= 0.01 \\
k_{21} = k'_{21} &= 0.25 \\
k_1 = k_2 = k_3 = k'_1 = k'_2 = k'_3 &= 9.768 \\
k_{-1} = k_{-2} &= 15.0 \\
k_{-3} &= 2022.38 \\
k'_{-1} = k'_{-2} &= 245.371 \\
k'_{-3} &= 29.77 \\
k_t = k'_t &= 0.5
\end{aligned}
$$

The MGS program below is the straightforward translation of the biochemical reactions, where the various molecular complexes are abstractly represented by symbols.

```
trans Phage = {
 \\Rules for CI
 'CI            ={ C = 0.005   }=> undef ;
 #2 'CI        ={ C = 0.01    }=> 'CI2 ;
 'CI2           ={ C = 0.25    }=> #2 'CI ;
 'D0, 'CI2      ={ C = 9.768   }=> 'D1 ;
 'D1            ={ C = 15.0    }=> 'D0, 'CI2 ;
 'D1, 'CI2      ={ C = 9.768   }=> 'D2 ;
 'D2            ={ C = 15.0    }=> 'D1, 'CI2 ;
 'D2, 'CI2      ={ C = 9.768   }=> 'D3 ;
 'D3            ={ C = 2022.38 }=> 'D2, 'CI2 ;
 'D2, 'P        ={ C = 0.5     }=> 'D2, 'P, 'CI ;

 \\Rules for CRO
 'CRO           ={ C = 0.005   }=> undef ;
 #2 'CRO       ={ C = 0.01    }=> 'CRO2 ;
 'CRO2          ={ C = 0.25    }=> #2 'CRO ;
 'D0, 'CRO2     ={ C = 9.768   }=> 'D'3 ;
 'D'3           ={ C = 29.77   }=> 'D0, 'CRO2 ;
 'D'3, 'CRO2    ={ C = 9.768   }=> 'D'2 ;
 'D'2           ={ C = 245.371 }=> 'D'3, 'CRO2 ;
 'D'2, 'CRO2    ={ C = 9.768   }=> 'D'1 ;
 'D'1           ={ C = 245.371 }=> 'D'2, 'CRO2 ;
 'D0, 'P        ={ C = 0.5     }=> 'D0, 'P, 'CRO ;
 'D'3, 'P       ={ C = 0.5     }=> 'D'3, 'P, 'CRO };;
```

The bacterium, seen as a chemical solution where the molecules are uniformly distributed following a brownian motion, is represented by an MGS multi-set. The initial state consists of a molecule of DNA with some RNA polymerase. As it has been noticed in the biological reality, the probability for CI to gain over CRO is low. In order to have the simulation evolve in favor of CI, we put some additional CI proteins to the initial state:

```
Bact := 'D :: #10 'P :: #15 'CI :: bag:();;
```

Figure 4 gives the results of eight executions of the MGS program above (the instructions required for the `Gnuplot` output have not been included). The three first pictures show the system in a state that has not yet evolved into a lytic or lysogenic phase; the three following pictures show the system after a switch has occurred in a lytic phase (only molecules of `CRO` remain); the last two pictures show the system in the lysogenic phase (only molecules of `CI` remain).

6 Conclusion

We have shown in this paper the additional expressivity brought by the introduction of a new transformation application strategy on topological collections. A first version if a *standard* rule application strategy that allows to implement a large class of algorithms: the *randomized algorithms*. A second version which implements Gillespie's SSA allows MGS to take into account, in

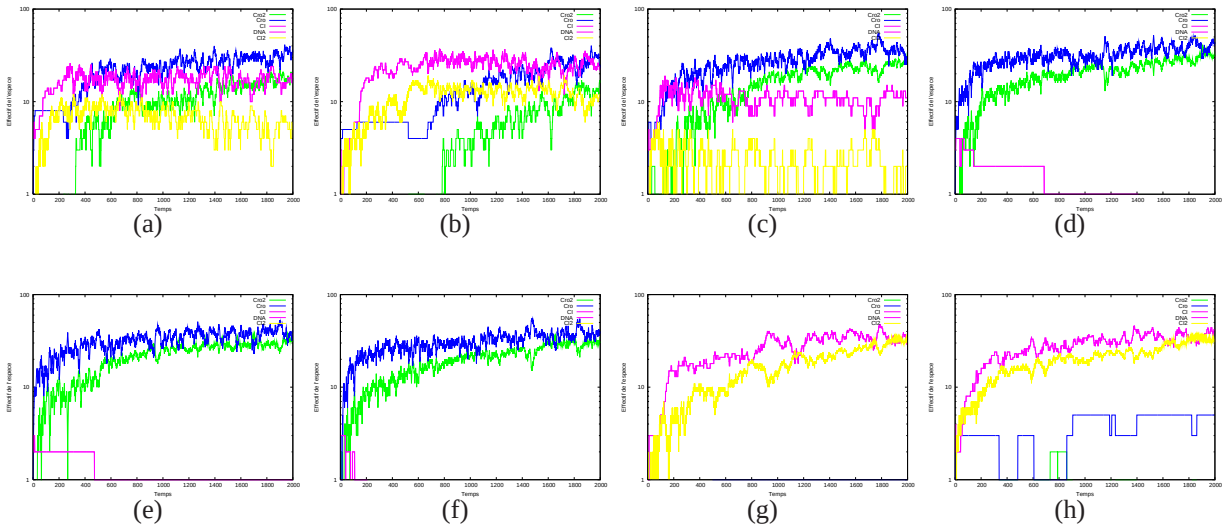


Figure 4: Results of the simulation, plotted with `Gnuplot` of the switch of the $lambda$ phage. The three first show a state of the system where no decision has occurred; the three following pictures show the system in a lytic phase; the two last pictures show the system in a lysogeny phase. All these simulations are of 2000 iterations on an initial state of a multi-set consisting of `(# 3 'CI, 'D0, #10'P)`. The diversity of results perfectly shows the stochastic nature of the switch phenomenon.

a very straightforward manner, the modeling and simulation of stochastic bio-chemical processes. It therefore shows the adequation of rewriting to the modeling and simulation of dynamical systems. The reader should keep in mind that we do not pretend here to compete with special purpose tools (like `E-Cell` [Tak], `Cellware` [DMS+04] or `SBW` [FBS+02]) but only to show that it is possible to quickly develop in MGS realistic models, like the one of the lambda phage, and to make short simulations.

This work has opened many perspectives. The first one concerns the necessary study of the matching and reconstruction algorithms of MGS. Indeed, if simulations on longer time-span and on larger samples (that is on multi-sets of many millions elements) are required, it becomes mandatory to specialize the algorithms (which are generic and the same for *all topological collections* of MGS). A first work in that direction has already been made (integration of the syntactic form `#n 'x` to express the matching of n occurrences of the `'x` constant) but it has to be pursued. Furthermore, it should be necessary to integrate in MGS a more optimized version of SSA [Gil00, Gil01, GB00, PK04].

A second path, more speculative, is about the extension of the work initiated by Gillespie on multi-sets. Indeed, what does it mean to drop the hypothesis of having a spatial homogeneous soup and to apply SSA on topologies that are not complete graphs ? We'd like to study, for example, bio-chemical reactions on non-homogeneous, structured and compartmented spaces (see for example [TAT05, section 2.5][LDVS05]).

References

[BFM01] Jean-Pierre Banâtre, Pascal Fradet, and Daniel Le Métayer. Gamma and the chemical reaction model: Fifteen years after. *Lecture Notes in Computer Science*, 2235:17–44, 2001.

[BK02] Olivier Bournez and Claude Kirchner. Probabilistic rewrite strategies. applications to *ELAN*. *j-LECT-NOTES-COMP-SCI*, 2378:252–??, 2002.

[DCBD03] Xueying De Cock, Katrienand Zhang, Mónica F. Bugallo, and Petar M. Djuric. Stochastic simulation and parameter estimation of first order chemical reactions. In *12th European Signal Processing Conference (EUSIPCO-2004)*, 2003.

[DJ90] N. Dershowitz and J.-P. Jouannaud. *Handbook of Theoretical Computer Science*, volume B, chapter Rewrite systems, pages 244–320. Elsevier Science, 1990.

[DMS+04] Pawan Dhar, Tan Chee Meng, Sandeep Somani, Li Ye, Anand Sairam, Mandar Chitre, Hao Zhu, and Kishore Sakharkar. Cellware-a multi-algorithmic software for computational systems biology. *Bioinformatics*, 20(8):1319–1321, 2004.

[EKL+02] S. Eker, M. Knapp, K. Laderoute, P. Lincoln, J. Meseguer, and J. Sonmez. Pathway logic: Symbolic analysis of biological signaling. In *Proceedings of the Pacific Symposium on Biocomputing*, pages 400–412, January 2002.

[FBS+02] A. Finney, H. Bolouri, H. M. Sauro, J. Doyle, and M. Hucka. The erato systems biology workbench: Enabling interaction and exchange between software tools for computational biology, September 27 2002.

[FMP00] Michael Fisher, Grant Malcolm, and Raymond Paton. Spatio-logical processes in intracellular signalling. *BioSystems*, 55:83–92, 2000.

[GB00] Michael A. Gibson and Jehoshua Bruck. Efficient exact stochastic simulation of chemical systems with many species and many channels. *J. Chem. Physics*, 104:1876–1889, 2000.

[GGMP02] J.-L. Giavitto, C. Godin, O. Michel, and P. Prusinkiewicz. *Modelling and Simulation of biological processes in the context of genomics*, chapter "Computational Models for Integrative and Developmental Biology". Hermes, July 2002. Also republished as an high-level course in the proceedings of the Dieppe spring school on "Modelling and simumation of biological processes in the context of genomics", 12-17 may 2003, Dieppes, France.

[Gia03] J.-L. Giavitto. Invited talk: Topological collections, transformations and their application to the modeling and the simulation of dynamical systems. In *Rewriting Technics and Applications (RTA'03)*, volume LNCS 2706 of *LNCS*, pages 208 – 233, Valencia, June 2003. Springer.

[Gil77] Daniel T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *J. Phys. Chem.*, 81(25):2340–2361, 1977.

[Gil00] Daniel T. Gillespie. Chemical langevin equation. *Journal of chemical physics*, 113:297–306, 2000.

[Gil01] Daniel T. Gillespie. Approximate accelerated stochastic simulation of chemically reacting systems. *Journal of chemical physics*, 115:1716–1733, 2001.

[GM02] J.-L. Giavitto and O. Michel. Data structure as topological spaces. In *Proceedings of the 3nd International Conference on Unconventional Models of Computation UMC02*, volume 2509, pages 137–150, Himeji, Japan, October 2002. Lecture Notes in Computer Science.

[KN05] Céline Kuttler and Joachim Niehren. Gene regulation in the pi calculus : simulation coperativity at the lambda switch. *Transactions on Computational Systems Biology*, 2005.

[LDVS05] Caroline Lemerle, Barbara Di Ventura, and Luis Serrano. Space as the final frontier in stochastic simulations of biological systems. *FEBS Letters*, 579:1789–1794, 2005.

[LJ92] A. Lindenmayer and H. Jürgensen. Grammars of development: discrete-state models for growth, differentiation, and gene expression in modular organisms. In G. Ronzenberg and A. Salomaa, editors, *Lindenmayer Systems, Impacts on Theoretical Computer Science, Computer Graphics and Developmental Biology*, pages 3–21. Springer Verlag, February 1992.

[Man01] Vincenzo Manca. Logical string rewriting. *Theoretical Computer Science*, 264:25–51, 2001.

[Mes06] Denis Mestivier. Modélisation déterministe de réseaux de gènes : le répresseur λ du bactériophage λ. Notes to the spring school "Modélisation de systèmes biologiques complexes dans le contexte de la génomique" on various modeling of the λ phage, Bordeaux, april 2006.

[MSD04] Tan Chee Meng, Sandeep Somani, and Pawan Dhar. Modeling and simulation of biological systems with stochasticity. *In Silico Biology*, 4, 2004.

[Pău01] G. Păun. From cells to computers: Computing with membranes (P systems). *Biosystems*, 59(3):139–158, March 2001.

[PK04] Jacek Puchalka and Andrzej Kierzek. Bridging the gap between stochastic and deterministic regimes in the kinetic simulations of the biochemical reaction networks. *Biophysical Journal*, March 2004.

[Pta92] Mark Ptashne. *A genetic switch : phage lambda and highet organisms*. 1992.

[RS92] Grzegorz Rozenberg and Arto Salomaa. *Lindenmayer Systems*. Springer, Berlin, 1992.

[Tak] Kouichi Takahashi. E-cell, available at `http://ecell.sourceforge.net/`.

[TAT05] Kouichi Takahashi, Satya Nanda Vel Arjunan, and Masaru Tomita. Space in systems biology of signaling pathways–towards intracellular molecular crowding in silico. *FEBS Letters*, 579:1783–1788, 2005.

[ZDCBD03] Xueying Zhang, Katrien De Cock, Mónica F. Bugallo, and Petar M. Djuric. Stochastic simulation and parameter estimation of enzyme reaction models. In *IEEE Workshop on Statistical Signal Processing*, 2003.