Model-checking for parametric transition systems based on symbolic execution*

Extended version

Matthieu Manceny[1,2], *Jean-Paul Comet*[3], *Jean-Pierre Gallois*[4] *and Pascale Le Gall*[1,2]

1. École Centrale Paris, Mas Laboratory, France
2. Epigenomics Project, University of Évry, France
3. I3S, University of Nice, France
4. CEA LIST Saclay, France

Abstract

In this paper, we present a new specification framework called Parametric Transition Systems (PTS) which represents a set of models as Kripke structures in a concise manner, each model corresponding to an interpretation of the PTS parameters. In order to determine models with a certain behaviour (described by an LTL temporal formula) we propose a method based on symbolic execution and directly inspired by LTL model checking technics involving Büchi automata. Our method determines constraints that the PTS parameters must verify such that the associated Kripke structure satisfies the considered formula. We then show that PTS are useful in the context of modelling of genetic regulatory network, which are highly parametric biological interacting networks. Indeed, this framework allows us to simplify the first step of modelling process which consists in determining the parameter interpretations leading to a behaviour compatible with known temporal properties.

Keywords: Formal methods, Model checking, Symbolic execution, Linear Temporal Logic, Presburger Arithmetic, Genetic regulatory networks

1 Introduction

To understand the functioning of Complex Systems, modelling and simulation are often useful or even mandatory since the complexity of the interleaved interactions between constituents of the system makes intuitive reasoning too difficult. In some particular fields, the lack of reliable quantitative available data about a system leads to a typical difficulty of the modelling approach. To overpass this limitation, qualitative models have been developed: they abstract unuseful details of the system although they preserve qualitative observations.

Unfortunately, modelling activity requires the determination of parameters which play a major role for the set of possible behaviours. It has been proved that formal methods like model checking can be useful for this task: after having formally specified temporal properties (using a temporal logic), it is possible to verify if the constructed

*This work is performed within the European project GENNETEC (STREP 34952).

model satisfies the specification. If one wants to know all models compatible with the specification, this leads to generate all the possible models, and to select the ones satisfying the temporal property [11].

In this paper, we present an approach inspired by classical model checking technics developped for Linear Temporal Logic (LTL) [16, 17] but instead of checking a model given as an usual Kripke structure, we check Parametric Transition Systems (PTS) which represent a set of models (one for each parameter interpretation). The idea is to use symbolic execution techniques [7] to generate all possible behaviours. Each behaviour is associated with a path condition which represents the constraints on parameters allowing this behaviour. Adapting model checking techniques, all paths verifying a specified behaviour are selected and the associated path conditions are reduced into a single constraint. Parameters verifying the resulting constraint are those leading to a model which satisfies the specified behaviour. The different behaviours are computed by the AGATHA tool which is also used for verification of industrial specifications [2].

The paper is organised as follows. Section 2 presents the *PTS* which abstract sets of *models*. Section 3 deals with *Linear Temporal Logic* whose atoms are built from Presburger arithmetic. In particular, variables used to parameterize PTS can occur in those formulas. For the purpose of model checking technics, a *Büchi automation* can be associated to any LTL formula on Presburger terms. Section 4 explains our method for the search of parameter interpretations defining models that verify the specification: the *Symbolic Execution Tree* which represents behaviours of all possible models, allows us to express the *path conditions* that a parameter interpretation must verify to lead to the considered behaviour. Section 5 shows how PTS modelling has been useful for the study of genetic regulatory networks. Finally, Section 6 contains some concluding remarks.

2 Parametric Transition Systems

2.1 Preliminaries

Presburger Arithmetic. Presburger Arithmetic is a first-order theory interpreted in the set $\mathbb{Z}$ of integers. $\mathcal{T}(V)$ denotes the set of *linear terms over the set V of variables*. They are of the form $c + \sum_{i=1}^{n} a_i x_i$ where a_i and c are integer constants, and x_i are variables of V.

The set $Pres(V)$ of *quantifier free Presburger formulas over V*, is the set of formulas recursively defined by usual Boolean connectives in $\{\neg, \vee, \wedge\}$ and atomic formulas of the form $t_1 \sim t_2$ where t_1 and t_2 are linear terms over V and $\sim$ is an operator in $\{=, <, \leq, >, \geq\}$.

Mappings, assignments and interpretations. We use the classical notation B^A for denoting the set of *mappings* from a set A to a set B. A mapping f in B^A is also denoted $f : A \to B$.

In the following, given a set A and a set V of variables including A, an *assignment* is a mapping from A to $\mathcal{T}(V)$ which assigns a term over V to elements of A. id_A will denote the identity assignment over A. An assignment α is usually represented by giving only the elements of A for which α differs from the identity assignment. For instance, $\alpha : a \mapsto v$ represents the assignment for which $\alpha(a) = v$ and $\forall a' \neq a, \alpha(a') = a'$.

An *interpretation* is a mapping from a set A to $\mathbb{Z}$ and will be generically denoted ν_A. Any interpretation ν_A can be canonically extended to terms over A as $\nu_A : \mathcal{T}(A) \to \mathbb{Z}$. Given A and A' two sets such that $A \subseteq A'$ and φ a formula of $Pres(A')$, then $\nu_A(\varphi)$ is a formula of $Pres(A' \backslash A)$ obtained by replacing each variable of $a \in A$ by $\nu_A(a)$.

Satisfaction of formulas. For a Presburger formula φ over V, for an interpretation $\nu_V : V \to \mathbb{Z}$, we note $\nu_V \models_V \varphi$ if the formula φ is interpreted as true by applying the assignment ν_V and interpreting quantifiers, connectives and binary operators as usual.

The satisfiability problem[1] is decidable for Presburger formulas [8] and we will use the Omega library [6] as decision procedure to solve constraints expressed as Presburger formulas.

A formula φ defined over V is *valid* iff $\forall \nu_V \in \mathbb{Z}^V, \nu_V \models_V \varphi$. Valid formulas can be equivalently denoted by the truth symbol $\top$.

2.2 Syntax

2.2.1 Definition.

Parametric Transition Systems, PTS for short, are specifications defining a set of models as Kripke structures. Given a set P of integer variables called parameters, PTS are built over a set A of integer variables called *attributes*. A PTS is defined by initial conditions on both attributes and parameters expressed by means of a quantifier free Presburger formula over the set of variables[2] $P \coprod A$, by a set of locations, by an initial location and by a set of transitions. Each transition is defined by source and target locations, by a guard expressed as a quantifier free Presburger formula over $P \coprod A$ and by an assignment of attributes.

Definition 1 (PTS)
Let A be a set of attributes and let P be a set of parameters. Let us note $V = P \coprod A$ the set of variables.

A parametric transition system $\mathscr{S}(P)$ *over A is a tuple (L, l_i, C_i, Tr) such that :*

- *L is a set of* locations*;*
- *$l_i \in L$ is the* initial location*;*
- *$C_i \in Pres(V)$ represents the* initial constraints*;*
- *Tr is the set of* transitions *of the form (l, g, α, l'), also denoted $l \xrightarrow{(g)\alpha} l'$, where*
  - *l and l' are elements of L, called resp.* source *and* target *locations*
  - *g is a formula of $Pres(V)$ called the* guard
  - *α is an assignement in $\mathcal{T}(V)^A$*

Moreover, PTS are required to be complete*: for each location $l \in L$, by denoting $Tr_l^+ = \{(l, g_i, \alpha_i, l_i') \mid (l, g_i, \alpha_i, l_i') \in Tr\}$ the set of outgoing transitions from l, $\bigvee_{(l,g_i,\alpha_i,l_i') \in Tr_l^+} g_i$ is a valid formula.*

2.2.2 Parameters and attributes.

The initial constraints define all the possible eligible combinations of initial values for parameters and attributes. The assignment associated to each transition expresses the modification of attribute values when a transition is passed over. Let us point out that assignments are used to modify states and thus do not change parameter values. In a manner of speaking, provided that initial constraints are satisfiable, parameters play the role of global variables that influence expected behaviours associated to PTS by means of initial constraints and guards but whose interpretation remains fixed.

In the sequel, the notations P and A denote disjoint sets of respectively parameters and attributes.

[1] That is the problem of the existence of at least an assignment interpreting as true a given formula.

[2] Recall that $\coprod$ denotes the disjoint union.

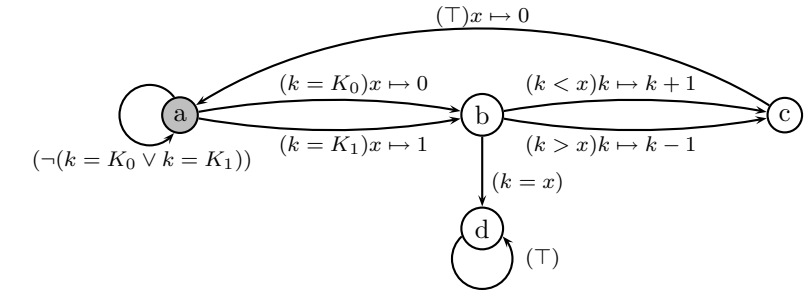


Fig. 1: $\mathscr{S}^0(P^0)$, an example of PTS

2.2.3 Completeness.

Any not complete PTS can be systematically complemented by applying a stutter rule, that is, by adding for each location l for which the disjunction of all guards associated to outgoing transitions from l would not be valid, a new transition of the form $(l, \neg \bigvee_{(l,g_i,\alpha_i,l'_i)\in Tr_l^+} g_i, id_A, l)$. The stutter rule is commonly considered for ensuring that any sequence of successive states can be extended in an infinite sequence [5].

Example 1 (PTS $\mathscr{S}^0(P^0)$ over A^0)
Let us consider a set $P^0 = \{K_0, K_1\}$ of parameters and a set $A^0 = \{k, x\}$ of attributes. $V^0 = P^0 \coprod A^0 = \{K_0, K_1\} \coprod \{k, x\} = \{K_0, K_1, k, x\}$ is the set of corresponding variables.

Fig. 1 presents the PTS $\mathscr{S}^0(P^0)$ over A^0 we will work on. $\mathscr{S}^0(P^0)$ is defined by $\mathscr{S}^0(P^0) = (L^0, {l_i}^0, {C_i}^0, Tr^0)$ with

- *$L^0 = \{a, b, c, d\}$ the set of locations;*
- *$l_i^0 = a$ the initial location (in gray in Fig. 1);*
- *$C_i^0 \in Pres(V^0)$ the initial constraints defined by*

$$C_i^0 = (K_0 = 0 \vee K_0 = 1) \wedge (K_1 = 0 \vee K_1 = 1) \wedge (k = 0 \vee k = 1) \wedge (x = 0)$$

- *Tr^0 the set of transitions defined in Fig. 1.*

2.2.4 PTS paths.

A path of a PTS $\mathscr{S}(P) = (L, l_i, C_i, Tr)$ is an infinite sequence $\lambda = (tr_n)_{n\in\mathbb{N}}$ where for each $n \in \mathbb{N}$, tr_n is a transition of Tr of the form $(l_n, g_n, \alpha_n, l'_n)$ with $l_0 = l_i$ and $\forall n \in \mathbb{N}, l'_n = l_{n+1}$. Such a path is also denoted: $\lambda = l_0 \xrightarrow{(g_0)\alpha_0} l_1 \xrightarrow{(g_1)\alpha_1} \ldots \xrightarrow{(g_{k-1})\alpha_{k-1}} l_k \xrightarrow{(g_k)\alpha_k} \ldots$

2.3 Semantics

Variables in $V = P \coprod A$ can be either attributes or parameters. So, two interpretations ν_A and ν_P define a variable interpretation denoted[3] $(\nu_P, \nu_A) : V \to \mathbb{Z}$ by $\forall a \in A, (\nu_P, \nu_A)(a) = \nu_A(a)$ and $\forall p \in P, (\nu_P, \nu_A)(p) = \nu_P(p)$.

[3] Let us remark that (ν_P, ν_A) can be viewed as element of $\mathbb{Z}^P$ or $\mathbb{Z}^A$, and thus, by abuse of notation, element of $\mathbb{Z}^P \times \mathbb{Z}^A$

2.3.1 PTS Models

Intuitively, a parametric transition system $\mathscr{S}(P)$ characterizes a set of *models*, one by parameter interpretation. More precisely, a parameter interpretation defines a model if the initial constraints are satisfiable taking into account the parameter interpretation.

Definition 2 (PTS models)
Let $\mathscr{S}(P) = (L, l_i, C_i, Tr)$ be a PTS over A and let $\nu_P : P \to \mathbb{Z}$ be a parameter interpretation such that $\nu_P(C_i)$ is satisfiable.

A model *of $\mathscr{S}(P)$ is a triple $S(\nu_P, \mathscr{S}(P)) = (Q, Q_i, \tau)$ where*

- *$Q_i = \{(l_i, \nu_A) | \nu_A \in \mathbb{Z}^A, (\nu_P, \nu_A) \models_V C_i\}$ is the set of initial states;*
- *$Q \subset L \times \mathbb{Z}^A$ is the set of* states*;*
- *$\tau \subset Q \times Q$ is the set of* transitions *of the form $((l, \nu_A), (l', \nu'_A))$, also denoted $(l, \nu_A) \to (l', \nu'_A)$;*
- *Q and τ are mutually defined by:*
  - *$Q_i \subseteq Q$*
  - *for all $(l, \nu_A) \in Q$ and for all $(l, g, \alpha, l') \in Tr$ s.t. $(\nu_P, \nu_A) \models_V g$, then*
    - *$(l', (\nu_P, \nu_A) \circ \alpha) \in Q$*
    - *$((l, \nu_A), (l', (\nu_P, \nu_A) \circ \alpha) \in \tau$*

$S(\nu_P, \mathscr{S}(P))$ is often denoted $S(\nu_P)$ by abuse of notation.

Definition 3 (Runs over (A, L), model runs)
Let L be a set of locations.

A run *over (A, L) is a sequence $(l_n, \nu_{An})_{n \in \mathbb{N}}$ with for all n in $\mathbb{N}$, $l_n \in L$ and $\nu_{An} \in \mathbb{Z}^A$. Such a run is also denoted: $(l_0, \nu_{A0}) \to (l_1, \nu_{A1}) \to ... \to (l_k, \nu_{Ak}) \to ...$*

Let $S(\nu_P) = (Q, Q_i, \tau)$ be a model of a PTS $\mathscr{S}(P)$ defined over A.

A model run *$\sigma(\nu_P)$ of $S(\nu_P)$ is a run $(l_n, \nu_{An})_{n \in \mathbb{N}}$ over (A, L) such that $(l_0, \nu_{A0}) \in Q_i$ and for all $n \in \mathbb{N}$, $(l_n, \nu_{An}) \to (l_{n+1}, \nu_{An+1}) \in \tau$.*

2.3.2 Semantics of PTS

It directly follows from the definition of PTS models that the semantics of a PTS is the set of models built over parameter interpretations compatible with the initial constraints of the PTS.

Definition 4 (Semantics of a PTS)
The semantics of a PTS $\mathscr{S}(P) = (L, l_i, C_i, Tr)$ over A is the set $[\![\mathscr{S}(P)]\!]$defined by

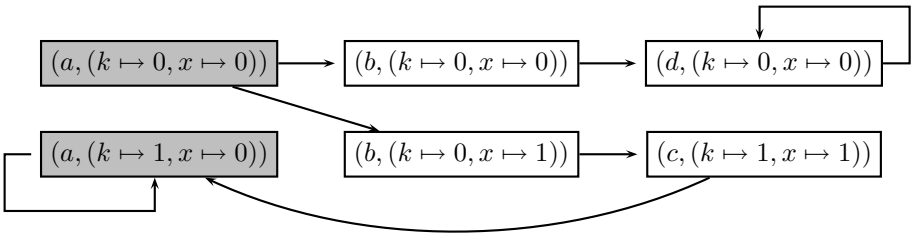
$$[\![\mathscr{S}(P)]\!] = \{S(\nu_P, \mathscr{S}(P)) \mid \nu_P \in \mathbb{Z}^P, \exists \nu_A \in \mathbb{Z}^A, (\nu_P, \nu_A) \models_V C_i\}$$

Example 2 (Models of $\mathscr{S}^0(P^0)$)
There are 4 different parameter interpretations compatible with the initial constraints $C_i^0 = (K_0 = 0 \vee K_0 = 1) \wedge (K_1 = 0 \vee K_1 = 1) \wedge (k = 0 \vee k = 1) \wedge (x = 0)$:

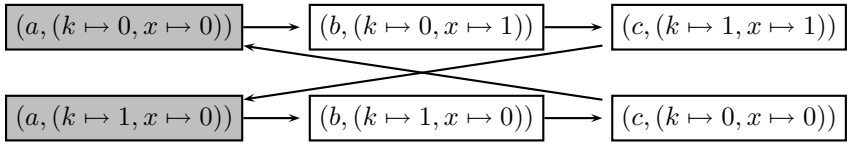
$$\nu_{P0} : (K_0 \mapsto 0, K_1 \mapsto 0) \qquad \nu_{P1} : (K_0 \mapsto 1, K_1 \mapsto 0)$$
$$\nu_{P2} : (K_0 \mapsto 0, K_1 \mapsto 1) \qquad \nu_{P3} : (K_0 \mapsto 1, K_1 \mapsto 1)$$

For each parameter interpretation ν_P, there are 2 attribute interpretations ν_A such that $(\nu_P, \nu_A) \models C_i^0$: $(k \mapsto 0, x \mapsto 0)$ and $(k \mapsto 1, x \mapsto 0)$.

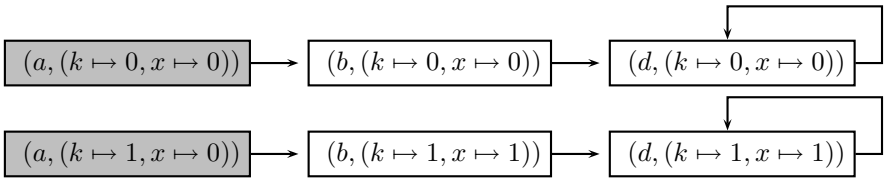
Thus, the semantics of $\mathscr{S}^0(P^0)$ contains four models depicted in Fig. 2.



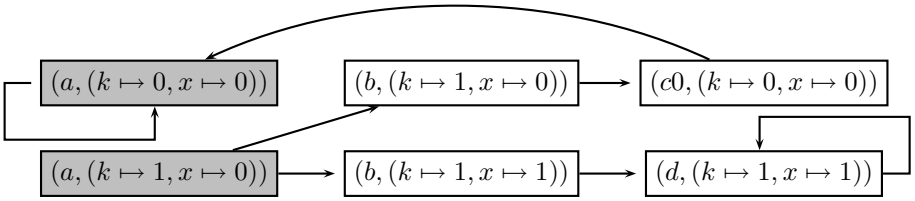
Model of $\mathscr{S}^0(P^0)$ for the interpretation $\nu_{P0} : (K_0 \mapsto 0, K_1 \mapsto 0)$



Model of $\mathscr{S}^0(P^0)$ for the interpretation $\nu_{P1} : (K_0 \mapsto 1, K_1 \mapsto 0)$



Model of $\mathscr{S}^0(P^0)$ for the interpretation $\nu_{P2} : (K_0 \mapsto 0, K_1 \mapsto 1)$



Model of $\mathscr{S}^0(P^0)$ for the interpretation $\nu_{P4} : (K_0 \mapsto 1, K_1 \mapsto 1)$

Fig. 2: The four models of $\mathscr{S}^0(P^0)$

As a matter of fact, the semantics of a PTS is defined as a family of Kripke structures. Indeed, each state is defined by an attribute interpretation which, associated to the underlying parameter interpretation defining the considered PTS model, allows one to interprete as true or false any Presburger formula defined over V.

We will provide PTS with temporal logic. For that, we chose formulas based on Linear Temporal Logic whose atomic formulas are quantifier free Presburger formulas defined over the set of attributes. In particular, let us remark that these Presburger-LTL formulas do not concern parameters since parameters no longer exist in PTS models.

3 Presburger-LTL formulas

3.1 Syntax and Semantics

3.1.1 Syntax.

Presburger-LTL formulas are built from a set of atomic propositions using the usual Boolean connectives and the temporal operators **X** (for neXt time) and **U** (for Until) [16, 17]. In our settings, atomic formulas are quantifier free Presburger formulas defined over A.

Definition 5 (Syntax of Presburger-LTL)
The set Φ_A of Presburger-LTL formulas over A is defined as follows: for $p \in Pres(A)$,

$$\Phi_A ::= p|\neg\Phi_A|\Phi_A \vee \Phi_A|\mathbf{X}\Phi_A|\Phi_A\mathbf{U}\Phi_A$$

3.1.2 Semantics.

Presburger-LTL formulas will be interpreted on runs.

Definition 6 (Semantics of Presburger-LTL)
Let L be a set of locations and let $\sigma = (l_n, \nu_{An})_{n\in\mathbb{N}}$ be a run over (A, L). Let φ and ψ be two Presburger-LTL formulas over A.

The satisfaction relation $\models$ is defined as follows[4]*:*

- *if $\varphi \in Pres(A)$, then $\sigma \models \varphi$ iff $\nu_{A0} \models_A \varphi$;*
- *$\sigma \models \neg\varphi$ iff $(\sigma \not\models \varphi)$;*
- *$\sigma \models \varphi \vee \psi$ iff $\sigma \models \varphi$ or $\sigma \models \psi$;*
- *$\sigma \models \mathbf{X}\varphi$ iff $\sigma^1 \models \varphi$;*
- *$\sigma \models \varphi\mathbf{U}\psi$ iff there exists $i \in \mathbb{N}$ such that $\sigma^i \models \psi$ and for all $j < i$, $\sigma^j \models \varphi$.*

To summarize, a formula without a temporal operator (**X**, **U**) as principal operator refers to the current state, the formula $\mathbf{X}\varphi$ means that φ is verified for the run issued from the next state, and $\varphi\mathbf{U}\psi$ means that φ is true until the condition ψ becomes true which is required to happen.

3.1.3 Additional operators.

Connectives. We can introduce other usual Boolean connectives: $\wedge$ ("and"), $\Rightarrow$ ("implies") and truth symbols $\top$ ("true") and $\bot$ ("false") by deriving them with the following definitions:

$$\begin{array}{rcl@{\qquad}rcl} \varphi \wedge \psi & \equiv & \neg(\neg\varphi \vee \neg\psi) & \top & \equiv & \varphi \vee \neg\varphi \\ \varphi \Rightarrow \psi & \equiv & \neg\varphi \vee \psi & \bot & \equiv & \neg\top \end{array}$$

[4] For a sequence $\sigma = (\sigma_n)_{n\in\mathbb{N}}$, σ^i is $(\sigma_{n+i})_{n\in\mathbb{N}}$.

Temporal operators. Many interesting properties can be simply expressed using **F** (finally), **G** (globally) and **R** (Release) operators. These operators may be expressed using the previous ones.

- The **F** operator expresses that a formula is true in one state of the path. Thus, $q \models \mathbf{F}\varphi$ iff there exists $i \in \mathbb{N}$ such that $q^i \models \varphi$.
- The **G** operator expresses that a formula is true in every states of the path. Thus, $q \models \mathbf{G}\varphi$ iff for all $i \in \mathbb{N}$, $q^i \models \varphi$.
- The **R** operator expresses (for $\varphi\mathbf{R}\psi$) that a formula ψ is true until the first state in which φ is true (or whenever if such a state does not exist). Thus $q \models \varphi\mathbf{R}\psi$ iff for all $i \in \mathbb{N}$, ($q^i \models \psi$ or there exists j such that $j < i$ and $q^j \models \varphi$).

In other words:

$$\mathbf{F}\varphi \equiv \top\mathbf{U}\varphi \qquad \mathbf{G}\varphi \equiv \neg\mathbf{F}(\neg\varphi) \qquad \varphi\mathbf{R}\psi \equiv \neg(\neg\varphi\mathbf{U}\neg\psi)$$

Example 3 (Presburger-LTL formula over A_0)
The Presburger-LTL formula $\varphi^0 = \mathbf{GF}(x = 1)$ means that in every position of the run, there exists a future position where $x = 1$ is true.

3.1.4 Normal form.

As usual LTL formulas, Presburger-LTL formulas can be translated into normal form which does not contain the temporal operators **F** and **G** and such that all negation connectives are adjacent to atomic formulas.

Let us remark that, in our case, the negation connectives will be included at the level of Presburger formulas. Thus, any Presburger-LTL formula over A is equivalent[5] to a formula belonging to the set Ψ_A defined by: for $p \in Pres(A)$,

$$\Psi_A ::= p \mid \Psi_A \vee \Psi_A \mid \Psi_A \wedge \Psi_A \mid \mathbf{X}\Psi_A \mid \Psi_A\mathbf{U}\Psi_A \mid \Psi_A\mathbf{R}\Psi_A$$

To transform a Presburger-LTL formula over A into normal form, one can use the following equivalences for temporal operators: $\neg\mathbf{X}\varphi \equiv \mathbf{X}(\neg\varphi)$, $\neg(\varphi\mathbf{U}\psi) \equiv (\neg\varphi)\mathbf{R}(\neg\psi)$, $\neg(\varphi\mathbf{R}\psi) \equiv (\neg\varphi)\mathbf{U}(\neg\psi)$.

Example 4 (Normal form of φ^0)
Using **G** *and* **F** *definitions, we find:* $\varphi^0 = \mathbf{GF}(x = 1) \equiv \bot\mathbf{R}(\top\mathbf{U}(x = 1))$

3.2 Büchi Automaton

3.2.1 Definition.

Classical methods of LTL model checking [17] are based on the construction for an LTL formula φ of a so-called Büchi Automaton, BA for short, recognizing all runs satisfying φ and only them. We adapt the usual BA definition dedicated for propositional LTL by labelling transitions with a Presburger-LTL formula instead of a propositional variable.

Definition 7 (Büchi automaton over $Pres(A)$)
A Büchi automaton over $Pres(A)$ is a tuple (L_B, l_{Bi}, Tr_B, Ac) where

- L_B *is the set of* locations*;*
- $l_{Bi} \in L_B$ *is the* initial location*;*
- $Tr_B \subseteq L_B \times Pres(A) \times L_B$ *is the set of* transitions *of the form (l_B, ρ, l'_B), also denoted $l_B \xrightarrow{\rho} l'_B$;*

[5] Two formulas are said to be equivalent if they are satisfied by the same set of runs.

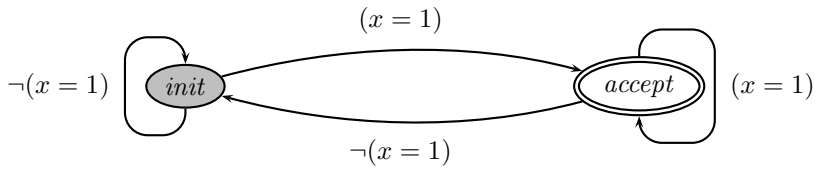


Fig. 3: The Büchi automaton $\mathscr{B}^0$

- $Ac \subseteq$ *is the set of* accepting locations.

Example 5 (Büchi automaton $\mathscr{B}^0$)
Fig. 3 describes the associated Büchi automaton $\mathscr{B}^0$ over $Pres(A^0)$ where $L_B^0 = \{init, accept\}$ is the set of locations with init the initial location (in gray) and accept the accepting location (with double line).

3.2.2 Paths of Büchi automaton.

A path of a Büchi automaton (L_B, l_{Bi}, Tr_B, Ac) is a sequence $\mu = (tr_{Bn})_{n\in\mathbb{N}}$ where for all $n \in \mathbb{N}$, tr_{Bn} is a transition of Tr_B which can be written $(l_{Bn}, \rho_n, l_{Bn+1}) \in Tr_B$ with $l_{B0} = l_{Bi}$. μ is also denoted: $l_{B0} \overset{\rho_0}{\rightarrow} l_{B1} \overset{\rho_1}{\rightarrow} ... \overset{\rho_{k-1}}{\rightarrow} l_{Bk} \overset{\rho_k}{\rightarrow} ...$

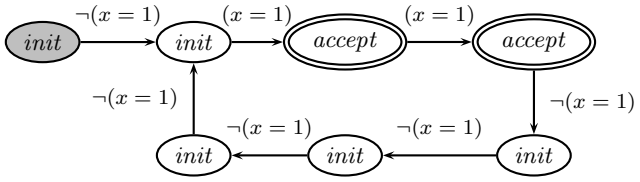
We note $inf(\mu)$ the set of locations which occur infinitely often in μ.

3.2.3 Accepted runs.

Given a set L of locations, a run $\sigma = (l_n, \nu_{An})_{n\in\mathbb{N}}$ over (A, L) is recognized or accepted by a Büchi automaton $\mathscr{B} = (L_B, l_{Bi}, Tr_B, Ac)$ over $Pres(A)$ if there exists a path $\mu = l_{B0} \overset{\rho_0}{\rightarrow} l_{B1} \overset{\rho_1}{\rightarrow} ... \overset{\rho_{k-1}}{\rightarrow} l_{Bk} \overset{\rho_k}{\rightarrow} ...$ of $\mathscr{B}$ such that μ contains at least an accepting state occurring infinitely often in it (*i.e.* $inf(\mu) \cap Ac \neq \emptyset$) and such that transition labellings are satisfied at each step (*i.e.* for all $k \in \mathbb{N}$, $\nu_{Ak} \models \rho_k$).

The set of runs accepted by $\mathscr{B}$ is called the language of $\mathscr{B}$ and denoted $\mathscr{L}(\mathscr{B})$.

Example 6 (Runs accepted by $\mathscr{B}^0$)
The run of $S(\nu_{P1})$ whose initial state is $(a, (k \mapsto 0, x \mapsto 0))$ is accepted by $\mathscr{B}^0$ considering the following path:



$S(\nu_{P2})$ admits two infinite runs. The run starting at $(a, (k \mapsto 0, x \mapsto 0))$ has the stationary state $(d, (k \mapsto 0, x \mapsto 0))$ and thus, is not accepted by $\mathscr{B}^0$. On the contrary, the other one is accepted by $\mathscr{B}^0$.

3.2.4 Büchi automaton associated to a Presburger-LTL formula.

The result of the existence of BA recognising run satisfying LTL formulas can be extended to Presburger-LTL.

Proposition 1 (BA for Presburger-LTL)
Let L be a set of locations. Let φ be a Presburger-LTL formula over A. A BA over $Pres(A)$, denoted $\mathscr{B}(\varphi)$ can be constructed such that $\mathscr{L}(\mathscr{B}(\varphi))$ is the set of runs over (A, L) satisfying φ.

Proof 1 *For φ a Presburger-LTL formula in normal form, let us introduce $PA(\varphi)$ the set of all Presburger formulas p occurring in φ as the largest sub-formulas of φ without temporal operators. We associate to $PA(\varphi)$ a set $PA^*(\varphi) = \{p^* \mid p \in PA(\varphi)\}$ of usual propositional atoms. We now consider φ^* the propositional LTL formula derived from φ by replacing elements of $PA(\varphi)$ by elements of $PA^*(\varphi)$.*

Let $\mathscr{B}^*(\varphi^*)$ be the BA recognizing the propositional runs $s^* = (s^*_n)_{n\in\mathbb{N}}$ satisfying φ^* and only them: s^*_n are sets of propositional atoms in $PA^*(\varphi)$. We consider $\mathscr{B}(\varphi)$ the BA over A deduced from $\mathscr{B}^*(\varphi^*)$ by replacing propositional labelling in $PA^*(\varphi)$ by labelling in $PA(\varphi) \subset Pres(A)$.

For any attribute interpretation ν_A, let us consider the state $s(\nu_A, PA^*(\varphi)) = \{p^* | p^* \in PA^*(\varphi), \nu_A \models_A p\}$.

For a run $\sigma = (l_n, \nu_{An})$, let us consider $\sigma^* = (s^*_n)_{n\in\mathbb{N}}$ the propositional run defined by $s^*_n = s(\nu_{An}, PA^*(\varphi))$. Then by construction, σ is recognised by $\mathscr{B}(\varphi)$ iff σ^* is recognised by $\mathscr{B}^*(\varphi^*)$.

Example 7 (BA associated to φ^0)
$\mathscr{B}^0$ is an example of BA associated to φ^0.

3.2.5 Product between PTS and BA.

In order to unify PTS and BA, let us introduce the notion of Accepting PTS, APTS for short,which are PTS equipped with a set $Ac \subseteq L$ of accepting locations and denoted by (L, l_i, C_i, Tr, Ac). Thus, both PTS and BA can be viewed as Accepting PTS: PTS are APTS where all locations are accepting, BA are APTS with no parameters and where all the assignments are the identity function, and the initial constraint is true. These completion mechanisms will be left implicit in the sequel and all notions and notations concerning either PTS or BA still hold for APTS.

To characterise paths of PTS verifying a given Presburger-LTL formula φ, we consider, as usual, the product between the PTS and $\mathscr{B}(\varphi)$.

Definition 8 (Product)
Let $\mathscr{S}(P) = (L, l_i, C_i, Tr)$ be a PTS over A. Let $\mathscr{B}(\varphi) = (L_B, l_{Bi}, Tr_B, Ac)$ be a BA over $Pres(A)$.

The product $\mathscr{S}_\varphi(P) = \mathscr{S}(P) \otimes \mathscr{A}(\varphi)$ is an APTS $(L \times L_B, (l_i, l_{Bi}), C_i, Tr_\varphi, L \times Ac)$ over A such that

$$Tr_\varphi = \{((l, l_B), g \wedge \rho, \alpha, (l', l'_B)) | (l, g, \alpha, l') \in Tr, (l_B, \rho, l'_B) \in Tr_B\}$$

Example 8 (product between $\mathscr{S}^0(P^0)$ and $\mathscr{B}^0(\varphi^0)$)
Fig. 4 presents $\mathscr{S}^0_{\varphi^0}(P^0)$, the APTS resulting from the product between $\mathscr{S}^0(P^0)$ and $\mathscr{B}^0$. The initial location is in gray, and the accepting ones are in double line.

As it is well known that product characterises language intersection, the resulting APTS $\mathscr{S}_\varphi(P)$ holds the interesting property that accepted runs of $\mathscr{S}_\varphi(P)$ define paths of $\mathscr{S}(P)$ satisfying φ. The question to find PTS models $\mathscr{S}(P)$ which admit paths verifying a Presburger-LTL formula is reduced to find models that allow the existence of infinite paths in the product. In the next section we will use symbolic execution to analyse paths of the product, in order to find parameter interpretations ν_P defining these models.

4 Searching parameters based on symbolic execution

Symbolic execution was first introduced for program analysis and test generation ([7]). The basic idea of symbolic execution is to associate to each variable of a system a *symbol* which represents its initial value and to express all the computations performed

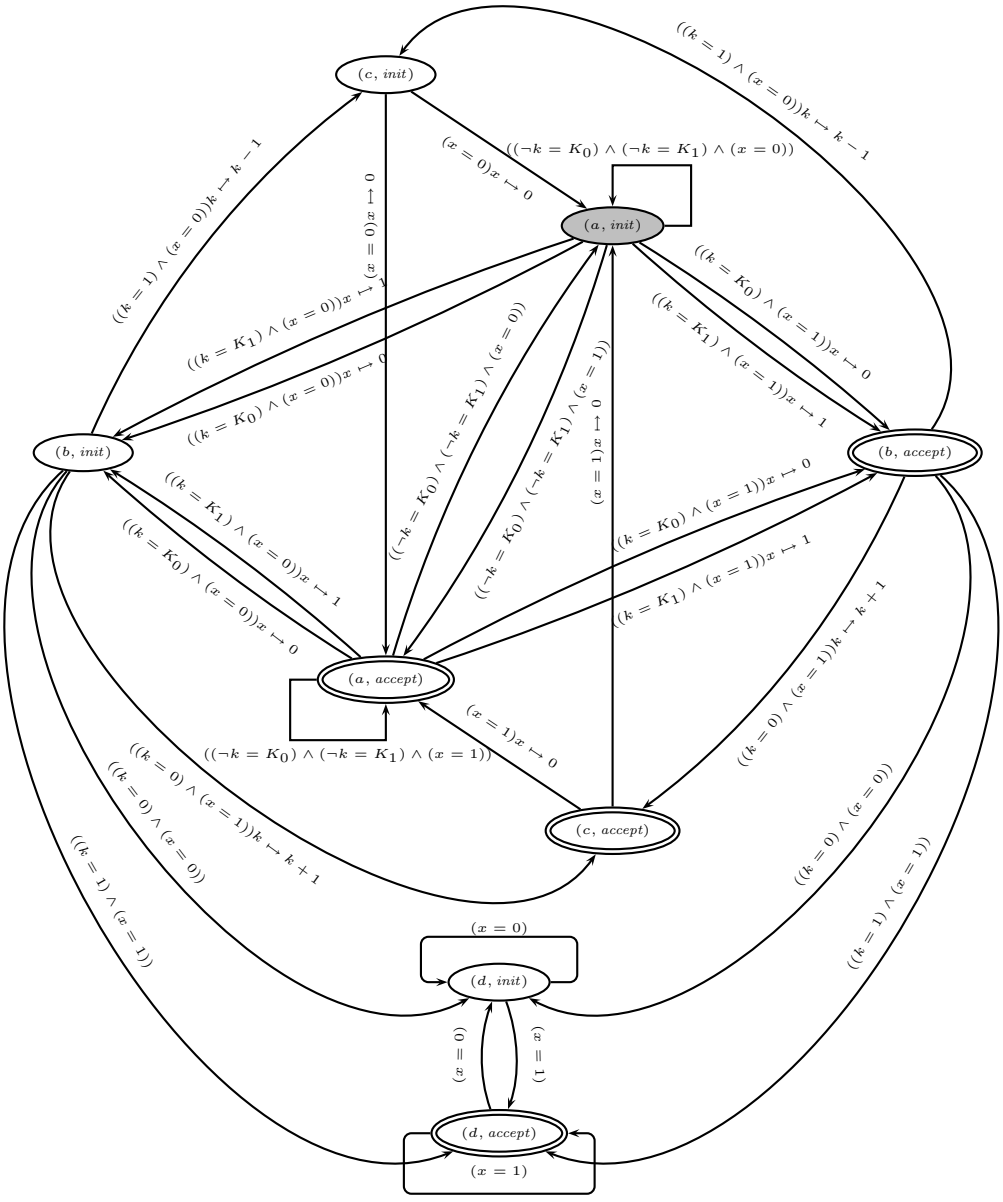


Fig. 4: $\mathscr{P}^0$, the product between $\mathscr{S}^0(P^0)$ and $B^0(\varphi^0)$

during the program execution using these symbols. An infinite *symbolic execution tree* can be built to represent all possible behaviours of the program: a path condition on the symbols expresses on which conditions on the initial values the computation denoted by the path can be executed. This technique has been previously used for symbolic transition systems [2]. In the case of PTS, we are interested in finding parameters which allow a certain behaviour, then parameters will be handled by symbolic execution techniques while attributes will be computed. Thus, as symbolic execution will hold on symbols denoting parameters, resulting path conditions will express constraints on parameters.

To make the computation on attributes possible, we restrict PTS assignments under the form $A \rightarrow \mathcal{T}(A)$ (instead of $\mathcal{T}(P \coprod A)$). To be able to enumerate all states of PTS models, we assume that values of attributes are bounded, that is for any attribute a, $\nu_A(a) \in [min_a, max_a]$. We denote by $\mathbb{B}$ the set of possible values for attributes.

4.1 Symbolic Execution Tree (SET)

A Symbolic Execution Tree, SET for short, is an infinite tree whose nodes are called *symbolic states* and edges *symbolic transitions*.

4.1.1 Symbolic states.

Given $\mathscr{S}(P) = (L, l_i, C_i, Tr, Ac)$ an APTS over A, a symbolic state is a triple (l, pc, ν_A) such that:

- $l \in L$ is a *location*;
- $pc \in Pres(P)$ is the *path condition*;
- $\nu_A : A \rightarrow \mathbb{B}$ is the *attribute interpretation*.

(l, pc, ν_A) is said *accepting* if $l \in Ac$.

The semantics of (l, pc, ν_A) is the set $[\![(l, pc, \nu_A)]\!]$ defined by:

$$[\![(l, pc, \nu_A)]\!] = \bigcup_{\nu_P \in \mathbb{Z}^P} \{(l, (\nu_P, \nu_A)) \in L \times (\mathbb{Z}^P \times \mathbb{B}^A) | \nu_P \models_V pc\}$$

In other words, $(l, (\nu_p, \nu_A)) \in [\![(l, pc, \nu_A)]\!]$ represents the state (l, ν_A) in the model $S(\nu_p)$. Thus, a symbolic state (l, pc, ν_A) represents the same model state (l, ν_A) in several PTS models $S(\nu_P)$, these models being the ones whose parameter interpretation ν_P satisfies the path condition pc of the symbolic state.

Example 9 (Symbolic state s^0)
Let us consider $\mathscr{S}^0_{\varphi^0}(P^0)$ the APTS introduced in Example 8. Let $s^0 = (l^0, pc^0, \nu^0_A)$ be a symbolic state of $\mathscr{S}^0_{\varphi^0}(P^0)$ with:

- $l^0 = (b, init)$;
- $pc^0 = (K_0 = 0 \vee K_0 = 1) \wedge (K_1 = 0)$;
- $\nu^0_A : k \mapsto 0, x \mapsto 0$.

By definition the semantics of s^0 is:

$$[\![((b, init), pc^0, \gamma^0)]\!] = \{ \quad ((b, init), (K_0 \mapsto 0, K_1 \mapsto 0, k \mapsto 0, x \mapsto 0)), \\ ((b, init), (K_0 \mapsto 1, K_1 \mapsto 0, k \mapsto 0, x \mapsto 0))\}$$

Thus, the semantics of s^0 corresponds to the state $((b, init), (k \mapsto 0, x \mapsto 0))$ in the two models $S(K_0 \mapsto 0, K_1 \mapsto 0)$ and $S(K_0 \mapsto 1, K_1 \mapsto 0)$ that is the two models $S(\nu_{P_0})$ and $S(\nu_{P_1})$.

4.1.2 Symbolic Execution Tree.

Nodes and edges of a SET are inductively defined, starting with the root. The root is the initial symbolic state, that is a symbolic state composed of the initial location of the PTS, the initial constraint, and an initial attribute interpretation.

Definition 9 (Symbolic Execution Tree)
Let $\mathscr{S}(P) = (L, l_i, C_i, Tr, Ac)$ be an APTS over A and let $\nu_{A_i} : A \to \mathbb{B}$ be an initial attribute interpretation s.t. $\nu_{A_i}(C_i)$ is satisfiable.

The symbolic execution tree associated to $\mathscr{S}(P)$ and ν_{A_i} is the tree $\mathscr{T}(\mathscr{S}(P), \nu_{A_i}) = (N, r, E)$ where:

- *$N \subseteq L \times Pres(P) \times \mathbb{B}^A$ is a set of symbolic states for $\mathscr{S}(P)$ called nodes;*
- *$r = (l_i, \nu_{A_i}(C_i), \nu_{A_i})$ is the root node;*
- *$E \subseteq N \times N$ is the set of edges.*
- *N and E are inductively defined by:*
  - *$r \in N$*
  - *for all $(l, pc, \nu_A) \in N$, for all $(l, g, \alpha, l') \in Tr$ s.t. $pc \wedge \nu_A(g)$ is satisfiable then*
    - *$(l', pc \wedge \nu_A(g), \nu_A \circ \alpha) \in N$*
    - *$((l, pc, \nu_A), (l', pc \wedge \nu_A(g), \nu_A \circ \alpha)) \in E$*

Let us remark that, for a given PTS, we can build as many SETs as ways to initially interpret attributes, considering that the PTS initial constraints remain satisfiable once the attributes have been instanciated.

Semantically, a SET for a given ν_{A_i} represents the set of runs of all PTS models $S(\nu_P)$ starting at the initial state (l_i, ν_{A_i}), i.e. s.t. $\nu_P \models_P \nu_{A_i}(C_i)$.

Note that, by construction of the SET, if there exists a path from $s = (l, pc, \nu_A)$ to $s' = (l', pc', \nu'_A)$, then there is a Presburger constraint pc^* such that $pc' = pc \wedge pc^*$, and then every parameter interpretation which satisfies pc' also satisfies pc. And so, $[\![s']\!] \subseteq [\![s]\!]$.

4.2 Folding of Symbolic Execution Tree

4.2.1 Return nodes.

Any SET of a PTS verify the two following properties: firstly, due to the completeness property of PTS, every node of a SET has a successor, and thus SET paths are infinite. Secondly, by hypothesis, the set of states of a PTS model is finite, and so each infinite path contains at least two nodes $s = (l, pc, \nu_A)$ and $s' = (l', pc', \nu'_A)$ s.t. $l = l'$ and $\nu_A = \nu'_A$. For each path of a SET, we focus on such nodes, more particularly on the nearest ones of the root node.

Definition 10 (Return and copy nodes of a SET)
Let $\mathscr{T}(\mathscr{S}(P), \nu_{A_i})$ be a SET. For any infinite path $(s_k)_{k\in\mathbb{N}}$ of $\mathscr{T}(\mathscr{S}(P), \nu_{A_i})$ with $s_k = (l_k, pc_k, \nu_{Ak})$ for all k, we call return node *the symbolic state $s_n = (l_n, pc_n, \nu_{An})$ s.t. n is the least integer verifying there exists $p < n$ s.t. $s_p = (l_n, pc_p, \nu_{An})$. s_p is called the* copy node *of s_n.*

Return nodes (l, pc, ν_A) hold the interesting property that parameter interpretations ν_P satisfying pc ensure that the model $S(\nu_P)$ admits at least a run containing infinitely often the state (l, ν_A). Intuitively, it suffices to consider a run built by repeating the states of the path between copy and return nodes, with the same successive values of locations and attributes.

Lemma 1 (Run associated to a return node)
Let $\mathscr{S}(P)$ be an APTS over A and let $(s_k)_{k\in\mathbb{N}}$ be a path of a SET of $\mathscr{S}(P)$ with $s_n = (l_n, pc_n, \nu_{An})$ its return node, and s_p its copy node.

If $\nu_P : P \to \mathbb{Z}$ is a parameter interpretation such that $\nu_P \models_V pc_n$, then $\sigma(s_n) = (\sigma_k)_{k\in\mathbb{N}}$ is a run of $S(\nu_P, \mathscr{S}(P))$ with for all $k \leq n$, $\sigma_k = (l_k, \nu_{Ak})$ and for all $k > n$, $\sigma_k = (l_{\eta_{p,n}(k)}, \nu_{A\,\eta_{p,n}(k)})$ where[6]

$$\eta_{p,n}(k) = p + 1 + ((k - (n+1))[n-p])$$

Proof 2 *Let $\nu_P : P \to \mathbb{Z}$ be a parameter interpretation such that $\nu_P \models_V pc_n$, and let $\sigma(s_n) = (\sigma_k)_{k\in\mathbb{N}}$ be defined as above. Then, $(\sigma_k)_{k\leq n}$ is a partial run of $S(\nu_P)$ since pc_n is satisfied. As successors of σ_n are included in the successors of σ_p by construction, then $\sigma(s_n)$ is a run of $S(\nu_P)$.*

4.2.2 Folding the SET.

Lemme 1 expresses that a return node characterises runs as soon as its path condition is satisfiable. In fact, the loop defined by the copy and return nodes allows us to consider runs following a minimal construction. Moreover, because the semantics of a return node is included in the semantics of the associated copy node, then necessarily, the set of return node's successors is included in the set of copy node's successors. Thus, any SET infinite path may be viewed as a sequence of several finite paths from a copy node to a return node.

We can fold the SET by cutting it at the level of return nodes, and considerind a kind of backtrack edges from the return nodes to the successors of their copy node. In other words, a path issued from the symbolic execution process can be seen either as an infinite path of the whole SET or a path on the folded structure. Let us remark that by construction, the path condition of an infinite path of the SET is equivalent to the conjunction of the path condition of all return nodes crossed in the traversal of the associated folded structure.

Example 10 (SETs for $\mathscr{S}^0_{\varphi^0}(P^0)$)
Fig. 5 and Fig. 6 show the two folded SETs of $\mathscr{S}^0_{\varphi^0}(P^0)$ for the initial attribute interpretations $(k \mapsto 0, x \mapsto 0)$ and $(k \mapsto 1, x \mapsto 0)$. The corresponding initial path condition is then $C = (K_0 = 0 \vee K_0 = 1) \wedge (K_1 = 0 \vee K_1 = 1)$.

Dashed transitions indicate that a return node has been reached, and precise the copy node. Accepting return nodes (see 4.3) are underlined.

4.3 Parametric model checking

4.3.1 Accepting return nodes.

In the context of APTS, we have to focus on accepting nodes; a return node will be considered as accepting if there is an accepting node in the loop defined by the copy and return nodes. Indeed, all nodes in this loop have the same role to make the runs accepted.

Definition 11 (Accepting return nodes)
Let $\mathscr{S}_\varphi(P)$ be an APTS and let $\mathscr{T}(\mathscr{S}_\varphi(P), \nu_{A_i})$ be a SET. A return node s of $\mathscr{T}(\mathscr{S}_\varphi(P), \nu_{Ai})$ is said to be accepting *iff at least one symbolic state between the copy node of s and s itself is accepting. The path condition of s is then called* accepting path condition.

[6] $n[k]$, also called n modulo k, is the remainder after numerical division of n by k.

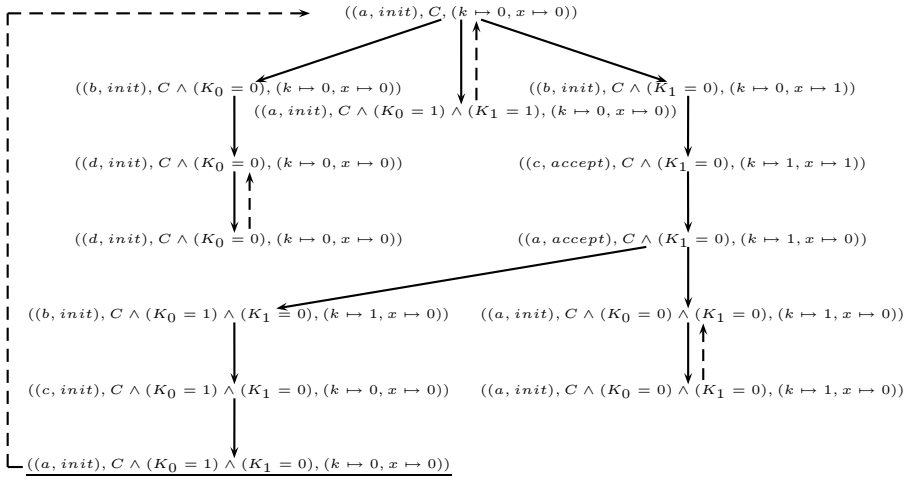


Fig. 5: SET of $\mathscr{S}^0_{\varphi^0}(P^0)$ for the initial attribute interpretation $(k \mapsto 0, x \mapsto 0)$

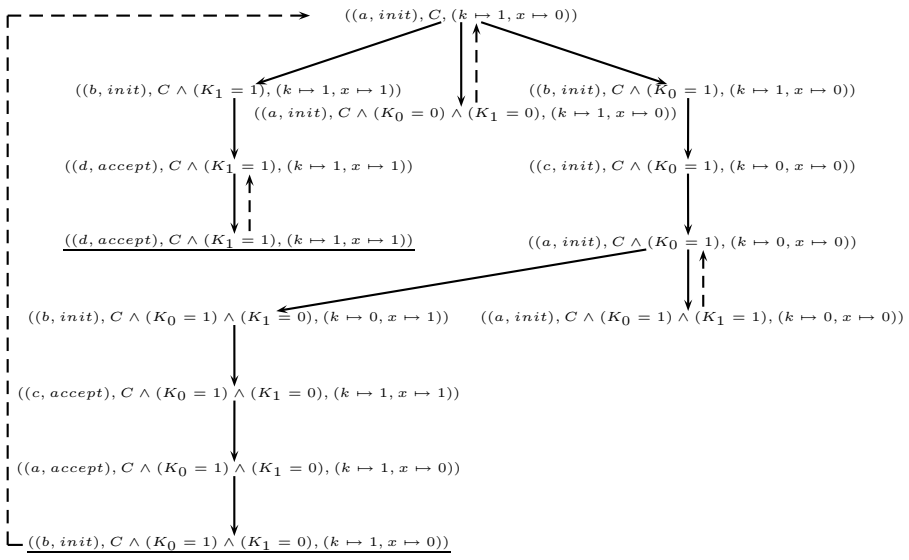


Fig. 6: SET of $\mathscr{S}^0_{\varphi^0}(P^0)$ for the initial attribute interpretation $(k \mapsto 1, x \mapsto 0)$

Example 11 (Accepting return nodes of $\mathscr{P}^0$ SETs)
Accepting return nodes of the two SETs of $\mathscr{S}^0_{\varphi^0}(P^0)$ are the underlined return nodes from Fig. 5 and Fig. 6

Let $s = (l, pc, \nu_A)$ be an accepting return node. Then, if $\nu_P \in \mathbb{Z}^P$ is a parameter interpretation such that $\nu_P \models_V pc$, then the run $\sigma(s)$ as built in Lemma 1 is an accepted run of $S(\nu_P)$.

4.3.2 Condition for an accepted path.

The following theorem expresses a necessary and sufficient condition on parameter and accepting path conditions of a SET to ensure that one path of the model associated to the parameter interpretation is accepted.

Theorem 1 (Disjonction of path conditions)
Let $\mathscr{S}(P) = (L, l_i, C_i, Tr)$ be a PTS, let φ be a Presburger-LTL formula, and let $\mathscr{S}_\varphi(P)$ be the APTS associated to $\mathscr{S}(P)$ and φ.

Let $\mathscr{T}(\mathscr{S}_\varphi(P), \nu_{Ai})$ be a SET with $pc_1, ..., pc_m$ be all its accepting path conditions. Let ν_P be a parameter interpretation, then there is a path in the model $S(\nu_P)$ starting at (l_i, ν_{Ai}) and satisfying φ iff $\nu_P \models_P \bigvee_{1 \leq i \leq m} pc_i$.

Proof 3 *We have already established that any ν_P satisfying an accepting path condition defined a model with at least an accepting run.*

Let σ be an accepted path in the model $S(\nu_P)$. Then it exists an accepting state $s = (l, \nu_A)$ of the model which appears infinitely often in σ. The folded SET being finite, there is a finite number of occurrences of s with different path conditions, and the SET path corresponding to σ passes infinitely often through at least one of these occurrences. Let s_j be such an occurrence, and let $r_1, ..., r_m$ the return nodes which are under s_j. At least one of the return nodes has a copy node which is over s_j (otherwise, the path cannot pass infinitely often through s_j) and at least one of these particular return nodes is passed infinitely often. Let r_m be such a node. Then, s_j is part of the loop between r_m and its copy node, and r_m is an accepting return node whose path condition is satisfied by ν_P.

4.3.3 Considering all the SETs.

A SET is associated to an initial attribute interpretation and describes the paths of all models, starting in the initial state defined by the initial interpretation. Then to verify all the paths of a model, we have to consider the accepting path conditions of all the SETs. There exists a model with (at least) one path verifying an LTL formula iff there exists a parameter interpretation which satisfies one accepting path condition of one SET. By contraposition: there exists a model with all paths verifying an LTL formula iff there exists a parameter interpretation which satisfies the conjunction of the negation of all non accepting path conditions for all the SETs.

In other words, by denoting $R(\nu_{Ai})$ the set of return nodes of $\mathscr{T}(\mathscr{S}(P), \nu_{Ai})$, $AR(\nu_{Ai})$ the set of accepting return nodes of $\mathscr{T}(\mathscr{S}(P), \nu_{Ai})$ and I_{ν_A} the set of possible initial attribute interpretations, we have:

- $S(\nu_p, \mathscr{S}(P))$ admits (at least) one run verifying φ iff

$$\nu_P \models_V \bigvee_{\nu_{Ai} \in I_{\nu_A}} \bigvee_{(l, pc, \nu_A) \in AR(\nu_{Ai})} pc$$

- All the runs of $S(\nu_p, \mathscr{S}(P))$ verify φ iff

$$\nu_P \models_V \bigwedge_{\nu_{Ai} \in I_{\nu_A}} \bigwedge_{(l, pc, \nu_A) \in R(\nu_{Ai}) \setminus AR(\nu_{Ai})} \neg pc$$

In section 5 we will focus on these conditions to model biological constraints.

Example 12 (Conditions on parameters of $\mathscr{S}^0(P^0)$)
A model $S(\nu_P)$ of $\mathscr{S}^0(P^0)$ admits (at least) one run verifying φ^0 iff

$$\nu_P \models_V (C \wedge (K_0 = 1) \wedge (K_1 = 0)) \vee (C \wedge (K_1 = 1)) \vee (C \wedge (K_0 = 1) \wedge (K_1 = 0))$$

Thus, $\nu_{P1} : (k \mapsto 1, x \mapsto 0)$, $\nu_{P2} : (k \mapsto 0, x \mapsto 1)$ and $\nu_{P3} : (k \mapsto 1, x \mapsto 1)$ lead to models with one run verifying φ^0.

A model $S(\nu_P)$ of $\mathscr{S}^0(P^0)$ is such that all its runs verify φ iff

$$\nu_P \models_V \neg(C \wedge (K_0 = 1)) \wedge \neg(C \wedge (K_0 = 0) \wedge (K_1 = 0)) \wedge \neg(C \wedge (K_0 = 1) \wedge (K_1 = 1))$$

And thus, $\nu_{P1} : (k \mapsto 1, x \mapsto 0)$ is the only parameter interpretation leading to a model with all paths verifying φ^0.

5 PTS for biological systems

In this section we focus on modelling of complex biological systems, more precisely on genetic regulatory networks. Generally, genetic regulatory networks are modelled by continuous differential systems in which parameters have to be inferred from a set of biological experiments. Unfortunately these parameters are rarely measurable and modelling process has to focus on the search of parameters values which lead to a dynamics which is coherent with experiments.

In order to address this kind of questions, we first focus on a discrete modelling of genetic regulatory network [14, 15]. The biological system is represented by an oriented graph (see Fig.7) where nodes abstract the proteins which play a role in the system and edges abstract the known interactions inside the considered system. With each node is associated a discrete concentration level which abstract its continuous concentration. Interaction $(a \to b)$ can be activation (the increase of a leads to an increase of b, the edge is labelled by the sign $+$ and a is an activator of b) or an inhibition (the increase of a leads to a decrease of b, the edge is labelled by the sign $-$ and a is an inhibitor of b). Moreover, when a regulator a acts on several targets, on b and c for exemple, it is often known that the level of a mandatory for an action on c is higher than the level necessary for the action of a on b. Thus edges are also labelled with a threshold below which the action is not effective. But such information is not sufficient to construct the complete dynamics of the systems since its dynamics is governed by a set of discrete parameters which express the directions of evolution of each concentration according to the current state.

Modeling framework. Several dynamics can be built from an interaction graph. The set of dynamic states are deduced from the labels of out-going edges. In fact, the set of the thresholds of edges out-going from a node x constitutes a set of the form $\{1, 2, ..., n\}$ and thus concentration of x can take $n + 1$ values : $0, 1, ...n$.

The evolution of the concentration of a node x depends on the value of a parameter depending of the state: $K(x, state)$. If the concentration of x is below $K(x, state)$, x can increase, if the concentration of x is above $K(x, state)$, x can decrease, otherwise x does not change. Actually parameters $(K(x, state))_{x,state}$ depend only on the set of present regulators (predecessors whose concentration is above the corresponding threshold).

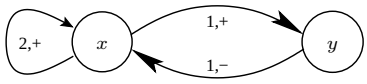


Fig. 7: Interaction graph for the mucus production system in *P. aeruginosa*

Model checking for determining suitable models. To go further, it can be useful to take into account known properties of the system. For example it can be known that if a particular protein is not expressed (its discrete level is equal to 0) then an event is not possible. Such a property can be translated into LTL formulas: $(a = 0) \Rightarrow \neg F(event)$. The main activity of modelling, consisting in searching parameters values which lead to behaviours coherent with observed properties, can so be aided by model-ckecking of parametric transition systems presented hereinbefore.

We then present the use of PTS through the example of mucus production in *Pseudomonas aeruginosa*. We give the main results for the classical epigenetic switch in bacteriophage lambda.

5.1 Mucus production system in P. aeruginosa

Pseudomonas aeruginosa are bacteria that secrete mucus (alginate) in lungs affected by cystic fibrosis, but not in common environment. As this mucus increases respiratory defficiency, this phenomenon is a major cause of mortality. Details of the regulatory network associated with the mucus production by *Pseudomas aeruginosa* are described by Govan and Deretic [3] but a simplified genetic regulatory network has been proposed by Guespin and Kaufman [4], see Fig.7.

It has been observed that mucoid *P. aeruginosa* can continue to produce mucus isolated from infected lungs. It is commonly thought that the mucoid state of *P. aeruginosa* is due to a mutation which cancels the inhibition of gene x. An alternative hypothesis has been made: this mucoid state can occur by reason of an epigenetic modification, *i.e.* without mutation [4]. The models compatible with this hypothesis are constructed in [1]. We show here that PTS are useful for extracting such set of models.

5.1.1 Parametric Transition System.

The PTS associated with the interaction graph is defined as follows.

- The set of attributes is constituted by the concentrations associated with nodes x and y, which are also noted x and y by abuse.
- The set of locations is $\{T, S\}$. The location S specifies the stable steady states where no concentration evolves and so the condition for entering into S is that neither x nor y can evolve.
- The set of parameters is $\{K(x,\{\}), K(x,\{x\}), K(x,\{y\}), K(x,\{x,y\}), K(y,\{\}), K(y,\{x\})\}$, since x has two predecessors and y only one.
- The initial location is "T" and the initial constraints are:

$$\begin{array}{lllll} & 0 \leq K(x,\{\}) \leq 2 & \wedge & 0 \leq K(x,\{x\}) \leq 2 \\ \wedge & 0 \leq K(x,\{y\}) \leq 2 & \wedge & 0 \leq K(x,\{x,y\}) \leq 2 \\ \wedge & 0 \leq K(y,\{\}) \leq 1 & \wedge & 0 \leq K(y,\{x\}) \leq 1 \end{array}$$

- Finally transitions are represented in Fig. 8 where guards are expressed using the notation $K(x, state)$ which denotes the parameter in the current state *state*.

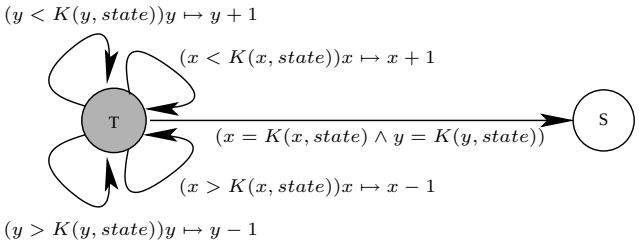


Fig. 8: PTS for the mucus production system in *P. aeruginosa*

Each guard can be translated into a semantically equivalent Presburger formula. For example, $x < K(x, state)$ is equivalent to

$$\begin{array}{ll} & (x < K(x, \{\}) \wedge x < 2 \wedge y < 1) \\ \vee & (x < K(x, \{x\}) \wedge x = 2 \wedge y < 1) \\ \vee & (x < K(x, \{y\}) \wedge x < 2 \wedge y = 1) \\ \vee & (x < K(x, \{x, y\}) \wedge x = 2 \wedge y = 1) \end{array}$$

Remarks. The initial constraints can specify some information available on the system. For exemple, it is possible to code in these initial constraints that we are looking for parameterizations which satisfy the Snoussi's constraints[12]: larger the set of effective activators, larger the focal point. In other words, if in a configuration the concentration of a node is increasing (resp. decreasing), then the addition of another activator (resp. inhibitor) does not lead to a decreasing (resp. increasing) of the concentration.

5.1.2 Specification of behaviours by Presburger-LTL formulas.

From biological knowledge we can state that *P. aeruginosa* does not produce mucus in a common environment, so there is no path from a state where $x = 0$ to a state where $x = 2$. In other words, the formula $\varphi \equiv \neg((x = 0) \wedge \mathbf{F}(x = 2))$ has to be evaluated to true on all paths. The associated constraint generated by our method is $K(x, \emptyset) \leq 1$.

5.1.3 Results.

If the hypothesis of an epigenetic change in mucoid *P. aeruginosa* is verified, bacteria which produce mucus can continue to produce mucus in a common environment. A path beginning with $(x = 2)$ which turns back forever to $(x = 2)$ is described by $((x = 2) \wedge G(F(x = 2)))$ which has to be evaluated to true on all paths. Our method leads to the constraints $((K(x, \{x, y\}) = 2 \wedge K(y, \{x\}) = 1) \vee (K(x, \{x\}) = 2 \wedge K(y, \{x\}) = 0))$. If we assume that $K(x, \{y\}) = K(y, \{\}) = 0$ (when no activator is present and when all the inhibitors are present, then the concentration of the considered variable is decreasing to 0) and that parameters verify the Snoussi constraints [12], 8 interesting models are to be considered for further analysis [10].

5.2 Bacteriophage lambda

Bacteriophage lambda is a virus whose DNA can integrate into bacterial chromosome and be faithfully transmitted to the bacterial progeny. After infection, most of the bacteria display a *lytic* response and liberate new phages, but some display a *lysogenic* response, *i.e.* survive and carry lambda genome, becoming immune to infection.

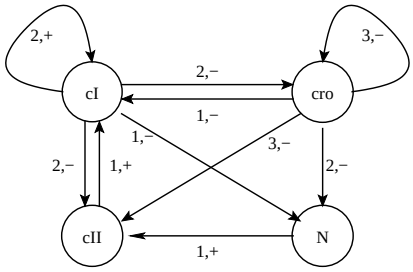


Fig. 9: Interaction graph for the lambda switch

Figure 9 presents the interaction graph described by Thieffry and Thomas [13] which has also been studied in [11]. 4 genes (called cI, cro, cII and N) and 10 interactions are involved, with 24 parameters. The states, represented by a vector (cI, cro, cII, N), are in $\{0,1,2\} \times \{0,1,2,3\} \times \{0,1\} \times \{0,1\}$. The set of possible instantiations of parameters is enormous ($3^8 \times 4^4 \times 2^8 \times 2^4 = 6.879.707.136$) since cI (resp. cro, cII, N) has 3 regulators (resp. 2, 3, 2). Thus we are in the situation where the parametric transition system represents a large number of rather small models (each model is a transition system with $3 \times 4 \times 2 \times 2 = 48$ states).

5.2.1 Parametric Transition System.

The PTS associated with the interaction graph of Fig. 9 is defined as follows.

- The set of attributes is constituted by the concentrations cI, cro, cII and N associated to the different nodes.
- The set of locations is $\{T, S\}$.
- The set of parameters is

$$\left\{\begin{array}{llll} K(cI,\{\}), & K(cI,\{cI\}), & K(cI,\{cro\}), & K(cI,\{cII\}), \\ K(cI,\{cI,cro\}), & K(cI,\{cI,cII\}), & K(cI,\{cro,cII\}), & K(cI,\{cI,cro,cII\}), \\ K(cro,\{\}), & K(cro,\{cI\}), & K(cro,\{cro\}), & K(cro,\{cI,cro\}) \\ K(cII,\{\}), & K(cII,\{cI\}), & K(cII,\{cro\}), & K(cII,\{N\}), \\ K(cII,\{cI,cro\}), & K(cII,\{cI,N\}), & K(cII,\{cro,N\}), & K(cII,\{cI,cro,N\}), \\ K(N,\{\}), & K(N,\{cI\}), & K(N,\{cro\}), & K(N,\{cI,cro\}) \end{array}\right\}$$

- The initial location is "T" and the initial constraints are:

$$\begin{array}{l} \bigwedge_{\omega \subseteq G^{-1}(cI)} 0 \leq K(cI,\omega) \leq 2 \quad \wedge \\ \bigwedge_{\omega \subseteq G^{-1}(cro)} 0 \leq K(cro,\omega) \leq 3 \quad \wedge \\ \bigwedge_{\omega \subseteq G^{-1}(cII)} 0 \leq K(cII,\omega) \leq 1 \quad \wedge \\ \bigwedge_{\omega \subseteq G^{-1}(N)} 0 \leq K(N,\omega) \leq 1 \end{array}$$

  where $G^{-1}(x)$ denotes the set of regulators of x.
- Finally transitions are represented in the figure 10. Actually, the guards are expressed using the notation $K(x, state)$ which denotes the parameter to be applied to x in the current state $state$. Each guard of the figure can be translated into a semantically equivalent formula of $Pres(V)$ (see example of *P. aeruginosa*).

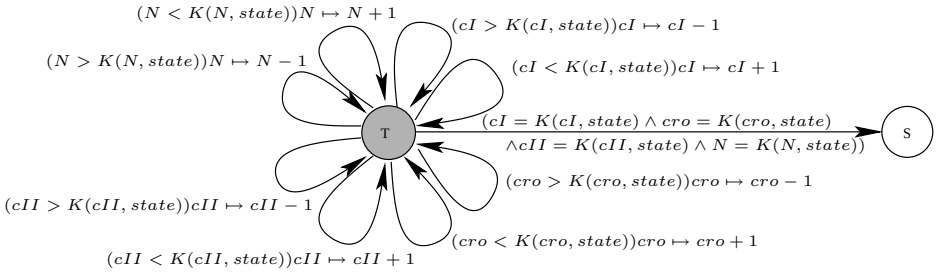


Fig. 10: PTS for the lytic/lisogenic switch in phage lambda

5.2.2 Specification of behaviors by Presburger-LTL formulas.

We used Presburger-LTL formulas to specify two kinds of biological knowledge: static and dynamic.

Static knowledges describe the states of interest, that is the protein concentration corresponding to lytic or lysogenic states, and the fact that viral proteins are absent when the viral genome integrates a cell.

- The lytic response leads to the states where *cro* is fully expressed, and other genes repressed. To specify that the system is in one of these states, we use the following formula, called *lytic*:

$$lytic = (cI = 0 \wedge cro \geq 2 \wedge cII = 0 \wedge N = 0)$$

- The lysogenic response leads to the state where cI is fully expressed, and the repressor produced by cI blocks the expression of the other viral genes, leading to immunity. This state corresponds to following formula, called *lysogenic*:

$$lysogenic = (cI = 2 \wedge cro = 0 \wedge cII = 0 \wedge N = 0)$$

- The viral proteins are initially absent when the viral genome integrates a cell. The system is in this initial state if it verifies the following *init* formula:

$$init = (cI = 0 \wedge cro = 0 \wedge cII = 0 \wedge N = 0)$$

Dynamic knowledges express that lytic and lysogenic states are stable, and that these states can be reached from the initial state.

- When a lytic state is reached, the system does not leave the set of lytic states, thus any path of searched models must verify:

$$\mathcal{P}_1 : \quad \neg(lytic \wedge F(\neg lytic))$$

- Similarly, the stability of the lysogenic state is equivalent to the following property which must be verified by any path:

$$\mathcal{P}_2 : \quad \neg(lysogenic \wedge F(\neg lysogenic))$$

- As lytic and lysogenic responses are possible from the initial state, it means that there exists at least a path from initial state to lytic states, and at least a path from initial state to lysogenic state. These properties are translated into:

$$\mathcal{P}_3 : \quad init \wedge F(lytic)$$

$$\mathcal{P}_4 : \quad init \wedge F(lysogenic)$$

5.2.3 Results.

The models we are interested in are the ones which admit at least one path verifying $\mathcal{P}_3$ and $\mathcal{P}_4$, and all paths verifying $\mathcal{P}_1$ and $\mathcal{P}_2$.

- The condition on $\mathcal{P}_1$ leads to following constraints on parameters:

  $$C_1 = (K(cI, \{cro\}) = 0) \wedge (K(cro, \{\}) > 1) \wedge (K(cII, \{\}) = 0) \wedge (K(N, \{cro\}) = 0)$$

- Similarly, $\mathcal{P}_2$ leads to the constraint:

  $$C_2 = (K(cI, \{cI\}) = 2) \wedge (K(cro, \{cI\}) = 0) \wedge (K(cII, \{cI\}) = 0) \wedge (K(N, \{cI\}) = 0)$$

- Focusing on $\mathcal{P}_3$, we remark that all models satisfying C_1 and C_2 have a path verifying $init \wedge F(lytic)$, in other words, the information given by $\mathcal{P}_3$ is already contained in $\mathcal{P}_1$ and $\mathcal{P}_2$.
- Finally property $\mathcal{P}_4$ leads to the constraint:

  $$C_4 = (K(cI, \{\}) = 2) \vee ((K(cI, \{cII\}) = 2) \wedge (K(cII, \{N\}) = 1) \wedge (K(N, \{\}) = 1))$$

The set of interesting parameter interpretations is then reduced from more than 6.8 billions to 2156 [9], leading scientists to focus directly on models which are coherent with current knowledge on the dynamics of the system.

6 Concluding remarks

A Parametric Transition System represents a set of models as Kripke structures in a concise manner. Thus, PTS are particularly useful when the modelled system is highly parametrical that is when the number of possible parameter interpretations is greater than the number of attribute interpretations. If we consider the example of Bacteriophage lambda, we have to deal with 24 parameters and 4 attributes, what leads to a large number of rather small models. This situation is common for genetic regulatory networks, which makes PTS perfectly adapted for their studies. Moreover, results are easy to read and manipulate due to their symbolic representations for both formulas and constraints.

A promising follow-up is to model and analyse larger biological networks, in order to prove that our approach can avoid the state space combinatorial explosion, common in the study of genetic regulatory networks. We are also interested in developing the applications of PTS to other domains, issued from Software Engineering for example. A remaining challenge is to avoid, or at least to delay, the states enumeration, that is to make the SET more symbolic.

References

[1] 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, 2004.

[2] C. Bigot, A. Faivre, J.-P. Gallois, A. Lapitre, D. Lugato, J.-Y. Pierron, and N. Rapin. Automatic test generation with agatha. In H. Garavel and J. Hatcliff, editors, *TACAS*, volume 2619 of *Lecture Notes in Computer Science*, pages 591–596. Springer, 2003.

[3] J. Govan and V. Deretic. Microbial pathogenesis in cystic fibrosis: mucoid pseudomonas aeruginosa and burkholderia cepacia. *Microbiol rev.*, 60(3):539–74, 1996.

[4] J. Guespin-Michel and M. Kaufman. Positive feedback circuits and adaptive regulations in bacteria. *Acta Biotheor.*, 49(4):207–218, 2001.

[5] G. Holzmann. *SPIN model checker : primer and reference manual*. Addison-Wesley, 2003.

[6] W. Kelly, V. Maslov, W. Pugh, E. Rosser, T. Shpeisman, and D. Wonnacott. *The Omega library interface guide*. College Park, MD, USA, http://www.cs.umd.edu/projects/omega, 1995.

[7] J. C. King. Symbolic execution and program testing. *Communication of the ACM*, 19(7):385–394, 1976.

[8] G. Kreisel and J. Krivine. *Elements of mathematical logic*. North-Holland Publishing Co., 1967.

[9] D. Mateus, J.-P. Comet, J.-P. Gallois, and P. Le Gall. *Proc. of the Evry Spring school on Modelling and simulation of biological processes in the context of genomics*, chapter Inferring parameters of genetic regulatory networks with symbolic formal methods, pages 51–70. EDP Science, ISBN : 978-2-7598-0019-3, 2007.

[10] D. Mateus, J.-P. Gallois, J.-P. Comet, and P. Le Gall. Symbolic modeling of genetic regulatory networks. *Journal of Bioinformatics and Computational Biology*, 2007.

[11] A. Richard, J.-P. Comet, and G. Bernot. *Modern Formal Methods and Applications*, chapter Formal methods for modeling biological regulatory networks, pages 83–122. Springer, 2006.

[12] E. Snoussi. Qualitative dynamics of a piecewise-linear differential equations : a discrete mapping approach. *Dynamics and stability of Systems*, 4:189–207, 1989.

[13] D. Thieffry and R. Thomas. Dynamical behaviour of biological regulatory networks immunity control in bacteriophage lambda. *Bull. Math. Biol.*, 57(2):277–97, 1995.

[14] R. Thomas. Logical analysis of systems comprising feedback loops. *J. Theor. Biol.*, 73(4):631–56, 1978.

[15] R. Thomas and R. d'Ari. *Biological Feedback*. CRC Press, 1990.

[16] M. Y. Vardi. An automata-theoretic approach to linear temporal logic. In *Banff Higher Order Workshop*, volume 1043 of *Lecture Notes in Computer Science*, pages 238–266. Springer, 1995.

[17] P. Wolper. Constructing automata from temporal logic formulas: A tutorial. In Springer, editor, *Lectures on Formal Methods and Performance Analysis: First EEF/Euro Summer School*, volume 2090 of *LNCS*, pages 261–277, 2001.