Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 272 (2011) 43–56
www.elsevier.com/locate/entcs

Abstract Interpretation of Dynamics of Biological Regulatory Networks

Loïc Paulevé[1] Morgan Magnin[2] Olivier Roux[3]

IRCCyN, UMR CNRS 6597, École Centrale de Nantes, France

Abstract

Analysing dynamics of large biological regulatory networks (BRNs) calls for innovative methods to cope with the state space explosion. Static analysis and abstract interpretation techniques seem promising approaches. In this paper, we address the *Process Hitting* framework, that has been shown of interest to model dynamics of BRNs with discrete values. We propose to take profit from the particular structures of Process Hitting to build efficient static analyses. We introduce a novel and original method to decide the reachability of the state of a component within a BRN modelled in Process Hitting. The decision is achieved by abstract interpretation and static analysis of Process Hittings. The scalability of our approach is illustrated by its application to the analysis of a BRN with 40 components.

Keywords: biological regulatory networks, model checking, abstract interpretation, static analysis.

1 Introduction

Biological regulatory networks (BRNs) are a common way to model regulations between biological components (RNA, proteins, etc.). These regulations are often represented as interaction graphs, where nodes are components of the system, and edges state the regulation between them, either positive (activation) or negative (inhibition). To each node is also assigned a numerical value representing the state (e.g. the concentration) of the component of the network, at a given time. Then, this value evolves in response to the various regulations the component is subject to. In 1973, the biologist René Thomas proposed a formalisation of BRNs where the value of components are boolean [11]. This formalisation uses an interaction graph and René Thomas' parameters (or equivalently, boolean functions between nodes inputs) to describe dynamics of a BRN. The full formalisation of BRNs with discrete values for components can be found in [2].

[1] Email: loic.pauleve@irccyn.ec-nantes.fr
[2] Email: morgan.magnin@irccyn.ec-nantes.fr
[3] Email: olivier.roux@irccyn.ec-nantes.fr

doi:10.1016/j.entcs.2011.04.004

The derivation of dynamical properties from the interaction graph of BRNs has been the motivation of various mathematical works. Twenty years ago, René Thomas conjectured that the presence of positive circuits within the interaction graph is a necessary condition to achieve systems with multi-stationnarity. The conjecture has been proven in several frameworks, notably in discrete dynamical systems [9]. By using more elaborated interaction graph analyses, the maximum number of fixed points within boolean networks has been characterised [1]. Similarly, the presence of negative circuits in interaction graphs has been proved necessary for sustained oscillations in the dynamics [8].

To produce more precise analyses of BRNs dynamics, it is then required to take into account the boolean functions specified together with the interaction graph (as in [3], for instance). The majority of current techniques use standard model-checking methods [10], that are based on state space explorations of the model. However, such methods suffer from the state space explosion, and are intractable on large regulatory networks.

The Process Hitting [7] is a recently introduced framework suitable to model BRNs with discrete values. Basically, each discrete component value is modelled as a process; at any time, one and only one process of each component (referred to as *sort*) is present; this process stands for the current state of the component. A component changes of process on the *hit* of at most one other process. Static analyses have already been developed in the Process Hitting framework, notably for obtaining all the fixed points of dynamics of a Process Hitting [7].

We present a novel abstract interpretation method on Process Hittings to decide the reachability of a process, i.e. the level of a component within a BRN. This decision may be inconclusive, however. Our approach, illustrated in Fig. 1, builds an over-approximation of the reachability decision, allowing to quickly detect negative cases. Then, concretions of this approximation are derived. Under specific conditions, the concretions are proved to be correct under-approximations of the reachability decision, leading to a positive decision. This reasoning prevents the explicit expression of the state space.

This original approach takes advantage of the particular structure of Process Hitting models. Its scalability is shown by its application to the decision of reachability of gene expression levels within a BRN of 40 components.

This paper is structured as follows. The Process Hitting framework is formally defined in Sect. 2. The abstract interpretation of process reachability is detailed in Sect. 3. The application of the presented method to the analysis of a large BRN is done in Sect. 4. We discuss our contribution in Sect. 5.

2 The Process Hitting Framework

This section presents the *Process Hitting* framework on which the methods presented in this paper apply. More detailed definitions can be found in [7].

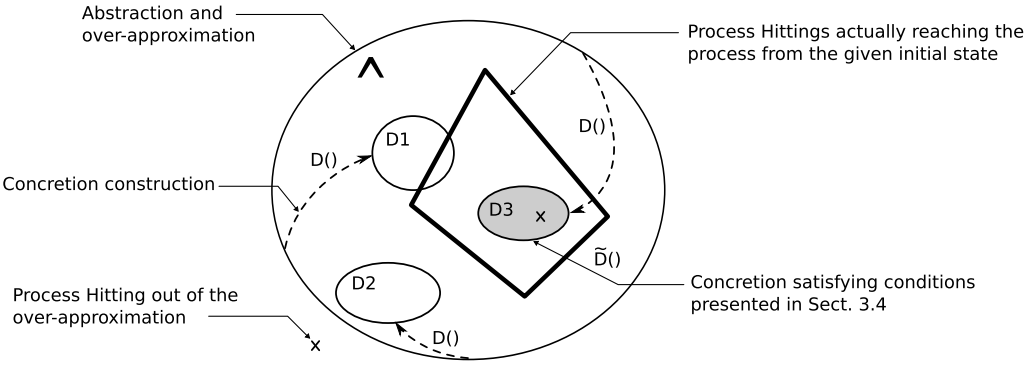

Figure 1. The space of Process Hittings. The polygon is the set of Process Hittings that can reach the given process from a given initial state. An over-approximation of this set (marked "$\widehat{\ }$") is first built using an abstraction operation. Based on this abstraction, concretions are constructed (here, D1, D2 and D3). If a concretion satisfies the condition presented in Sect. 3.4, it lies within the polygon (here, D3, greyed).

2.1 Process Hitting Definition

The Process Hitting gathers a finite number of concurrent *processes* grouped into a finite set of *sorts*. A process belongs to one and only one sort and is noted a_i where a is the sort and i the identifier of the process within the sort a. At any time, one and only one process of each sort is present, forming a state of the Process Hitting. The concurrent interactions between processes are defined by a set of *actions*. Actions describe the replacement of a process by another of the same sort conditioned by the presence of at most one other process in the current state of the Process Hitting. An action is denoted by $a_i \to b_j \not\uparrow b_k$ where a_i, b_j, b_k are processes of sorts a and b. It is required that $b_j \neq b_k$ and that if $a = b$, then $a_i = b_j$. An action $\alpha = a_i \to b_j \not\uparrow b_k$ is read as "a_i *hits* b_j to make it bounce to b_k", and a_i, b_j, b_k are called respectively *hitter*, *target* and *bounce* of the action, and can be referred to as $hitter(\alpha), target(\alpha), bounce(\alpha)$ where $hitter(a_i \to b_j \not\uparrow b_k) = a_i$, $target(a_i \to b_j \not\uparrow b_k) = b_j$ and $bounce(a_i \to b_j \not\uparrow b_k) = b_k$.

Definition 2.1 [Process Hitting] A *Process Hitting* $\mathcal{PH}$ is a triple $(\Sigma, L, \mathcal{H})$:

- $\Sigma = \{a, b, \dots\}$ is the finite countable set of sorts,
- $L = \prod_{a \in \Sigma} L_a$ is the set of states for $\mathcal{PH}$, with $L_a = \{a_0 \dots a_{l_a}\}$ the finite and countable set of processes of sort $a \in \Sigma$ and l_a a positive integer, $a \neq b \Rightarrow a_i \neq b_j \ \forall (a_i, b_j) \in L_a \times L_b$,
- $\mathcal{H} = \{a_i \to b_j \not\uparrow b_k, \cdots \mid (a, b) \in \Sigma^2,\ (a_i, b_j, b_k) \in L_a \times L_b \times L_b, b_j \neq b_k, a = b \Rightarrow a_i = b_j\}$, is the finite set of actions.

Given a state $s \in L$, the process of sort $a \in \Sigma$ present in s is denoted by s_a, that is the a-coordinate of the state s. The sort of a process a_i is referred to as $\Sigma(a_i)$ and the set of sorts present in an action $\alpha = a_i \to b_j \not\uparrow b_k$ as $\Sigma(a_i \to b_j \not\uparrow b_k) = \{\Sigma(a_i), \Sigma(b_j)\}$.

An action $\alpha = a_i \to b_j \not\uparrow b_k \in \mathcal{H}$ is *playable* in $s \in L$ if and only if $s_a = a_i$ and $s_b = b_j$. In such a case, $(s \cdot \alpha)$ stands for the state resulting from the play of the action α in s, that is $(s \cdot \alpha)_b = b_k$ and $\forall c \in \Sigma, c \neq b, (s \cdot \alpha)_c = s_c$. For the sake of clarity, $((s \cdot \alpha) \cdot \beta)$ is abbreviated as $(s \cdot \alpha \cdot \beta)$.

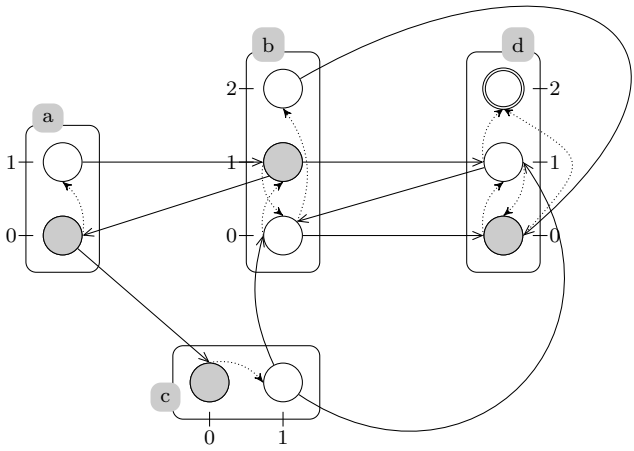

Figure 2. A toy Process Hitting. Sorts are represented by labeled boxes, and processes by circles (ticks are the identifiers of the processes within the sort, for instance, a_0 is the process ticked 0 in the box a). An action (for instance $b_1 \rightarrow a_0 \nearrow a_1$) is represented by a directed hyperedge, having the hit part (b_1 to a_0) in plain line and the bounce part (a_0 to a_1) in dotted line. The reachability of the process d_2 (double circled) is studied in next sections. The current state is represented by the grey processes, i.e. $\langle a_0, b_1, c_0, d_0 \rangle$.

Fig. 2 represents a Process Hitting $(\Sigma, L, \mathcal{H})$ where $\Sigma = \{a, b, c, d\}$, $L = \{a_0, a_1\} \times \{b_0, b_1, b_2\} \times \{c_0, c_1\} \times \{d_0, d_1, d_2\}$ and $\mathcal{H} = \{a_0 \rightarrow c_0 \nearrow c_1, a_1 \rightarrow b_1 \nearrow b_0, c_1 \rightarrow b_0 \nearrow b_1, b_1 \rightarrow a_0 \nearrow a_1, b_0 \rightarrow d_0 \nearrow d_1, b_1 \rightarrow d_1 \nearrow d_2, d_1 \rightarrow b_0 \nearrow b_2, c_1 \rightarrow d_1 \nearrow d_0, b_2 \rightarrow d_0 \nearrow d_2\}$. Playing the action $b_1 \rightarrow a_0 \nearrow a_1$ in the state $\langle a_0, b_1, c_0, d_0 \rangle$ results in the state $\langle a_1, b_1, c_0, d_0 \rangle$.

2.2 Scenarios and Bounce Sequences

This subsection defines two specific compositions of actions: scenarios and bounce sequences. Both are sequences of actions, i.e. an ordered list of actions. Briefly, a scenario is a sequence of actions that are successively playable in some states of the Process Hitting. A bounce sequence is a sequence of actions that permits to make a process bounce to another of the same sort.

Definition 2.2 [Sequence of Actions] Given a Process Hitting $(\Sigma, L, \mathcal{H})$, a *sequence of actions* is a series $A = \alpha_1, \ldots, \alpha_x$, where $\forall y, 1 \leq y \leq x, \alpha_y \in \mathcal{H}$. $|A|$ denotes the length of the sequence (i.e. $|A| = x$). By notation, $A_y = \alpha_y, 1 \leq y \leq |A|$. An empty sequence is referred to as ε.

$A_{i..j}$ is the subsequence of actions $A_i, \ldots, A_j$. The set of sorts present in a sequence A is denoted by $\Sigma(A) = \bigcup_{y=1}^{|A|} \Sigma(A_y)$. A process a_i is said to be present in the sequence A, noted $a_i \in A$, if and only if there exists an action in A where a_i is either the hitter or the target or the bounce. Given a sequence of actions A, $prev_a^y(A)$ denotes the latter process of sort a that is either a bounce or a hitter of

an action $A_u, u < y$, formally defined below.

$$prev_a^y(A) = \begin{cases} \varnothing & \text{if } a \notin \Sigma(A_{1..y-1}), \\ bounce(A_u) & \text{if } u = \max\{v \mid a \in \Sigma(A_v) \wedge 1 \leq v < y\} \\ & \wedge \Sigma(bounce(A_u)) = a, \\ hitter(A_u) & \text{if } u = \max\{v \mid a \in \Sigma(A_v) \wedge 1 \leq v < y\} \\ & \wedge \Sigma(bounce(A_u)) \neq a. \end{cases}$$

A scenario δ is a sequence of actions where the hitter (resp. the target) a_i of the y^{th} action is either occurring for the first time ($prev_a^y(\delta) = \varnothing$), or is the latter process of sort a occurring in preceding actions ($prev_a^y(\delta) = a_i$).

Definition 2.3 [Scenario] A *scenario* δ is a sequence of actions such that for all y, $1 \leq y \leq |\delta|$, $a_i = hitter(\delta_y)$ (resp. $target(\delta_y)$) $\Rightarrow prev_a^y(\delta) \in \{\varnothing, a_i\}$.

A scenario δ is said to be playable in a state $s \in L$, if and only if δ_1 is playable in s and for all y, $1 \leq y < |\delta|$, δ_{y+1} is playable in the state $(s \cdot \delta_1 \cdot \ldots \cdot \delta_y)$. The state resulting from the sequential play of the scenario in s is denoted by $s \cdot \delta$. The definition of the reachability of a process from a given state can be formalised using scenarios:

Definition 2.4 [Process Reachability] Given a state $s \in L, s_z \neq z_l$, s can reach the process z_l if and only if there exists a scenario δ playable in s such that $(s \cdot \delta)_z = z_l$.

Bounce sequences result from a local reasoning on a single sort a. Bouncing from a_i to a_j may require the play of several actions on processes of sort a, that form a *bounce sequence*. By notation, a_i and a_j are called respectively the *target* and the *bounce* of the *objective* $a_i \Rsh^* a_j$; the sort of the objective is $\Sigma(a_i \Rsh^* a_j) = a$. The set of all objectives within any sort is denoted by $\mathcal{O}bj$.

Definition 2.5 [Objective; Bounce Sequence] The reach of process a_j from a_i is called an *objective*, noted $a_i \Rsh^* a_j$. A *bounce sequence* ζ is a sequence of actions such that for all $y, 1 \leq y < |\zeta|$, $bounce(\zeta_y) = target(\zeta_{y+1})$. It *resolves* an objective $a_i \Rsh^* a_j$ if and only if $target(\zeta_1) = a_i$ and $bounce(\zeta_{|\zeta|}) = a_j$.

In a bounce sequence ζ, target and bounces of all actions share the same sort $\Sigma(\zeta)$. In the scope of this paper, bounce sequences do not contain cycles between targets and bounces of actions. In that way, the maximum length of a bounce sequence for a sort a is the number of processes of sort a.

Definition 2.6 [$\mathcal{BS}$] $\mathcal{BS}(a_i \Rsh^* a_j)$ denotes the set of all bounce sequences resolving the objective $a_i \Rsh^* a_j$. Because bounce sequences do not admit cycles, this set is finite. Obviously, $\mathcal{BS}(a_i \Rsh^* a_i) = \{\varepsilon\}$; and $\mathcal{BS}(a_i \Rsh^* a_j) = \emptyset$ if there is no possibility to reach a_j from a_i.

Remark 2.7 Bounce sequences are not necessarily scenarios (e.g. $b_i \rightarrow a_i \Rsh a_j, b_j \rightarrow a_j \Rsh a_k$ is a bounce sequence but not a scenario if $b_i \neq b_j$).

For the Process Hitting example in Fig. 2, $\zeta = a_1 \to b_1 \not\Rsh b_0, d_1 \to b_0 \not\Rsh b_2$ is the only bounce sequence resolving the objective $b_1 \not\Rsh^* b_2$ (i.e. $\mathcal{BS}(b_1 \not\Rsh^* b_2) = \{\zeta\}$). $\delta = a_0 \to c_0 \not\Rsh c_1, b_1 \to a_0 \not\Rsh a_1, a_1 \to b_1 \not\Rsh b_0, b_0 \to d_0 \not\Rsh d_1, c_1 \to b_0 \not\Rsh b_1$ is a scenario playable in the state $s = \langle a_0, b_1, c_0, d_0 \rangle$, and $s \cdot \delta = \langle a_1, b_1, c_1, d_1 \rangle$.

3 Abstract Interpretation of Process Reachability

The *Process Reachability* problem consists in deciding, within a Process Hitting, if there exists a scenario playable in a given state leading to a state containing a given process (Def. 2.4). This section assumes a Process Hitting $(\Sigma, L, \mathcal{H})$, a state $s \in L$ and a process $z_l \in L_z, z \in \Sigma$ for which the reachability from s has to be decided.

We propose an abstract interpretation that aims at replying quickly the process reachability problem. Based on an abstraction of bounce sequences, we first build an over-approximation of the decision (responding either by the negative or inconclusive); then, concretions are derived from the abstraction and, if satisfying a certain property, are shown to be valid under-approximations of the decision, allowing then to potentially give a positive answer.

3.1 Abstraction of Bounce Sequences

A bounce sequence ζ resolving an objective P is abstracted into the set of hitters of its actions having a different sort than that of P. This abstraction is noted $\zeta^\wedge$ and is given by (1). The abstraction $\mathcal{BS}^\wedge(P)$, P being an objective, is then the set of its abstracted sequences (2). For the sake of efficiency, only minimal abstracted sequences are kept.

$$\zeta^\wedge = \{hitter(\zeta_y) \mid 1 \leq y \leq |\zeta|, \Sigma(hitter(\zeta_y)) \neq \Sigma(P)\} \ , \tag{1}$$

$$\mathcal{BS}^\wedge(P) = \{\zeta^\wedge \mid \zeta \in \mathcal{BS}(P), \nexists \zeta' \in \mathcal{BS}(P), \zeta'^\wedge \subsetneq \zeta^\wedge\} \ . \tag{2}$$

It is worth noticing that $\mathcal{BS}^\wedge(P)$ can be computed directly (without computing $\mathcal{BS}(P)$), providing still a computation exponential in the number of processes in the sort of P, but yet more efficient because only minimal sets are kept.

3.2 Over-Approximation of Process Reachability

The proposed over-approximation defines the satisfiability of a bounce sequence ζ as the (independent) reachability from the initial state s of the involved hitters (i.e. processes in $\zeta^\wedge$). The set of objectives from the initial state to each of these hitters is denoted by $objs(\zeta^\wedge)$:

$$objs(\zeta^\wedge) = \{s_a \not\Rsh^* a_j \mid a_j \in \zeta^\wedge\} \ . \tag{3}$$

Hence, $objs(\zeta^\wedge)$ is said to be *satisfiable* if for each of its objectives $P \in objs(\zeta^\wedge)$, the abstraction of bounce sequences $\mathcal{BS}^\wedge(P)$ is satisfiable. In a natural way, $\mathcal{BS}^\wedge(P)$ is said to be *satisfiable* if and only if there exists an abstracted bounce sequence

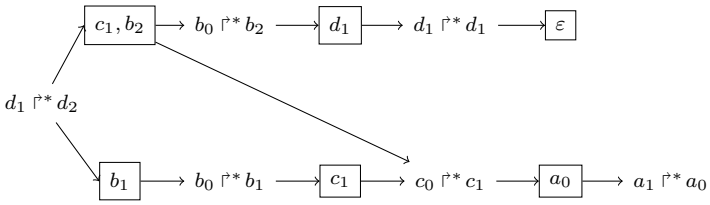

Figure 3. Graph of relations between $\mathcal{BS}\hat{}(P)$ and $objs(\zeta\hat{})$ sets that are connected to $\mathcal{BS}\hat{}(d_1 \Rsh^* d_2)$ for the Process Hitting example of Fig. 2 in the state $\langle a_1, b_0, c_0, d_1\rangle$. Boxed nodes represents the abstracted sequences $\zeta\hat{}$ resolving the parent objective. In this case, $\mathcal{BS}\hat{}(d_1 \Rsh^* d_2) \equiv \bot$ so d_2 is not reachable from $\langle a_1, b_0, c_0, d_1\rangle$.

$\zeta\hat{} \in \mathcal{BS}\hat{}(P)$ such that $objs(\zeta\hat{})$ is satisfiable. The non-satisfiability of $\mathcal{BS}\hat{}(P)$ is noted $\mathcal{BS}\hat{}(P) \equiv \bot$ and is formally defined by the equations below.

$$\mathcal{BS}\hat{}(P) \equiv \bot \iff \forall \zeta\hat{} \in \mathcal{BS}\hat{}(P), objs(\zeta\hat{}) \equiv \bot \tag{4}$$

$$objs(\zeta\hat{}) \equiv \bot \iff \exists P \in objs(\zeta\hat{}), \mathcal{BS}\hat{}(P) \equiv \bot \tag{5}$$

It is an over-approximation since it requires the reachability of an unordered set of hitters which is necessary but not sufficient to resolve the objective.

Lemma 3.1 *$\mathcal{BS}\hat{}(s_z \Rsh^* z_l) \equiv \bot \Longrightarrow z_l$ is not reachable from s.*

Proof $\exists$ scenario δ, $(s \cdot \delta)_z = z_l \Rightarrow \mathcal{BS}\hat{}(s_z \Rsh^* z_l) \not\equiv \bot$, by induction on δ (see Appendix A.1). □

Testing if $\mathcal{BS}\hat{}(s_z \Rsh^* z_l) \equiv \bot$ is done by simply traversing the graph of relations between $\mathcal{BS}\hat{}(P)$ and $objs(\zeta\hat{})$ sets that are connected to $\mathcal{BS}\hat{}(s_z \Rsh^* z_l)$.

Fig. 3 illustrates this lemma on the Process Hitting example of Fig. 2.

3.3 Concretions of Abstraction of Bounce Sequences

This subsection presents a concretion $\mathcal{D}$ of the abstraction $\mathcal{BS}\hat{}$ previously defined. The concretion $\mathcal{D}$ is a partial order relation between objectives, and its construction may lead to several solutions. Once the satisfiability of $\mathcal{D}$ is defined, the question of the link between this satifiability and process reachability is tackled in the next subsection.

The concretion $\mathcal{D}$ of $\mathcal{BS}\hat{}$. The concretion is first done by fixing for each objective P one bounce sequence ζ to execute, i.e. by reducing every $\mathcal{BS}\hat{}$ to one element. Finally, each objective $Q \in objs(\zeta\hat{})$ is put in relation with P in $\mathcal{D} \subset \mathcal{O}bj \times \mathcal{O}bj$. It is worth noticing that relation cycles can be prohibited, as it is discussed in the satisfaction of $\mathcal{D}$. In that way, $\mathcal{D}$ has the structure of a partial order between objectives, with $s_z \Rsh^* z_l$ as minimal element. The set of objectives that are in relation by transitivity with the objective P is noted $\mathcal{C}(P)$. The following steps illustrate the concretion of $\mathcal{BS}\hat{}$ into $\mathcal{D}$:

(i) for each $\mathcal{BS}\hat{}(P) \neq \emptyset$, choose $\zeta\hat{} \in \mathcal{BS}\hat{}(P)$, and add (P, Q) to $\mathcal{D}$ for each $Q \in objs(\zeta\hat{})$;

(ii) remove from $\mathcal{D}$ any (P, Q) such that $P \notin \mathcal{C}(s_z \Rsh^* z_l)$;

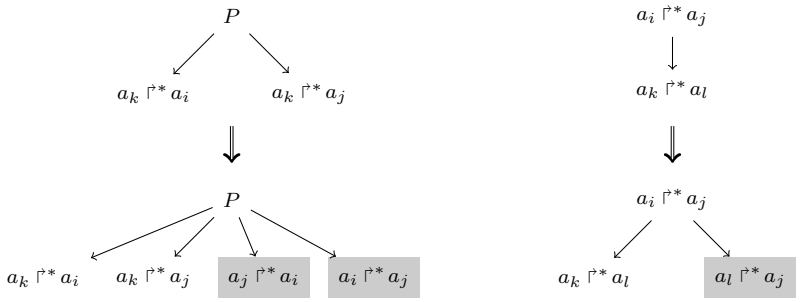

Figure 4. Sketch of the saturation procedure. There is an edge from P to Q if P depends on Q. (left) Concurrent objectives: objectives ensuring a cycle between bounces are added as dependencies. (right) Target redirection: objective from the bounce of the dependence to the bounce of the root is added as a dependency.

(iii) ignore solution if there exists a relation cycle.

Computing a concretion $\mathcal{D}$ is linear in the number of objectives connected to $s_z \Rsh^* z_l$. The number of possible concretions is finite and is exponential in the cardinalities of $\mathcal{BS}^\wedge$ sets. If $(P, Q) \in \mathcal{D}$, P is said to *depend* on Q (and Q is a dependency of P).

$\mathcal{D}$ satisfiability. The concretion $\mathcal{D}$ of an objective P is said to be satisfiable if and only if, either the empty bounce sequence resolves P ($\mathcal{BS}^\wedge(P) = \{\varepsilon\}$), i.e. P does not depend on any other objective; or each objective Q that is a dependence of P is satisfiable. This is denoted $\mathcal{D}(P) \not\equiv \bot$:

$$\mathcal{D}(P) \not\equiv \bot \iff \mathcal{BS}^\wedge(P) = \{\varepsilon\} \text{ or } \forall Q, (P, Q) \in \mathcal{D}, \mathcal{D}(Q) \not\equiv \bot \ . \tag{6}$$

Hence, it is obvious that if $\mathcal{D}$ contains a cycle, $\mathcal{D}(s_z \Rsh^* z_l) \equiv \bot$.

3.4 Under-Approximation of Process Reachability

If two different objectives having a same sort are in relation with $s_z \Rsh^* z_l$ in $\mathcal{D}$, resolving one of both may result in a state in which the resolution of the other may be impossible. This is basically an issue of objective resolutions interleaving. This subsection proposes a construction of a relation $\widetilde{\mathcal{D}}$ upon $\mathcal{D}$ that is satisfiable when the order of objective resolutions does not influence the process reachability.

The main principle, illustrated in Fig. 4, is to saturate the relation $\mathcal{D}$ in order to ensure that every possible scheduling of objective resolution is satisfiable. For instance, if $a_k \Rsh^* a_i$ and $a_k \Rsh^* a_j$ have to be concurrently resolved, the saturation procedure requires that both $a_i \Rsh^* a_j$ and $a_j \Rsh^* a_i$ are satisfiable, i.e. there is a cycle between the presence of a_i and a_j. Besides, if resolving $a_i \Rsh^* a_j$ implies the prior resolution of $a_k \Rsh^* a_l$, the saturation procedure requires $a_l \Rsh^* a_j$ to be satisfiable too.

First, let us denote by $TOP_a(P)$ the first objectives in $\mathcal{D}$ of sort a in relation with P. It is defined by the following equation where $Succ(P) = \{Q \in \mathcal{O}bj \mid (P, Q) \in \mathcal{D}\}$:

$$TOP_a(P) = \begin{cases} \{P\} & \text{if } \Sigma(P) = a; \\ \bigcup_{Q \in Succ(P)} TOP_a(Q) & \text{otherwise.} \end{cases} \tag{7}$$

The relation $\mathcal{D}$ fully saturated for the objective P is denoted by $\widetilde{\mathcal{D}}(P)$. Three

properties are verified by $\widetilde{\mathcal{D}}(P)$:

(i) $\forall Q \in Succ(P), \widetilde{\mathcal{D}}(Q) \subset \widetilde{\mathcal{D}}(P)$;

(ii) For all sorts a, $\forall Q, R \in Succ(P)$ where Q and R have a different sort than P and $TOP_a(Q)$ and $TOP_a(R)$ are not empty, then, there exist $a_i \Rsh^* a_j, a_{j'} \Rsh^* a_{i'} \in Succ(P)$ where a_i and $a_{i'}$ (resp. a_j and $a_{j'}$) are bounces in $TOP_a(Q)$ (resp. $TOP_a(R)$).

(iii) If a is the sort of P, $\forall Q \in Succ(P), \Sigma(Q) \neq a$, if there is at least one $a_k \Rsh^* a_l \in TOP_a(Q)$, then $a_l \Rsh^* bounce(P) \in Succ(P)$.

The saturation procedure works by recursive saturation of dependencies of $s_z \Rsh^* z_l$, by using the rules sketched in Fig. 4 to satisfy the properties established above. As the maximum number of objectives is the sum of the square of the number of processes for each sort, the complexity of the saturation procedure is polynomial with the number of processes.

Lemma 3.2 *$\widetilde{\mathcal{D}}(s_z \Rsh^* z_l) \not\equiv \bot \Longrightarrow z_l$ is reachable from s.*

Proof By induction on the partial order $\widetilde{\mathcal{D}}$ (see Appendix A.2). □

Testing the satisfiability of a concretion is done linearly with the number of objectives in relation in the concretion (6). In this way, performing the analysis by applying the Lemma 3.2 is very efficient.

The saturation procedure fails as soon as a needed objective Q has no solving bounce sequences ($\mathcal{BS}(Q) = \emptyset$). This failure uncovers a partial order constraint upon the occurrence of processes: if $\mathcal{BS}(a_i \Rsh^* a_j) = \emptyset$, the process a_j is always present before a_i. A future work may make use of this knowledge to refine the static decision.

To illustrate this scheduling analysis, let us consider the reachability of the process d_2 in the Process Hitting in Fig. 2. Considering first $\langle a_1, b_1, c_1, d_0\rangle$ as initial state, a satisfiable saturated concretion is obtained, as shown in Fig. 5(left), so d_2 is reachable from this state. However, when considering the initial state $\langle a_0, b_1, c_1, d_0\rangle$, the saturation of concretions is not satisfiable because of the empty $\mathcal{BS}(a_1 \Rsh^* a_0)$ (Fig. 5(right)), uncovering a partial order (a_0 can not be reached after a_1).

4 Application to a Large BRN

From BRNs to Process Hittings. We first sketch how to model a discrete BRN in the Process Hitting framework. Basically, to each component corresponds a sort, and to each state of components corresponds a process. If a component a at state i activates a component b at state j, an action $a_i \rightarrow b_j \Rsh b_k$ is added, where b_k is the state of b after activation. The inhibition is modelled similarly. The realisation of boolean functions between nodes are modelled using a dedicated sort, and is illustrated in Fig. 6. The full formalisation of this translation can be found in [7].

T-Cell Receptor Signalling Pathway. Introduced in [4], this biological system models the T-Cell Receptor (TCR) signalling pathway, the behaviours of which

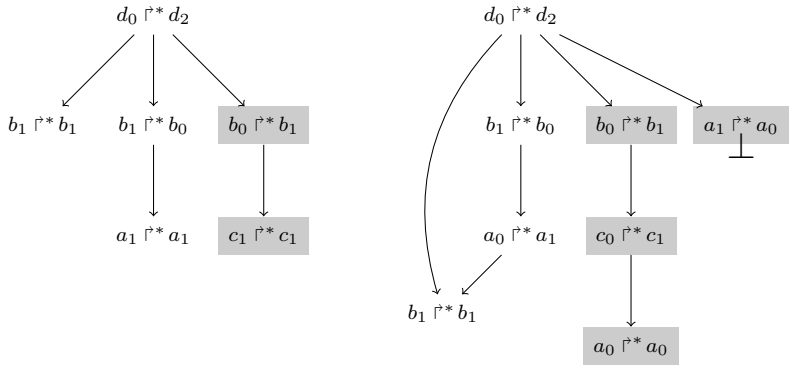

Figure 5. Saturated concretions obtained when testing the reachability of d_2 in the Process Hitting in Fig. 2. Nodes added by the saturation procedure are greyed. (left) Concretion computed from $\langle a_1, b_1, c_1, d_0\rangle$ that is satisfiable (appearing as D3 in Fig. 1). (right) Concretion computed from $\langle a_0, b_1, c_0, d_0\rangle$ that is not satisfiable (appearing as D1 or D2 in Fig. 1).

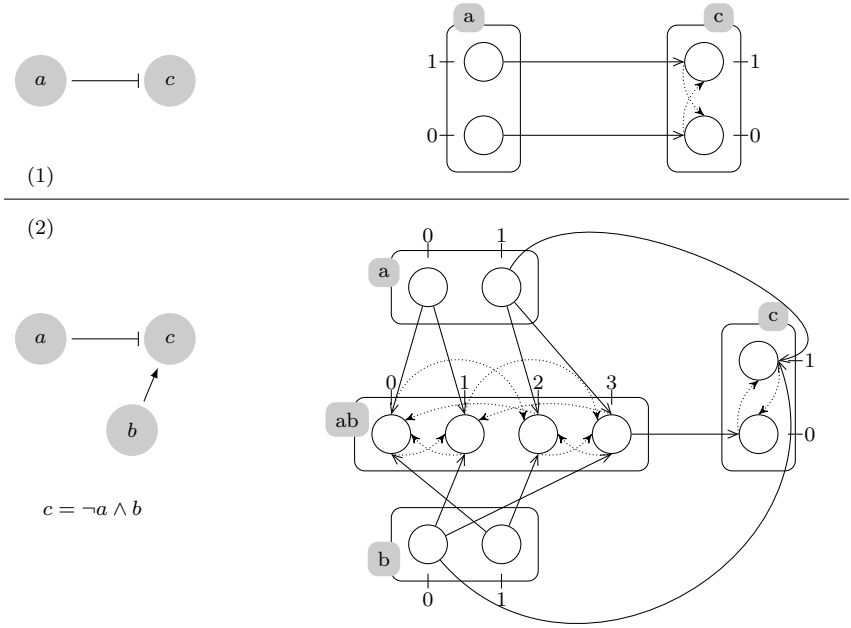

Figure 6. Examples of Process Hittings (right) from BRNs, having interaction graphs at left. (1) simple inhibition of c by a. (2) boolean function between a (inhibitor) and b (activator) on c ab reflects the state of sorts a and b. In this case, ab_3 reflects the state $\langle a_0, b_1\rangle$ (and, ab_0 the state $\langle a_1, b_0\rangle$, ab_1 the state $\langle a_1, b_1\rangle$, ab_2 the state $\langle a_0, b_0\rangle$).

reveal an activation of transcription factors controlling the cell's fate, e.g. whether it proliferates or not. The interaction graph (reproduced in Appendix B.1), together with logical rules can be found in [6], which studies the feedbacks circuits and stable states of this network.

The Process Hitting model [4] of this system is composed of 416 actions between 176 processes split in 53 sorts (the largest sort has 32 processes). The total number

[4] Model and implementation available at http://www.irccyn.ec-nantes.fr/~pauleve/sasb10-ProcessHitting.tar.bz2

of states of this model is 2^{73} ($\approx 10^{22}$).

Reachability decisions have been experimented from all possible inputs combinations (components CD45, CD8, TCRlig) to each output (components CRE, AP1, NFkB, NFAT). All result in conclusive decisions. The response times are around the second on a 3GHz processor with 2GB of RAM. To give a comparison, we did the same experiments with a model-checking method using state-space compression: the libddd framework [5], known for its good performances. For many reachability decisions, the program run out of memory, for others, response times range from some seconds to hours. This shows the remarkable efficiency of our method, based on abstract interpretation.

5 Discussion

The Process Hitting is a recently proposed framework suitable for modelling dynamics of BRNs with discrete values. In Process Hitting, components are represented as sorts, and their levels as processes; at any time, one and only one process of each sort is present. The successive states of a component within the system are enclosed in a so-called sort. The replacement of a process by another of the same sort (i.e. level change of a component), is conditioned by the presence of at most one other process, of any sort.

Thanks to the particular structure of Process Hittings models, a powerful static analysis and abstract interpretation method has been developed to decide the reachability of a process, hence of a component level in the scope of BRNs modelling. The computation is done by over- and under-approximation of the decision, and may reveal to be inconclusive. Further improvements of our procedure are currently being done to limit the number of inconclusive cases (Sect. 3.4).

This new and original approach has been applied to the analysis of a large BRN (40 components). Response times are really fast (around the second on a desktop computer), showing the scalability of our method. Application to ever much larger networks (100 components) are expected shortly.

Our next research direction is the incorporation of quantitative aspects within the presented decision of process reachability, such as the probability of reaching a given process in a given time interval.

References

[1] Aracena, J., *Maximum number of fixed points in regulatory boolean networks*, Bulletin of Mathematical Biology **70** (2008), pp. 1398–1409.

[2] Bernot, G., F. Cassez, J.-P. Comet, F. Delaplace, C. Müller and O. Roux, *Semantics of biological regulatory networks*, Electronic Notes in Theoretical Computer Science **180** (2007), pp. 3 – 14.

[3] Bernot, G., J.-P. Comet and Z. Khalis, *Gene regulatory networks with multiplexes*, in: *European Simulation and Modelling Conference Proceedings*, 2008, pp. 423–432.

[4] Klamt, S., J. Saez-Rodriguez, J. Lindquist, L. Simeoni and E. Gilles, *A methodology for the structural and functional analysis of signaling and regulatory networks*, BMC Bioinformatics **7** (2006), p. 56.

[5] LIP6/Move, *the libDDD environment* (2007), http://ddd.lip6.fr.

[6] Naldi, A., D. Thieffry and C. Chaouiya, "Computational Methods in Systems Biology," 2007 pp. 233–247.

[7] Paulevé, L., M. Magnin and O. Roux, *Refining Dynamics of Gene Regulatory Networks in a Stochastic π-Calculus Framework*, Transactions in Computational Systems Biology **To appear** (2010), preprint: http://www.irccyn.ec-nantes.fr/~pauleve/refining-revised.pdf.

[8] Remy, É., P. Ruet and D. Thieffry, *Graphic requirements for multistability and attractive cycles in a boolean dynamical framework*, Advances in Applied Mathematics **41** (2008), pp. 335 – 350.

[9] Richard, A. and J.-P. Comet, *Necessary conditions for multistationarity in discrete dynamical systems*, Discrete Applied Mathematics **155** (2007), pp. 2403 – 2413.

[10] Richard, A., J.-P. Comet and G. Bernot, "Modern Formal Methods and Applications," 2006 pp. 83–122.

[11] Thomas, R., *Boolean formalization of genetic control circuits*, Journal of Theoretical Biology **42** (1973), pp. 563 – 585.

A Proofs

Proofs assume z_l as the process to which the reachability from an initial state s has to be decided.

A.1 Lemma 3.1

Let us assume that z_l is reachable from s using the scenario δ.

If $|\delta| = 0$ then $s_z = z_l$ and $\varepsilon \in \mathcal{O}bj(s_z \Rsh^* z_l)$, therefore $\mathcal{BS}\hat{}(s_z \Rsh^* z_l) \not\equiv \bot$.

If $|\delta| > 0$ then there exists a bounce sequence ζ resolving $s_z \Rsh^* z_l$ such that to each $i, 1 \leq i \leq |\zeta|$ corresponds a $x_i, 1 \leq x_i \leq |\delta|$ where $\zeta_i = \delta_{x_i}$ and $x_i < x_j, \forall i < j$. Moreover, $\forall i, 1 \leq i \leq |\zeta|, \mathit{hitter}(\zeta_i)$ is reachable from s using the scenario $\delta' = \delta_1, \ldots, \delta_{x_i - 1}$ (induction with $s_z = \mathit{hitter}(\zeta_i)$ and $\delta = \delta'$, where $|\delta'| < |\delta|$). □

A.2 Lemma 3.2

Let EXECUTE be the following recursive function building a scenario to reach the bounce of an objective $a_i \Rsh^* a_j$ from a given state s:

function EXECUTE($s, a_i \Rsh^* a_j$)
  $\zeta \leftarrow$ bounce sequence matching the selected $\zeta\hat{} \in \mathcal{BS}\hat{}(a_i \Rsh^* a_j)$
  for $x \leftarrow 1$ **to** $|\zeta|$ **do**
    $b \leftarrow \Sigma(\mathit{hitter}(\zeta_x))$
    if $s_b \neq \mathit{hitter}(\zeta_x)$ **then**
      $s \leftarrow$ EXECUTE($s, s_b \Rsh^* \mathit{hitter}(\zeta_x)$)
      if $s_a \neq \mathit{target}(\zeta_x)$ **then** */s_a has changed/*
        return EXECUTE($s, s_a \Rsh^* a_j$)
      end if
    end if
    $s_a \leftarrow \mathit{bounce}(\zeta_x)$ */play of action ζ_x/*
  end for
  return s
end function

By construction of $\widetilde{\mathcal{D}}(s_z \Rsh^* z_l)$, every objectives of recursive calls to EXECUTE are in relation with $s_z \Rsh^* z_l$ in $\widetilde{\mathcal{D}}(s_z \Rsh^* z_l)$. As $\widetilde{\mathcal{D}}(s_z \Rsh^* z_l)$ is a partial order, no looping recursive calls are possible, so the function always terminates. Moreover, if $\widetilde{\mathcal{D}}(s_z \Rsh^* z_l) \not\equiv \bot$, ζ is always defined. Therefore, by induction on the partial order $\widetilde{\mathcal{D}}(s_z \Rsh^* z_l)$, it is then proved that EXECUTE$(s, s_z \Rsh^* z_l)$ returns a state where z_l is present. □

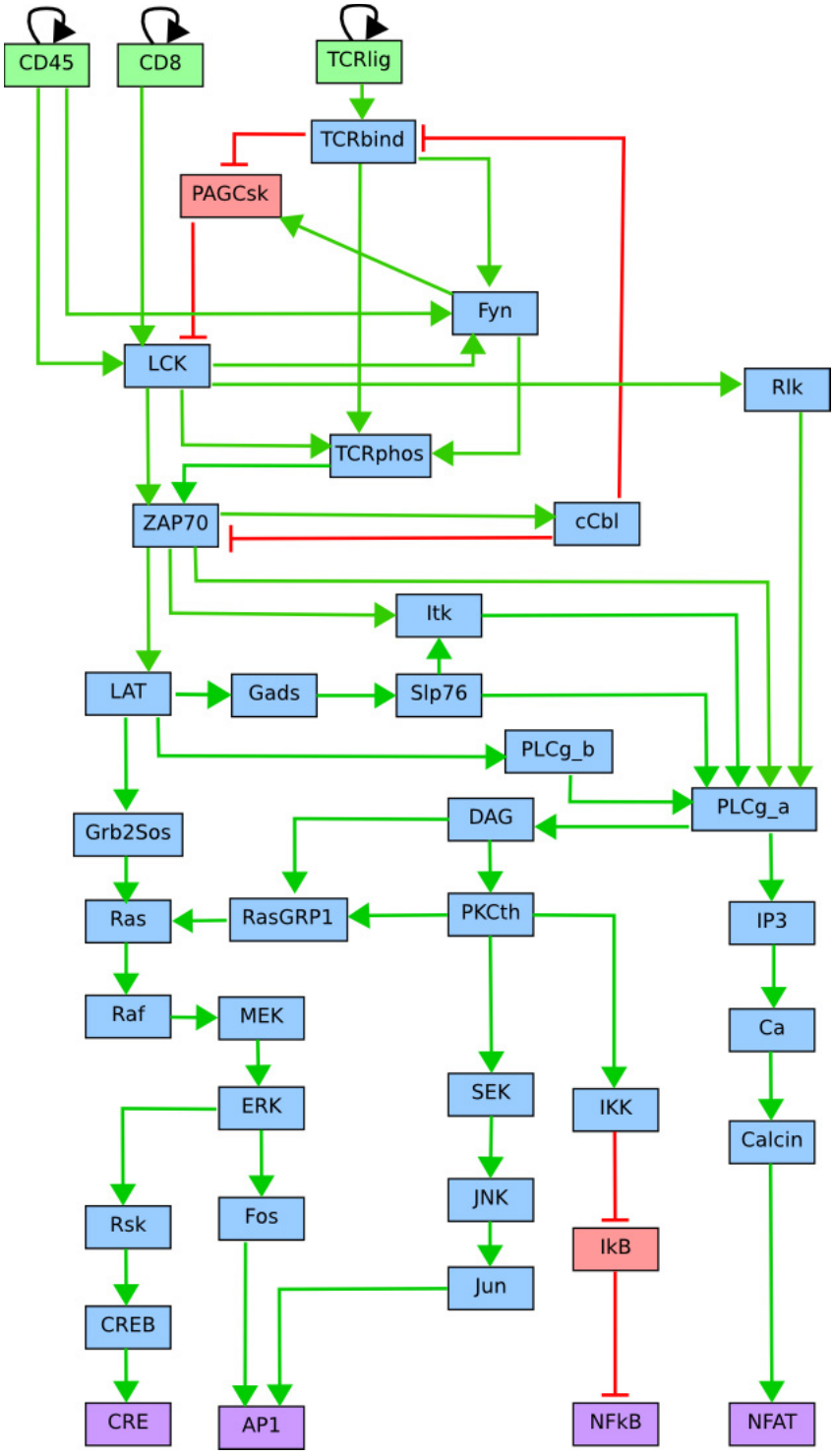

Figure B.1. Interaction graph for the TCR signalisation pathway [6].