Static Analysis by Abstract Interpretation of Biological Regulatory Networks Dynamics

Loïc Paulevé, Morgan Magnin, Olivier Roux

Under consideration for publication in Math. Struct. in Comp. Science

Static Analysis by Abstract Interpretation of Biological Regulatory Networks Dynamics

LOÏC PAULEVÉ, MORGAN MAGNIN, OLIVIER ROUX

IRCCyN, UMR CNRS 6597
École Centrale de Nantes, France.
{loic.pauleve,morgan.magnin,olivier.roux}@irccyn.ec-nantes.fr

Received 7th March 2011

The analysis of dynamics of Biological Regulatory Networks (BRNs) requires innovative methods to cope with the state space explosion. This paper settles an original approach for deciding reachability properties based on the *Process Hitting*, a framework suitable to model dynamical complex systems. In particular, the Process Hitting has been shown of interest to model dynamics of BRNs with discrete values. The Process Hitting describes the way each process is able to act upon (i.e. to "hit") an other one (or itself) in order to "bounce" it as another process further acting. By using complementary abstract interpretations of the succession of actions, we build a very efficient static analysis to over- and under-approximate reachability properties within Process Hittings. Applied to a large BRN of 94 components, our method replies quasi-instantaneously to reachability questions, overtaking the state-of-the-art approaches and showing a very promising scalability.

Contents

1. Introduction

Biological regulatory networks (BRNs) are a common framework to model the concurrent regulations between biological components (RNA, proteins, etc.). These regulations are generally 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. 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 (Thomas, 1973). His formalisation uses an interaction graph and René Thomas' parameters (or equivalently, boolean functions between nodes inputs) to describe dynamics of a BRN. A full description of BRNs formalism with discrete values for components can be found in (Bernot, Cassez, Comet, Delaplace, Müller and Roux, 2007).

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 (Richard and Comet, 2007). By using more elaborated interaction graph analyses, the maximum number of fixed points within boolean networks can be characterised (Aracena, 2008). Under strong conditions, particular fixed points (qualified as topological) can be fully extracted from the interaction graph (Paulevé and Richard, 2010): these points are fix in all possible dynamics matching the interaction graph. Finally, the presence of negative circuits in interaction graphs has been proven necessary for sustained oscillations in the dynamics (Remy, Ruet and Thieffry, 2008; Richard, 2010).

To produce more precise analyses of BRNs dynamics, it is required to take into account the boolean functions specified together with the interaction graph (as in (Bernot, Comet and Khalis, 2008), for instance). The majority of current techniques use standard model-checking methods (Richard, Comet and Bernot, 2006) based on the (explicit or symbolic) exploration of the state space of the model. Such methods suffer from the state space explosion, and are intractable on large regulatory networks. We propose here a novel and original method relying on the Process Hitting framework to address this scalability issue.

The Process Hitting (Paulevé, Magnin and Roux, 2011) 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 sort 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 (Paulevé et al., 2011). Being a particular

restriction of Communicating Finite-State Machines (Brand and Zafiropulo, 1983), the Process Hitting can be applied to less specific dynamical complex systems.

The static analysis by abstract interpretation (Cousot and Cousot, 1977) aims at providing an efficient analysis of a program without executing it. This is achieved by constructing one or several sound abstractions of the semantics of the program; these abstractions are then interpreted to decide the validity of a given property, resulting in over- or under-approximations of the validity of the property in the concrete program. Hereafter, we refer to this validity as the concretizability of the abstract property.

In this paper, we present a novel static analysis by abstract interpretation of Process Hittings. We address the decision of a successive reachability of processes. Our approach is based on two complementary abstractions of a succession of actions within a Process Hitting. Several refinement operators upon these two abstractions are then defined. These refinements detail an abstraction, with the aim of simplifying the concretizability decision. By using the abstraction refinement operators, over- and under-approximations of the process reachability decision are developed. Their implementations rely on the analysis of an abstract structure, that can be represented as a graph. We show that this abstract structure has always a reasonable size (i.e., polynomial in the total number of processes); and, while its computation can be exponential in the number of processes within a single sort, the approximations are always linear or polynomial in its size.

The scalability of our approach is illustrated by its application to the decision of reachability of gene expression levels within a BRN of 94 components. Our methods responds very fast to the decision, while a well established symbolic model checking technique (SDD, (Hamez, Thierry-Mieg and Kordon, 2009)) regularly fails because of the state space explosion.

A preliminary version of these results have been presented in (Paulevé, Magnin and Roux, 2010). This paper extends them in several ways: the general framework for presenting the analyses has been deeply reworked and unified; the conclusiveness of the overall method has been largely improved; the BRN application case has been extended to 94 components, instead of 40.

This paper is structured as follows. The Process Hitting framework is defined in Sect. 2. Sect. 3 presents complementary abstractions of scenarios (i.e., sequences of actions) and defines the abstraction refinements operators. These refinements operators are then applied to the over- and under-approximation of process reachability in Sect. 4; their implementation and complexity is also detailed. Sect. 5 briefly presents the encoding of BRNs dynamics into Process Hittings and applies the above methods to a large BRN relating 94 components. Finally, Sect. 6 summarises and discusses the contributions of this paper.

Notations: Given a countable *set* $S = \{e_1, \ldots, e_n\}$, $|S| = n$; $\wp(S)$ is the power set. Given a finite *sequence* of elements $A = e_1 :: \ldots :: e_n$, $|A| = n$ is the length of the sequence; $\mathbb{I}^A = \{1, \ldots, |A|\}$ is the set of A indexes; $A_i = e_i, \forall i \in \mathbb{I}^A$; ε is the empty sequence; $A_{i..j}$ is the subsequence $A_i, \ldots, A_j$; $A_{i..j} = \varepsilon$ if $j < i$. $\mathrm{lfp}\{x_0\}\ (x \mapsto x')$ is the least fix point of the function $f(x) = x'$ initially applied to x_0.

2. The Process Hitting Framework

This section presents the *Process Hitting* framework (Paulevé et al., 2011) on which the methods presented in this paper rely.

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 $a_i \rightarrow b_j \not\uparrow b_k$ 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 $a = b \Rightarrow a_i = b_j$. An action $h = a_i \rightarrow b_j \not\uparrow b_k$ 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 $\text{hitter}(h), \text{target}(h), \text{bounce}(h)$, respectively.

Definition 1 (Process Hitting). A *Process Hitting* 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, 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, $a \neq b \Rightarrow a_i \neq b_j \; \forall (a_i, b_j) \in L_a \times L_b$,
— $\mathcal{H} = \{a_i \rightarrow b_j \not\uparrow b_k, \cdots \mid (a, b) \in \Sigma^2,\ (a_i, b_j, b_k) \in L_a \times L_b \times L_b, b_j \neq b_k, a = b \Rightarrow a_i = b_j\}$, is the finite set of actions.

Proc refers to the set of all processes ($\mathbf{Proc} = \{a_i \mid a \in \Sigma \wedge a_i \in L_a\}$).

The sort of a process a_i is referred to as $\Sigma(a_i) = a$ and the set of sorts present in an action $h \in \mathcal{H}$ as $\Sigma(h) = \{\Sigma(\text{hitter}(h)), \Sigma(\text{target}(h))\}$. 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. We define the following notations: if $a_i \in L_a$, $a_i \in s \Leftrightarrow s[a] = a_i$; and if $ps \in \wp(Procs)$, $ps \subseteq s \Leftrightarrow \forall a_i \in ps, a_i \in s$.

An action $h = a_i \rightarrow b_j \not\uparrow b_k \in \mathcal{H}$ is *playable* in $s \in L$ if and only if $s[a] = a_i$ and $s[b] = b_j$. In such a case, $(s \cdot h)$ stands for the state resulting from the play of the action h in s, that is $(s \cdot h)[b] = b_k$ and $\forall c \in \Sigma, c \neq b, (s \cdot h)[c] = s[c]$. For the sake of clarity, $((s \cdot h) \cdot h')$, $h' \in \mathcal{H}$ is abbreviated as $(s \cdot h \cdot h')$.

If A is a sequence of actions, the set of sorts present in A is given by $\Sigma(A) = \bigcup_{n \in \mathbb{I}^A} \Sigma(A_n)$. The first (resp. last) process of sort a appearing in the sequence is referred to as $\text{fst}_a(A)$ (resp. $\text{last}_a(A)$):

$$\text{fst}_a(A) = \begin{cases} \varnothing & \text{if } a \notin \Sigma(A), \\ \text{hitter}(A_m) & \text{if } m = \min\{n \in \mathbb{I}^A \mid a \in \Sigma(A_n)\} \wedge \Sigma(\text{hitter}(A_m)) = a, \\ \text{target}(A_m) & \text{else if } \Sigma(\text{target}(A_m)) = a \ ; \end{cases} \quad (1)$$

$$\text{last}_a(A) = \begin{cases} \varnothing & \text{if } a \notin \Sigma(A), \\ \text{bounce}(A_m) & \text{if } m = \max\{n \in \mathbb{I}^A \mid a \in \Sigma(A_n)\} \\ & \qquad \wedge \Sigma(\text{bounce}(A_m)) = a, \\ \text{hitter}(A_m) & \text{else if } \Sigma(\text{hitter}(A_m)) = a \ . \end{cases} \tag{2}$$

Amongst sequences of actions, the particular sequences composed only of successively playable actions form *scenarios* (Def. 2). 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 $n \in \mathbb{I}^\delta, n < |\delta|$, δ_{n+1} is playable in the state $(s \cdot \delta_1 \cdot \ldots \cdot \delta_n)$; or equivalently δ is playable in s if and only if $\text{support}(\delta) \subseteq s$ (Eq. (3)). The state resulting from the sequential play of the scenario in s is denoted by $s \cdot \delta$. One can easily show that $\forall a_i \in \text{end}(\delta), (s \cdot \delta)[a] = a_i$ and $\forall b \in \Sigma \setminus \Sigma(\delta), (s \cdot \delta)[b] = s[b]$; where $\text{end}(\delta)$ is defined in Eq. (4).

Definition 2 (Scenario (Sce)). Given a Process Hitting $(\Sigma, L, \mathcal{H})$, a *scenario* δ is a sequence of actions in $\mathcal{H}$ such that for all $n \in \mathbb{I}^\delta$, $a_i = \text{hitter}(\delta_n)$ (resp. $\text{target}(\delta_n)$) $\Rightarrow \text{last}_a(\delta_{1..n-1}) \in \{\varnothing, a_i\}$. The set of all scenarios is denoted by **Sce**. $\text{support}(\delta)$ and $\text{end}(\delta)$ give the first and last processes of each sort, respectively:

$$\text{support}(\delta) = \{p \in \mathbf{Proc} \mid \Sigma(p) \in \Sigma(\delta) \wedge p = \text{fst}_{\Sigma(p)}(\delta)\} \ , \tag{3}$$

$$\text{end}(\delta) = \{p \in \mathbf{Proc} \mid \Sigma(p) \in \Sigma(\delta) \wedge p = \text{last}_{\Sigma(p)}(\delta)\} \ . \tag{4}$$

Fig. 1 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 $\mathcal{H} = \{a_0 \to c_0 \Rsh c_1, a_1 \to b_1 \Rsh b_0, c_1 \to b_0 \Rsh b_1, b_1 \to a_0 \Rsh a_1, b_0 \to d_0 \Rsh d_1, b_1 \to d_1 \Rsh d_2, d_1 \to b_0 \Rsh b_2, c_1 \to d_1 \Rsh d_0, b_2 \to d_0 \Rsh d_2\}$. Playing the action $b_1 \to a_0 \Rsh a_1$ in the state $\langle a_0, b_1, c_0, d_0 \rangle$ results in the state $\langle a_1, b_1, c_0, d_0 \rangle$. $\delta = a_0 \to c_0 \Rsh c_1 :: b_1 \to a_0 \Rsh a_1 :: a_1 \to b_1 \Rsh b_0 :: b_0 \to d_0 \Rsh d_1 :: d_1 \to b_0 \Rsh b_2$ is a scenario playable in the state $s = \langle a_0, b_1, c_0, d_0 \rangle$, and $s \cdot \delta = \langle a_1, b_2, c_1, d_1 \rangle$.

Remark 1. The Process Hitting framework can be considered as a class of Communicating Finite-State Machines (Brand and Zafiropulo, 1983) where at most two machines (sorts) share a synchronization label (action) and one and only one machine changes its state (process) at each synchronization (action play).

3. Abstract Interpretation of Scenarios

After having introduced preliminary definitions (Subsect. 3.1), this section establishes two orthogonal abstractions of scenarios: by *objective sequences* (Subsect. 3.2) and by *bounce sequences* (Subsect. 3.3). The former describes a succession of process changes per sort (called objectives), while the latter details the actions actually played to resolve these objectives. While objective sequences can be seen as a sparse representation of a scenario, bounce sequences emphases the necessary actions required to resolve an objective.

Upon these two complementary abstractions, several objective sequence refinements operators are derived in Subsect. 3.4. The aim of these refinements is at providing more

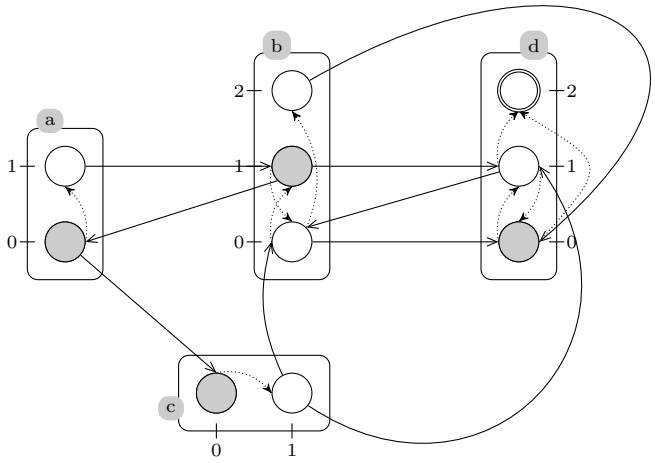


Figure 1. A Process Hitting example. Sorts are represented by labeled boxes, and processes by circles (ticks are the identifiers of the processes within the sort, for instance, a_0 is the process ticked 0 in the box a). An action (for instance $a_0 \rightarrow c_0 \uparrow c_1$) is represented by a pair of directed arcs, having the hit part (a_0 to c_0) in plain line and the bounce part (c_0 to c_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 grayed processes: $\langle a_0, b_1, c_0, d_0 \rangle$.

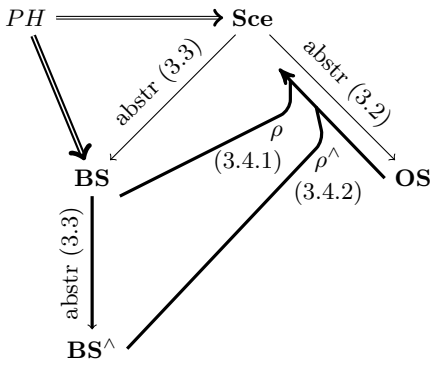


Figure 2. Derivation relations between a Process Hitting (PH), scenarios (**Sce**), objective sequences (**OS**, Subsect. 3.2), bounce sequences (**BS** and **BS**$^\wedge$, Subsect. 3.3) and refinement operators (e.g., ρ and $\rho^\wedge$, Subsect. 3.4). The thicked relations are used in Sect. 4 to decide the concretizability of an objective sequence.

precise abstractions on which the concretizability may be easier to decide. Fig. 2 summarizes the possible derivations between a Process Hitting and the different representations of scenarios.

3.1. *Preliminaries*

This subsection introduces the notion of *objective* and *context* used by the developed abstractions.

The reachability of the process a_j from a process a_i is called an objective and is denoted by $a_i \Rsh^* a_j$ (Def. 3).

Definition 3 (Objective (Obj)). The reachability of process a_j from a_i is called an *objective*, noted $a_i \Rsh^* a_j$. The set of objectives is denoted by $\mathbf{Obj} = \{a_i \Rsh^* a_j \mid a \in \Sigma \wedge (a_i, a_j) \in L_a^2\}$. Given an objective $P \in \mathbf{Obj}$, where $P = a_i \Rsh^* a_j$, $\Sigma(P) = a$, $\text{target}(P) = a_i$, $\text{bounce}(P) = a_j$. An objective P is *trivial* if $\text{target}(P) = \text{bounce}(P)$.

We extend the notion of state by the notion of context (Def. 4). A context references the set of processes per sort that can serve as initial state.

Definition 4 (Context ς (Ctx)). A *context* ς associates with each sort in Σ a non-empty subset of its processes: $\forall a \in \Sigma, \varsigma[a] \subseteq L_a \wedge \varsigma[a] \neq \emptyset$. **Ctx** refers to the set of contexts.

Given a context ς, we note $a_i \in \varsigma \Leftrightarrow a_i \in \varsigma[a]$; $ps \in \wp(\mathbf{Proc}), ps \subseteq \varsigma \Leftrightarrow \forall a_i \in ps, a_i \in \varsigma$. The override of a context ς by a set of processes ps is noted $\varsigma \Cap ps$ (Def. 5). For instance, $\langle a_1, a_2, b_1, c_1\rangle \Cap \{a_3, b_2, b_3\} = \langle a_3, b_2, b_3, c_1\rangle$.

Definition 5 ($\Cap : \mathbf{Ctx} \times \wp(\mathbf{Proc}) \mapsto \mathbf{Ctx}$). Given a context $\varsigma \in \mathbf{Ctx}$ and $ps \in \wp(\mathbf{Proc})$, the override of ς by ps is noted $\varsigma \Cap ps$ and is defined as

$$\forall a \in \Sigma, (\varsigma \Cap ps)[a] = \begin{cases} \{p \in ps \mid \Sigma(p) = a\} & \text{if } \exists p \in ps, \Sigma(p) = a, \\ \varsigma[a] & \text{otherwise.} \end{cases}$$

A scenario $\delta \in \mathbf{Sce}$ is playable in the context ς if and only if $\text{support}(\delta) \subseteq \varsigma$. The play of δ in ς is denoted by $\varsigma \cdot \delta$ where $\varsigma \cdot \delta = \varsigma \Cap \text{end}(\delta)$.

3.2. *Abstraction of Scenarios into Objective Sequences*

During the execution of a scenario, processes of different sorts bounce one after the other, following the play of the actions. An abstraction of such an execution is a succession of objectives: the process a_j is reached (after a certain number of actions) from a_i, then the process b_j is reached from b_i, etc. This forms an objective sequence (Def. 6). The append of an objective to an objective sequence is specified in Def. 7.

Definition 6 (Objective Sequence (OS)). An *Objective Sequence* is a sequence $\omega = P_1 :: \ldots :: P_{|\omega|}$, where $\forall n \in \mathbb{I}^\omega, \omega_n \in \mathbf{Obj}$ and $a_i = \text{target}(\omega_n) \Rightarrow \text{last}_a(\omega_{1..n-1}) \in \{\varnothing, a_i\}$. The set of objective sequences is referred to as **OS**. The definitions of last_a (Eq. (2)), fst_a (Eq. (1)), support (Eq. (3)) and end (Eq. (4)) are straightforwardly derived to objective sequences by omitting the hitter case.

Definition 7 ($\oplus : \mathbf{OS} \times \mathbf{Obj} \mapsto \mathbf{OS}$)**.** The join between an objective sequence $\omega \in \mathbf{OS}$ and an objective $a_i \Rsh^* a_j \in \mathbf{Obj}$ is defined as:

$$\omega \oplus a_i \Rsh^* a_j = \begin{cases} \omega :: a_i \Rsh^* a_j & \text{if } a \notin \Sigma(\omega), \\ \omega :: \text{last}_a(\omega) \Rsh^* a_j & \text{otherwise.} \end{cases}$$

A scenario can be abstracted by several objective sequences, describing process changes more or less sparsely. For instance, the scenario $a_0 \to \mathbf{c}_0 \Rsh \mathbf{c}_1 :: b_1 \to \mathbf{a}_0 \Rsh \mathbf{a}_1 :: a_1 \to \mathbf{b}_1 \Rsh \mathbf{b}_0 :: b_0 \to \mathbf{d}_0 \Rsh \mathbf{d}_1 :: d_1 \to \mathbf{b}_0 \Rsh \mathbf{b}_2$ can be abstracted to $c_0 \Rsh^* c_1 :: \mathbf{a}_0 \Rsh^* \mathbf{a}_1 :: \mathbf{b}_1 \Rsh^* b_0 :: d_0 \Rsh^* d_1 :: b_0 \Rsh^* \mathbf{b}_2$; or, more sparsely, to $a_0 \Rsh^* a_1 :: \mathbf{b}_1 \Rsh^* \mathbf{b}_2$; or to $b_1 \Rsh^* b_2$; etc. (bolded processes are the ones kept in the succeeding abstraction).

The set of scenarios concretizing an objective sequence ω in a context ς is given by $\gamma_\varsigma(\omega)$ (Def. 8). We also define the concretization of a set of objective sequences as the union of their concretizations (Def. 9).

Definition 8 ($\gamma_\varsigma : \mathbf{OS} \mapsto \wp(\mathbf{Sce})$)**.** Given $\omega \in \mathbf{OS}$, $\gamma_\varsigma(\omega)$ is the set of scenarios concretizing ω in the context ς:

$$\begin{aligned} \gamma_\varsigma(\omega) = \{\delta \in \mathbf{Sce} \mid & (\omega^\Delta = \varepsilon \wedge \delta = \varepsilon) \vee \omega^\Delta \neq \varepsilon \wedge \text{support}(\delta) \subseteq \varsigma \\ & \wedge \exists \phi : \mathbb{I}^\omega \mapsto \mathbb{I}^\delta, (\forall n, m \in \mathbb{I}^\omega, n < m \Leftrightarrow \phi(n) \leq \phi(m)) \\ & \wedge \forall n \in \mathbb{I}^\omega, \text{bounce}(\omega_n) \in \varsigma \cdot \delta_{1..\phi(n)}\} \ , \end{aligned}$$

where ω^Δ refers to the objective sequence ω where trivial objectives have been removed.

Definition 9 ($\gamma_\varsigma : \wp(\mathbf{OS}) \mapsto \wp(\mathbf{Sce})$)**.** $\gamma_\varsigma(\Omega) = \{\delta \in \gamma_\varsigma(\omega) \mid \omega \in \Omega\}$.

It is worth noticing that the concretization of an objective sequence ω does not depend on its support $\text{support}(\omega)$ as it is imposed by the context. In that way, we use $\star \Rsh^* a_i$ to denote an objective where the target can be any process of sort a present in the context:

$$\gamma_\varsigma(\star \Rsh^* a_i) = \gamma_\varsigma(a_j \Rsh^* a_i), \forall a_j \in \varsigma[a] \ . \tag{5}$$

We finally refer to α_ς as the reverse operation of concretization of a set of scenarios (Def. 10). This straightforwardly provides the Galois connection in Property 1.

Definition 10 ($\alpha_\varsigma : \wp(\mathbf{Sce}) \mapsto \wp(\mathbf{OS})$)**.**

$$\alpha_\varsigma(\Delta) = \{\omega \in \mathbf{OS} \mid \exists \delta \in \Delta, \delta \in \gamma_\varsigma(\omega)\} \ .$$

Property 1. $(\wp(\mathbf{Sce}), \subseteq) \underset{\alpha_\varsigma}{\overset{\gamma_\varsigma}{\leftrightarrows}} (\wp(\mathbf{OS}), \subseteq)$ is a Galois connection.

Proof. $\forall \Delta \in \wp(\mathbf{Sce}), \Omega \in \wp(\mathbf{OS}), \alpha_\varsigma(\Delta) \subseteq \Omega \Leftrightarrow \Delta \subseteq \gamma_\varsigma(\Omega)$. □

3.3. *Abstraction of Scenarios into Bounce Sequences*

Bounce sequences result from a local reasoning on a single sort a. Bouncing from a_i to a_j (i.e., resolving the objective $a_i \Rsh^* a_j$) may require the play of several actions on processes of sort a, forming a *bounce sequence* (Def. 11). Remark that bounce sequences

are generally not scenarios: e.g., $b_i \rightarrow a_i \upharpoonright a_j :: b_j \rightarrow a_j \upharpoonright a_k$ is a bounce sequence but not a scenario if $b_i \neq b_j$.

Definition 11 (Bounce Sequence (BS)). A *bounce sequence* ζ is a sequence of actions such that $\forall n \in \mathbb{I}^{\zeta}, n < |\zeta|, \text{bounce}(\zeta_n) = \text{target}(\zeta_{n+1})$. **BS** denotes the set of bounce sequences. We refer to the set of bounce sequences *resolving* the objective P as $\mathbf{BS}(P)$:

$$\mathbf{BS}(a_i \upharpoonright^* a_j) = \{\zeta \in \mathbf{BS} \mid \text{target}(\zeta_1) = a_i \wedge \text{bounce}(\zeta_{|\zeta|}) = a_j\} \ .$$

Obviously, $\mathbf{BS}(a_i \upharpoonright^* a_i) = \{\varepsilon\}$; and $\mathbf{BS}(a_i \upharpoonright^* a_j) = \emptyset$ if there is no possibility to reach a_j from a_i.

In a bounce sequence ζ, target and bounces of all actions share the same sort $\Sigma(\zeta)$. In the scope of this paper, we do not consider bounce sequences that 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.

The full set of bounce sequences can be computed directly from the set of actions $\mathcal{H}$ of the Process Hitting without any enumeration of scenarios. Given an objective $a_i \upharpoonright^* a_j$, the computation of bounce sequences $\mathbf{BS}(a_i \upharpoonright^* a_j)$ (Def. 11) works by a depth-first research between actions on the sort a to form a bounce sequence without cycle. Such a computation is exponential in the number of actions on the sort a, and is then efficient when this number is small in front of the total number of actions.

We also consider a sparse representation of a bounce sequence ζ resolving an objective P by considering the set of hitters of its actions that have a different sort than that of P. We denote by $\mathbf{BS}^{\wedge}(P)$ the set of such abstracted bounce sequences (Def. 12).

Definition 12 ($\mathbf{BS}^{\wedge} : \mathbf{Obj} \mapsto \wp(\mathbf{Proc})$).

$$\mathbf{BS}^{\wedge}(P) = \{\zeta^{\wedge} \mid \zeta \in \mathbf{BS}(P), \nexists \zeta' \in \mathbf{BS}(P), \zeta'^{\wedge} \subsetneq \zeta^{\wedge}\} \ ,$$

where $\zeta^{\wedge} = \{\text{hitter}(\zeta_n) \mid n \in \mathbb{I}^{\zeta} \wedge \Sigma(\text{hitter}(\zeta_n)) \neq \Sigma(P)\}$.

It is worth noticing that $\mathbf{BS}^{\wedge}(P)$ can be computed directly from the Process Hitting in the same manner as $\mathbf{BS}(P)$, but yet more efficiently since only minimal sets of hitters are kept, pruning redundant explorations.

The relations of abstractions and concretizations between scenarios and (abstracted) bounce sequences can be derived easily, and are not detailed here.

Looking at the Process Hitting example in Fig. 1, $\zeta = a_1 \rightarrow b_1 \upharpoonright b_0 :: d_1 \rightarrow b_0 \upharpoonright b_2$ is the only bounce sequence resolving the objective $b_1 \upharpoonright^* b_2$ (i.e. $\mathbf{BS}(b_1 \upharpoonright^* b_2) = \{\zeta\}$, and $\mathbf{BS}^{\wedge}(b_1 \upharpoonright^* b_2) = \{\{a_1, d_1\}\}$).

3.4. *Objective Sequence Refinements*

Before introducing objective sequence refinements operators, we define the relation $\preccurlyeq_{\mathbf{OS}}$ between two objective sequences (Def. 13). Basically, if $\omega \preccurlyeq_{\mathbf{OS}} \omega'$, we say ω is a more precise abstraction than ω', hence $\gamma_{\varsigma}(\omega) \subseteq \gamma_{\varsigma}(\omega')$ (Property 2). In such a setting, an objective sequence joined to an objective is a more precise abstraction than the objective alone (Property 3).

Definition 13 ($\preccurlyeq_{\mathbf{OS}} \subset \mathbf{OS} \times \mathbf{OS}$). $\omega \preccurlyeq_{\mathbf{OS}} \omega'$ if and only if the following properties are satisfied:

— $|\omega| \geq |\omega'|$;
— there exists a mapping $\phi : \mathbb{I}^{\omega'} \mapsto \mathbb{I}^{\omega}$ such that $\forall n \in \mathbb{I}^{\omega'}, \text{bounce}(\omega'_n) = \text{bounce}(\omega_{\phi(n)})$ and $\forall n, m \in \mathbb{I}^{\omega'}, n < m \Leftrightarrow \phi(n) < \phi(m)$.

Property 2. $\omega \preccurlyeq_{\mathbf{OS}} \omega' \Longrightarrow \gamma_\varsigma(\omega) \subseteq \gamma_\varsigma(\omega')$.

Property 3. Given $\omega \in \mathbf{OS}$ and $P \in \mathbf{Obj}$, $\omega \oplus P \preccurlyeq_{\mathbf{OS}} P$.

Finally, given an objective P, $\mathbf{BS}_\varsigma(P)$ (Def. 14) and $\mathbf{BS}^\wedge_\varsigma(P)$ (Def. 15) generalise $\mathbf{BS}(P)$ and $\mathbf{BS}^\wedge(P)$ to the scope of the context ς, respectively.

Definition 14 ($\mathbf{BS}_\varsigma : \mathbf{Obj} \mapsto \wp(\mathbf{BS})$).

$$\mathbf{BS}_\varsigma(\star \Rsh^* a_j) = \bigcup_{a_i \in \varsigma[a]} \mathbf{BS}(a_i \Rsh^* a_j) \ .$$

Definition 15 ($\mathbf{BS}^\wedge_\varsigma : \mathbf{Obj} \mapsto \wp(\wp(\mathbf{Proc}))$).

$$\mathbf{BS}^\wedge_\varsigma(\star \Rsh^* a_j) = \bigcup_{a_i \in \varsigma[a]} \mathbf{BS}^\wedge(a_i \Rsh^* a_j) \ .$$

3.4.1. *Objective refinement by* $\mathbf{BS}$ *(ρ).* We build the function β such that, given an objective sequence P, a bounce sequence $\zeta \in \mathbf{BS}(P)$ is abstracted by $\beta(\zeta)$ to the objective sequence describing the successive reachability of its hitters (Def. 16). From $\mathbf{BS}_\varsigma$ definition, if a scenario concretizes P in context ς, it necessarily concretizes one bounce sequence $\zeta \in \mathbf{BS}_\varsigma(P)$, thus $\beta(\zeta)$. The refinement operator $\rho(P, \mathbf{BS}_\varsigma(P))$ extends P to the set of objective sequences where P is prefixed by each $\beta(\zeta)$, $\zeta \in \mathbf{BS}_\varsigma(P)$ (Def. 17). Finally, Lemma 1 states the correctness of this refinement, ensuring the preservation of the concretization set.

Definition 16 ($\beta : \mathbf{BS} \mapsto \wp(\mathbf{OS})$).

$$\beta(\zeta) = \{\omega \in \mathbf{OS} \mid |\omega| = |\zeta| \wedge \forall n \in \mathbb{I}^\zeta, \text{bounce}(\omega_n) = \text{hitter}(\zeta_n)\} \ .$$

Definition 17 ($\rho : \mathbf{Obj} \times \wp(\mathbf{BS}) \mapsto \wp(\mathbf{OS})$).

$$\rho(P, zs) = \{\omega \oplus P \mid \omega \in \beta(\zeta), \zeta \in zs\} \ .$$

Lemma 1. $\gamma_\varsigma(P) = \gamma_\varsigma(\rho(P, \mathbf{BS}_\varsigma(P)))$.

Proof. ($\supseteq$) $\forall \omega \in \rho(P, \mathbf{BS}_\varsigma(P)), \omega \preccurlyeq_{\mathbf{OS}} P$, therefore $\gamma_\varsigma(P) \supseteq \gamma_\varsigma(\rho(P, \mathbf{BS}_\varsigma(P)))$;
($\subseteq$) By definition of $\mathbf{BS}_\varsigma(P)$, $\forall \delta \in \gamma_\varsigma(P), \exists \omega \in \rho(P, \mathbf{BS}_\varsigma(P)), \delta \in \gamma_\varsigma(\omega)$, thus $\gamma_\varsigma(P) \subseteq \gamma_\varsigma(\rho(P, \mathbf{BS}_\varsigma(P)))$. □

3.4.2. *Objective refinement by* $\mathbf{BS}^\wedge$ *($\rho^\wedge$).* The refinement of an objective P by $\mathbf{BS}^\wedge$ is done in a similar way. A set of hitters $ps \in \mathbf{BS}^\wedge(P)$ is abstracted by $\beta^\wedge(ps)$ into the set of objective sequences describing any ordering of reach of these hitters. The relation between objective sequences in $\beta(\zeta)$ and in $\beta^\wedge(ps)$ is emphased in Property 4. The refinement $\rho^\wedge(P, \mathbf{BS}^\wedge_\varsigma(P))$ is presented in Def. 19 and the preservation of concretizations is stated in Lemma 2.

Definition 18 ($\beta^\wedge : \wp(\mathbf{Proc}) \mapsto \wp(\mathbf{OS})$)**.**

$$\beta^\wedge(ps) = \{\omega \in \mathbf{OS} \mid |\omega| = |ps| \wedge \forall p \in ps, \exists n \in \mathbb{I}^\omega, \text{bounce}(\omega_n) = p\} \ .$$

Property 4. $\forall \zeta \in \mathbf{BS}(P), \forall \omega \in \beta(\zeta), \exists \omega' \in \beta^\wedge(\zeta^\wedge), \omega \preccurlyeq_{\mathbf{OS}} \omega'$.

Definition 19 ($\rho^\wedge : \mathbf{Obj} \times \wp(\wp(\mathbf{Proc})) \mapsto \wp(\mathbf{OS})$)**.**

$$\rho^\wedge(P, pss) = \{\omega \oplus P \mid \omega \in \beta^\wedge(ps), ps \in pss\} \ .$$

Lemma 2. $\gamma_\varsigma(P) = \gamma_\varsigma(\rho^\wedge(P, \mathbf{BS}^\wedge_\varsigma(P))) \ .$

Proof. ($\supseteq$) $\forall \omega \in \rho^\wedge(P, \mathbf{BS}^\wedge_\varsigma(P)), \omega \preccurlyeq_{\mathbf{OS}} P$; ($\subseteq$) by Property 4, $\forall \omega \in \rho(P, \mathbf{BS}_\varsigma(P))$, $\exists \omega' \in \rho^\wedge(P, \mathbf{BS}^\wedge_\varsigma(P)), \omega \preccurlyeq_{\mathbf{OS}} \omega'$, thus $\gamma_\varsigma(\rho(P, \mathbf{BS}_\varsigma(P)) \subseteq \gamma_\varsigma(\rho^\wedge(P, \mathbf{BS}^\wedge_\varsigma(P))$. □

3.4.3. *Objective sequence refinements ($\widetilde{\rho}$).* Finally, to generalize the refinements defined on objective to objective sequence, we exhibit an objective sequence refinement operator that uses any of the above refinements. We took the operator ρ as an example and define the refinement $\widetilde{\rho}(\omega, \mathbf{BS})$ (Def. 21). Basically, this refinement chooses any objective ω_n of the objective sequence, refines it using ρ, and returns all the interleaving of the obtained refined sequences with the objective sequence $\omega_{1..n-1}$ (Def. 20); the concretization set is then preserved (Lemma 3).

Definition 20 (interleave : $\mathbf{OS} \times \mathbf{OS} \mapsto \wp(\mathbf{OS})$)**.**

$$\begin{aligned}\text{interleave}(\omega^1, \omega^2) = \{\omega \in \mathbf{OS} \mid |\omega| = |\omega^1| + |\omega^2| \wedge \exists \phi^1 : \mathbb{I}^{\omega^1} \mapsto \mathbb{I}^\omega, \phi^2 : \mathbb{I}^{\omega^2} \mapsto \mathbb{I}^\omega, \\ (\forall n, m \in \mathbb{I}^{\omega^1}, n < m \Leftrightarrow \phi^1(n) < \phi^1(m)) \\ \wedge (\forall n, m \in \mathbb{I}^{\omega^2}, n < m \Leftrightarrow \phi^2(n) < \phi^2(m)) \\ \wedge (\nexists n^1 \in \mathbb{I}^{\omega^1}, n^2 \in \mathbb{I}^{\omega^2}, \phi^1(n^1) = \phi^2(n^2))\} \ .\end{aligned}$$

Definition 21 ($\widetilde{\rho} : \mathbf{OS} \times \wp(\mathbf{BS}) \mapsto \wp(\mathbf{OS})$)**.**

$$\begin{aligned}\widetilde{\rho}(\omega, \mathbf{BS}) = \{\varpi \oplus \omega_{n..|\omega|} \mid n \in \mathbb{I}^\omega, \omega' \oplus \omega_n \in \rho(\omega_n, \mathbf{BS}_\varsigma(\omega_n)), \\ \varpi \in \text{interleave}(\omega_{1..n-1}, \omega')\} \ .\end{aligned}$$

Lemma 3. $\gamma_\varsigma(\omega) = \gamma_\varsigma(\widetilde{\rho}(\omega, \mathbf{BS}))$.

Proof. ($\supseteq$) $\forall \omega' \in \widetilde{\rho}(\omega, \mathbf{BS}), \omega' \preccurlyeq_{\mathbf{OS}} \omega$;
($\subseteq$) by Lemma 1 and Def. 20, $\delta \in \gamma_\varsigma(\omega) \Rightarrow \exists \omega' \in \widetilde{\rho}(\omega, \mathbf{BS}), \delta \in \gamma_\varsigma(\omega')$. □

4. Over- and Under-approximations of Process Reachability

We define the *Process Reachability* problem as deciding if a given objective sequence $\omega \in \mathbf{OS}$ is concretizable for a given Process Hitting in a context ς; i.e. if the set $\gamma_\varsigma(\omega)$ is not empty. The process reachability problem can also be formulated in a subclass of the *CTL* (Clarke and Emerson, 1981), restricted to the following form:

$$\Phi ::= a_i \mid a_i \wedge \varphi \qquad \varphi ::= \texttt{EF}\ \Phi \ ,$$

where $a_i \in \mathbf{Proc}$ is true if the current state contains the process a_i and $\mathtt{EF}$ stands for the usual *exists finally* predicate. Given an objective sequence ω, one can encode it into CTL using the following recursive definition of $[|\cdot|]$:

$$[|a_j \Rsh^* a_i :: \varepsilon|] = \mathtt{EF}\ a_i \qquad [|a_j \Rsh^* a_i :: \omega|] = \mathtt{EF}\ (a_i \wedge [|\omega|]) \text{ if } \omega \neq \varepsilon \ .$$

Based upon the refinements operators defined in the previous section, we establish several necessary or sufficient conditions for the concretizability of an objective sequence in a given context ς. These objective sequences can be either given by a user (to check some temporal properties), or extracted from **BS** (with β, Def. 16) to refine this set of bounce sequences. Indeed, if a bounce sequence is not concretizable in ς, it can be ignored in all analyses in the scope of this context.

These approximations aim at being very fast to compute, overcoming the state space explosion problem inherent to such a kind of dynamics analysis. While inconclusive in some cases, the application section (Sect. 5) shows the very good suitability of our analyses to biological regulatory networks dynamics with a very promising scalability.

The remainder of this section is structured as follows. Subsect. 4.1 presents a first over-approximation based on an un-ordered analysis of objectives required to concretize the given objective sequence; Subsect. 4.2 refines this former approximation by exploiting the sequentiality of objectives to concretize; then Subsect. 4.3 uses order constraints between process occurrences to complete these over-approximations. Finally, Subsect. 4.4 sets up an under-approximation of the process reachability decision.

4.1. *Un-ordered Over-approximation*

Given a context ς and an objective sequence ω, we obtain that ω is concretizable only if each objective $\omega_n, n \in \mathbb{I}^\omega$, is independently concretizable in the same context (Proposition 1). In this way, one can approximate the concretizability of a bounce sequence by recursively applying Proposition 1 to extract objectives from the given objective sequence, and use the refinement operator $\rho^\wedge$ to extend the objective into several objective sequences.

Proposition 1. $\gamma_\varsigma(\omega) \neq \emptyset \Longrightarrow \forall n \in \mathbb{I}^\omega, \gamma_\varsigma(\omega_n) \neq \emptyset$.

Proof. By Def. 13 and Property 2, $\omega \preccurlyeq_{\mathbf{OS}} \omega_i$. □

Given an objective P, $\mathrm{minCont}_\varsigma(P)$ (Def. 22) is the set of re-targeted objectives Q, with $\mathrm{target}(P) \neq \mathrm{target}(Q)$ and $\mathrm{bounce}(P) = \mathrm{bounce}(Q)$, which are always derived from a recursive application of $\rho^\wedge(P, \mathbf{BS}^\wedge(P))$ with Proposition 1 (Lemma 4). The recursive procedure to check necessary conditions for the concretizability of an objective is then summarised by Proposition 2. If P is concretizable, then there is an execution of this procedure without cycle, and all tested objectives have at least one solution (Theorem 1).

Definition 22 ($\text{minCont}_\varsigma : \mathbf{Obj} \mapsto \wp(\mathbf{Obj})$)**.**

$$\text{minCont}_\varsigma(\star \Rsh^* a_j) = \{a_k \Rsh^* a_j \mid a_k \neq a_j \wedge \forall a_i \in \varsigma[a], a_k \in \text{minCont}_\varsigma^{Obj}(a, a_i \Rsh^* a_j)\}$$
$$\text{minCont}_\varsigma^{Obj} : \Sigma \times \mathbf{Obj} \mapsto \wp(\mathbf{Proc})$$
$$\text{minCont}_\varsigma^{Obj}(a, P) = \emptyset \qquad \text{if } \mathbf{BS}^\wedge(P) = \emptyset, \text{ otherwise,}$$
$$\text{minCont}_\varsigma^{Obj}(a, P) = \{p \in \mathbf{Proc} \mid \forall ps \in \mathbf{BS}^\wedge(P), \exists q \in ps, p \in \text{minCont}_\varsigma^{Proc}(a, q)\}$$
$$\text{minCont}_\varsigma^{Proc} : \Sigma \times \mathbf{Proc} \mapsto \wp(\mathbf{Proc})$$
$$\text{minCont}_\varsigma^{Proc}(a, b_i) = \begin{cases} \{b_i\} & \text{if } a = b, \\ \{p \in \mathbf{Proc} \mid \forall b_j \in \varsigma[b], & \\ \quad p \in \text{minCont}_\varsigma^{Obj}(a, b_j \Rsh^* b_i)\} & \text{otherwise.} \end{cases}$$

Lemma 4. $a_k \Rsh^* a_j \in \text{minCont}_\varsigma(\star \Rsh^* a_j) \Longrightarrow \gamma_\varsigma(\star \Rsh^* a_j) = \gamma_\varsigma(\star \Rsh^* a_k :: a_k \Rsh^* a_j)$.

Proof. By induction on minCont_ς, a_k occurs in all recursive refinements of $\star \Rsh^* a_j$ by $\rho^\wedge$. □

Proposition 2. $\gamma_\varsigma(P) \neq \emptyset \Longrightarrow \exists ps \in \mathbf{BS}_\varsigma^\wedge(P), \forall p \in ps, \gamma_\varsigma(\star \Rsh^* p) \neq \emptyset$, and $\forall Q \in \text{minCont}_\varsigma(P), \gamma_\varsigma(Q) \neq \emptyset$, with $\mathbf{BS}_\varsigma^\wedge(\star \Rsh^* a_j) = \{ps \in \mathbf{BS}^\wedge(a_i \Rsh^* a_j) \mid a_i \in \varsigma[a]\}$.

Proof. By Lemma 2, Lemma 4 and Proposition 1. □

Theorem 1 (Un-ordered over-approximation). $\gamma_\varsigma(\omega) \neq \emptyset \Longrightarrow \forall n \in \mathbb{I}^\omega$, there exists a finite recursive application of Proposition 2 to $\gamma_\varsigma(\omega_n) \neq \emptyset$ such that for all tested objective P, $\mathbf{BS}_\varsigma^\wedge(P) \neq \emptyset$.

Proof. By induction, if $\gamma_\varsigma(P) \neq \emptyset$ requires $\gamma_\varsigma(P') \neq \emptyset, P' \neq P$, and if $\gamma_\varsigma(P') \neq \emptyset$ requires again $\gamma_\varsigma(P) \neq \emptyset$, then $\gamma_\varsigma(P) = \emptyset$; therefore, by Proposition 1, there exists a finite recursion of Proposition 2 application on $\gamma_\varsigma(\omega_n) \neq \emptyset, \forall n \in \mathbb{I}^\omega$. Finally, Proposition 2 implies the non-emptyness of $\mathbf{BS}_\varsigma^\wedge(P)$ for each tested objective P. □

Implementation. Given a context ς and an objective sequence ω, we define an abstract structure $\mathcal{A}_\varsigma^\omega$ (Def. 23) which mimics the relations between objectives during the execution of Proposition 2 (Lemma 5). $\mathcal{A}_\varsigma^\omega$ gathers three relations $(\text{Req}_\varsigma^\omega, \text{Sol}_\varsigma^\omega, \text{Cont}_\varsigma^\omega)$, respectively the requirements, solutions, and minimal continuity (or re-targeting).

Definition 23 ($\mathcal{A}_\varsigma^\omega$)**.** Given a context ς and an objective sequence ω, we define the abstract structure $\mathcal{A}_\varsigma^\omega = (\text{Req}_\varsigma^\omega, \text{Sol}_\varsigma^\omega, \text{Cont}_\varsigma^\omega)$, where $\text{Req}_\varsigma^\omega$, $\text{Sol}_\varsigma^\omega$ and $\text{Cont}_\varsigma^\omega$ are defined as follows:

$$\begin{aligned} \text{Req}_\varsigma^\omega = &\{(a_i, a_j \Rsh^* a_i) \in \mathbf{Proc} \times \mathbf{Obj} \mid a_j \in \varsigma[a] \wedge (\exists (P, ps) \in \text{Sol}_\varsigma^\omega, a_i \in ps \\ &\qquad \vee \exists n \in \mathbb{I}^\omega, \text{bounce}(\omega) = a_i)\} \\ \text{Sol}_\varsigma^\omega = &\{(P, ps) \in \mathbf{Obj} \times \wp(\mathbf{Proc}) \mid \exists (a_i, P) \in \text{Req}_\varsigma^\omega \wedge ps \in \mathbf{BS}^\wedge(P)\} \\ \text{Cont}_\varsigma^\omega = &\{(P, Q) \in \mathbf{Obj} \times \mathbf{Obj} \mid \exists (P, ps) \in \text{Sol}_\varsigma^\omega \wedge Q \in \text{minCont}_\varsigma(p)\} \ . \end{aligned}$$

Lemma 5. Given an objective P referenced in $\mathcal{A}_\varsigma^\omega$,

— $\omega' \oplus P \in \rho^{\wedge}(P, \mathbf{BS}^{\wedge}(P)) \iff (P, \{\text{bounce}(\omega'_n) \mid n \in \mathbb{I}^{\omega'}\}) \in \text{Sol}^{\omega}_{\varsigma} \wedge \forall n \in \mathbb{I}^{\omega'}, a_j = \text{bounce}(\omega'_n), \forall a_i \in \varsigma[a], (a_j, a_i \Rsh^* a_j) \in \text{Req}^{\omega}_{\varsigma}$;
— $Q \in \text{minCont}_{\varsigma}(P) \iff (P, Q) \in \text{Cont}^{\omega}_{\varsigma}$.

Proof. By construction of $\mathcal{A}^{\omega}_{\varsigma}$. □

$\mathcal{A}^{\omega}_{\varsigma}$ has a graph structure, with cycles, potentially. We remark that, as $|\mathbf{Obj}| = \sum_{a \in \Sigma} |L_a|^2$, the size of $\text{Req}^{\omega}_{\varsigma}$ and $\text{Cont}^{\omega}_{\varsigma}$ sets are polynomial in the number of processes in the Process Hitting. The size of $\text{Sol}^{\omega}_{\varsigma}$ also depends on the cardinality of $\mathbf{BS}^{\wedge}$ which follows the maximum number of combinations of $|L_a|$ different processes, far below an exponential growth.

Finally, Algorithm 1 details the computation of the Theorem 1 decision.

Algorithm 1 (Un-ordered over-approximation). Given a context ς, an objective sequence $\omega \in \mathbf{OS}$ and the abstract structure $\mathcal{A}^{\omega}_{\varsigma}$:

1 Initialise $\Theta = \{P \in \mathbf{Obj} \mid (P, \emptyset) \in \text{Sol}^{\omega}_{\varsigma}\}$.
2 Repeat until fix-point:
  (a) $\Upsilon = \{p \in \mathbf{Proc} \mid \exists P \in \Theta, (p, P) \in \text{Req}^{\omega}_{\varsigma}\}$;
  (b) $\Theta = \{P \in \mathbf{Obj} \mid \exists (P, ps) \in \text{Sol}^{\omega}_{\varsigma}, ps \subseteq \Upsilon \wedge \forall (P, Q) \in \text{Cont}^{\omega}_{\varsigma}, Q \in \Theta\}$.
3 $\gamma_{\varsigma}(\omega) \neq \emptyset \Longrightarrow \forall n \in \mathbb{I}^{\omega}, \exists P \in \Theta, \text{target}(P) \in \varsigma \wedge \text{bounce}(P) = \text{bounce}(\omega_n)$.

Complexity. The computation of $\mathcal{A}^{\omega}_{\varsigma}$ is done by iteratively adding the required processes and objectives. The steps of Algorithm 1 are done polynomially in the size of $\mathcal{A}^{\omega}_{\varsigma}$. Putting aside the $\mathbf{BS}^{\wedge}$ computation complexity, the proposed over-approximation can then be achieved by a number of operations polynomial in the size of the abstract structure.

Examples. Fig. 3 represents graphically the abstract structure that has been extracted from the Process Hitting example in Fig. 1 for a particular objective sequence and context. In this case, Theorem 1 is satisfied. Fig. 4 applies Theorem 1 to the Process Hitting example in Fig. 1 for the decision of the objective $d_1 \Rsh^* d_2$ concretizability. In this case, the necessary condition is not satisfied.

4.2. *Ordered Over-approximation*

This subsection exploits the ordering of objectives in an objective sequence to increase the conclusiveness of the over-approximation for its concretizability.

Given an objective P which is not trivial in the given context ς, we denote by $\text{ends}(P)$ the set of processes a scenario concretizing P may lead to (Def. 24, Proposition 3). This set is computed from $\mathbf{BS}(P)$ by taking the hitter and bounce of the last action in each bounce sequence.

Definition 24 ($\text{ends}_{\varsigma} : \mathbf{Obj} \mapsto \wp(\wp(\mathbf{Proc}))$)**.**

$$\text{ends}_{\varsigma}(\star \Rsh^* a_i) = \{\text{end}(h) \mid \exists a_j \in \varsigma[a], \exists \zeta \in \mathbf{BS}(a_j \Rsh^* a_i), \zeta \neq \varepsilon \wedge h = \zeta_{|\zeta|}\} \ .$$

Proposition 3. $\gamma_{\varsigma}(\star \Rsh^* a_i) \neq \emptyset \wedge a_i \notin \varsigma[a] \Longrightarrow \exists \delta \in \gamma_{\varsigma}(\star \Rsh^* a_i), \exists eps \in \text{ends}_{\varsigma}(\star \Rsh^* a_i)$ such that $eps \subseteq \text{end}(\delta)$.

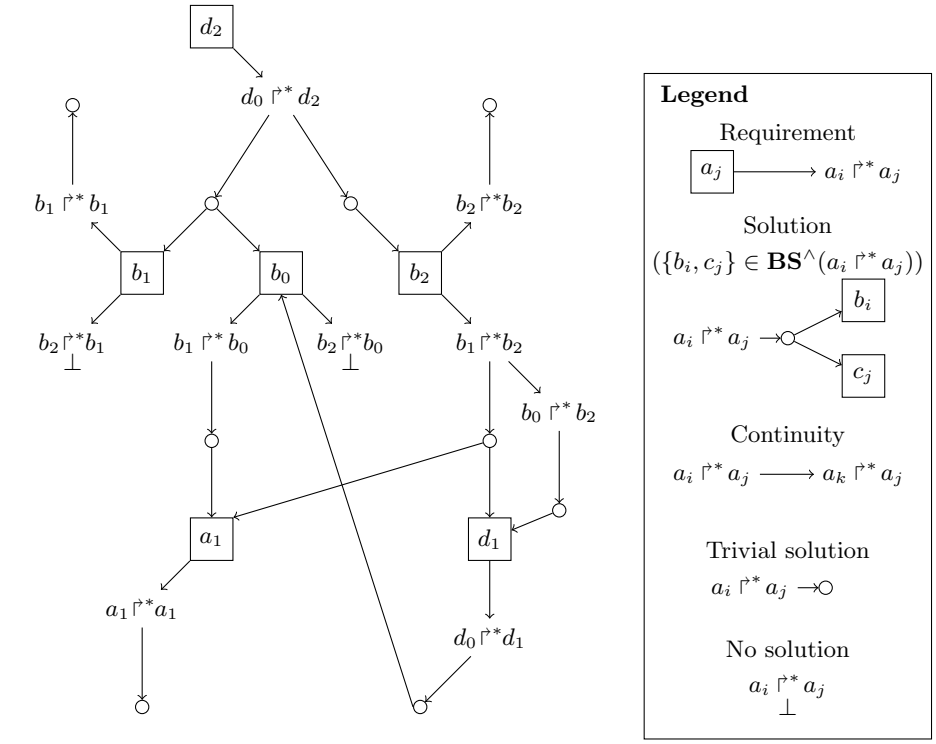


Figure 3. Graphical representation of the abstract structure $\mathcal{A}^\omega_\varsigma$ extracted from the Process Hitting in Fig. 1, with $\omega = d_0 \Rsh^* d_2$ and $\varsigma = \langle a_1, b_1, b_2, d_0 \rangle$.

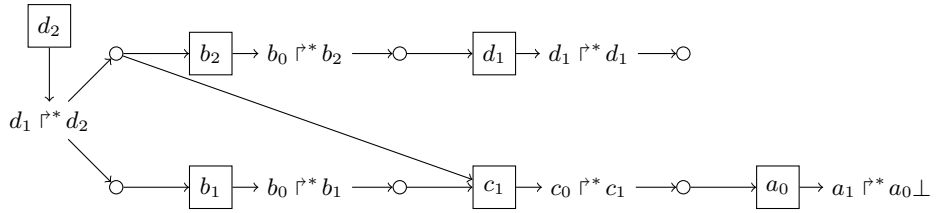


Figure 4. Abstract structure $\mathcal{A}^\omega_\varsigma$ for the Process Hitting example in Fig. 1 with $\omega = d_1 \Rsh^* d_2$ and $\varsigma = \langle a_1, b_0, c_0, d_1 \rangle$. By Theorem 1, the objective $d_1 \Rsh^* d_2$ is *not concretizable*.

Proof. As $a_i \notin \varsigma[a]$, there exists $n \in \mathbb{I}^\delta$ such that $\text{bounce}(\delta_n) = a_i$; hence $\delta_{1..n} \in \gamma_\varsigma(\star\Gamma^*a_i)$ and $\text{end}(\delta_n) \subseteq \text{end}(\delta_{1..n})$. By Def. 24, $\text{end}(\delta_n) \in \text{ends}_\varsigma(\star\Gamma^*a_i)$. □

Given an abstract structure $\mathcal{A}^\omega_\varsigma \in \mathbb{A}$, $\text{procs}(\mathcal{A}^\omega_\varsigma)$ is the set of processes referenced in $\mathcal{A}^\omega_\varsigma$ (Def. 25) from which derives Proposition 4. We then define $\text{maxprocs}_\varsigma(\omega)$ (Def. 26) as the set of processes present in the abstract structure having its context saturated (i.e., such that $\varsigma \Cap \text{procs}(\mathcal{A}^\omega_\varsigma) = \varsigma$). Note a particular optimisation when $\omega = P$, in which case the process $\text{bounce}(P)$ can be ignored by the context.

Definition 25 ($\text{procs} : \mathbb{A} \mapsto \wp(\mathbf{Proc})$**).**

$$\begin{aligned}\text{procs}((\text{Sol}^\omega_\varsigma, \text{Req}^\omega_\varsigma, \text{Cont}^\omega_\varsigma)) = \{p \in \mathbf{Proc} \mid &\exists (P, ps) \in \text{Sol}^\omega_\varsigma, p \in ps \\ &\vee\, p = \text{target}(P) \\ &\vee\, (P \neq \omega \Rightarrow p = \text{bounce}(P))\}\ .\end{aligned}$$

Proposition 4. For each objective $Q \neq \omega$ tested by Proposition 2 during the checking of Theorem 1 on $\gamma_\varsigma(\omega)$, $\text{bounce}(Q) \in \text{procs}(\mathcal{A}^\omega_\varsigma)$ and $\varsigma[\Sigma(Q)] \subseteq \text{procs}(\mathcal{A}^\omega_\varsigma)$.

Proof. By Lemma 5. □

Definition 26 ($\text{maxprocs}_\varsigma : \mathbf{OS} \mapsto \wp(\mathbf{Proc})$**).**

$$\text{maxprocs}_\varsigma(\omega) = \text{procs}(\lceil \mathcal{A}^\omega_\varsigma \rceil) \qquad \text{where } \lceil \mathcal{A}^\omega_\varsigma \rceil = \text{lfp}\{\mathcal{A}^\omega_\varsigma\}\ \left(\mathcal{A}^\omega_\varsigma \mapsto \mathcal{A}^\omega_{\varsigma \Cap \text{procs}(\mathcal{A}^\omega_\varsigma)}\right)\ .$$

By intersecting $\text{maxprocs}_\varsigma(\omega)$ and $\text{ends}_\varsigma(\omega_1)$, we obtain a context into which $\omega_{2..|\omega|}$ concretizability should satisfy Theorem 1, if ω is concretizable in ς. This is stated by Theorem 2, which gives a straightforward refinement of Theorem 1 by taking the sequentiality of objectives in ω into account.

Theorem 2 (Ordered over-approximation). Given a context ς and $\omega = P :: \omega' \in \mathbf{OS}$ such that $\text{bounce}(P) \notin \varsigma$, $\gamma_\varsigma(P :: \omega') \neq \emptyset \Longrightarrow$ Theorem 1 is satisfied with $\gamma_{max\varsigma}(\omega') \neq \emptyset$, where $max\varsigma = \varsigma \Cap \text{maxprocs}_\varsigma(\omega) \Cap eps$ and $eps \in \text{ends}_\varsigma(P)$.

Proof. $\delta \in \gamma_\varsigma(P :: \omega') \Rightarrow \exists n \in \mathbb{I}^\delta$ such that $\text{bounce}(\delta_n) = \text{bounce}(P), eps \subseteq \text{end}(\delta_{1..n})$, and $\delta_{n+1..|\delta|} \in \gamma_{\varsigma'}(\omega')$ with $\varsigma' = \text{support}(\delta)$. Therefore, Theorem 1 is satisfied with $\gamma_{\varsigma'}(\omega') \neq \emptyset$. Let Q be an objective tested by Proposition 2 when checking Theorem 1 satisfaction with $\gamma_{max\varsigma}(\omega') \neq \emptyset$. By Proposition 2, $\text{target}(Q) \in max\varsigma$. If $\text{target}(Q) \in eps$ then $\text{target}(Q) \in \varsigma'$, thus by hypothesis, Proposition 2 is satisfied. In the case of $\text{target}(Q) \notin eps$, by Def. 26 and Proposition 4, $\text{bounce}(Q) \in \text{maxprocs}_\varsigma(\omega)$, thus Proposition 2 holds. □

By defining $\text{maxCtx}(\varsigma, \omega, n)$ as the maximum context after the resolution of $\omega_{1..n}$ (Def. 27), the above theorem can be straightforwardly extended to Corollary 1.

Definition 27 ($\text{maxCtx} : \mathbf{Ctx} \times \mathbf{OS} \times \mathbb{N} \mapsto \mathbf{Ctx}$**).** (we assume $n \in \{0\} \cup \mathbb{I}^n$):

$$\text{maxCtx}(\varsigma, \omega, n) = \begin{cases} \varsigma & \text{if } n = 0, \\ \varsigma \Cap \text{maxprocs}_\varsigma(\omega) & \text{if } \text{bounce}(\omega_n) \in \text{maxCtx}(\varsigma, \omega, n-1) \\ \varsigma \Cap \text{maxprocs}_\varsigma(\omega) \Cap eps & \text{otherwise, where } eps \in \text{ends}_\varsigma(\omega_n) \end{cases}$$

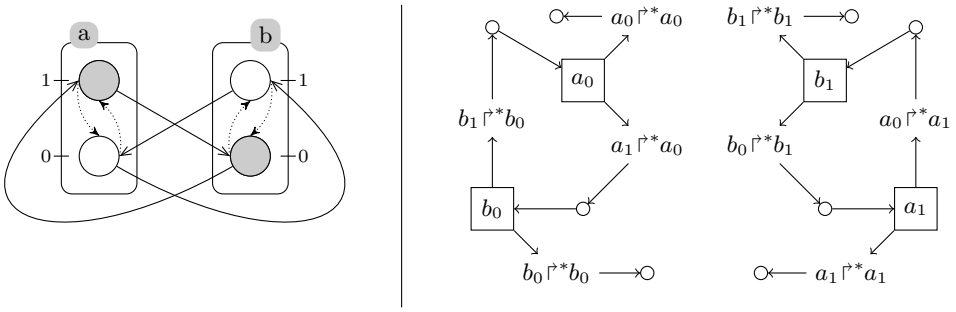


Figure 5. (right) Saturated abstract structure $\lceil \mathcal{A}^{\omega}_{\varsigma} \rceil$ of the (left) Process Hitting with $\varsigma = \langle a_1, b_0 \rangle$ and $\omega = a_1 \Rsh^* a_0 :: b_0 \Rsh^* b_1$. Theorem 2 concludes this objective sequence is not concretizable.

Corollary 1. $\gamma_{\varsigma}(\omega) \neq \emptyset \Longrightarrow \forall n \in \mathbb{I}^{\omega}, \gamma_{max\varsigma}(\omega_n) \neq \emptyset$, with $max\varsigma = \text{maxCtx}(\varsigma, \omega, n-1)$.

Implementation. The computation of $\text{maxprocs}_{\varsigma}(\omega)$ requires at most $|\mathbf{Proc}|$ iterations, giving a number of steps polynomial in the number of processes. Regarding the computation of $\text{ends}_{\varsigma}(\star \Rsh^* a_i)$, it can either be derived from prior $\mathbf{BS}(\star \Rsh^* a_i)$ computations; or, if **BS** are too costly to compute, it can be approximated by either $\{\{\text{bounce}(P)\}\}$, or by $\{\text{end}(h) \mid h \in \mathcal{H} \wedge \text{bounce}(h) = a_i \wedge \exists a_j \in \varsigma[a], \exists ps \in \mathbf{BS}^{\wedge}(a_j \Rsh^* a_i), \text{target}(h) \in ps\}$.

Example. Given the Process Hitting defined in Fig. 5, with $\varsigma = \langle a_1, b_0 \rangle$, Theorem 1 is inconclusive on the concretizability of $\omega = a_1 \Rsh^* a_0 :: b_0 \Rsh^* b_1$. By applying Theorem 2, it appears that $b_0 \Rsh^* b_1$ is not concretizable in $max\varsigma = \text{maxCtx}(\varsigma, \omega, 1) = \langle a_0, b_0 \rangle$ (in such a case, the $\mathcal{A}^{b_0 \Rsh^* b_1}_{max\varsigma}$ forms a unique cycle between b_1 and a_1).

4.3. *Over-approximation using Process Occurrences Order Constraints*

An objective $a_i \Rsh^* a_j$ having no solution ($\mathbf{BS}^{\wedge}(a_i \Rsh^* a_j) = \emptyset$) informs that the process a_j never occurs after a_i. This order constraints between process occurrences is referred to as $a_j \lhd a_i$ (Def. 28, Property 5).

Definition 28 ($\lhd$). The binary relation $\lhd \subset \mathbf{Proc} \times \mathbf{Proc}$ is a partial pre-order such that $a_j \lhd a_i$ (i.e., $(a_j, a_i) \in \lhd$) if and only if there exists no scenario where a_j occurs after a_i: $a_j \lhd a_i \iff \nexists \delta \in \mathbf{Sce}$ such that $\exists n, m \in \mathbb{I}^{\delta}, n \leq m, \text{target}(\delta_n) = a_i \wedge \text{bounce}(\delta_m) = a_j$.

Property 5 (Order constraint uncovering). $\mathbf{BS}(a_i \Rsh^* a_j) = \emptyset \Longrightarrow a_j \lhd a_i$.

Knowing some order constraints on process occurrences, we want to check if some sequence of objectives doesn't contradict such constraints. This can be achieved by computing the processes that always occur when resolving an objective: given an objective sequence ω, if a process a_i is required by ω_n and a process a_j by ω_m, $n < m$, then the constraint $a_j \lhd a_i$ should not exist. This is illustrated by Fig. 6.

In a similar fashion to $\text{minCont}_{\varsigma}$ (Def. 22), $\text{minProc}_{\varsigma}(P)$ refers to the set of processes of *any* sort that occur in all refinements of P in the context ς (Def. 29, Lemma 6). By

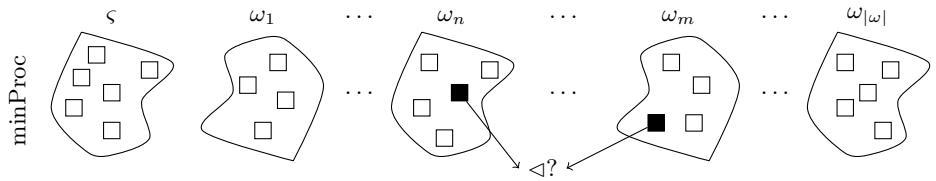


Figure 6. Illustration of the method developed in Subsect. 4.3 to over-approximate the concretizability of an objective sequence ω in the state ς: for each objective, the minimal set of occurring processes (represented as squares) are computed using minProc (Def. 30); the sequence is not concretizable as soon as two processes (in black) occurring in distinct objectives resolution contradict the process occurrences order $\lhd$.

using the previously defined maxCtx (Def. 27), we define $\text{minProc}(\varsigma, \omega, n)$ (Def. 30) the set of processes occurring in the resolution of ω_n after having resolved $\omega_{1..n-1}$. From this definition, Theorem 3 states the over-approximation illustrated in Fig. 6.

Definition 29 ($\text{minProc}_\varsigma : \mathbf{Obj} \mapsto \wp(\mathbf{Proc})$**).** Given a context ς,

$$\begin{aligned}
&\text{minProc}_\varsigma(\star \Rsh^* a_i) = \{p \in \mathbf{Proc} \mid \forall a_j \in \varsigma[a], \\
&\qquad\qquad \mathbf{BS}(a_j \Rsh^* a_i) \neq \emptyset \Rightarrow p \in \text{minProc}_\varsigma^{Obj}(a_j \Rsh^* a_i)\} \\
&\text{minProc}_\varsigma^{Obj} : \mathbf{Obj} \mapsto \wp(\mathbf{Proc}) \\
&\text{minProc}_\varsigma^{Obj}(a_j \Rsh^* a_i) = \{a_i\} \cup \{p \in \mathbf{Proc} \\
&\qquad\qquad \mid \forall ps \in \mathbf{BS}^\wedge(a_j \Rsh^* a_i), \exists q \in ps, p \in \text{minProc}_\varsigma(\star \Rsh^* q) \\
&\qquad\qquad \vee \exists a_k \Rsh^* a_i \in \text{minCont}_\varsigma^{Obj}(a, a_j \Rsh^* a_i), \\
&\qquad\qquad\quad p \in \text{minProc}_\varsigma^{Obj}(a_i \Rsh^* a_k) \cup \text{minProc}_\varsigma^{Obj}(a_k \Rsh^* a_i)))\} \ .
\end{aligned}$$

Lemma 6. $\forall \delta \in \gamma_\varsigma(P), \forall p \in \text{minProc}_\varsigma(P), p \in \delta$.

Proof. By induction on minProc_ς, p occurs in all recursive refinements of P by $\rho^\wedge$. □

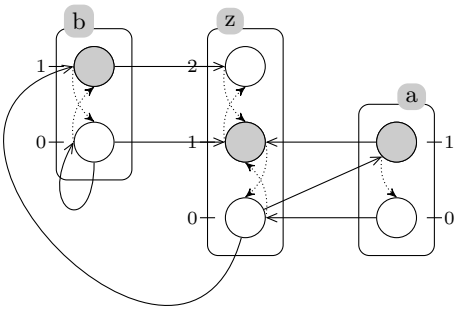
Definition 30 ($\text{minProc} : \mathbf{Ctx} \times \mathbf{OS} \times \mathbb{N} \mapsto \wp(\mathbf{Proc})$**).** (we assume $n \in \{0\} \cup \mathbb{I}^\omega$):

$$\text{minProc}(\varsigma, \omega, n) = \begin{cases} \{a_i \in \varsigma\} & \text{if } n = 0 \\ \text{minProc}_{max\varsigma}(\omega_n) & \text{otherwise,} \\ & \text{with } max\varsigma = \text{maxCtx}(\varsigma, w, n-1) \ . \end{cases}$$

Theorem 3 (Ordered over-approximation refined by $\lhd$). $\gamma_\varsigma(\omega) \neq \emptyset \Longrightarrow \nexists n, m \in \{0\} \cup \mathbb{I}^\omega, n < m, \exists p \in \text{minProc}(\varsigma, \omega, n), \exists q \in \text{minProc}(\varsigma, \omega, m), q \lhd p$.

Proof. By Lemma 6, Corollary 1 and Def. 30. □

Implementation. The implementation is very similar to the one presented in Subsect. 4.2. The uncovering of $\lhd$ is done linearly in the size of the saturated abstract structure $\lceil \mathcal{A}_\varsigma^\omega \rceil$.



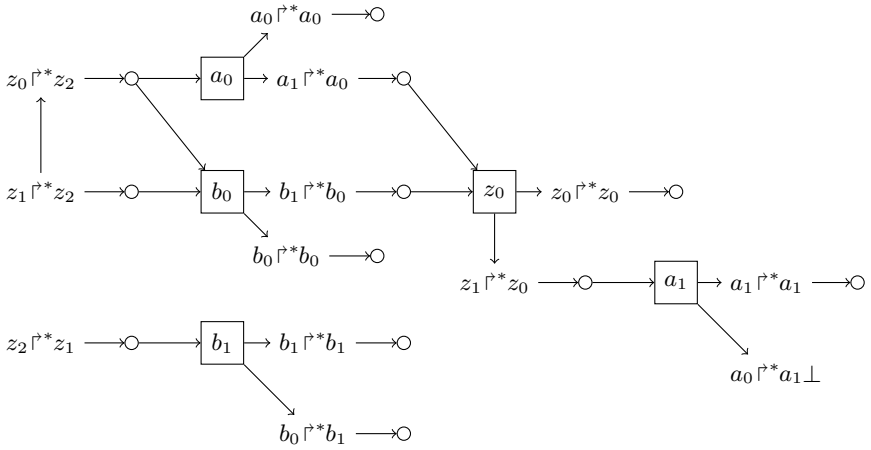


Figure 7. (bottom) Saturated abstract structure $\lceil \mathcal{A}_\varsigma^\omega \rceil$ of the (top) Process Hitting with $\varsigma = \langle a_1, b_1, z_1 \rangle$ and $\omega = z_1 \Rsh^* z_2 :: z_2 \Rsh^* z_1 :: z_1 \Rsh^* z_2$. Theorem 3 concludes this objective sequence is not concretizable.

Example. Let us define the Process Hitting as in Fig. 7, and its saturated abstract structure $\lceil \mathcal{A}_\varsigma^\omega \rceil$, with $\varsigma = \langle a_1, b_1, z_1 \rangle$ and $\omega = z_1 \Rsh^* z_2 :: z_2 \Rsh^* z_1 :: z_1 \Rsh^* z_2$, for which the concretizability has to be decided. The evaluation of $\text{minCont}(\varsigma, \omega, n)$ and $\text{maxCtx}(\varsigma, \omega, n)$ give the following:

	$n = 0 - \varsigma$	$n = 1 - z_1 \Rsh^* z_2$	$n = 2 - z_2 \Rsh^* z_1$	$n = 3 - z_1 \Rsh^* z_2$
$\text{minProc}(\varsigma, \omega, n)$	a_1, b_1, z_1	$\mathbf{a_0}, a_1, b_0, z_0, z_2$	b_1, z_1	$a_0, \mathbf{a_1}, b_0, z_0, z_2$
$\text{maxCont}(\varsigma, \omega, n)$	a_1, b_1, z_1	a_0, a_1, b_0, z_2	a_0, a_1, b_1, z_1	-

As $a_1 \lhd a_0$, ω is not concretizable in ς.

4.4. *Under-approximation*

The under-approximation procedure presented in this subsection takes advantage of a variant of the abstract structure used in the above over-approximations. If certain conditions on this abstract structure are verified, then it is shown that a scenario concretizing

the given objective sequence exists. The proposed construction of the scenario is made by a so-called *top-down* resolution: given an objective sequence ω, we first build the scenario concretizing ω_1 by preempting the resolution of the objective sequence $\omega_{2..|\omega|}$ (and hence, ignoring any objective interleaving that may be required to concretize ω). As stated by the refinement operators in Subsect. 3.4, the concretization of ω_1 involves the concretization of a refinement of ω_1, resulting in a recursive procedure of scenario construction.

We first define an alternative definition of the set of scenarios concretizing an objective sequence ω in a context ς: $\ell_\varsigma(\omega)$ is empty unless, for each state $s \in L$ included in ς, there exists a scenario $\delta \in \gamma_\varsigma(\omega)$ such that δ is playable in s; in that case, $\ell_\varsigma(\omega) = \gamma_\varsigma(\omega)$ (Def. 31). Property 6 is directly derived from this definition and the extension of ℓ_ς to a set of objective sequence is given in Def. 32.

Definition 31 ($\ell_\varsigma : \mathbf{OS} \mapsto \wp(\mathbf{Sce})$).

$$\ell_\varsigma(\omega) = \begin{cases} \gamma_\varsigma(\omega) & \text{if } \forall s \in L, s \subseteq \varsigma, \exists \delta \in \gamma_\varsigma(\omega), \text{support}(\delta) \subseteq s \\ \emptyset & \text{otherwise.} \end{cases}$$

Property 6. $\varsigma' \subseteq \varsigma \wedge \ell_\varsigma(\omega) \neq \emptyset \Longrightarrow \ell_{\varsigma'}(\omega) \neq \emptyset$.

Definition 32 ($\ell_\varsigma : \wp(\mathbf{OS}) \mapsto \wp(\mathbf{Sce})$). $\ell_\varsigma(\Omega) = \{\delta \in \ell_\varsigma(\omega) \mid \omega \in \Omega\}$

Given an objective P, $\text{maxCont}_\varsigma(\Sigma(P), P)$ (Def. 33) is the set of processes of sort $\Sigma(P)$ that may be encountered during the resolution of P. We then define the saturated abstract structure $\lceil \mathcal{B}^\omega_\varsigma \rceil = (\lceil \text{Req}^\omega_\varsigma \rceil, \lceil \text{Sol}^\omega_\varsigma \rceil, \lceil \text{Cont}^\omega_\varsigma \rceil)$ (Def. 34) similarly to $\lceil \mathcal{A}^\omega_\varsigma \rceil$ (Def. 23), except that $\lceil \text{Sol}^\omega_\varsigma \rceil$ can arbitrarily selects bounce sequences to resolve an objective, and that $\lceil \text{Cont}^\omega_\varsigma \rceil$ reflects maxCont_ς instead of minCont_ς. It appears that if $\lceil \mathcal{B}^\omega_\varsigma \rceil$ contains no cycle, and if all referenced objectives have at least one solution, then a top-down resolution of any referenced objective succeeds in every state of ς. This is stated by Theorem 4, which provides sufficient conditions for the concretization of an objective sequence in a given context.

Definition 33 ($\text{maxCont}_\varsigma : \Sigma \times \mathbf{Obj} \mapsto \wp(\mathbf{Proc})$).

$$\begin{aligned} \text{maxCont}_\varsigma(a, P) = \{p \in \mathbf{Proc} \mid & \exists ps \in \mathbf{BS}^\wedge(P), \exists b_i \in ps, b = a \wedge p = b_i \\ & \vee b \neq a \wedge p \in \text{maxCont}_\varsigma(a, b_j \Rsh^* b_i) \wedge b_j \in \varsigma[b])\} \ . \end{aligned}$$

Definition 34 ($\lceil \mathcal{B}^\omega_\varsigma \rceil$). The abstract structure $\lceil \mathcal{B}^\omega_\varsigma \rceil = (\lceil \text{Req}^\omega_\varsigma \rceil, \lceil \text{Sol}^\omega_\varsigma \rceil, \lceil \text{Cont}^\omega_\varsigma \rceil)$ is defined as $\lceil \mathcal{B}^\omega_\varsigma \rceil = \text{lfp}\{\mathcal{B}^\omega_\varsigma\} \left(\mathcal{B}^\omega_\varsigma \mapsto \mathcal{B}^\omega_{\varsigma \Cap \text{procs}(\mathcal{B}^\omega_\varsigma)}\right)$, with $\mathcal{B}^\omega_\varsigma = (\overline{\text{Req}^\omega_\varsigma}, \overline{\text{Sol}^\omega_\varsigma}, \overline{\text{Cont}^\omega_\varsigma})$:

$$\begin{aligned} \overline{\text{Req}^\omega_\varsigma} = & \{(a_i, a_j \Rsh^* a_i) \in \mathbf{Proc} \times \mathbf{Obj} \mid a_j \in \varsigma[a] \wedge (\exists (P, ps) \in \overline{\text{Sol}^\omega_\varsigma}, a_i \in ps \\ & \vee \exists n \in \mathbb{I}^\omega, \text{bounce}(\omega) = a_i\} \\ \overline{\text{Sol}^\omega_\varsigma} \subseteq & \{(P, ps) \in \mathbf{Obj} \times \wp(\mathbf{Proc}) \mid \exists (a_i, P) \in \overline{\text{Req}^\omega_\varsigma} \wedge ps \in \mathbf{BS}^\wedge(P)\} \\ \overline{\text{Cont}^\omega_\varsigma} = & \{(P, q \Rsh^* \text{bounce}(P)) \in \mathbf{Obj} \times \mathbf{Obj} \mid \exists (P, ps) \in \overline{\text{Sol}^\omega_\varsigma} \\ & \wedge q \in \text{maxCont}_\varsigma(\Sigma(P), P)\} \ . \end{aligned}$$

Theorem 4 (Under-approximation). If the graph $\lceil \mathcal{B}^\omega_\varsigma \rceil$ has no cycle and all referenced objectives have at least one solution, then $\ell_\varsigma(\omega) \neq \emptyset$.

Proof. We denote by $max\varsigma = \varsigma \Cap \text{procs}(\lceil \mathcal{B}^\omega_\varsigma \rceil)$ the context handled by $\lceil \mathcal{B}^\omega_\varsigma \rceil$. By induction on the acyclic graph $\lceil \mathcal{B}^\omega_\varsigma \rceil$, we prove that $\forall s \in L, s \subseteq max\varsigma$, for each objective P referenced in $\lceil \mathcal{B}^\omega_\varsigma \rceil$ such that target$P \in s$, $\exists \delta \in \ell_s(\omega)$ and $\text{end}(\delta) \subseteq max\varsigma$.

— $(P, \emptyset) \in \lceil \text{Sol}^\omega_\varsigma \rceil \Rightarrow$ either $\text{target}(P) = \text{bounce}(P)$ (thus $\delta = \varepsilon$), or $\forall \zeta \in \mathbf{BS}(P), \zeta \in \mathbf{Sce} \wedge \Sigma(\zeta) = \{\Sigma(P)\}$, thus $\delta = \zeta$.

— we assume all objectives children of P concretizables (no cycles). If $\exists Q \in \lceil \text{Cont}^\omega_\varsigma \rceil$, then by hypothesis, $\ell_s(\text{target}(P) \Rsh^* \text{target}(Q) :: Q) \neq \emptyset$, thus $\ell_s(P) \neq \emptyset$. Otherwise, by Def. 33, concretizations of children of P do not require any process of sort $\Sigma(P)$. Also, there exists $\zeta \in \mathbf{BS}(P)$ such that $(P, \zeta^\wedge) \in \lceil \text{Sol}^\omega_\varsigma \rceil$. By hypothesis, $\forall n \in \mathbb{I}^\zeta, \exists \delta^n \in \ell_{s^{n-1}}(\star \Rsh^* \text{hitter}(\zeta_n))$ with either $s^{n-1} = s$ if $n = 1$, or $s^{n-1} = s \cdot \delta^1 \cdot \ldots \cdot \delta^{n-1}$; and $\Sigma(P) \notin \Sigma(\delta^n)$ (by Def. 33). Therefore, $\delta = \delta^1 :: \zeta^1 :: \ldots \delta^n :: \zeta^n \in \ell_s(P)$ and $\text{end}(\delta) \subseteq max\varsigma$.

Finally, as $\ell_{max\varsigma}(\omega) \neq \emptyset$, $\ell_\varsigma(\omega) \neq \emptyset$ (Property 6). □

From the proof of the above theorem, we define $\text{endProc}(\lceil \mathcal{B}^\omega_\varsigma \rceil, P)$ (Def. 35) as the maximum set of processes a scenario built by Theorem 4 may end to (Corollary 2). This allows a straightforward extension of Theorem 4 to take the sequentiality of objectives into account (Corollary 3).

Definition 35 ($\text{endProc} : \mathbb{B} \times \mathbf{Obj} \mapsto \wp(\mathbf{Proc})$)**.**

$$\begin{aligned}\text{endProc}(\lceil \mathcal{B}^\omega_\varsigma \rceil, P) = \{p \in \mathbf{Proc} \mid\ & \Sigma(p) = \Sigma(P) \Rightarrow p = \text{bounce}(P) \\ & \wedge (\exists (P, Q) \in \lceil \text{Cont}^\omega_\varsigma \rceil, p \in \text{endProc}(\lceil \mathcal{B}^\omega_\varsigma \rceil, \text{target}(P) \Rsh^* \text{target}(Q)) \\ & \qquad \vee p \in \text{endProc}(\lceil \mathcal{B}^\omega_\varsigma \rceil, Q) \\ & \vee \exists (P, ps) \in \lceil \text{Sol}^\omega_\varsigma \rceil, \exists b_i \in ps, \exists b_j \in \varsigma[b], p \in \text{endProc}(\lceil \mathcal{B}^\omega_\varsigma \rceil, b_j \Rsh^* b_i))\}\end{aligned}$$

Corollary 2. Given $P \in \mathbf{Obj}$, if Theorem 4 is satisfied with $\ell_\varsigma(P) \neq \emptyset$, then $\forall s \in L, s \subset \varsigma, \exists \delta \in \ell_\varsigma(P)$ such that $\text{end}(\delta) \subseteq \text{endProc}(\mathcal{B}^P_\varsigma, P)$.

Corollary 3. Given $P \in \mathbf{Obj}$, and $\omega \in \mathbf{OS}$, if Theorem 4 is satisfied with $\ell_\varsigma(P) \neq \emptyset$, and if Theorem 4 is satisfied with $\ell_{\varsigma'}(\omega) \neq \emptyset$, where $\varsigma' = \varsigma \Cap \text{endProc}(\mathcal{B}^P_\varsigma, P)$, then Theorem 4 is satisfied with $\ell_\varsigma(P \oplus \omega) \neq \emptyset$.

Implementation. The computation of the saturated abstract structure $\lceil \mathcal{B}^\omega_\varsigma \rceil$ works by progressive addition of relations between objectives and processes, found by several traversing of the abstract structure, giving a maximal complexity polynomial in the size of the abstract structure. Checking for Theorem 4 conditions is done linearly in the size of the obtained abstract structure. It is worth noticing that arbitrarily selecting solutions for objectives in $\lceil \mathcal{B}^\omega_\varsigma \rceil$ prevents spurious saturations and may increase the satisfaction of the above theorem, but potentially increases the complexity of checking (as several combinations of solutions can be tested).

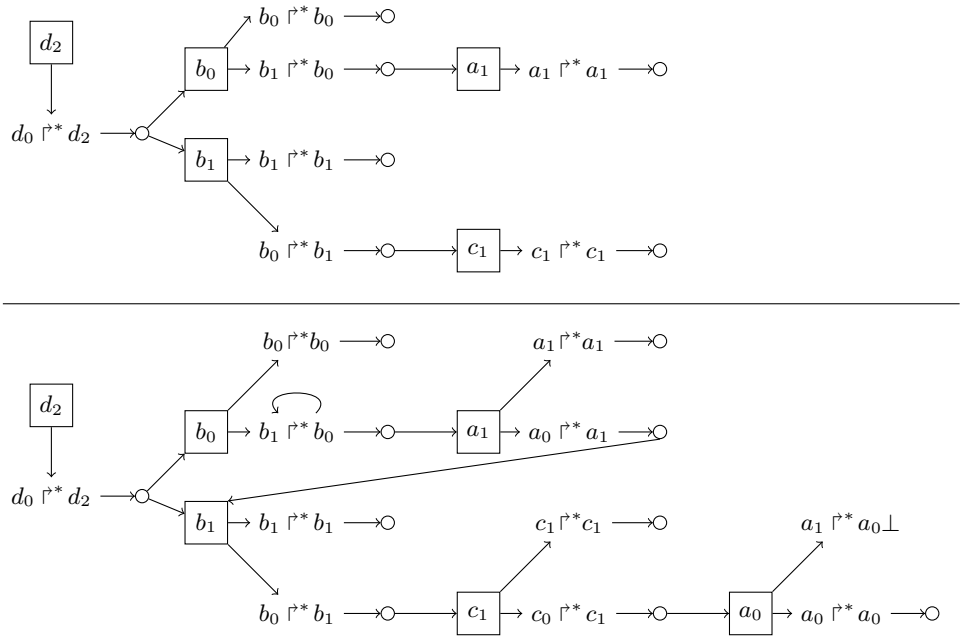


Figure 8. Saturated abstract structure $\lceil \mathcal{B}^{\omega}_{\varsigma} \rceil$ from the Process Hitting in Fig. 1 with $\omega = d_0 \Rsh^* d_2$ and $\varsigma = \langle a_1, b_1, c_1, d_0 \rangle$ (top), and $\varsigma = \langle a_0, b_1, c_0, d_0 \rangle$ (bottom). By Theorem 4, ω is concretizable in $\langle a_1, b_1, c_1, d_0 \rangle$. At this stage, our procedure is inconclusive for the concretizability of ω in $\langle a_0, b_1, c_0, d_0 \rangle$.

Examples. Fig. 8 shows two examples of the application of Theorem 4 on the Process Hitting example in Fig. 1.

Discussion. Corollary 3 suggests that testing a refined objective sequence may reveal more conclusive than testing the original given objective sequence. A future work may use of graph analysis of $\lceil \mathcal{B}^{\omega}_{\varsigma} \rceil$ to determine which refined objective sequences are good candidates to satisfy Theorem 4, and then increase the conclusiveness of the method.

5. Application to Biological Regulatory Networks

5.1. *From Biological Regulatory Networks to Process Hittings*

We first sketch the modelling of 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. 9. The full formalisation of this translation can be found in (Paulevé et al., 2011).

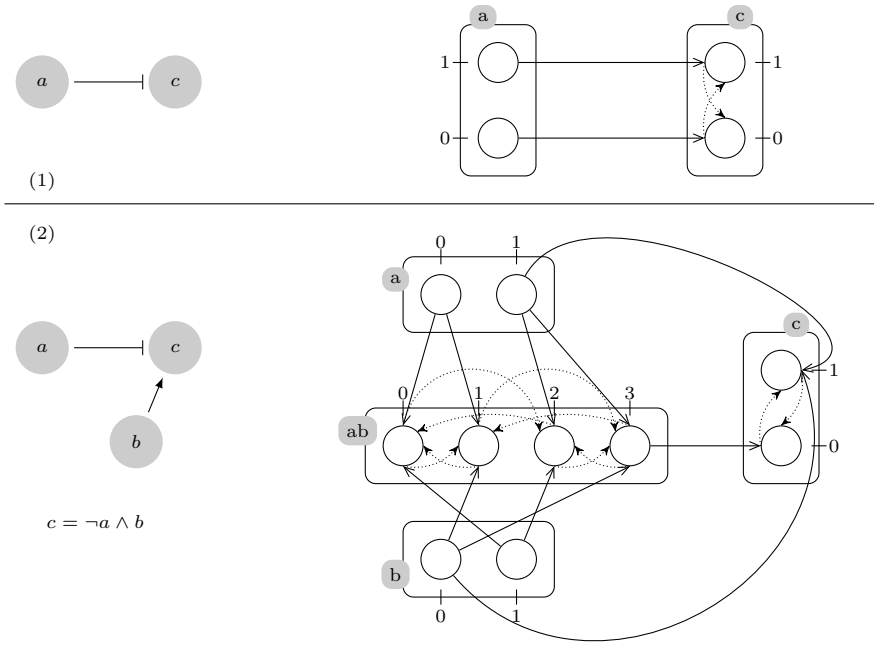


Figure 9. Examples of Process Hittings (right) from BRNs, having interaction graphs at left. (1) simple inhibition of c by a. (2) boolean function between a (inhibitor) and b (activator) on c ab reflects the state of sorts a and b. In this case, ab_3 reflects the state $\langle a_0, b_1 \rangle$ (and, ab_0 the state $\langle a_1, b_0 \rangle$, ab_1 the state $\langle a_1, b_1 \rangle$, ab_2 the state $\langle a_0, b_0 \rangle$).

5.2. *T-Cell Receptor Signalling Pathway (94 components)*

Presented in (Saez-Rodriguez, Simeoni, Lindquist, Hemenway, Bommhardt, Arndt, Haus, Weismantel, Gilles, Klamt and Schraven, 2007), this biological system models the T-Cell Receptor (TCR) signalling pathway, the behaviours of which reveal an activation of transcription factors controlling the cell's fate, e.g. whether it proliferates or not. This model is an extension of a former presented BRN relating 40 components (Klamt, Saez-Rodriguez, Lindquist, Simeoni and Gilles, 2006).

The Process Hitting model[†] of this system is composed of 1124 actions between 448 processes splitted in 133 sorts (the largest sort has 16 processes). The total number of states of this model is 2^{194} ($\approx 2 \cdot 10^{58}$).

Independant reachability decisions have been experimented from all possible inputs combinations (components CD45, CD8, TCRlig) to each output (components SRE, AP1, CRE, NFkB, NFAT, Cyc1, p21c p27k, FKHR, BclXL). *All result in conclusive decisions.* It is worth noticing that only approximations defined by Theorem 1 (Subsect. 4.1) and

[†] Model and implementation available at `http://processhitting.wordpress.com`

Theorem 4 (Subsect. 4.4) have been exploited, showing quite simple dynamics for the independant reachability of components.

Computation times are around the hundredth of a second on a 3GHz processor with 2GB of RAM. To give a comparison, we did the same experiments with a standard symbolic model-checking method using state-space compression based on Hierarchical Set Decision Diagrams (SDDs) (Hamez et al., 2009): the libddd framework (LIP6/Move, 2007), known for its good performances. For the majority of reachability decisions, the program runs out of memory, for others, computation times range from some seconds to hours. This shows the remarkable efficiency of our method, based on abstract interpretation.

6. 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 the 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 by abstract interpretation has been developed to decide the successive reachability of processes, hence of component levels in the scope of BRNs modelling. The computation is done by over- and under-approximation of the decision, and may reveal to be inconclusive. It exploits two complementary abstractions of scenarios in Process Hittings (by objective and bounce sequences). Several refinements of an objective sequence are then defined to detail the required steps necessary to its concretizability. These refinements are exploited to derive necessary or sufficient conditions for the process reachability satisfaction. Further work may improve the conclusiveness of our method, as it is discussed in Subsect. 4.4. Also, the link between the conclusiveness of the developed method and the structure of the interaction graph of the BRN could be studied.

The implementation of the presented approximations use of an abstract structure that is ensured to have a size nearly polynomial in the total number of processes. On the one hand, the computation of the refinements can be exponential in the number of processes within a single sort; on the other hand the computations of the decisions are polynomial in the size of the abstract structure. Hence, it is expected efficient analyses when the number of processes within a sort is limited, while a large amount of sorts should be handled.

This new and original approach has been applied to the analysis of a large BRN relating 94 components. Response times are really fast (around the hundredth of a second on a desktop computer), showing a promising scalability of our method. It has been compared to a standard symbolic model checking techniques which regularly fails to analyse the model, because of the state space explosion. To our knowledge, this is the first successful application of model checking to BRNs of such a size.

Related work. There has been recent work on the fast computation the set of reachable components within Kappa models (Danos, Feret, Fontana and Krivine, 2008) (a rule

based language). In the general case, this set is over-approximated and such an analysis does not permit to decide the successive reachability of processes, as it is done in this paper. The work presented in (Alimonti, Feuerstein, Laura and Nanni, 2011), on so-called T-Paths within Petri Nets, establishes structural properties within Petri Nets to derive either necessary or sufficient conditions for place marking reachability. This follows a similar approach to ours. Finally, (Pilegaard, Nielson and Nielson, 2005) applies static analysis techniques of bioambient models to study the behaviour of biological systems.

Future work investigate other applications of the presented refinements of the abstraction interpretations of Process Hitting. In particular, as they extract the causality between process changes, they point up processes required for a certain process reachability. This kind of analysis may lead to a control of the studied system by acting upon these *key* processes: e.g. knockdowning a key gene to prevent a cascade of gene activation (gene therapy).

Another 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

Alimonti, P., Feuerstein, E., Laura, L. and Nanni, U. (2011). Linear time analysis of properties of conflict-free and general petri nets, *Theoretical Computer Science* **412**(4-5): 320 – 338.

Aracena, J. (2008). Maximum number of fixed points in regulatory boolean networks, *Bulletin of Mathematical Biology* **70**(5): 1398–1409.

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

Bernot, G., Comet, J.-P. and Khalis, Z. (2008). Gene regulatory networks with multiplexes, *European Simulation and Modelling Conference Proceedings*, pp. 423–432.

Brand, D. and Zafiropulo, P. (1983). On communicating finite-state machines, *Journal of the ACM* **30**: 323–342.

Clarke, E. M. and Emerson, E. A. (1981). Design and synthesis of synchronization skeletons using branching-time temporal logic, *Logic of Programs*, Springer-Verlag, London, UK, pp. 52–71.

Cousot, P. and Cousot, R. (1977). Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints, *Proceedings of the 4th ACM SIGACT-SIGPLAN symposium on Principles of programming languages*, POPL '77, ACM, New York, NY, USA, pp. 238–252.

Danos, V., Feret, J., Fontana, W. and Krivine, J. (2008). Abstract interpretation of cellular signalling networks, *in* F. Logozzo, D. Peled and L. Zuck (eds), *Verification, Model Checking, and Abstract Interpretation*, Vol. 4905 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 83–97.

Hamez, A., Thierry-Mieg, Y. and Kordon, F. (2009). Building efficient model checkers using hierarchical set decision diagrams and automatic saturation, *Fundam. Inf.* **94**(3-4): 413–437.

Klamt, S., Saez-Rodriguez, J., Lindquist, J., Simeoni, L. and Gilles, E. (2006). A methodology for the structural and functional analysis of signaling and regulatory networks, *BMC Bioinformatics* **7**(1): 56.

LIP6/Move (2007). the libDDD environment. `http://ddd.lip6.fr`.

Paulevé, L., Magnin, M. and Roux, O. (2010). Abstract Interpretation of Dynamics of Biological Regulatory Networks, *The First International Workshop on Static Analysis and Systems Biology (SASB 2010)*, Vol. To appear. Preprint: `http://www.irccyn.ec-nantes.fr/~pauleve/sasb10.pdf`.

Paulevé, L., Magnin, M. and Roux, O. (2011). Refining Dynamics of Gene Regulatory Networks in a Stochastic π-Calculus Framework, *Transactions on Computational Systems Biology* **XIII (in press)**. Preprint: `http://www.irccyn.ec-nantes.fr/~pauleve/refining-revised.pdf`.

Paulevé, L. and Richard, A. (2010). Topological Fixed Points in Boolean Networks, *Comptes Rendus de l'Académie des Sciences - Series I - Mathematics* **348**(15-16): 825 – 828.

Pilegaard, H., Nielson, F. and Nielson, H. R. (2005). Static analysis of a model of the LDL degradation pathway, *in* G. Plotkin (ed.), *Third International Workshop on Computational Methods in Systems Biology (CMSB'05)*, University of Edinburgh.

Remy, É., Ruet, P. and Thieffry, D. (2008). Graphic requirements for multistability and attractive cycles in a boolean dynamical framework, *Advances in Applied Mathematics* **41**(3): 335 – 350.

Richard, A. (2010). Negative circuits and sustained oscillations in asynchronous automata networks, *Advances in Applied Mathematics* **44**(4): 378 – 392.

Richard, A. and Comet, J.-P. (2007). Necessary conditions for multistationarity in discrete dynamical systems, *Discrete Applied Mathematics* **155**(18): 2403 – 2413.

Richard, A., Comet, J.-P. and Bernot, G. (2006). *Modern Formal Methods and Applications*, chapter Formal Methods for Modeling Biological Regulatory Networks, pp. 83–122.

Saez-Rodriguez, J., Simeoni, L., Lindquist, J. A., Hemenway, R., Bommhardt, U., Arndt, B., Haus, U.-U., Weismantel, R., Gilles, E. D., Klamt, S. and Schraven, B. (2007). A logical model provides insights into t cell receptor signaling, *PLoS Comput Biol* **3**(8): e163.

Thomas, R. (1973). Boolean formalization of genetic control circuits, *Journal of Theoretical Biology* **42**(3): 563 – 585.