LaMI

Laboratoire de Méthodes Informatiques

Modeling Multi-Valued Genetic Regulatory Networks Using High-Level Petri Nets

J.-P. Comet, H. Klaudel, S. Liauzu

e-mail : {comet, klaudel}@lami.univ-evry.fr

Rapport de Recherche n° 107-2004

Décembre 2004

Modeling Multi-Valued Genetic Regulatory Networks Using High-Level Petri Nets

Jean-Paul Comet[1], Hanna Klaudel[1], and Stéphane Liauzu[1]

LaMI, UMR CNRS 8042, Université d'Evry-Val d'Essonne,
Boulevard François Mitterrand, 91025 Evry Cedex, France.
tel : (33) 1.60.87.39.17 - fax : (33) 1.60.87.37.89
`comet@lami.univ-evry.fr`

keywords: HLPN, modeling of regulatory networks, model checking.

Abstract Regulatory networks are at the core of all biological functions from bio-chemical pathways to gene regulation and cell communication processes. Because of the complexity of the interweaving retroactions, the overall behavior is difficult to grasp and the development of formal methods is needed in order to confront the supposed properties of the biological system to the model. We revisit here the tremendous work of R. Thomas and show that its binary and also its multi-valued approach can be expressed in a unified way with high-level Petri nets.

A compact modeling of genetic networks is proposed in which the tokens represent gene's expression levels and their dynamical behavior depends on a certain number of biological parameters. This allows us to take advantage of techniques and tools in the field of high-level Petri nets. A developed prototype lets a biologist to verify systematically the coherence of the system under various hypotheses. These hypotheses are translated into temporal logic formulae and the model-checking techniques are used to retain only the models whose behavior is coherent with the biological knowledge.

1   Introduction

To elucidate the principles that govern biological complexity, computer modeling has to overcome *ad hoc* explanations in order to make emerge novel and abstract concepts[10]. Computational *system biology*[31] tries to establish methods and techniques that enable us to understand biological systems as systems, including their robustness, design and manipulation[13,9]. It means to understand: the structure of the system, such as gene/metabolic/signal transduction networks, the dynamics of such systems, methods to control, design and modify systems in order to cope with desired properties[14].

Biological regulatory networks place the discussion at a biological level instead of a biochemical one, that allows one to study behaviors more abstractly. They model interactions between biological entities, often macromolecules or genes. They are statically represented by oriented graphs, where vertices abstract the biological entities and arcs their interactions. Moreover, at a given stage, each vertex has a numerical value to describe the level of concentration of the corresponding entity. The dynamics correspond to the evolutions of these concentration levels and can be represented, for instance, using differential equation systems.

R. Thomas introduced in the 70's a boolean approach for regulatory networks to capture the qualitative nature of the dynamics. He proved its usefulness in the context of immunity in bacteriophages[26,25]. Later on, he generalized it to multi-valued levels of concentration, so called "generalized logical" approach. Moreover, the vertices of R. Thomas' regulatory networks are abstracted into "variables" allowing the cohabitation of heterogeneous information (e.g., adding environmental variables to genetic ones).
The R. Thomas boolean approach has been justified as a discretization of the continuous differential equation system[21], then has been confronted to the more classical analysis in terms of differential equations[11]. Taking into account "singular states", Thomas and Snoussi showed that all steady states can be found *via* the discrete approach[23]. More recently Thomas and Kaufman have shown that the discrete description provides a qualitative fit of the differential equations with a small number of possible combinations of values for the parameters[27].
A direct or indirect influence of a gene on itself corresponds to a closed oriented path which constitutes a feedback circuit. Feedback circuits are fundamental because they decide the existence of steady states of the dynamics: it has been stated then proved [17,22,6,24] that at least one positive regulatory circuit is necessary to generate multistationarity whereas at least one negative circuit is necessary to obtain a homeostasis or a stable oscillatory behavior[28].

These static properties (number of stationary states) can be reinforced by introducing some properties on the dynamics of the system extracted from the biological knowledge or hypotheses. It becomes necessary to construct models which are coherent not only with the previous static conditions but also with the dynamical ones. Formal methods from computer science should be able to help modeler to automatically perform this verification. In [19,2] the machinery of formal methods is used to revisit R. Thomas' regulatory networks: all possible state graphs are generated and model checkers help to select those which satisfy the temporal properties. All this approach is based on the semantics of the regulatory graph, *i.e.*, its dynamics, which has to be computed before. The state explosion phenomenon in the transition graph limits the readability of these modelings and the possible extensions like, for instance, the introduction of delays for transitions. These observations motivated our interest for applying in this context the Petri net theory.

In this article we present a modeling of the R. Thomas' regulatory networks in terms of high-level Petri nets. To ensure the adequation between both formalisms, we first present formally the biological regulatory graphs which describe the interactions between biological entities, the parameters which pilot the behaviors of the system and the associated dynamics (section 2). Then, after a brief introduction to the high-level Petri nets, a modeling of regulatory graphs is introduced in section 3. In section 4 we show how model-checking can be used to determine which models have to be considered. Sections 5 and 6 illustrate our approach with the LTL model-checker Maria, and describe our prototype for computer aided modeling in the context of genetic regulatory networks. The last section 7 discusses the results and the possible extensions.

2   Genetic regulatory graphs and associated semantics

In this section we formally define the biological regulatory networks. We first introduce the biological regulatory graph which represents interactions between biological entities. A vertex represents a variable (which can abstract a gene or its protein for instance). The interactions between genes are represented by arcs and each arc is labelled with the sign of the interaction: "$-$" for an inhibition and "$+$" for an activation.

The interactions have almost always a sigmoid nature (see Fig. 1): for each positive interaction of i on j, if variable i has a concentration below a certain threshold (defined by the inflection point of the sigmoid), then the variable j is not influenced by i, and above this threshold, j is controlled by i in a same way for any concentration of i above the threshold. Figure 1 assumes that the variable i acts positively on j and negatively on j'; each curve represents the concentration of the target after a sufficient delay for the regulator i to act on it. Three regions are relevant: the first one corresponds to the situation where i does not act neither on j nor on j', the second to the situation where i acts only on j, and the last one corresponds to the situation where i acts on both targets j and j'. This justifies the discretization of the concentration of i into three abstract levels (0, 1 and 2) corresponding to the previous regions and constituting the only relevant information from a qualitative point of view.
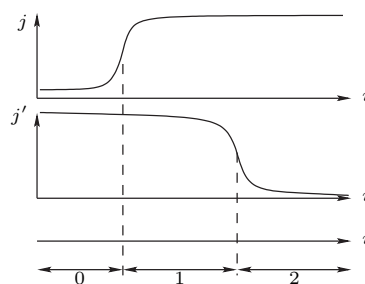
Figure 1. The discretization is supervised by the thresholds of actions on targets.

For a variable which has τ targets (itself possibly included), $\tau + 1$ abstract levels have to be considered if all thresholds are distinct, but possibly less in the case where two or more thresholds are equal.

Definition 1. *A biological regulatory graph is a labelled directed graph $G = (V, E)$ where each vertex i of V, called a variable, is provided with a boundary $\beta_i \in \mathbb{N}$ less or equal to the out-degree of i except if the out-degree is 0 in which case $\beta_i = 1$. Each edge $(i \longrightarrow j)$ is labelled with a pair (t_{ij}, ϵ_{ij}) where t_{ij}, called the threshold of the interaction, is a natural number between 1 and β_i and $\epsilon_{ij} \in \{-, +\}$ is its sign.*

The threshold t_{ij} of a positive interaction $(i \longrightarrow j)$ determines the conditions which allow the variable i to stimulate j: if variable i has an abstract level below t_{ij}, the interaction is not active and j is not stimulated, otherwise, it is. For negative interactions, the conditions are symmetrical.

At a given stage, each variable of a regulatory graph has a unique abstract concentration level. So, the *state* of the system may be represented as a vector of concentration levels n_i of each variable i.

Definition 2. *A state of a biological graph is a tuple $n = (n_1, n_2, ..., n_p)$, where p is the number of variables and n_i is the abstract concentration level of the variable i with $n_i \in \mathbb{N}$ and $n_i \leq \beta_i$.*
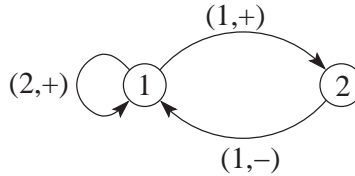
Figure 2. Regulatory graph for mucus production in *pseudomonas aeruginosa*.

Running example: We take as running example the mucus production in *Pseudomonas aeruginosa*. These bacteria are commonly present in the environment and secrete mucus only in lungs affected by cystic fibrosis. As this mucus increases the respiratory deficiency of the patient, it is the major cause of mortality. The regulatory network which controls the mucus production has been elucidated [8]. The main regulator for the mucus production, AlgU, supervises an operon which is made of 4 genes among which one codes for a protein that is a repressor of AlgU. Moreover AlgU favors its own synthesis. The regulatory network can then be simplified into the regulatory graph of Fig. 2, where variable 1 represents AlgU, and variable 2 its repressor. The order of thresholds t_{12} and t_{11} is not deductible from biological knowledge and in fact both orderings have to be considered [1]. Figure 2 assumes that $t_{12} < t_{11}$. Variable 1 can take three different abstract concentration levels: 0,1 or 2. Similarly, variable 2 which is an inhibitor of variable 1, can take two levels: 0 and 1. Consequently, there are 6 possible states $(0,0)$, $(0,1)$, $(1,0)$, $(1,1)$, $(2,0)$ and $(2,1)$.

Up to now, the discretization of continuous concentrations into the abstract levels allows us to define when a regulator has an influence on its targets, but we need to determine towards which abstract levels the targets are attracted. To answer this question, one has to know for each state n which regulators are actually effective on the considered target i, in other words, which are the "*resources*" of i in the state n.

Definition 3. *Given a biological regulatory graph $G = (V, E)$ and a possible abstract state $n = (n_1, n_2, ..., n_p)$, the set of* resources *of a variable i is the set*

$$R_i(n) = \left\{ j \in V \mid (j \overset{(t_{ji}, \epsilon_{ji})}{\longrightarrow} i) \in E, (((n_j \geq t_{ji}) \wedge (\epsilon_{ji} = +)) \vee ((n_j < t_{ji}) \wedge (\epsilon_{ji} = -))) \right\}.$$

$R_i(n)$ contains the activators of i whose abstract level is above the threshold and the inhibitors of i whose abstract level is below the threshold. A resource is either the presence of an activator or the *absence* of an inhibitor.

It remains to define towards which abstract level a variable i is attracted when its resources are ω. We call this level the *attractor* of i for resources ω and denote it by $k_{i,\omega}$.

Having the values of attractors, it is straightforward to define the *synchronous state graph* of the biological network. For a state $n = (n_1, n_2, ..., n_p)$ where p is the number of variables, we take for each variable i, the attractor $k_{i,R_i(n)}$, where $R_i(n)$ is the set of resources of variable i when the system is in the state n. The synchronous state graph is obtained by setting for the unique possible next state the state towards which the system is attracted $n' = (k_{1,R_1(n)}, k_{2,R_2(n)}, ...k_{p,R_p(n)})$. However, this definition has at least two drawbacks:

- first, it allows two or more variables to change simultaneously, while the probability that several variables pass through their respective thresholds at the same time is negligible *in vivo*. But we do not know which one will pass through its threshold first;

---

[1] Two regulatory graphs should be considered; for simplicity we explain the concepts on one of them.

– and second, it does not prevent that a variable passes directly two or more thresholds, which is not realistic because an abstract concentration level should evolve gradually.

Then, an improved semantics is defined in terms of an *asynchronous state graph* which:

– replaces each diagonal transition of the synchronous state graph (transition with 2 or more variables changing their concentration levels) by the collection of transitions each of them modifying only one of the involved variables,
– replaces a transition of length greater or equal to 2 (which passes two or more thresholds at once) by a transition of length 1 in the same direction.

We then introduce the evolution operator which allows us to define formally the asynchronous state graph.

Definition 4. *Let $x, k \in \mathbb{N}$. The evolution operator* $\blacktriangleright$ *is defined as follows:*

$$x \blacktriangleright k = \begin{cases} x - 1 & \text{iff } x > k \\ x + 1 & \text{iff } x < k \\ x & \text{otherwise.} \end{cases}$$

Definition 5. *Let $G = (V, E)$ a regulatory graph with p variables. Its asynchronous state graph is defined as follows :*

– *the set of vertices is the set of states $\Pi_{i \in V}[0, \beta_i] = \{(n_1, ..., n_p) \in \mathbb{N}^p \mid \forall i \in [1, p], n_i \leq \beta_i\}$*
– *there is a transition from the state $n = (n_1, ..., n_p)$ to $m = (m_1, ..., m_p)$ iff*

$$\begin{cases} \exists! \ i \ such \ that \ m_i \neq n_i \\ m_i = (n_i \blacktriangleright k_{i, R_i(n)}) \end{cases} \quad or \quad \begin{cases} m = n \\ \forall i \in [1, p], \ n_i = (n_i \blacktriangleright k_{i, R_i(n)}) \end{cases}$$

From the R. Thomas modeling towards a modeling with Petri nets. A natural way to define an equivalent Petri net, *i.e.*, whose dynamics is exactly the same as the asynchronous approach of R. Thomas, consists in introducing inhibitor arcs [4]. A place is associated to each gene, and a transition to each parameter. The input places of the transition corresponding to $k_{i,\omega}$ are all the predecessors of i, with the places of ω connected to the transition by standard arcs and the predecessors of i not included in ω connected to the transition by inhibitor arcs.

In such a modeling, there are as many transitions as parameters. For a non trivial regulatory graph, it leads to a Petri net which is difficult to interpret because of its size. Moreover, using inhibitor arcs changes the complexity of the Petri net class and may lead to introduce difficulties in proofs of some properties.

3 Modeling with high-level Petri nets

3.1 Introduction to high-level Petri nets

Multiset notations. A *multiset* over a set E is a function $\mu : E \to \mathbb{N}$; μ is finite if $\{e \in E \mid \mu(e) > 0\}$ is finite. We denote by $\mathcal{M}_f(E)$ the set of finite multisets over E.

Definition 6. *A (low-level) Petri net is a triple $L = (S, T, W)$, where S is a set of* places, *T is a set of* transitions, *such that $S \cap T = \emptyset$ and $W : (S \times T) \cup (T \times S) \to \mathbb{N}$ is a* weight function.

A *marking* of a Petri net (S, T, W) is a mapping $M : S \to \mathbb{N}$, which associates to each place a natural number of *tokens*. The behavior of such a net, starting from an arbitrary *initial marking*, is determined by the usual definitions for place/transition Petri nets.

High-level nets that we consider can be viewed as simple abbreviations of the low-level ones.

Let *Val* and *Var* be fixed but suitably large disjoint sets of *values* and *variables*, respectively. The set of all well-formed *predicates* built from the sets *Val*, *Var* and a suitable set of operators is denoted by Pr.

Definition 7. *A high-level Petri net, HLPN for short, is a triple (S, T, ι), where S and T are disjoint sets of* places *and* transitions, *and ι is an inscription function with domain $S \cup (S \times T) \cup (T \times S) \cup T$ such that:*

– *for every place $s \in S$, $\iota(s) \subseteq Val$, is the* type *of s, i.e., the set of possible values the place may carry;*
– *for every transition $t \in T$, $\iota(t)$ is the* guard *of t, i.e., a predicate from* Pr;

– *for every arc $(s,t) \in (S \times T) : \iota((s,t)) \in \mathcal{M}_f(Val \cup Var)$ is a multi-set of variables or values (analogously for arcs $(t,s) \in (T \times S)$). The inscriptions $\iota((s,t))$ and $\iota((t,s))$ will generally be abbreviated as $\iota(s,t)$ and $\iota(t,s)$, respectively.*

A *marking* of a high-level Petri net (S,T,ι) is a mapping $M : S \to \mathcal{M}_f(Val)$ which associates to each place $s \in S$ a multi-set of values from its type $\iota(s)$. A *binding* is a mapping $\sigma : Var \to Val$ and an *evaluation* of an entity η (which can be a variable, a vector or a (multi-)set of variables, etc.) through σ is defined as usual and denoted by $\eta[\sigma]$.

The transition rule specifies the circumstances under which a marking M' is reachable from a marking M. A transition t is *activated* at a marking M if there is an *enabling binding* σ for variables in the inscription of t (making the guard true) and in inscriptions of arcs around t such that $\forall s \in S : \iota(s,t)[\sigma] \leq M(s)$, *i.e.*, there are enough tokens of each type to satisfy the required flow. The effect of an occurrence of t, under an enabling binding σ, is to remove tokens from its input places and to add tokens to its output places, according to the evaluation of arcs' annotations under σ.
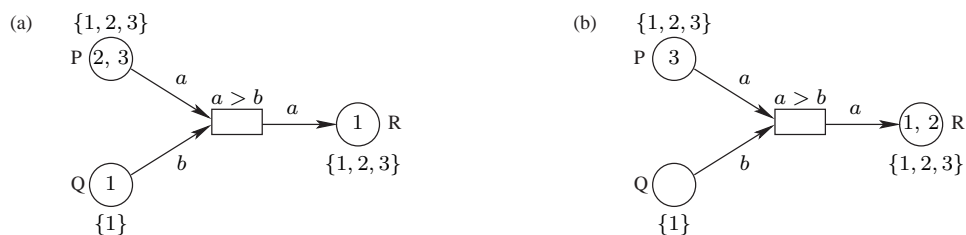
Figure 3. A simple marked high-level Petri net before (a) and after (b) the firing of the transition.

For the example of Fig. 3-(a), the marking is given by: $M(P) = \{2, 3\}$, $M(Q) = \{1\}$, $M(R) = \{1\}$. Bindings are $\sigma_1 = \begin{cases} a \to 1 \\ b \to 1 \end{cases}$, $\sigma_2 = \begin{cases} a \to 2 \\ b \to 1 \end{cases}$ and $\sigma_3 = \begin{cases} a \to 3 \\ b \to 1 \end{cases}$. Only σ_2 and σ_3 are enabling (σ_1 does not make the guard true). At the marking M, the transition t is activated for both σ_2 and σ_3. Figure 3-(b) shows the new marking if σ_2 is chosen.

An important property of high-level nets is that they may be unfolded to low-level ones, which may be helpful when using various verification tools. The unfolding operation associates a low-level net $\mathcal{U}(N)$ with every high-level net N, as well as a marking $\mathcal{U}(M)$ of $\mathcal{U}(N)$ with every marking M of N.

Definition 8. *Let $N = (S,T,\iota)$; then $\mathcal{U}(N) = (\mathcal{U}(S), \mathcal{U}(T), W)$ is defined as follows:*

– $\mathcal{U}(S) = \{s_v \mid s \in S \text{ and } v \in \iota(s)\}$;
– $\mathcal{U}(T) = \{t_\sigma \mid t \in T \text{ and } \sigma \text{ is an enabling binding of } t\}$;
– $W(s_v, t_\sigma) = \sum\limits_{\substack{a \in \iota(s,t) \\ a[\sigma] = v}} \iota(s,t)(a)$, *where $\iota(s,t)(a)$ is the number of occurrences of a in the multiset $\iota(s,t)$, and*

analogously for $W(t_\sigma, s_v)$.

Let M be a marking of N. The unfolding of a marking $\mathcal{U}(M)$ is defined as follows: for every place $s_v \in \mathcal{U}(S)$, $(\mathcal{U}(M))(s_v) = (M(s))(v)$ where $(M(s))(v)$ is the number of occurrences of v in the marking $M(s)$ of s. Thus, each elementary place $s_v \in \mathcal{U}(S)$ contains as many tokens as the number of occurrences of v in the marking $M(s)$. Figure 4 presents the Petri net obtained by unfolding of the high-level Petri net of Fig. 3-(a).

3.2 Modeling of genetic regulatory networks

We can represent a regulatory network by a high-level Petri net which has a unique transition and as many places as genes in the regulatory graph. Each place corresponds to a gene i and carries one token: its abstract concentration level n_i. The marking of this net corresponds thus to an abstract state $n = \{n_1, \dots, n_p\}$.
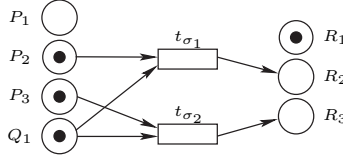
Figure 4. Unfolded Petri net of high-level Petri net of Fig. 3-(a).

The transition can fire at a marking n leading to the marking n' if its guard

$$\text{asyn\_guard}(n, n') = \left\{ \begin{array}{c} \Big(\exists i \in [1, p], \ (n_i \neq k_{i, R_i(n)}) \wedge (n'_i = n_i \blacktriangleright k_{i, R_i(n)}) \wedge (\forall j \neq i, n'_j = n_j) \Big) \\ \bigvee \\ \Big(\forall i \in [1, p], \ (n_i = k_{i, R_i(n)}) \wedge (n'_i = n_i) \Big) \end{array} \right\}$$

is true. This guard translates directly the asynchronous semantics of R. Thomas. Indeed, the marking represents a stable steady state for the asynchronous semantics, when the attractors $k_{i, R_i(n)}$ equal the current concentrations n_i for all variables $i \in V$. The guard gives in this case the same marking for the next one. If the marking does not correspond to any stable steady state, then some variables are not equal to their attractors. Following Thomas' semantics, a possible next marking is a marking for which the variables do not change unless one of them which changes (plus or minus one) in the direction of its attractor.

Figure 5 represents the high-level Petri net for the running example. Let us consider furthermore that the attractors of the running example are given: $k_{1, \{\}} = 0$, $k_{1, \{1\}} = 2$, $k_{1, \{2\}} = 2$, $k_{1, \{1,2\}} = 2$, $k_{2, \{\}} = 0$ and $k_{2, \{1\}} = 1$. Let us choose the state $(1, 0)$ as the initial marking. The resources of variable 1 are $R_1((1, 0)) = \{2\}$ since both inhibitor and activator are absent, and the resources of variable 2 are $R_2((1, 0)) = \{1\}$ since the activator is present. Both variables are attracted towards values different from their current values. Two possible new markings are possible: $(2, 0)$ if the variable 1 evolves first, and $(1, 1)$ otherwise. Globally, the sequence of markings during an execution corresponds to a particular possible path.
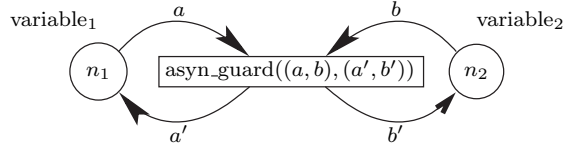
Figure 5. HLPN modeling a genetic regulatory network with 2 genes. Each place abstracts a gene and carries one token: its abstract concentration level. The guard expresses the relationship between the current marking $(n_{\text{variable}_1}, n_{\text{variable}_2})$ and the possible updated marking $(n'_{\text{variable}_1}, n'_{\text{variable}_2})$.

Thus, for implementation reasons, we propose a more compact modeling which is in fact a folding of the previous one. It consists in a unique place called *cell*, which abstracts the cell in which each token represents a specific gene and its expression level. A particular structured type *gene* is needed: it represents a couple $(gene, level)$ where *level* is the abstract level of the variable *gene*. The arc inscriptions are also modified: the input one becomes $\{(1, a_1), \ldots, (p, a_p)\}$, with $a_i \neq a_j$ for $i \neq j$, and the output one becomes $\{(1, a'_1), \ldots, (p, a'_p)\}$, with $a'_i \neq a'_j$ for $i \neq j$. The state of the system is now represented by the set of tokens present in the unique place. Let be $\nu = \{(i, n_i), i \in [1, p]\}$ and $\nu' = \{(i, n'_i), i \in [1, p]\}$. The guard of the unique transition can then be written:

$$\text{asyn\_guard}(\nu, \nu') = \left\{ \begin{array}{c} \Big(\exists i \in [1, p], \ (n_i \neq k_{i, R_i(n)}) \wedge (n'_i = n_i \blacktriangleright k_{i, R_i(n)}) \wedge (\forall j \neq i, n'_j = n_j) \Big) \\ \bigvee \\ \Big(\forall i \in [1, p], \ (n_i = k_{i, R_i(n)}) \wedge (n'_i = n_i) \Big) \end{array} \right\}$$

Then, for every biological/genetic regulatory network, the high-level Petri net has a unique place and a unique transition. The only things that distinguish different nets are the number and the type of tokens, and

6

the parameters $k_{i,R_i(n)}$ which are present in the guard. This property is very useful in practice because it allows us to have a generic description (and so a generic source file) and to generate automatically the high-level Petri net for an arbitrary genetic regulatory network.

As an illustration, Fig. 6 presents the high-level Petri net for the regulatory graph of our running example.
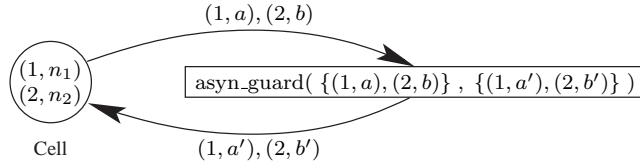
Figure 6. Second HLPN modeling of a genetic regulatory network with 2 genes.

4 Determination of valuable models

Parameters $\{k_{i,\omega}, i \in V$ and $\omega \subseteq G^{-1}(i)\}$ play a major role on the dynamics of the regulatory networks. In fact, they code entirely for their dynamics. For this reason, a set of particular parameter values is called a *model* of the system. Unfortunately, most often they cannot be deduced from experiments and the modeler has to consider the different possible values of parameters. For a given regulatory graph, the number of different sets of parameter values, *i.e.*, the number of models, is exponential with the number of predecessors of each variable. More precisely, this number is equal to $\prod_{i \in V} 2^{|G^{-1}(i)|}$, where $|G^{-1}(i)|$ denotes the number of predecessors of variable i in the regulatory graph. This enormous number prevents us to construct all possible dynamics of the regulatory network and to let biologist select only interesting ones. Some interesting results can therefore be used for reducing the number of models to be considered. In [23] the modeling of Thomas is seen as a discretization of a particular class of continuous differential equation systems, and parameters $k_{i,\omega}$ reflect a discretization of sums of ratios of positive constants. In such a case, the parameters in the set $\{k_{i,\omega} \mid i \in V$ and $\omega \subseteq G^{-1}(i)\}$ have to satisfy the following constraints

$$k_{i,\emptyset} = 0 \text{ and } \omega \subset \omega' \implies k_{i,\omega} \leq k_{i,\omega'}.$$

Nevertheless, it is possible to enlarge the set of models which can be described by this discrete formalism leading to slacken these previous constraints which cannot be added arbitrarily.

Then, the modeling activity focuses on the determination of a suitable class of models, *i.e.*, parameter values that lead to a dynamics which is coherent with the experimental knowledge. Biological knowledge about the behavior can then be used as indirect criteria constraining the set of models. For instance, multistationarity or homeostasis which are experimentally observable, are useful to reduce the set of parameter values. This relies on notions of positive/negative functional circuits and of their characteristic states [29].

But, it is possible to take into account not only such conditions as homeostasis and multistationarity, but also particular temporal properties extracted from biological knowledge or hypotheses[2]. These knowledge or hypotheses may be translated into a formal temporal language as LTL (Linear Temporal Logic) or CTL (Computational Tree Logic) in order to be manipulated automatically by computer. The coherence of the model may then be verified automatically by model checking.

For the running example, the presence of a positive circuit in the regulatory graph of *Pseudomonas aeruginosa* makes possible a dynamics with two stable steady states which would correspond, from a biological point of view, to an epigenetic switch (stable change of phenotype without mutation) from the non-mucoid state (the bacterium does not produce mucus) to the mucoid state (it does). In other words, the question is to find at least one model of the bacteria, which is compatible with the known biological results and which has a multi-stationarity where one stable steady state produces mucus and the other one does not. It turns out that the mucus production is triggered by a high level of variable 1. Then, a recurrent production of mucus is equivalent to the fact that the concentration level n_1 of variable 1 is repeatedly equal to 2. So, the stationarity of the mucoid state can be expressed as:

$$(n_1 = 2) \implies XF(n_1 = 2) \tag{1}$$

where $XF\varphi$ means that φ will be satisfied in the future. Moreover, we know that the bacteria never produce mucus by them-selves when starting from a basal state (second stable steady state):

$$(n_1 = 0) \implies G(\neg(n_1 = 2)). \tag{2}$$

However, even if it may be easy to express in this way particular properties of a given system, proposing a general method allowing to express formally a biological hypothesis remains a difficult open problem.

Nevertheless the formal properties being given, it becomes possible to design a general approach for selecting models of a given regulatory graph which lead to a dynamics coherent with the considered temporal properties. The stages are as follows (see Fig. 7):

1. Design the regulatory graph corresponding to the biological system. Because of the partial information on the system, the biological regulatory network can be represented by several regulatory graphs. For this step it is not necessary to describe all details of the system but only the key concepts. In particular, positive and negative circuits have to be present as well as their intertwined interactions.
2. Design the temporal logic formulae which express formally dynamical knowledge or hypotheses that biologist want to take into account. This step should be performed with care in order not to forget important information.
3. Generate, from the regulatory graphs, all potential models (set of parameter values).
4. Construct the high-level Petri net for each of them.
5. For each Petri net, call the model checker for verifying if the temporal properties are satisfied. Return only the models and associated state graphs which satisfy the formulae.
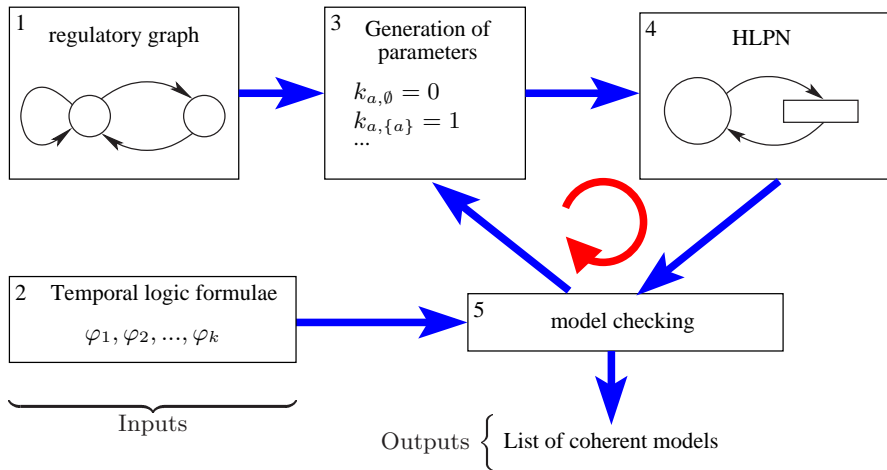
Figure 7. Computer aided modeling approach. The first step consists in designing the regulatory graph (1) and the temporal logic formulae expressing temporal properties of the underlying biological system (2). Generate a potential model (3), then construct the HLPN (4). The model checker is then called for verifying the temporal properties (5). If the model satisfies the properties, the model is stored. Back to (3) for generating another potential model.

5   Implementation with Maria

For implementing, we chosen the model checker MARIA [16] (Modular Reachability Analyzer for Algebraic System Nets) which takes as input a high-level Petri net described in a particular language (see Fig. 13 in the appendix for an example), a LTL formula and performs the checking.

The execution of regulatory networks depends on the initial state and the checkings have to be done from each possible initial state unless the formulae specify the opposite. Maria makes possible to specify a unique initial marking. Then, it is necessary to include a mechanism allowing to take into account all interesting initial markings. In order to do this, we add two places and a new transition (see Fig. 8).

– The first place, *Inits*, contains all initial states that have to be considered for the current checking. If no restriction has to be taken into consideration, all states are added in this place.

– The second place, $Bool_1$, contains only a boolean token initially set to false, which means that the initial state has to be chosen. Then the token becomes true.
– The transition G_1 takes an initial marking from $Inits$ and the value false from $Bool_1$ and generates the corresponding tokens in the place $Cell$, the value true in $Bool_1$ and gives back the initial marking to $Inits$.
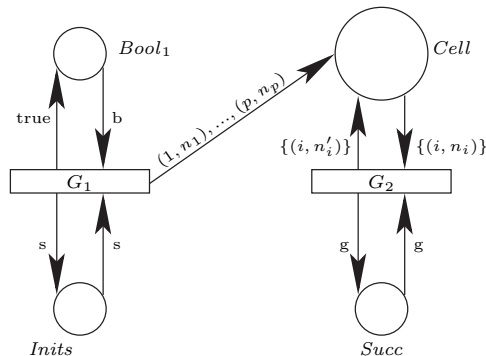
Figure 8. Implementation in MARIA of the HLPN. The guard $G_1 = (b = false) \wedge (s = ((1, n_1), (2, n_2), ..., (p, n_p)))$ is fired only once for the initialization step. G_2 is the asynchronous guard completed with the number of the gene which evolves (see text).

At the beginning, the place $Cell$ is empty. The transition G_2 is not enabled but the transition G_1 can fire. It chooses a possible initial state and generates the corresponding tokens in the place $Cell$. The token of the place $Bool_1$ becomes true and the transition G_1 is disabled. The fact that G_1 gives back to $Inits$ the chosen initial state prevents to generate supplementary states of the Petri net which do not correspond to anything in the regulatory network.

The high-level Petri net which models an asynchronous state graph contains some non-determinism due to the fact that several successor states may be reachable from a given state. Representing this in Maria assumes the definition of a supplementary place $Succ$ initialized with all natural numbers from 0 to p where p is the number of genes present in the regulatory graph. The transition G_2 reads the current state, chooses a particular token from the place $Succ$ and generates the next state according to the $Succ$' token. For example, if the token 3 is chosen, the next marking corresponds to the state where only gene 3 has changed. If 0 is chosen, it means that no gene evolves.

The guard of G_2 is almost as asyn_guard seen before, the only change concerns the token g read from the place $Succ$. It can be written as follows:

$$
G_2(\nu, \nu', g) = \left\{
\begin{array}{c}
\left((g \neq 0) \wedge (n_g \neq k_{g, R_g(n)}) \wedge (n'_g = n_g \blacktriangleright k_{g, R_g(n)}) \wedge (\forall j \neq g, n'_j = n_j) \right) \\
\bigvee \\
(g = 0) \wedge \left(\forall i \in [1, p], \; (n_i = k_{i, R_i(n)}) \wedge (n'_i = n_i) \right)
\end{array}
\right\}
$$

With this modeling it is possible to implement the global approach described in section 4. In such an approach the majority of models are rejected. It could be interesting to test directly a set of models which are considered as non suitable in order to reject them in only one call to the model checker. We construct now a new modeling which is able to manage a set of models.

Three supplementary places and a transition are added (see Fig. 9). The place $Models$ is initialized with all models considered as suitable, the place $Model$ is initially empty and carries the current model to check, and $Bool_2$ contains a boolean token which is false iff $Model$ is empty. The transition G'_3 takes a possible model from $Models$ and the value false from $Bool_2$ and generates the corresponding token in the place $Model$ as well as the value true in $Bool_2$. The transition G'_2 which simulates the evolution of the regulatory network reads now the parameters from the place $Model$, and the guard is modified accordingly. The size of this Petri net does not depend on the number of genes nor on the number of models.

This modeling does not replace the previous one but completes it. It becomes possible to verify the temporal properties on a set of models similarly as it was possible to do it for a set of initial markings.

9

Figure 9. Implementation a HLPN manipulating a set of models for a regulatory network.

This approach could also be used for finding a first model that is compatible with the temporal properties. Let us assume that we are looking for a model that satisfies a set of formulae $\varphi_1, \varphi_2, ..., \varphi_n$. The model checker MARIA tries to validate the formulae for all possible paths and for all possible models. If no path contradicts the formulae, Maria answers that the Petri net satisfies them. In our current modeling, it would mean that all possible models are coherent with the biological temporal properties. If one path refutes the conjunction of the formulae, then the computation stops.

Let us initialize the Petri net with all possible models and give to the model checker the formula $XX(\neg(\varphi_1 \wedge \varphi_2 \wedge ... \wedge \varphi_n))$. The double X expresses that the formulae are checked after the initialization of parameters and after the initialization of markings. The model checker is looking for a model which contradicts the negation of the conjunctions of formulae. If there exists such a path, Maria stops and gives the associated parameters. This direct way does not allow us to exhibit all models that we are looking for but only the "first" one.

Unfortunately, the used temporal logic limits this approach. Let us assume that we are looking for models satisfying the temporal property φ. If we give the model checker the negation of the formula, Maria tries to exhibit a path that contradicts the negation, *i.e.*, that satisfies the formula. This path corresponds to a model, but for this parameterization, there could exist another path that refutes the formula. In fact, in order to develop such an approach, one needs to express in the formulae that something is possible or that something is true for all possible choices, in other words, the suitable temporal logic would be rather CTL. For example, if we have to check the formula on all paths and for all initial markings, we are looking for a path that satisfies the formula $EX\ AX\ \varphi$. It means that there exists a model for which all initial markings lead to a state where φ is satisfied. The model checker is given the negation of the previous formula, $AX\ EX\ \neg\varphi$, and it will answer, if any, a path that refutes the previous formula, *i.e.*, a model such that for all initial markings, we have φ.

As mentioned before this modeling permits the user to exhibit, if any, a model which satisfies the temporal properties. The existence of such a model proves the coherence between the regulatory graph and the temporal properties, but it is not sure that the biological system works in the same way. From a biological point of view it is more useful to exhibit all possible models which are coherent with the temporal properties since it permits the biologist to explain the behaviors by various models. It would be interesting to develop a model checker which enumerate all counterexamples.

6 Prototype for computer aided modeling

We have designed a prototype for computer aided modeling which implements our general approach described above. It contains principally 3 modules (see Fig. 10):

- The first module permits the user to design the regulatory graph with a user friendly interface (see Fig. 12 in the appendix). `NetworkEditor` is enable to generate a XML file to represent the regulatory graph according to the GINML document type definition [3].
- `FormulaEditor` helps the user to write the LTL formula which describes the biological knowledge or hypothesis on the dynamics of the system. The editor translates the LTL formula into the Maria LTL format.
- `Marianne` takes as input a GINML file or a text file representing a biological regulatory graph and an LTL formula. Marianne computes then for each model the corresponding HLPN. It selects the models

which satisfy the LTL formula using MARIA. It offers also the possibility to generate the corresponding asynchronous state graph (see Fig. 11 in the appendix).
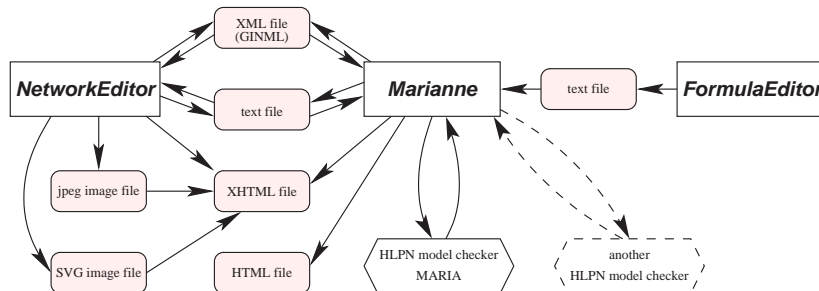

Figure 10. A prototype for computer aided modeling.

7   Conclusion and future work

We have defined a modeling of regulatory networks in terms of high-level Petri nets. Applied to the *Pseudomonas aeruginosa*, this modeling approach selected 4 models leading to 4 different asynchronous state graphs for each regulatory graph corresponding to two possible orderings of t_{11} and t_{12} (see footnote 1). These 8 models prove that the proposed regulatory graphs of *Pseudomonas aeruginosa* are coherent with the hypothesis of epigenesis. Figure 13 presents one model which satisfies the temporal specifications and Fig. 11 the corresponding state graph obtained directly from Maria. If *Pseudomonas aeruginosa* is actually compatible with one of these models (no matter which model, because there are observationally equivalent), it could open new therapeutics in prospects. Since the formula 2 is known to be satisfied, one has just to confirm *in vivo* the formula 1. An experiment schema may be suggested by the structure of the formula: it consists in pulsing $variable_1$ up to saturation by an external signal, and in checking, after a transitory phase due to the pulse, if the mucus production persists [8].

Besides this biological case study, the contribution of the paper is also on a more abstract level. Indeed, our approach overpasses the pure application context and allows a computer aided manipulation of the semantics of the discrete modeling of R. Thomas. It consists in defining an automatic translation from regulatory graphs to high-level Petri nets and to provide the means to express and check some behavioral properties. In particular, temporal properties expressed in temporal logics can be checked in order to confirm or refute some biological hypotheses. These analyzes may be performed using various existent Petri net methods and tools.

Our compact and generic representation through high-level Petri nets opens up some extensions. First, recent extensions [1] taking into account non sigmoid character of the interaction function may easily be handled. Second, resources, like time or energy, may be introduced in the net model, for instance, using high-level buffers, as in [18,15]. Moreover, Petri net representation naturally leads to various kinds of semantics. In particular, one may consider non sequential ones, which allow on one hand combating the state explosion and on the other hand using more efficient verification techniques [12,7].

The paper presents also a user friendly environment we developed, helping biologists in modeling and analyzing regulatory networks and to express desired properties. Our experiments showed that it would be interesting to extend the model checker MARIA with CTL logics. Also, since Maria allows the user to unfold the model into the native input formats of PEP [7], LoLA [20] or Prod [30], it gives a possibility to use different analyze techniques offered by these tools. In particular, the problem of using a CTL model checker may be resolved with Prod.

This approach has been compared in terms of efficiency with another environment for regulatory networks, which uses the classical CTL model checker NuSMV [5] and the execution times were similar. It was not surprising because the semantics was explicitly sequential. We hope that for some extensions accepting truly concurrent behaviors, the verification could be more efficient if partial order representation and dedicated tools are used.

References

1. G. Bernot, F. Cassez, J.-P. Comet, F. Delaplace, Müller, O. Roux, and O.H. Roux. Semantics of biological regulatory networks. In *Molecular Biology BioConcur'2003*, 2003.
2. G. Bernot, J.-P. Comet, A. Richard, and J. Guespin. A fruitful application of formal methods to biological regulatory networks: Extending Thomas' asynchronous logical approach with temporal logic. *J. Theor. Biol.*, 229(3):339–347, 2004.
3. C. Chaouiya, E. Remy, B. Mossé, and D. Thieffry. GINML: towards a GXL based format for logical regulatory networks and dynamical graphs. http://www.esil.univ-evry.fr/~chaouiya/GINsim/ginml.html, 2003.
4. C. Chaouiya, E. Remy, P. Ruet, and D. Thieffry. Qualitative modelling of genetic networks: From logical regulatory graphs to standard petri nets. In *ICATPN 2004*, LNCS 3099, pages 137–156. Springer-Verlag, 2004.
5. A. Cimatti, E. Clarke, F. Giunchiglia, and M. Roveri. Nusmv: a reimplementation of smv. In *STTT'98*, BRICS Notes Series, NS-98-4, pages 25–31, 1998.
6. O. Cinquin and J. Demongeot. Positive and negative feedback: striking a balance between necessary antagonists. *J. Theor. Biol.*, 216(2):229–41, 2002.
7. B. Grahlman. The state of PEP. In *AMAST'98*, number 1548 in LNCS, pages 522–526. Springer-Verlag, 1999.
8. J. Guespin-Michel and M. Kaufman. Positive feedback circuits and adaptive regulations in bacteria. *Acta Biotheor.*, 49(4):207–18, 2001.
9. J. Hasty, D. McMillen, and J.J. Collins. Engineered gene circuits. *Nature*, 420(6912):224–30, 2002.
10. S. Huang. Genomics, complexity and drug discovery: insights from boolean network models of cellular regulation. *Pharmacogenomics.*, 2(3):203–22, 2001.
11. M. Kaufman and R. Thomas. Model analysis of the bases of multistationarity in the humoral immune response. *J. Theor. Biol.*, 129(2):141–62, 1987.
12. V. Khomenko, M. Koutny, and W. Vogler. Canonical prefixes of petri net unfoldings. *Acta Informatica*, 40:95–118, 2003.
13. H. Kitano. Computational systems biology. *Nature*, 420(6912):206–10, 2002.
14. H. Kitano. Looking beyond the details: a rise in system-oriented approaches in genetics and molecular biology. *Curr. Genet.*, 41(1):1–10, 2002.
15. H. Klaudel and F. Pommereau. Asynchronous links in the pbc and m-nets. In *ACSC'99*, volume 1742 of *LNCS*, pages 190 – 200. Springer-Verlag, 1999.
16. M. Mäkelä. Maria: Modular reachability analyser for algebraic system nets. In *ICATPN 2002*, number 2360 in LNCS, pages 434–444. Springer-Verlag, 2002.
17. E. Plathe, T. Mestl, and S.W. Omholt. Feedback loops, stability and multistationarity in dynamical systems. *J. Biol. Syst.*, 3:569–577, 1995.
18. Franck Pommereau. *Modèles composables et concurrents pour le temps-réel*. PhD thesis, Université Paris 12, 2002.
19. S. Pérès and J.-P. Comet. Contribution of computational tree logic to biological regulatory networks: example from pseudomonas aeruginosa. In *CMSB'03*, volume 2602 of *LNCS*, pages 47–56, February 24-26, 2003.
20. K. Schmidt. LoLA: a low level analyser. In M. Nielsen and D. Simpson, editors, *ICTPN 2000*, volume 1825 of *LNCS*, pages 465–474. Springer-Verlag, 2000.
21. E.H Snoussi. Qualitative dynamics of a piecewise-linear differential equations : a discrete mapping approach. *Dynamics and stability of Systems*, 4:189–207, 1989.
22. E.H. Snoussi. Necessary conditions for multistationarity and stable periodicity. *J. Biol. Syst.*, 6:3–9, 1998.
23. E.H. Snoussi and R. Thomas. Logical identification of all steady states : the concept of feedback loop caracteristic states. *Bull. Math. Biol.*, 55(5):973–991, 1993.
24. C. Soulé. Graphical requirements for multistationarity. *ComPlexUs*, 1:123–133, 2003.
25. R. Thomas. Logical analysis of systems comprising feedback loops. *J. Theor. Biol.*, 73(4):631–56, 1978.
26. R. Thomas, A.M. Gathoye, and L. Lambert. A complex control circuit. regulation of immunity in temperate bacteriophages. *Eur. J. Biochem.*, 71(1):211–27, 1976.
27. R. Thomas and M. Kaufman. Multistationarity, the basis of cell differentiation and memory. I. & II. *Chaos*, 11:170–195, 2001.
28. R. Thomas, D. Thieffry, and M. Kaufman. Dynamical behaviour of biological regulatory networks - I. *Bull. Math. Biol.*, 57(2):247–76, 1995.
29. R. Thomas, D. Thieffry, and M. Kaufman. Dynamical behaviour of biological regulatory networks - I. biological role of feedback loops an practical use of the concept of the loop-characteristic state. *Bull. Math. Biol.*, 57(2):247–76, 1995.
30. K. Varpaaniemi, J. Halme, K. Hiekkanen, and T. Pyssysalo. Prod: reference manual. Technical Report B13, Helsinki University of Technology, Finland, 1995.
31. O. Wolkenhauer. Systems biology: the reincarnation of systems theory applied in biology? *Brief Bioinform.*, 2(3):258–70, 2001.
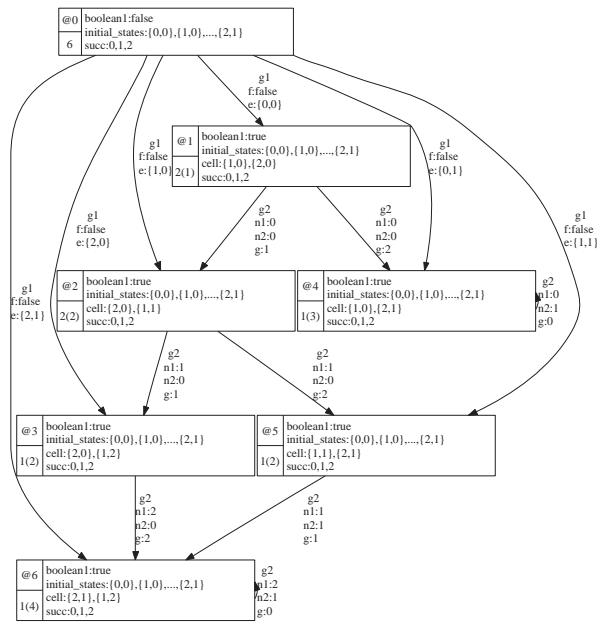
Appendix

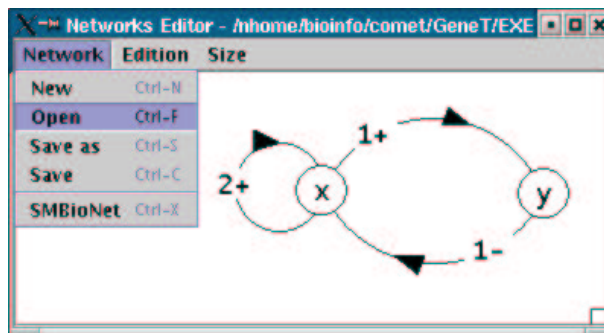**Figure 11.** State graph obtained directly from Maria.

Figure 12. The interface allows the user to specify the regulatory network in term of regulatory graph. The thresholds and the sign of the interactions are added on vertices.

```
///////////////////////////
// DEFINITION OF TYPES  //
///////////////////////////
typedef unsigned (1..2) numero_of_gene;
typedef unsigned (0..2) level;
typedef struct{ numero_of_gene g;
               level n;           } gene;
typedef unsigned (0..2) level_gene1;
typedef unsigned (0..1) level_gene2;
typedef struct{ level_gene1 n1;
               level_gene2 n2;    } state;
typedef unsigned (0..2) succt;
typedef bool flag;


//////////////////////////////
// DEFINITION OF FUNCTIONS  //
//////////////////////////////
state vector(gene g1,gene g2)
    is state({is level_gene1 g1.n,
              is level_gene2 g2.n});

state synchronous(state e) is state(e?
    {2,1}:  // attractor for the state {2,1}
    {2,1}:  // attractor for the state {1,1}
    {0,1}:  // attractor for the state {0,1}
    {2,1}:  // attractor for the state {2,0}
    {2,1}:  // attractor for the state {1,0}
    {2,1}   // attractor for the state {0,0});

state step1(state e) is state(
  (e.n1<synchronous(e).n1)  ?{+e.n1,e.n2}:
  ( (e.n1>synchronous(e).n1)?{|e.n1,e.n2}:e));

state step2(state e) is state(
  (e.n2<synchronous(e).n2)  ?{e.n1,+e.n2}:
  ( (e.n2>synchronous(e).n2)?{e.n1,|e.n2}:e));

gene gene1(state e) is gene({1,e.n1});
gene gene2(state e) is gene({2,e.n2});
```

```
///////////////////////////
// DEFINITION OF PLACES //
///////////////////////////
place boolean1 flag: false;
place initial_states state: state e  : e;
place cell gene;
place succ succt: succt g: g;


//////////////////////////////////
// DEFINITION OF TRANSITIONS   //
//////////////////////////////////

trans g1
in  {  place boolean1: f;
       place initial_states : e;          }
out {  place boolean1: true;
       place initial_states : e;
       place cell : (gene1(e),gene2(e)); }
gate (f==false);


trans g2
    {  state v=vector({1,n1},{2,n2});
       state v1=step1(v);
       state v2=step2(v);               }
in  {  place succ : g;
       place cell : {1,n1},{2,n2};       }
out {  place succ : g;
       place cell :  (b ?
           (gene1(v2),gene2(v2)): // b==2
           (gene1(v1),gene2(v1)): // b==1
           ({1,n1},{2,n2})      ); // b==0
    }
gate (
  (g==0 && v1.n1==v.n1 && v2.n2==v.n2 ) ||
  (g==1 && v1.n1!=v.n1) ||
  (g==2 && v2.n2!=v.n2) );

deadlock fatal;
```

Figure 13. File describing the high-level Petri net corresponding to the regulatory network of Fig. 12 for model checker Maria. The attractors are deduced from the following parameters: $k_{1,\emptyset} = 0$, $k_{1,\{1\}} = 2$, $k_{1,\{2\}} = 2$, $k_{1,\{1,2\}} = 2$, $k_{2,\emptyset} = 1$ and $k_{2,\{1\}} = 1$.

14