An LTL Model Checking Approach for Biological Parameter Inference

E Gallet, M Manceny, P Le Gall, Paolo Ballarini

An LTL Model Checking Approach for Biological Parameter Inference

E. Gallet[1], M. Manceny[2], P. Le Gall[1], and P. Ballarini[1]

[1] Laboratoire MAS, Ecole Centrale Paris, 92195 Châtenay-Malabry, France
email: {emmanuelle.gallet, pascale.legall, paolo.ballarini}@ecp.fr
[2] Laboratoire LISITE, ISEP, 28 Rue Notre-Dame-des-Champs 75006 Paris, France
email: matthieu.manceny@isep.fr

Abstract. The identification of biological parameters governing dynamics of Genetic Regulatory Networks (GRN) poses a problem of combinatorial explosion, since the possibilities of parameter instantiation are numerous even for small networks. In this paper, we propose to adapt LTL model checking algorithms to infer biological parameters from biological properties given as LTL formulas. In order to reduce the combinatorial explosion, we represent all the dynamics with one parametric model, so that all GRN dynamics simply result from all eligible parameter instantiations. LTL model checking algorithms are adapted by postponing the parameter instantiation as far as possible. Our approach is implemented within the SPuTNIk tool.

Keywords: LTL Model Checking, Parameter Identification, Symbolic Execution, Genetic Regulatory Network, Thomas Discrete Modeling.

1 Introduction

Gene expression is a biological process where proteins are synthesized from *genes*. These proteins can regulate the synthesis of other proteins provided that their concentrations are sufficient. A collection of regulatory inter-dependencies between genes/proteins is called a *Genetic Regulatory Network* (GRN). In this paper we consider a discrete-state formalism, the René Thomas' formalism [2,3,6,18,19], according to which the amounts of proteins in a GRN are discrete abstractions of continuous concentrations. Hence the overall evolution of protein concentration along time, called the *dynamics* of the network, is captured by a discrete-state transition system. Given a René Thomas GRN model, the main interest is in analyzing the possible dynamics that may be associated to it. However, from the dynamics point of view, a GRN on its own is an underspecified type of representation: it represents the dependencies between a set of genes, but it does not describe what effect the combination of all such dependencies has on a given state of the network, hence on its evolution. The mapping of a GRN model to a specific dynamic (i.e. a transition system) is achieved by considering an instantiation of so-called *biological parameters*, i.e., a specification of the combined effect that all *activated regulators* have on a given state of the network.

Since the number of possible instantiations of biological parameters is a double exponential function of the GRN *size* (in terms of number of genes and of interactions), the analysis of the possible dynamics associated to a GRN model is a complex task. In this paper we consider the application of formal methods, namely model checking [1], as a means to reason about the possible dynamics associated to a GRN specification. In particular we tackle the following problem: given a relevant behavior of interest, formally expressed in terms of a temporal logic property, say φ, we want to be able to automatically identify the biological parameters instances which give rise to dynamics complying with φ.

Related work. Model checking techniques have been widely advocated in several works to verify whether a given discrete Thomas model fulfills some relevant biological temporal properties. In [3] Bernot *et al.* expressed biological knowledge with *Computation Tree Logic* (CTL) formulas [1]. To exhaustively search the parameters' space, the set of all possible dynamics is generated and a CTL model checking procedure is iterated, one dynamics after the other. This approach is implemented in the SMBioNet tool [16] and has been illustrated in [8]. In [12], the approach has been extended to cope with the formation of complexes from proteins which allows modelers to express relationships between biological parameters leading to a reduced set of dynamics to be investigated. This work prefigures the interest of using constraints on the parameters. [13,2] define an approach based on an encoding technique to share computations between different dynamics. Sets of dynamics are encoded by a binary vector, one bit (or color) per dynamics, and LTL model checking algorithms [1] are extended with Boolean operations on vectors. In [6,5], the tool GNBox uses Constraint Logic Programming (CLP) techniques to identify parameters. GRN dynamics and biological knowledge are described by declarative rules and constraints on parameters, then target behaviors are expressed as some kind of finite paths that models have to verify. [9] also uses CLP techniques to adapt CTL model checking, but the encoding introduces a lot of fresh logical variables that hamper to scale up the method.

Our contribution. We propose a new approach that is based on a *parametric model*, called Parametric GRN (PGRN). This allows us to encompass all the dynamics of a GRN in a unique representation, biological parameters being processed as symbols, and to implement an efficient (on-the-fly) searching of (a symbolic representation of) the parameter's space. Similarly to [13], our approach is based on LTL model checking. While in, [13], LTL model checking algorithms are optimized for the particular case of *time series*, that are sequences of states made of one expression level per gene, observed one by one, we follow the same creed as the one advocated in [6,5]: model sets are handled through some logical language both to avoid combinatorial explosion and to take benefit of constraint solving techniques. A preliminary version of our approach has been described in [14]. In the present version, algorithms combining symbolic execution and constraint solving techniques have been reengineered and tuned to be more efficient and cope with GRN features. Thereby, we consider the full

LTL language while [13] essentially focuses on time series and [6,5] focuses on properties carrying on finite paths.

Paper organization. We reformulate the logical description of Thomas' modeling framework in section 2, and explain how we encode the set of dynamics of a GRN with Parametric GRN in section 3. Section 4 presents our adaptation of LTL model checking algorithms with symbolic execution techniques. In section 5, we briefly discuss the validity of our approach with our dedicated tool SPuTNIk. Finally, section 6 contains some concluding remarks.

2 Genetic Regulatory Networks

A Genetic Regulatory Network (GRN) is a collection of regulatory inter-dependencies between genes to represent the mechanism of gene expression, i.e. the biological process by means of which proteins are synthesized. Two kinds of interactions exist: activation or inhibition depending on whether the protein expressed from the source gene can enhance or reduce the expression of the target gene. Moreover, an interaction is effective only if the concentration of the source protein is sufficient, in other words if its *level of expression* is above a given threshold. From the modeling point of view, a gene g is assimilated to the protein it synthesizes. In particular, it inherits the protein's level of expression, denoted x_g, which, in this context, is abstractly represented by a non-negative integer ranging from 0 (absence of protein or very low concentration level) to a maximal value m_g. A GRN is classically represented by an *interaction graph*.

Definition 1 (Interaction graph). *An n-order* interaction graph *(IG) is a labeled directed graph* $\Gamma = (G, I)$ *where* G *is a finite set of* gene *nodes,* $n = |G|$ *and* $I \subseteq G \times \{+, -\} \times \mathbb{N}^+ \times G$ *is the set of* interactions. *Given* $(g_1, s, t, g_2) \in I$, s *indicates the effect of* g_1 *over* g_2 *(sign "+" for activation and "−" for inhibition) and* t *denotes the threshold of the interaction. Moreover, the following properties hold: i)* $\forall (g_1, g_2) \in G^2$, *there exists at most one interaction* $(g_1, s, t, g_2) \in I$; *ii)* $\forall (g_1, s, t, g_2) \in I, t > 1 \Rightarrow \exists (g_1, s', t', g_3) \in I, t' = t - 1$.

The threshold t of an interaction $(g_1, s, t, g_2) \in I$ indicates the minimal level that g_1 needs to be at in order to affect the expression of g_2. The condition on thresholds states that for any gene g_1 every intermediate threshold level must appear on at least an interaction arc originating in g_1. Since an interaction graph may contain at most one interaction between two genes, then for an interaction $(g_1, s, t, g_2) \in I$, we denote $s(g_1, g_2)$ its sign, and $t(g_1, g_2)$ its threshold. $m_g = max\{t \mid \exists (g, s, t, g') \in I\}$ is[3] the maximal level of expression of gene g, and $G^-(g) \subseteq G$ is the set of *regulators* of g (i.e. $G^-(g) = \{g'' | \exists (g'', s, t, g) \in I\}$).

A dynamics of a GRN corresponds to an evolution over time of the levels of expression of all genes, The *state space* describes the states that may be observed during such a possible evolution.

[3] m_g is equal to 1 if there does not exist edges outgoing from g.

Definition 2 (State space of an interaction graph). *For $\Gamma = (G, I)$ an interaction graph we define $\mathbb{X} = \prod_{g \in G} \mathbb{X}_g$ the state space underlying Γ, where $\mathbb{X}_g = \{0, ..., m_g\}$ is the set of possible levels of expression for gene g.*

Example 1 (Interaction graph and state space). Figure 1 presents a two-genes interaction graph Γ_0 where gene α is both an activator of β and of itself (self-activator) while β is an inhibitor of α. In particular α activates the expression of β whenever its level of expression is at least 1, while when its level of expression is at least 2 it activates both itself and β. The thresholds of Γ_0 induce the following sets of levels for the two genes, $\mathbb{X}_\alpha = \{0, 1, 2\}$ and $\mathbb{X}_\beta = \{0, 1\}$, hence the state space $\mathbb{X} = \{(0,0), (0,1), (1,0), (1,1), (2,0), (2,1)\}$.

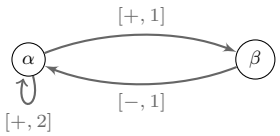

Fig. 1: Γ_0: an example of a two-genes interaction graph.

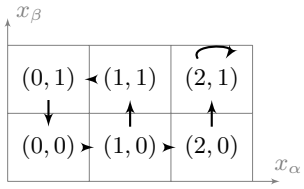

Fig. 2: *Dynamics* $\mathcal{D}_K$ induced by the parameters mapping K

For Γ an interaction graph with state space $\mathbb{X}$, we denote $\mathcal{D} = (\mathbb{X}, \rightarrow)$ a generic transition system[4] called *dymanics* of Γ. With respect to $\mathcal{D}$, Γ can be regarded as an underspecified formalism: from Γ, on its own, one cannot devise any concrete dynamics $\mathcal{D}$. To obtain a dynamics for Γ we need to describe for each gene g the effect that any subset of its regulators $\omega \subseteq G^-(g)$ would have on g. This is achieved by associating *biological parameters* with Γ.

Definition 3 (Biological parameters). *For g a gene of $\Gamma = (G, I)$, the set of biological parameters of g is $\mathbb{K}_g = \{K_g(\omega) \mid \omega \subseteq 2^{G^-(g)}\}$ and the set of all biological parameters of Γ is $\mathbb{K} = \cup_{g \in G} \mathbb{K}_g$.*

An instantiation of biological parameters is defined by any mapping $K : \mathbb{K} \rightarrow \cup_{g \in G} \mathbb{X}_g$ associating to any parameter $K_g(\omega)$ a value in $\mathbb{X}_g$. Any instantiation $K : \mathbb{K} \rightarrow \cup_{g \in G} \mathbb{X}_g$ defines a mapping $\mathcal{K} : \otimes_{g \in G} 2^{G^-(g)} \rightarrow \mathbb{X}$ verifying $\forall (\omega_{g_1}, \ldots, \omega_{g_n}) \in \otimes_{g \in G} 2^{G^-(g)}, \mathcal{K}((\omega_{g_1}, \ldots, \omega_{g_n})) = (K(K_{g_1}(\omega_{g_1})), \ldots, K(K_{g_n}(\omega_{g_n})))$ (with $G = \{g_1, \ldots, g_n\}$).

In the sequel, for simplicity purpose, an instantiation K associating the value x to the parameter $K_g(\omega)$ will be simply given by the equality $K_g(\omega) = x$.
The *biological parameters* of an IG indicate the values the genes of the GRN tend towards when a certain n-tuple of regulators is activated. Thus to obtain the *dynamics* $\mathcal{D}_K$ corresponding to parameters K we need to know what regulators

[4] A transition system (E, R) verifies that R is a binary relation on $E \times E$.

are activated in a state $x \in \mathbb{X}$. We say that a set of regulators $\omega \subseteq G^-(g)$ of gene g is activated in state $x = (x_1, \ldots, x_n) \in \mathbb{X}$, denoted $(x_1, \ldots, x_n) \models \omega$, iff $\left(\bigwedge_{g' \in \omega} x_{g'} \geq t(g', g) \wedge \bigwedge_{g' \in G^-(g) \setminus \omega} x_{g'} < t(g', g)\right)$, that is: if and only for every regulator $g' \in \omega$ the corresponding component $x_{g'}$ of state x is above the corresponding activation threshold (i.e., $x_{g'} \geq t(g', g)$) while no other regulator of g does. In the remainder we denote $ActR : \mathbb{X} \to \otimes_{g \in G} 2^{G^-(g)}$ the function that maps each state x into the corresponding n-tuples of activated regulators for the n genes of a GRN. In the remainder, for $x \in \mathbb{X}$, we denote $x[x_g\uparrow]$ (resp. $x[x_g\downarrow]$) the state resulting from x by increasing (resp. decreasing) the x_g component of one unit.

Definition 4 (dynamics induced by an instantiation of parameters). *For $\Gamma = (G, I)$ an n-order IG, $K : \mathbb{K} \to \cup_{g \in G} \mathbb{X}_g$ a set of biological parameters of Γ, we define $\mathcal{D}_K = (\mathbb{X}, \to_K)$ the dynamics (transition system) of Γ induced by K. The transition relation $\to_K \subseteq \mathbb{X} \times \mathbb{X}$ is minimally defined as follows:*
$\forall x = (x_1, \ldots x_n) \in \mathbb{X}$ let $x^* = (x_1^*, \ldots, x_n^*) = \mathcal{K} \circ ActR((x_1, \ldots, x_n))$:

- *if $x \neq x^*$ then $\forall i, 1 \leq i \leq n$*
  - *if $x_i < x_i^*$, then $x \to_K x[x_i\uparrow]$* (increment gene i)
  - *if $x_i > x_i^*$, then $x \to_K x[x_i\downarrow]$* (decrement gene i)
- *else if $x = x^*$ then $x \to_K x$* (self-loop)

Then for each state $x \in \mathbb{X}$ we determine the corresponding attractor state x^* (by application of the parameters mapping $\mathcal{K}$ to the regulators activated in x i.e., $x^* = \mathcal{K} \circ ActR(x)$). If the attractor state x^* is different from x then for each different component $x_i \neq x_i^*$ we add either an increment or a decrement transition in $\to_K$. Conversely if $x^* = x$ we add a self-loop in $\to_K$.

Example 2 (Biological parameters and Dynamics). The subsets of regulators for the two genes of Γ_0 (Figure 1) are $2^{G^-(\alpha)} = \{\{\}, \{\alpha\}, \{\beta\}, \{\alpha, \beta\}\}$, resp. $2^{G^-(\beta)} = \{\{\}, \{\alpha\}\}$ hence the biological parameters of Γ_0 are: $\mathbb{K} = \{K_\alpha(\{\}),\ K_\alpha(\{\alpha\}),\ K_\alpha(\{\beta\}),\ K_\alpha(\{\alpha, \beta\}),\ K_\beta(\{\}),\ K_\beta(\{\alpha\})\}$. According to $\models$ the association between states of $\mathbb{X}$ and n-tuples of activated regulators is: $(0,0) \models (\{\}, \{\})$, $(0,1) \models (\{\beta\}, \{\})$, $(1,0) \models (\{\}, \{\alpha\})$, $(1,1) \models (\{\beta\}, \{\alpha\})$, $(2,0) \models (\{\alpha\}, \{\alpha\})$, $(2,1) \models (\{\alpha, \beta\}, \{\alpha\})$. As an example we consider the following mapping (instantiation) of the parameters for gene α and β into corresponding target levels: $K_\alpha(\{\}) = 2$, $K_\alpha(\{\alpha\}) = 2$, $K_\alpha(\{\beta\}) = 0$, $K_\alpha(\{\alpha, \beta\}) = 2$, $K_\beta(\{\}) = 0$ and $K_\beta(\{\alpha\}) = 1$ yielding the combined mapping $K = K_\alpha \times K_\beta = \{(\{\}, \{\}) \to (2,0), (\{\{\}, \{\alpha\}) \to (2,1), \ldots, (\{\alpha, \beta\}, \{\alpha\}) \to (2,1)\}$ Figure 2 shows the *dynamics* $\mathcal{D}_K$ yielded by the parameters mapping K.

A parameters mapping K for an IG Γ yields a dynamics $\mathcal{D}_K$. However some mappings K may result into inconsistent dynamics. To rule out inconsistent dynamics, mapping must comply with the following constraints.

Definition 5 (Constraints for parameters mapping). *Let $\Gamma = (G, I)$ be an IG.* Definition constraint: *$\forall g \in G, \forall g' \in G^-(g), \forall \omega \subseteq G^-(g) \setminus \{g'\}$: if $s(g', g) = +$*

then $K_g(\omega) \leq K_g(\omega \cup \{g'\})$, *if* $s(g', g) = -$ *then* $K_g(\omega) \geq K_g(\omega \cup \{g'\})$. Observation constraint: $\forall g \in G$, $\forall g' \in G^-(g)$, *there exists* $\omega \subseteq G^-(g) \setminus \{g'\}$: *if* $S(g', g) = +$ *then* $K_g(\omega) < K_g(\omega \cup \{g'\})$, *if* $s(g', g) = -$ *then* $K_g(\omega) > K_g(\omega \cup \{g'\})$. Min/Max constraint: $\forall g \in G$, $K_g(\{g' | g' \in G^-(g), s(g', g) = -\}) = 0$ *and* $K_g(\{g' | g' \in G^-(g), s(g', g) = +\}) = m_g$.

The *Definition constraint* (or Snoussi constraint [17]) states that if the level of expression of a gene g' which activates (resp. inhibits) a gene g becomes greater than its threshold, then the expression level of g cannot decrease (resp. increase). The *Observation constraint* expresses how we identify regulators. If g' is an activator (resp. inhibitor) of g, then there exists at least one dynamic state where the increase of the level of expression of g' leads to an increase (resp. decrease) of the expression level of g. Finally, the *Min/Max constraint* states that in a dynamic state where all the activators (resp. inhibitors) of a gene are above the threshold and simultaneously none of the inhibitors (resp. activators) is, then the level of expression of the attractor of the gene is maximum (resp. minimum).

Example 3. The *Constraints* for IG Γ_0 (Figure 1) correspond to the following conditions: $K_\alpha(\{\alpha\}) = 2$, $K_\alpha(\{\beta\}) = 0$, $K_\beta(\{\}) = 0$, $K_\beta(\{\alpha\}) = 1$, $\big(K_\alpha(\{\}) < 2 \vee 0 < K_\alpha(\alpha, \beta)\big)$ and $\big(K_\alpha(\{\}) > 0 \vee 2 > K_\alpha(\alpha, \beta)\big)$. Notice that amongst the 324 possible parameter mappings[5] for Γ_0, only 7 are consistent with the *Constraints* for Γ_0.

Even if these constraints are well-founded, there are not always considered by biologists. In the sequel, by default, they will be considered and generically denoted as C_I, but they can be relaxed on demand.

3 Modeling dynamics with Parametric GRN

In order to study all the dynamics simultaneously, we represent them all through a single (meta)model, called *Parametric GRN* (PGRN), i.e. a facility of transition systems parameterized by the biological parameters.

Parametric GRN. A PGRN is a transition system associated with an interaction graph $\Gamma = (G, I)$. It involves two families of symbols: the biological parameters $\mathbb{K} = \{K_g(\omega) \mid g \in G, \omega \subseteq 2^{G^-(g)}\}$ and the state variables $\mathbb{G} = \{x_g | g \in G\}$. Note that, according to the context, x_g will denote either a state variable or a value representing a concentration level.

The main idea is to encode state evolution with transitions parameterized by parameters of $\mathbb{K}$. A PGRN is composed of two states: T (transient) corresponding to configurations such that at least one gene can change its current level, and S (stable) corresponding to situations where no change is possible for any gene. A transition of a PGRN is characterized by a guard (a condition

[5] The number of possible parameters instantiation is equal to $\prod_{g \in G} (m_g + 1)^{2^{|G^-(g)|}}$.

over parameters $\mathbb{K}$ and state variables of $\mathbb{G}$) and an assignment (an application $\mathbb{X} \to \mathbb{X}$ expressing how states of a dynamics evolve). For example, transition $T \xrightarrow{(x_\alpha<2 \wedge x_\beta=0 \wedge x_\alpha<K_\alpha(\{\}))[x_\alpha\uparrow]} T$ (see Fig. 3) indicates that for any (transient) state $x \in \mathbb{X}$ such that $x_\beta = 0$, $x_\alpha < 2$ and $x_\alpha < K_\alpha(\{\})$ then a transition corresponding to an increase of the level of α exists.

More precisely, for each gene g, there is a transition from T to T for each kind of variation (increase or decrease) of x_g. For $\omega \subseteq G^-(g)$ a subset of regulators of g, let us introduce the predicate $P_g(\omega) : \mathbb{X} \to \{\top, \bot\}$ defined by: $(\bigwedge_{g' \in \omega} x_{g'} \geq t(g', g)) \wedge (\bigwedge_{g' \in G^-(g) \setminus \omega} x_{g'} < t(g', g))$. $P_g(\omega)$ characterises the set of states in which regulators ω are the only effective ones on g. The transition associated to the increase of x_g is conditioned by the guard $Increase(g) = \vee_{\omega \subset G^-(g)}(P_g(\omega) \wedge x_g < K_g(\omega))$. Similarly, the transition associated to the decrease of x_g is conditioned by the guard $Decrease(g) = \vee_{\omega \subset G^-(g)}(P_g(\omega) \wedge x_g > K_g(\omega))$. Finally there is one transition from T to S when the expression level of all genes remains stable, *i.e.* if any gene g satisfies the condition $Stable(g) = \wedge_{\omega \subset G^-(g)} (P_g(\omega) \wedge x_g = K_g(\omega))$, and one last transition from S to S where the guard is always true.

Definition 6 (PGRN). *A PGRN associated to an interaction graph $\Gamma = (G, I)$ is a pair $P = (Q_P, \delta_P)$ with $Q_P = \{T, S\}$ the set of states and δ_P a set of transitions. A transition of δ_P is of the form (q_P, g_P, a_P, q'_P), also denoted $q_P \xrightarrow{(g_P)[a_P]} q'_P$, with q_P and q'_P states of Q_P, g_P a guard, i.e. a formula over $\mathbb{K} \cup \mathbb{G}$ and a_P an assignment, i.e. an application $\mathbb{X} \to \mathbb{X}$. More precisely, δ_P is the set of all following transitions:*

- $(T, Increase(g), x_g\uparrow, T)$ *with g in G,*
- $(T, Decrease(g), x_g\downarrow, T)$ *with g in G,*
- $(T, \wedge_{g \in G} Stable(g), id, S)$ *where id is the identity assignment,*
- $(S, \top, id, S)$ *where $\top$ indicates the guard always true.*

Let us remark that unfolded versions of guards can be rather long and complex, but in the best cases they can be simplified by application of the initial constraints C_I. Nevertheless, generally, the most complex guard is the one labeling the transition $T \to S$ since it corresponds to the conjunction of all $Stable(g)$ conditions. On the other hand, once in S, the guard of the only possible transition $(S \to S)$ is simply true $(\top)$. Moreover, transitions involving disjunctions in their guard can be split. Indeed, transition $(T, g_P \vee g'_P, a_P, T)$ can be equivalently split in (T, g_P, a_P, T) and (T, g'_P, a_P, T).

Example 4 (PGRN). Fig. 3 represents the PGRN associated with the interaction graph of Fig. 1. In relation with the different possible subsets ω, one can explicit the different guards: e.g. $Increase(\beta) \equiv (x_\alpha < 1 \wedge x_\beta < K_\beta(\{\})) \vee (x_\alpha \geq 1 \wedge x_\beta < K_\beta(\{\alpha\})))$. The Initial constraints C_I (cf. Def 5) can be used to simplify the guards: e.g. C_I implies $K_\beta(\{\}) = 0$ and $K_\beta(\{\}) = 1$ and then $Increase(\beta) \equiv (x_\alpha > 0 \wedge x_\beta = 0)$.

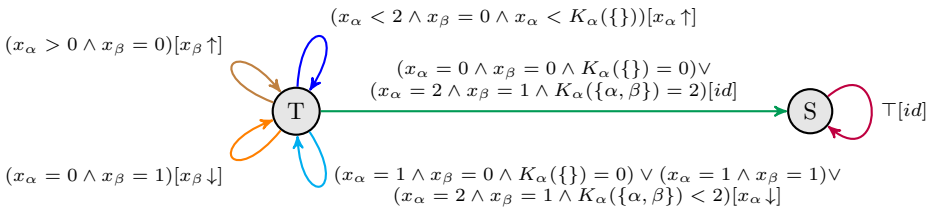

Fig. 3: PGRN associated to the interaction graph in Figure 1.

Annotated dynamics. A PGRN characterizes a set of dynamics, one for each possible instantiation of biological parameters, that is, for any parameter mapping $K : \mathbb{K} \to \cup_{g\in G}\mathbb{X}_g$. For g_P a transition guard of a PGRN, $x \in \mathbb{X}$ a state of the corresponding GRN, and K an instance of biological parameters, we denotes $[\![g_P]\!]_{x,K}$ the instance of g_P obtained by substituting g_P's state variables and g_P's biological parameters with the corresponding state value of x, and parameter values of K. Similarly, we denote $[\![g_P]\!]_x$ the resulting substitution only of g_P's state variables (parameters in $\mathbb{K}$ remain symbolic).

Definition 7 (Annotated Dynamics). *Let $P = (Q_P, \delta_P)$ be a PGRN associated with an interaction graph $\Gamma = (G, I)$, and let $K : \mathbb{K} \to \cup_{g\in G}\mathbb{X}_g$. The annotated dynamics associated to P and K is a pair $D_K = (Q_D, \delta_D)$ where the set of* states *$Q_D \subset Q_P \times \mathbb{X}$ and the set of transitions $\delta_D \subset Q_D \times Q_D$ are mutually defined by: $\forall x \in \mathbb{X}, (T, x) \in Q_D$ and for all $(q_P, x) \in Q_D$ and $(q_P, g_P, a_P, q'_P) \in \delta_P$ s.t. $[\![g_P]\!]_{x,K}$ is evaluated to True, then $(q'_P, a_P(x)) \in Q_D$ and $((q_P, x), (q'_P, a_P(x))) \in \delta_D$.*

Example 5 (Annotated Dynamics). Figure 4 presents one possible annotated dynamics for the PGRN represented in Figure 3, with the following instantiation of parameters: $K_\alpha(\{\}) = 2$, $K_\alpha(\{\alpha\}) = 2$, $K_\alpha(\{\beta\}) = 0$, $K_\alpha(\{\alpha,\beta\}) = 2$, $K_\beta(\{\}) = 0$ and $K_\beta(\{\alpha\}) = 1$.

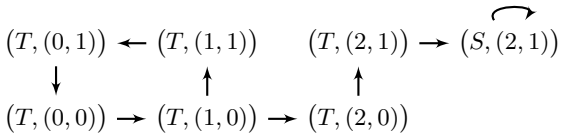

Fig. 4: A possible annotated dynamics for the PGRN in Figure 3.

By construction, for a given instantiation $K : \mathbb{K} \to \cup_{g\in G}\mathbb{X}_g$, the associated annotated dynamics D_K corresponds to the dynamics $\mathcal{D}_K$ of the underlying IG Γ induced by the instantiation K (cf Def 4). For a transition $((q_P, x), (q'_P, a_P(x)))$ in δ_D, it suffices to give up the first component and keep the second one, $x \to a_P(x)$, to retrieve a dynamics of Γ. Thus, the dynamics represented in Fig. 2 can be obtained from the annotated dynamics of Fig. 4. The first component (T

or S) is somehow a technical artifact annotating the presence of a stable state when building sequences of consecutive states. Depending on the context, we will assimilate $\mathcal{D}_K$ and D_K or will work with the most appropriate of the two forms. Motivated by efficiency considerations, we will apply a specific treatment for states x already recognized as stable, that is, annotated by S.

4 Adapting LTL model-checking to PGRN

The classical approach of LTL model-checking [15] consists in confronting a model (e.g. a dynamics) against an LTL formula. To do so, the negation of the LTL formula is transformed into a Büchi automaton and the product between the automaton and the dynamics is computed. We then look for accepting paths in the product by checking the existence of reachable cycles containing at least an accepting state. Model checking is usually time consuming, and since the number of dynamics is large, this method is not applicable in our case. To avoid the combinatorial explosion, we want to check all the dynamics simultaneously, i.e. we check directly the PGRN. To do so, we first build the Parametric Product between the PGRN and the Büchi Automaton associated to the LTL formula φ. We then use symbolic execution technics in order to search for accepting cycles. As a result, we obtain a set of constraints C that a parameter instantiation K must fulfill such that the associated dynamics $\mathcal{D}_K$ verifies φ.

Büchi Automaton and Parametric product Biological properties on a sequence of states can be expressed using LTL formulas built from a set of atomic propositions using the usual logical operators in $\{\top, \bot, \neg, \wedge, \vee\}$ and the temporal operators **X** (for neXt time), **G** (Globally), **F** (Finally) and **U** (Until) [1]. Since we need to express biological knowledge on levels of expression of genes, atomic propositions are of the form $x_g \bowtie c$ where x_g denotes the level of expression of a gene g, $\bowtie \in \{=, \neq, <, >, \leq, \geq\}$ and $c \in \mathbb{N}$. Any LTL formula φ can be translated into a Büchi automaton $B(\varphi)$.

Definition 8 (Büchi Automaton associated to an LTL formula). *Let Γ be a GRN and φ an LTL formula over the levels of expression of genes of Γ. A Büchi Automaton associated to φ is a tuple $B(\varphi) = (Q_B, q_B^0, A_B, \delta_B)$ where Q_B is the set of states, $q_B^0 \in Q_B$ is the initial state, $A_B \subseteq Q_B$ is the set of accepting states and δ_B is the set of transitions. A transition of δ_B is of the form (q_B, g_B, q'_B) with q_B and q'_B states of Q_B and g_B a non temporal formula over the levels of expressions of genes in Γ. Moreover, $B(\varphi)$ is such that an infinite sequence of states provided with truth values for all atomic propositions (a path) verifies φ iff this path is accepted by $B(\varphi)$, i.e. iff this path contains at least a so-called accepting state infinitely often.*

Example 6 (LTL formula and associated Büchi automaton). The existence of a steady state (*i.e.* a state which is itself its only own successor) in $(x_\alpha, x_\beta) = (2, 1)$ corresponds to the LTL formula $\mathbf{G}((x_\alpha = 2 \wedge x_\beta = 1) \rightarrow \mathbf{X}(x_\alpha = 2 \wedge x_\beta = 1))$. Fig. 5 presents a Büchi Automaton associated to the negation of this formula.

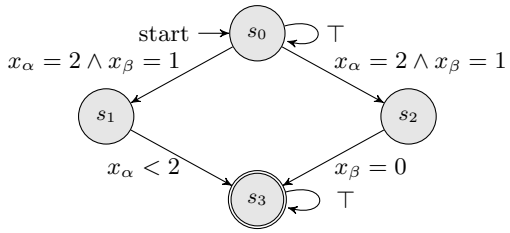

Fig. 5: $B(\neg\varphi)$ with $\varphi \equiv \mathbf{G}\big((x_\alpha = 2 \wedge x_\beta = 1) \Rightarrow \mathbf{X}(x_\alpha = 2 \wedge x_\beta = 1)\big)$.

Definition 9 (Parametric Product). *Let $P = (Q_P, \delta_P)$ be a PGRN and $B(\neg\varphi) = (Q_B, q_B^0, A_B, \delta_B)$ a Büchi Automaton associated to the LTL formula $\neg\varphi$. The product $\Pi = P \otimes B(\neg\varphi)$ is the tuple $(Q_\Pi, q_\Pi^0, A_\Pi, \delta_\Pi)$ with $Q_\Pi = Q_P \times Q_B$ the set of vertices, $q_\Pi^0 = (T, q_B^0)$ the initial vertex, $A_\Pi = Q_P \times A_B$ the set of accepting vertices, and δ_Π the set of transitions. A transition of δ_Π is of the form $(q_\Pi, g_\Pi, a_P, q'_\Pi)$ with $q_\Pi = (q_P, q_B)$, $q'_\Pi = (q'_P, q'_B)$, $g_\Pi = g_P \wedge g_B$ such that $(q_P, g_P, a_P, q'_P) \in \delta_P$, $(q_B, g_B, q'_B) \in \delta_B$ and g_Π is satisfiable.*

Example 7. The product of the PGRN in Fig. 3 and the Büchi Automaton in Fig. 5 is represented in Fig. 6. The product has been simplified by removing output transitions whose guard on expression levels is not satisfiable according to the guards and assignments of the input transitions of the same vertex; we also remove the transitions whose guard is not satisfiable according to the guards on parameters necessarily crossed (ϕ_{2_1} and ϕ_{2_2} here). Finally, we remove vertices which can not be reached and those belonging to a terminal cycle without accepting vertex.

Search for parametric accepting cycles The search for accepting cycles is based on symbolic execution techniques which are program analysis techniques. The key point is the substitution of actual values by symbolic variables in order to symbolically perform computations. Each execution (or path) of the program associates to each variable a symbolic computation together with a *path condition* that expresses what are the conditions on input values to execute the given path. Symbolic execution techniques has been extended to symbolic transition systems [11] by unfolding transition systems as symbolic trees. As symbolic execution is only applicable for finite paths, selection criteria are used to cut infinite paths when considering testing. In the sequel, we will take particular care to cut infinite paths in identifying situations of return on a node already encountered. Indeed such situations reveal the presence of cycles.

In the symbolic execution of the parametric product Π, the parameters $K_g(\omega)$ are handled as symbolic variables (i.e. not evaluated), and Π is unfolded leading to the construction of several *Symbolic Execution Trees* (SET), one for any $x \in \mathbb{X}$.

Definition 10 (Symbolic Execution Tree). *Let $\Pi = (Q_\Pi, q_\Pi^0, A_\Pi, \delta_\Pi)$ be a parametric product. The Symbolic Execution Tree associated to Π and $x \in \mathbb{X}$ is a transition system (Q_T, δ_T) where the set of* nodes *Q_T and the set of transitions*

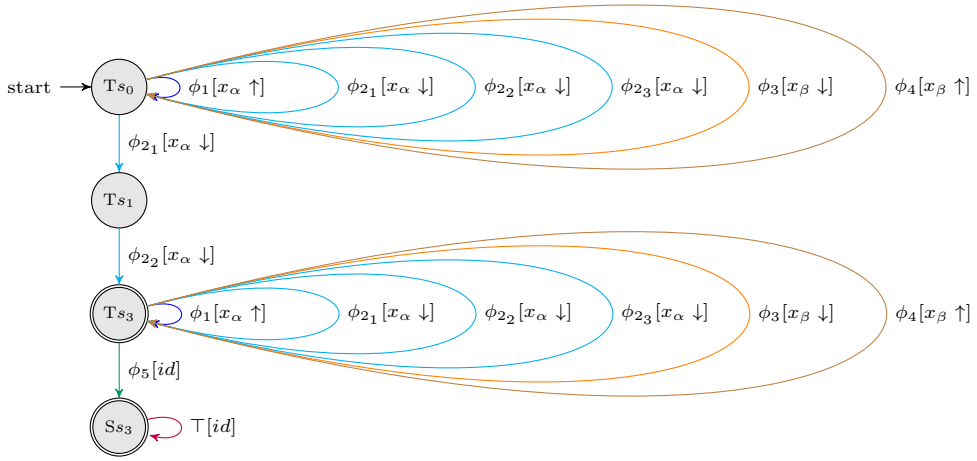

Fig. 6: Parametric product $P \otimes B(\neg\varphi)$ associated to the *Parametric GRN* in Figure 3 and the Büchi Automaton in Figure 5 (after simplification), with:
$\phi_1 \equiv x_\alpha < 2 \wedge x_\beta = 0 \wedge x_\alpha < K_\alpha(\{\})$ $\quad \phi_3 \equiv x_\alpha = 0 \wedge x_\beta = 1$
$\phi_{2_1} \equiv x_\alpha = 2 \wedge x_\beta = 1 \wedge K_\alpha(\{\alpha, \beta\}) < 2$ $\quad \phi_4 \equiv x_\alpha > 0 \wedge x_\beta = 0$
$\phi_{2_2} \equiv x_\alpha = 1 \wedge x_\beta = 1$ $\quad \phi_5 \equiv x_\alpha = 0 \wedge x_\beta = 0 \wedge K_\alpha(\{\}) = 0.$
$\phi_{2_3} \equiv x_\alpha = 1 \wedge x_\beta = 0 \wedge K_\alpha(\{\}) = 0$

$\delta_T \subset Q_T \times Q_T$ are mutually defined by: $(q_\Pi^0, x, \top) \in Q_T$ and for all $q_T = (q_\Pi, x, pc) \in Q_T$, for all $(q_\Pi, g_\Pi, a_P, q'_\Pi) \in \delta_\Pi$ such that $pc' = pc \wedge [\![g_\Pi]\!]_x \neq \bot$, then $q'_T = (q'_\Pi, a_P(x), pc') \in Q_T$ and $(q_T, q'_T) \in \delta_T$. If $q_\Pi \in A_\Pi$, then the node is said to be accepting.

For any node $(q_\Pi, x, pc) \in Q_T$ with $q_\Pi = (q_P, q_B) \in Q_P \times Q_B$, pc is the *path condition* in the form of a constraint over parameters in $\mathbb{K}$. It defines the set of annotated dynamics D that can reach the state (q_P, x) from the state in $Q_P \times \mathbb{X}$ associated to the ancestor of (q_Π, x, pc). With the process described in the section 3, D itself allows the definition of the set of dynamics of the corresponding GRN which can reach all the states associated to nodes of the node path along the same sequence of traversed states. By construction, path conditions expressed over parameters increase along paths of SET and reduce the number of dynamics compatible with the path under construction.

Biological properties are expressed along infinite sequences, and thus, paths of the product and paths of SET are also infinite. But, by disregarding path conditions, the number of possible nodes in a SET is finite[6]. So, when we are building a new node (q_Π, x, pc') whereas it is descendant of a node (q_Π, x, pc) (same vertex in Q_Π and same value in $\mathbb{X}$), we stop the analysis of the path; these two nodes are respectively called *child node* and *return node*. By construction, the path condition of the child node is included in the path condition of the

[6] it is bounded by the product of all combinations of levels of expression, the number of vertices of the Büchi automaton and the number of vertices (2) of the PGRN.

return node, i.e. all parameter instantiations satisfying the child path condition also satisfy the return path condition.

Thus, by performing a mixed symbolic and numerical execution (parameters in $\mathbb{K}$ remain unchanged and state variables in $\mathbb{X}$ are evaluated), we can stop the execution procedure of the product Π so that each path of the resulting SET is finite and contains a cycle (starting at the return node and ending with the transition leading to the child node). If there exists an accepting node between the return node and the child node, the path condition is said *accepting*.

Once all finite SET are built, it remains to compute for which parameter instantiations there exist accepting paths. For that, it suffices to consider every accepting path conditions of the SET associated to Π. Each accepting path condition can be satisfied by (at least) one instantiation of parameters in $\mathbb{K}$, it means that there exists a path in Π going infinitely often through the associated cycle, and thus passing infinitely often by an accepting state. And so there exists a path in the dynamics corresponding to this accepting path condition verifying $\neg\varphi$.

Thus, instantiations of parameters verifying the conjunction of the negation of every accepting path condition of the SET associated to Π correspond to the dynamics such that there is no path verifying $\neg\varphi$, in other words, all paths verify φ. Note that the obtained dynamics verify φ along *all* paths; if the model must verify φ only on *at least one* path, our approach remains adequate with a small adaptation: to do this, we have to get the disjunction of all accepting path conditions of the SET associated to the product $P \otimes B(\varphi)$.

Example 8. For the Product in Figure 6, there are two solutions after computation; the corresponding values of parameters are: $K_\alpha(\{\}) = 1$ or 2, $K_\alpha(\{\alpha\}) = 2$, $K_\alpha(\{\beta\}) = 0$, $K_\alpha(\{\alpha, \beta\}) = 2$, $K_\beta(\{\}) = 0$ and $K_\beta(\{\alpha\}) = 1$. One of the corresponding dynamics (with $K_\alpha(\{\}) = 2$) is represented in Figure 2.

Algorithm of traversal of SET. Algorithm 1, based on a *Depth First Search* schema, gives an overview of how we practically compute the accepting path conditions. We use three global variables: the parametric product Π, the list of accepting path conditions *acceptingPC*, and the list *nodesList* of SET nodes which have already been analyzed.

Starting with a SET node, line 2 to line 4 test and compute its successors, as explained in the "Symbolic Execution Trees" part of section 4. Three tests are then performed successively. Firstly, if the path condition of the successor node is already known, it cannot provide additional information (the *pc* becomes more specific every depth call), and we stop the study of this successor (line 5). Secondly, line 7 tests if one of the ancestors of the successor is a return node (ancestor with the same vertex and state). If it is the case, then there is an infinite cycle between them and, if there is an accepting node in that cycle, then the successor node is an accepting return node, and its path condition is added to the list *accepting_PCs* (lines 8 to 9). Thirdly, if the successor is not a return node, we check (line 10) if the node corresponds to a node in *nodesList* with the same vertex, the same state and the same or a more general path condition.

Algorithm 1: Overview of DFS((q_Π, x, pc), *ancestorsList*)

Data: global $\Pi = (Q_\Pi, {q^0}_\Pi, A_\Pi, \delta_\Pi)$, global *acceptingPC*, global *nodesList*

1 $ancestorsList$.add((q_Π, x, pc));
2 **forall** $(q_\Pi, g_\Pi, a_P, q'_\Pi) \in \delta_\Pi$ **do** // calculation of all successors
3 $pc' \leftarrow pc \wedge [\![g_\Pi]\!]_x$; // pc of the new node
4 **if** $pc' \neq \bot$ **then** // the transition can be crossed
5 **if** $pc' \not\subset acceptingPC$ **then** // pc not included in the analyzed *accepting pc*
6 $x' \leftarrow a_P(x)$; // state of the new node
7 **if** $\exists (q'_\Pi, x', pc'') \in ancestorsList$ **then** // an ancestor of the new node is a return node
8 **if** $\exists (q'''_\Pi \in A_\Pi, x''', pc''') \in [(q'_\Pi, x', pc''), \ldots, (q'_\Pi, x', pc')] \subset ancestorsList$ **then** // a descendant of the return node is accepting
9 $acceptingPC$.add(pc')
10 **else if** $\nexists (q'_\Pi, x', pc'''') \in nodesList$ *with* $pc' \subset pc''''$ **then** // no copy node: recall of DFS()
11 DFS((q'_Π, x', pc'), $ancestorsList$);
12 $nodesList$.add((q'_Π, x', pc'));

If it is not the case (no *copy node*), then the DFS function is recalled with the successor node in argument (line 11), which is then added to *nodesList* (line 12).

Transient and Stable. Nodes of the tree are of the form (q_Π, x, pc) with $q_\Pi = (q_P, q_B) \in Q_P \times Q_B$. According to the value of q_P (either T or S, from the PGRN), we say that the node is either *transient* or *stable*. By construction of the PGRN, the target vertices of all transitions outgoing from a stable vertex are vertices of the same type, hence the appellation *stable*. Furthermore, the guards of the transitions between the stable vertices are always of the form $g_\Pi = g_P \wedge g_B$ with $g_P = \top$, and the assignment of the transitions is $a_P = id$ (identity assignment). Thus a specific treatment can be provided for the stable nodes, briefly described in the sequel.

In the algorithm 1, the line 11 can be split in two calls, one for the current function and another for the specific treatment of stable nodes (called if the successor node is stable, called *stable root* in the sequel). In this case, the second argument of the function, the list of ancestors, is an empty list since none of the previous ancestors (all transient) can be a return node of a stable node. According to the characteristics of the transitions between stable vertices mentioned above, for the treatment of stable nodes there is no need to test if the path condition is already known (line 5, already tested with the corresponding stable root), there is no guard on parameters to symbolically verify and no substitution of levels of expression (lines 3 to 4), and there is no update of the state to do (line 6). Furthermore, if a node is accepting then all the explorations

of the SET from the stable root can be stopped; indeed, its path condition is identical to all path conditions of the nodes which can be built from it.

5 Assessment

The methodology described above has been implemented in a prototype software tool called SPuTNIk. SPuTNIk is written in Java and relies on the Z3 constraint solver [7] to check the satisfiability of path conditions during the traversal of SET and on the ltl2ba and LTL2BA4J libraries [10,4] to generate a Büchi Automaton of minimal size from an LTL formula. To validate our approach with SPuTNIk, we have considered a common biological case study: the analysis of the genetic network that controls the life cycle of the λ phage virus [19]. The λ phage can infect the *E. coli* bacterium with two different outcomes: either it integrates the genome of the host through a process called *lysogeny* or it enters a *lytic* phase where it kills the residing cell to reproduce itself. We based our approach on the λ phage model studied in [13] by Klarner *et al.* and composed of four genes, denoted cI, cII, cro and N, and ten interactions described Fig. 7.

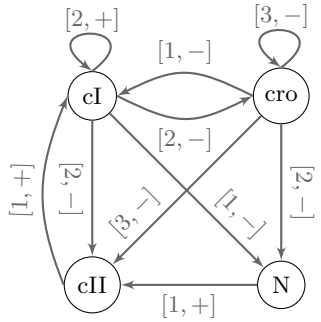

Fig. 7: The interaction graph G_λ for the λ phage.

Klarner *et al.* describe biological properties as *time series*: a sequence of specific states given in the form $\theta \equiv s_1, *, s_2, *, \ldots, *, s_n$ where s_i is the i^{th} observed state while $*$ denotes a possibly empty sequence of unspecified states. Times series are equivalent to LTL formulas of the form $\phi \equiv s_1 \wedge \mathbf{F}(s_2 \wedge \mathbf{F}(\cdots \wedge \mathbf{F}(s_n)\ldots))$ (i.e. only composed of $\wedge$ and $\mathbf{F}$ operators). Moreover, the states of time series are fully determined, each level of expression corresponds to a single value. For example for G_λ, each state is a quadruple $(x_{cI}, x_{cII}, x_{cro}, x_N) \in \{0,1,2\} \times \{0,1\} \times \{0,1,2,3\} \times \{0,1\}$). Given a time series θ and an interaction graph, the goal of Klarner *et al.* is to find out all models which contain at least one path matching θ (i.e. passing through the states of θ in the correct order). Klarner *et al.* distinguish the following states: $init \equiv [0000]$, $lyt_1 \equiv [0021]$, $lyt_2 \equiv [0020]$, $lyt_3 \equiv [0030]$, $lys_1 \equiv [2101]$ and $lys_2 \equiv [2000]$, belonging to time series $\theta_1 \equiv init, *, lyt_1, *, lyt_2, *, lyt_3$ and $\theta_2 \equiv init, *, lys_1, *, lys_2$, which correspond to

evolution towards lytic and lysogenic phases. The equivalent LTL counterparts for θ_1 and θ_2 are respectively $\phi_1 \equiv init \wedge \mathbf{F}(lyt_1 \wedge \mathbf{F}(lyt_2 \wedge \mathbf{F}(lyt_3 \wedge \mathbf{F}(lyt_2))))$ and $\phi_2 \equiv init \wedge \mathbf{F}(lys_1 \wedge \mathbf{F}(lys_2))$.

In order to reproduce the same experiment than Klarner *et al.*, we discard the Min/Max constraint for all genes (as it is not supported in [13]), and we relax the Observation constraint for the specific case where cI is activator of itself (as done in [13]). We then use SPuTNIk to find out the parameter instantiations corresponding to dynamics which are guaranteed to exhibit either a lytic or a lysogenic phenotype in compliance with series θ_1 and θ_2 (i.e. all models that contain at least one path that satisfies ϕ_1 and at least one that satisfies ϕ_2).

The obtained results are in accordance: amongst the 7 billions possible models, we obtain the same number (8759) of valid ones as in [13]. But unlike Klarner *et al.* our method is not restricted to time series: we can consider any form of LTL formulas and there is no need to fully specify all the levels of expression. For example, it is known that a lytic λ phage can not become lysogenic in the future (and conversely). This knowledge cannot be expressed with time series, but it corresponds to the following LTL formulas: $\phi_3 \equiv \mathbf{G}(lys_2 \Rightarrow \neg\mathbf{F}(lyt_3))$ and $\phi_4 \equiv \mathbf{G}(lyt_3 \Rightarrow \neg\mathbf{F}(lys_2))$. By adding these formulas to the previous, we reduce the number of solutions to 2390.

6 Conclusion

In this paper we introduced a new methodology for reverse-engineering of genetic network models, based on adaptation of classical LTL model-checking with symbolic execution. In order to find dynamics consistent with biological knowledge, we use the whole extent of LTL to express biological knowledge in terms of constraints over time. Instead of checking each dynamics of the GRN, we propose a method which performs checking with a novel formalism, the Parametric GRN, a compact (symbolic) representation of all the dynamics associated to an interaction graph within a single structure. From the Parametric GRN and LTL formulas, our algorithm processes parameters, defining the dynamics, as symbols in order to avoid combinatorial explosion. The solutions are in the form of a set of constraints that the parameters must fulfill. Such analysis has been carried out through the SPuTNIk tool, a prototype software of the proposed method. We are working on a parallel version of SPuTNIk, based on the splitting of the Parametric Product into strongly connected components in order to detect the accepting cycles in each component.

References

1. C. Baier and J.-P. Katoen. *Principles of Model Checking.* The MIT Press, 2008.
2. J. Barnat, L. Brim, A. Krejci, A. Streck, D. Safránek, M. Vejnar, and T. Vejpustek. On parameter synthesis by parallel model checking. *IEEE/ACM Trans. Comput. Biology Bioinform.*, 9(3):693–705, 2012.

3. G. Bernot, J.-P. Comet, A. Richard, and J. Guespin. Application of formal methods to biological regulatory networks: extending Thomas' asynchronous logical approach with temporal logic. *Journal of Theoretical Biology*, 229(3):339–347, August 2004.
4. Eric Bodden. *LTL2BA4J Software, http://www.sable.mcgill.ca/~ebodde/rv/ltl2ba4j/*. RWTH Aachen University, 2011.
5. F. Corblin, E. Fanchon, and L. Trilling. Applications of a formal approach to decipher discrete genetic networks. *BMC Bioinformatics*, 11:385, 2010.
6. F. Corblin, S. Tripodi, E. Fanchon, D. Ropers, and L. Trilling. A declarative constraint-based method for analyzing discrete genetic regulatory networks. *BioSystems*, 98:91–104, 2009.
7. Leonardo De Moura and Nikolaj Bjørner. Z3: An efficient smt solver. In *Tools and Algorithms for the Construction and Analysis of Systems*, volume 4963 of *Lecture Notes in Computer Science*, pages 337–340. Springer, 2008.
8. D. Filopon, A. Mérieau, G. Bernot, J.-P. Comet, R. Leberre, B. Guery, B. Polack, and J. Guespin. Epigenetic acquisition of inducibility of type III cytotoxicity in P. aeruginosa. *BMC Bioinformatics*, 7:272–282, 2006.
9. J. Fromentin, J.-P. Comet, P. Le Gall, and O. Roux. Analysing gene regulatory networks by both constraint programming and model-checking. In *EMBC'07, 29th IEEE Engineering in Medicine and Biology Society*, pages 4595–4598, 2007.
10. P. Gastin and D. Oddoux. Fast LTL to Büchi automata translation. In *Proceedings of the 13th International Conference on Computer Aided Verification (CAV'01)*, volume 2102 of *Lecture Notes in Computer Science*, pages 53–65, Paris, France, 2001. Springer.
11. C. Gaston, P. Le Gall, N. Rapin, and A. Touil. Symbolic execution techniques for test purpose definition. In *18th IFIP Int. Conf. TestCom*, volume 3964 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2006.
12. Z. Khalis, J.-P. Comet, A. Richard, and G. Bernot. The SMBioNet method for discovering models of gene regulatory networks. *Genes, Genomes and Genomics*, 3(special issue 1):15–22, 2009.
13. H. Klarner, A. Streck, D. Šafránek, J. Kolčák, and H. Siebert. Parameter identification and model ranking of thomas networks. In *Proceedings of the 10th international conference on Computational Methods in Systems Biology*, CMSB'12, pages 207–226, Berlin, Heidelberg, 2012. Springer-Verlag.
14. D. Mateus, J.-P. Gallois, J.-P. Comet, and P. Le Gall. Symbolic modeling of genetic regulatory networks. *Journal of Bioinformatics and Computational Biology*, 5(2B):627–640, 2007.
15. Amir Pnueli. The temporal logic of programs. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, SFCS '77, pages 46–57, Washington, DC, USA, 1977. IEEE Computer Society.
16. A. Richard. *SMBioNet User manual, http://www.i3s.unice.fr/~richard/smbionet/*, 2010.
17. E. Snoussi and R. Thomas. Logical identification of all steady states: the concept of feedback loop characteristic states. *Bull. Math. Biol.*, (55(5)):973–991, 1993.
18. D. Thieffry, M. Colet, and R. Thomas. Formalisation of regulatory networks : a logical method and its automation. *Math. Modelling and Sci. Computing*, 2:144–151, 1993.
19. D. Thieffry and R. Thomas. Dynamical behaviour of biological regulatory networks - II. immunity control in bacteriophage lambda. *Bull. Math. Biol.*, 57(2):277–97, 1995.