

Analyzing Large Network Dynamics with Process Hitting

Loïc Paulevé, Courtney Chancellor, Maxime Folschette, Morgan Magnin, Olivier Roux

▶ **To cite this version:**

Loïc Paulevé, Courtney Chancellor, Maxime Folschette, Morgan Magnin, Olivier Roux. Analyzing Large Network Dynamics with Process Hitting. Luis Fariñas del Cerro; Katsumi Inoue. Logical Modeling of Biological Systems, Wiley, pp.125 - 166, 2014, 978-1-84821-680-8. hal-01060490

HAL Id: hal-01060490
https://hal.science/hal-01060490

Submitted on 3 Sep 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Chapter 4

Analyzing Large Network Dynamics with Process Hitting

4.1. Introduction / State of the art

4.1.1. *The modeling challenge*

Regulation is a key aspect of biological systems, all the way from the molecular scale to the ecological one. Gaining a precise understanding of regulation is one of the main goals of systems biology. This discipline has emerged from the synergy between cell biology and cybernetics [WIE 48], from the collaboration of biologists, physicists and computer scientists [IDE 01]. The modeling approach presented in this chapter derives from this heritage by studying the interactions of components of a biological system and analyzing how these interactions impact the function and behavior of the system as a whole. Here, we will focus on applications to genetics, but the potential field of the approach is broader: the Process Hitting framework is relevant to any interactive system, whether it is a biological regulatory network, a logistic scheme or an embedded system.

The recent progress in molecular biology has made it possible to obtain a comprehensive map of the genomes of many living organisms. Simultaneously, the development of DNA micro array technology has given access to time series data of the expression of several thousands of genes. One of the main challenges now is to

---

Chapter written by Loïc PAULEVÉ and Courtney CHANCELLOR and Maxime FOLSCHETTE and Morgan MAGNIN and Olivier ROUX.

integrate this high-throughput data in order to infer genetic networks using this previously inaccessible time series data. Because the interactions occur on different scales (genes, proteins, biochemical components, cells, etc.), it is necessary to build methods that can formally learn from biological data at a system-level understanding.

The modeling of biological regulation can be decomposed into two main trends. The first is based on ordinary differential equations involving the quantitative expression of the interacting components. However, these equations are generally non-linear, preventing the design of an analytical solution. In addition, the biological data we get from experiments is generally quite noisy, thus needs to be filtered efficiently. By contrast, the second trend consists of addressing the problem using discrete methodology. Even though discrete modeling could be seen as a less faithful abstraction, it has been proven efficient in tackling many qualitative biological questions (e.g., understanding how biological systems evolve or determining the reachability of some states).

4.1.2. *Historial context: Boolean and discrete models*

As data in a biological context is often more qualitative than quantitative, it is meaningful that another trend, based on qualitative modeling, emerged in the late 1960s. The principle of this modeling framework, introduced as synchronous Boolean networks by Stuart Kauffman on the one hand [KAU 69], asynchronous René Thomas' networks on the other [THO 91], is to represent genes as Boolean variables. These variables, or nodes, can take only two states, on or off, modeling the fact that their influence is either active or not. Between the Boolean variables are activation and inhibition relations, respectively represented by positive and negative edges. In the following paragraph, we discuss the differences between Stuart Kauffman's and René Thomas' models.

Stuart Kauffman established an idealized representation of a gene regulatory network: he considered the state of genes as being described by Boolean variables, the next state resulting from the application of a synchronous Boolean function governing the state of each variable. Meanwhile, René Thomas proposed a comprehensive logical description of the mechanisms governing transcription regulation. He considered the influence of an activating (resp. inhibiting) gene on its target as depending on a threshold value: when the level of the gene is greater or equal to the threshold, the gene has a positive (resp. negative) effect; when the level of the gene is lower than the threshold, the gene has a negative (resp. positive) effect [RIC 06a]. Even if the specification of these two models is different, they both rely on the central idea that the next state depends on a Boolean function of the current state. The main difference between the two paradigms is found in the update function: while Boolean networks are mainly studied with respect to synchronous semantics, René Thomas' approach deals with an asynchronous one. Recent works [ARA 09, NOU 11a, NOU 11b] have discussed the impact of synchronism compared to asynchronism on the network behaviors.

Both models were considered of interest and, after their publication, were followed by numerous additional works. For example, for Boolean networks, the authors of [AKU 08] review a panel of algorithms to address inference, identification of attractors and control in Boolean networks. However, because the size of the state space grows exponentially with the number of components, classical approaches may be intractable for large systems. Therefore, a significant number of analysis were designed to make explicit the relationship between the interaction graph of the network and its dynamic properties. For instance, some innovative works focus on feedback loops, i.e., circular chains of interactions. It can be shown that the number of inhibition interactions in the loop is a sufficient condition for multiple steady states. In [PAU 11c], the authors give insight on the relationship between the interaction graph and the dynamical properties of Boolean networks. Another recently developed approach consists of analyzing Boolean networks as logical programs. In [INO 11], the authors demonstrate direct translations of Boolean networks into logic programs and exploiting methods of inferring these logic programs to compute the Boolean network trajectories and attractors.

These Boolean paradigms may appear to be simplified models, but they led to significant results about the behavior of networks, such as cyclicity or steady states. Moreover, these models have been extended over the years, for example, to consider additional levels of expressions, paving the way for discrete networks as we present them here.

4.1.3. *Analysis issues*

Thanks to qualitative approaches, it is easier — although still quite complicated — to get a deep understanding of the interactions involved in a particular network. This methodology aims at explaining how the components of gene regulatory networks are controlled or predict some previously unobserved behavior which can be experimentally tested. This is of crucial importance, especially in the field of synthetic biology and drug design. Challenges encountered in analysis and prediction can be thought of as falling into one of three major categories:

– Identification of parameters: discrete models such as Boolean networks or Thomas' networks require not only information about the topology of the network and the type of influences — activation or inhibition — between genes but also about the strength of each interaction. This information is needed to determine, for example, the net result when two genes have opposing effects on a single target. These parameters can be obtained from biological experimental observations via model-checking or constraint programming.

– Inference of the model: the problem of inference is not limited to parameters but is also found in the global structure of the network. Given some time series data which can be translated into a partial state transition system, one may want to build a discrete model that is consistent with the available data.

– Identification of steady states and attractors, key properties of most biological systems.

– Control of the model: closely linked to the analysis of key properties is the ability to control the model such that it demonstrates some desired behavior or, conversely, prevents some unwanted behavior. This is the most recent, yet challenging, topic in systems biology directly connected to gene therapy.

These difficulties in classical approaches come short of our overall modeling goals. Although the available data on the interaction graph between genes is more and more extensive, there is an overall lack of precise quantitative kinetic data. As previously mentioned, experimentally obtained data is generally noisy, making it difficult to infer parameter values. As a consequence, parameter identification requires some indirect reasoning, which becomes tricky when the model grows beyond ten interacting components; due to the combinatorial explosion, it is extremely difficult to to handle large, realistic regulatory networks.

4.1.4. *The Process Hitting framework*

The Process Hitting framework was introduced in order to address the scalability issue [PAU 11a]. Three key concepts (some of them inspired by π-calculus) make up the foundation of this qualitative approach:

– Biological components (e.g., genes) are abstracted as *sorts* which are in turn divided into different processes which correspond to the qualitative discrete levels that the sort may represent.

– Every interaction between genes is represented as a *hit* from one process of a sort to another process of another sort.

– Such a representation allows to build the most generalized dynamics then proceed by successive refinement, for example the introduction of *cooperative sorts* to represent the combined influences of multiple genes on a single target.

Process Hitting lends itself to modeling gene regulatory networks with different levels of abstraction by capturing the most general dynamics. Establishing relationships between the components at the most atomic level possible, the Process Hitting opens the way to many static analysis and abstract interpretation methods to study complex dynamic properties. Static analyses have already been developed in the Process Hitting framework, notably for obtaining all the fixed points of dynamics of a Process Hitting. Being a particular restriction of asynchronous automata networks or safe Petri nets [BER 92], Process Hitting can be applied to complex dynamical systems with a very large number of interacting components, where each of these components can be described with a few internal states. Throughout the chapter, we will illustrate the efficiency of the approach and show that it can be applied to large

networks (up to 10,000 components). Our methods produce very fast responses for the decision of successive reachability questions, while well-established symbolic model checking techniques regularly fail because of the state-space explosion. Thanks to this efficiency, it appears that the Process Hitting approach offers a promising perspective to the inference of gene regulatory networks even when a large amount of biological data must be processed.

4.1.5. *Outline*

This chapter is structured as follows. Section §4.2 introduces the Process Hitting framework, focusing on the associated methodology of constructing the most permissive dynamics (called the generalized dynamics) and then using successive refinements to fine tune the model. Section §4.3 presents static analysis methods designed to identify fixed points or answer successive reachability questions. Section §4.4 introduces the stochastic semantics of Process Hitting and draws links to available numerical techniques. The overall approach is illustrated by biological applications in section §4.5. Here, we present the Pint software designed to manipulate Process Hitting models, give some benchmarks and discuss their results. Finally, section §4.6 concludes on the merits of the framework and gives an overview of future works.

4.2. Discrete Modeling with the Process Hitting

4.2.1. *Motivation*

The modeling and analysis of the qualitative dynamics of large interaction networks face two important challenges. For modeling, it is very rare that all logical functions governing the dynamics are known, even if the interaction graph is considered as prior knowledge. One technique would then be to enumerate all logical networks that are compatible with the prior knowledge on the interaction graph and partial logical functions. But, with large networks, such an enumeration suffers from high combinatorics. For analysis, because the size of the state transition graph grows exponentially with the number of components, proving dynamical properties becomes intractable for networks with hundreds or thousands of elements.

In order to illustrate the challenge of modeling interactions with partial knowledge, let us consider a simple interaction graph where both b and c are the sole activators of a. Typically, we can assume that if both b and c are inactive (resp. active), then a should eventually become inactive (resp. active). However, the case where only one of either b or c is active is undefined without additional knowledge on the system. We cannot be sure that only one of the two activators will be enough to push a past its relative threshold. To conduct an exhaustive analysis would then require us to consider all possible Boolean functions compatible with the system.

An alternative approach would be to instead consider the dynamical (transition) system which includes the dynamics (transitions) of all compatible Boolean functions. In such a setting, if we prove that a behavior is impossible in the model, we have proved that it is impossible for any Boolean function compatible with the prior knowledge. This modeling by over-approximation allows us to reason on the dynamics of multiple models while only looking at one model. In our simple example of two activators, one could characterize the largest compatible dynamics with the simple nondeterministic function $f_a(x) = \begin{cases} 1 & \text{if } x[b] = 1 \vee x[c] = 1 \\ 0 & \text{if } x[b] = 0 \vee x[c] = 0 \end{cases}$ — where x is the state of the network, $x[b]$ is the value (0 or 1) of the component b in x, and f_a associates the state of the network with the next value of a. Such a function has a nondeterministic result when b is different from c, as no knowledge permits us to decide on a value for a, a phenomenon which cannot be represented using classical Boolean or multi-valued frameworks for modeling dynamics of interaction networks.

These modeling and analysis challenges motivated the definition of a new formalism, the *Process Hitting*. Thanks to the atomic description of possible transitions, the Process Hitting enables us to easily model interactions with partial knowledge (that is, nondeterministic logical functions). Moreover, as we explain in section §4.3, the particular structure of Process Hitting models makes possible the definition of powerful static analysis for deciding dynamical properties such as fixed points, reachability and cut sets for reachability. Those results make the modeling and analysis of dynamics of very large networks tractable.

In the next subsection, we give the formal definition of Process Hitting and its semantics. The modeling of the most permissive dynamics delimited by an interaction graph is detailed in section §4.2.3, and section §4.2.4 explains how such dynamics can be refined to take into account known cooperative or synergistic interactions between regulators. In those two subsections, we use the incoherent feed-forward loop interaction network to illustrate the modeling of dynamics using Process Hitting with partial or complete knowledge on cooperations between regulators. Finally, the relationship with the classical Boolean (and multi-valued) modeling is discussed in section §4.2.5.

4.2.2. *The Process Hitting Framework*

Definition and Semantics

Process Hitting gathers a finite number of concurrent *processes* grouped into a finite set of *sorts* (or automata). 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 $a_i \to b_j \Rsh b_k$ 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 $h = a_i \to b_j \Rsh b_k$ 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 the *hitter*, the *target* and the *bounce* of the action, or $\mathsf{hitter}(h), \mathsf{target}(h), \mathsf{bounce}(h)$.

DEFINITION.– *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;* $a \neq b \Rightarrow \forall (a_i, b_j) \in L_a \times L_b, a_i \neq b_j$.

– $\mathcal{H} = \{a_i \to b_j \Rsh b_k, \cdots \mid (a, b) \in \Sigma^2,\ (a_i, b_j, b_k) \in L_a \times L_b \times L_b, b_j \neq b_k, a = b \Rightarrow a_i = b_j\}$*, is the finite set of actions.*

$\mathbf{Proc}$ *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 $\mathsf{sort}(a_i) = a$ and the set of sorts present in an action $h \in \mathcal{H}$ as $\mathsf{sorts}(h) = \{\mathsf{sort}(\mathsf{hitter}(h)), \mathsf{sort}(\mathsf{target}(h))\}$. 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 $a_i \in L_a$, $a_i \in s \overset{\Delta}{\Leftrightarrow} s[a] = a_i$; and if $ps \in \wp(\mathbf{Proc})$, $ps \subseteq s \overset{\Delta}{\Leftrightarrow} \forall a_i \in ps, a_i \in s$.

An action $h = a_i \to b_j \Rsh b_k \in \mathcal{H}$ 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 $\forall c \in \Sigma, c \neq b, (s \cdot h)[c] = s[c]$. Among sequences of actions, the particular sequences only composed of successively playable actions form *scenarios*.

EXAMPLE.– Figure 4.1 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 $s = \langle a_0, b_1, c_0, d_0 \rangle$ which results in the state $\langle a_1, b_2, c_1, d_1 \rangle$.

4.2.3. *Generalized Dynamics of Interaction Graphs*

The starting point for modeling dynamics of interaction networks is typically the Interaction Graph (IG). In this section we assume that the IG is our only prior knowledge, that is, we have no constraints on logical functions. In addition, we assume that

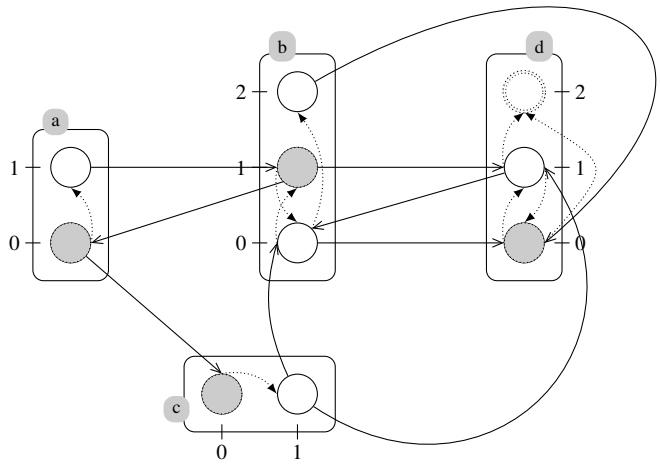


Figure 4.1: A Process Hitting example. Sorts are represented by labeled boxes, and processes by circles (ticks are the identifiers of the processes within the sort, for instance, a_0 is the process ticked 0 in the box a). The current state is represented by the grayed processes: $\langle a_0, b_1, c_0, d_0 \rangle$. An action (for instance $a_0 \rightarrow c_0 \not\uparrow c_1$) 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.

for any component, if all of its activators are present (resp. absent) and all its inhibitors are absent (resp. present), the level of the component cannot decrease (resp. increase); we refer to this constraint as the *extreme values assumption*.

Due to the extreme values assumption, the IG already delimits the dynamics (possible transitions) of the regulatory network. In this section, we show how to build the most permissive dynamics with respect to an IG using Process Hitting which we call the *generalized dynamics* of the IG. These dynamics include, in term of transitions, the dynamics of any asynchronous Boolean/multi-valued network sharing the same IG. A formal link is addressed in section §4.2.5.

We must first settle on one possible formal definition of an Interaction Graph (IG) for discrete networks. In such IGs, an edge is either positive (activation) or negative (inhibition) and cannot be both (there are no unsigned edges). In addition, each regulation receives a discrete threshold: if the regulator is below (resp. greater or equal to) the threshold, the effective sign of the regulation is the negative of (resp. same as) the sign of the edge. For each edge $a \rightarrow b$, $\mathsf{levels}_+(a \rightarrow b)$ and $\mathsf{levels}_-(a \rightarrow b)$ are respectively the set of levels of a where a is an effective activator and inhibitor of b.

DEFINITION.– *An* Interaction Graph *(IG) is a triple* (Γ, E_+, E_-) *where* Γ *is a finite number of* components*, and* E_+ *(resp.* E_-*)* $\subset \{a \xrightarrow{t} b \mid a, b \in \Gamma \wedge t \in [1; l_a]\}$ *is the set of positive (resp. negative)* regulations *between two nodes, labeled with a* threshold*. A regulation from* a *to* b *is uniquely referenced: if* $a \xrightarrow{t} b \in E_+$ *(resp.* E_-*),* $\nexists a \xrightarrow{t'} b \in E_+$ *(resp.* E_-*),* $t' \neq t$ *and* $\nexists a \xrightarrow{t'} b \in E_-$ *(resp.* E_+*),* $t' \in \mathbb{N}$*.*

DEFINITION.– *Let* (Γ, E_+, E_-) *be an IG and* $a, b \in \Gamma$ *two of its components:*

– *if* $a \xrightarrow{t} b \in E_+$*,* $\mathsf{levels}_+(a \to b) \triangleq [t; l_a]$ *and* $\mathsf{levels}_-(a \to b) \triangleq [0; t-1]$*;*

– *if* $a \xrightarrow{t} b \in E_-$*,* $\mathsf{levels}_+(a \to b) \triangleq [0; t-1]$ *and* $\mathsf{levels}_-(a \to b) \triangleq [t; l_a]$*;*

– *otherwise,* $\mathsf{levels}_+(a \to b) \triangleq \mathsf{levels}_-(a \to b) \triangleq \emptyset$*;*

where l_a *is the highest level for* a*.*

The *generalized dynamics* of the IG in Process Hitting can be encoded by following this simple rule: the level of a component can increase (resp. decrease) if and and only if at least one of its regulator activates (resp. inhibits) it. If the component has no regulator, we consider that the component can freely increase or decrease in order to be as general as possible. We denote by $\mathbf{PH}(\mathcal{G})$ the Process Hitting of the generalized dynamics of the IG $\mathcal{G}$, which is formally defined as follows.

DEFINITION.– *Given an interaction graph* $\mathcal{G} = (\Gamma, E_+, E_-)$*, its generalized dynamics in Process Hitting is given by* $\mathbf{PH}(\mathcal{G})$*, where*

$$\mathbf{PH}(\mathcal{G}) = (\Gamma, \prod_{a \in \Gamma} L_a, \bigcup_{(a,b) \in \Gamma^2} \mathcal{H}_a^b) \quad \text{, with } L_a = \{a_0, \ldots, a_{l_a}\}\text{, and}$$

– *if* $b \xrightarrow{t} a \in E_+ \cup E_-$*,*

$$\mathcal{H}_a^b \triangleq \{b_k \to a_i \Rsh a_j \mid b_k \in L_b, a_i, a_k \in L_a, |i-j| = 1 \wedge \\ i < j \Rightarrow k \in \mathsf{levels}_+(b \to a) \wedge i > j \Rightarrow k \in \mathsf{levels}_-(b \to a)\} \ ;$$

– *otherwise, if* $a = b$ *and* $\nexists c \in \Gamma, c \xrightarrow{t} a \in E_+ \cup E_-$*,*

$$\mathcal{H}_a^a \triangleq \{a_i \to a_i \Rsh a_{i-1} \mid 0 < i \leq l_a\} \cup \{a_i \to a_i \Rsh a_{i+1} \mid 0 \leq i < l_a\} \ ;$$

– *otherwise,* $\mathcal{H}_a^b = \emptyset$*.*

Part of this construction is exposed by figure 4.2 for the Boolean case. Note that the construction is linear with the number of edges and nodes in the IG.

Generalized Dynamics of the Incoherent Feed-forward Loop

In order to illustrate the modeling process of a biological network using the Process Hitting, we selected a common motif of regulatory and signalling networks, the

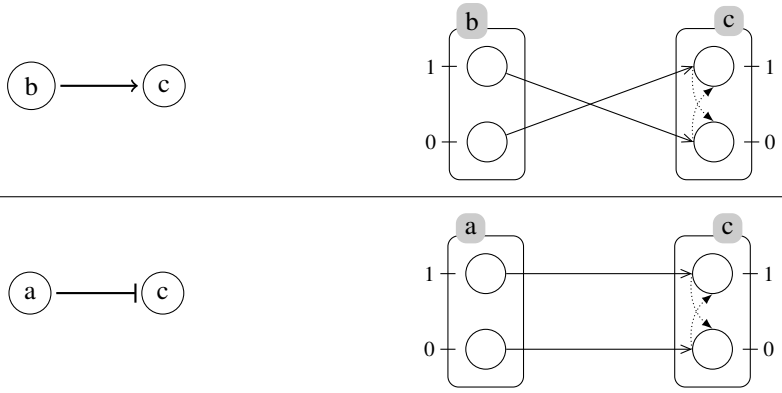


Figure 4.2: Rules for encoding the generalized dynamics of an IG in Process Hitting where each component is assumed Boolean and the threshold of the edges to be 1. Positive edges end with an arrow, while negative edges end with a bar.

Incoherent Feed-forward Loop [MAN 03] whose Interaction Graph (IG) is given in figure 4.3. The network has three components: a (assumed here to be constant) which activates b, and c which is both activated by b and inhibited by a. It is called incoherent as c is both inhibited (directly) and activated (through b) by a.

Thanks to the rules depicted above, the generalized Boolean dynamics of the IG in figure 4.3 can be automatically encoded in Process Hitting, resulting in the actions summarized in figure 4.4(left).

Figure 4.4(right) draws the possible transitions from the state $\langle a_1, b_0, c_0 \rangle$ of the generalized dynamics. First b is activated by a. Then, as there is no knowledge on the cooperation between a and b and c, there cannot be any consensus on the value of c. As a result, the value of c oscillates due to the successive independent activations by b and inhibitions by a.

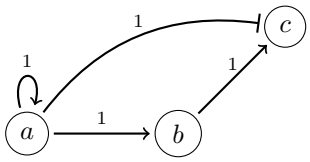


Figure 4.3: Interaction Graph of the Incoherent feed-forward loop

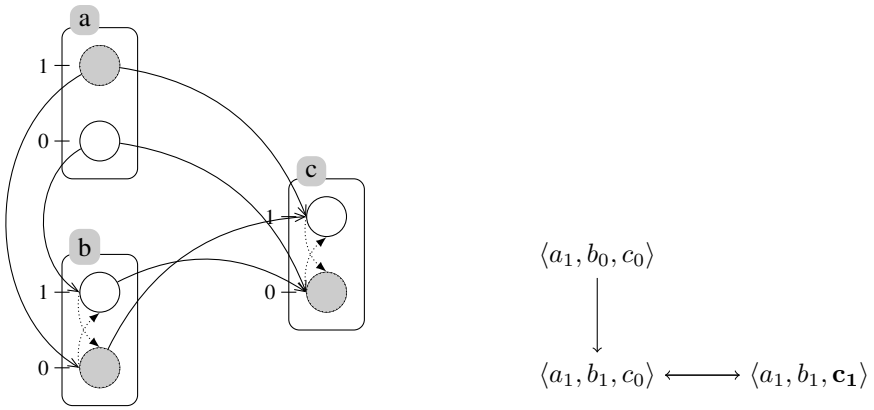


Figure 4.4: (left) Generalized Boolean dynamics of the Incoherent feed-forward loop in Process Hitting. (right) possible transitions from the state $\langle a_1, b_0, c_0 \rangle$.

4.2.4. *Refining Dynamics with Cooperativity*

In Process Hitting, the local state change of a sort is controlled by one and only one proces, the *hitter*. In such a setting, we may ask how to encode local state changes that should be controlled by the presence of at least two active processes. For instance, we may want c_0 to bounce to c_1 only if a_0 *and* b_1 are active. We call such a behavior a *cooperativity* between sorts a and b to act on c_0. Cooperativities are typically specified using logical functions [RIC 06b, BER 08]. In this section, we show how to interpret such cooperativities in Process Hitting.

Given a Process Hitting $(\Sigma, L, \mathcal{H})$, let us define $\sigma \subset \Sigma$ a subset of sorts that cooperate to make c_j bounce to c_k. The set of states of cooperating sorts is denoted by $S = \prod_{z \in \sigma} L_z$. The subset of those states where the cooperativity is effective is denoted by $\top \subset S$.

In Process Hitting, a cooperativity can be encoded with a new sort which will act as the logical function. Such a *cooperative sort* contains one process per state of the cooperating sorts. Let us call υ such a sort. The processes of sort υ are $L_\upsilon = \{\upsilon_\varsigma, \forall \varsigma \in S\}$. The intuition is that the active process of sort υ will reflect the state of all the cooperating sorts. To do so, each cooperating process z_i, $z \in \sigma$, hits all processes υ_ς where $\varsigma[z] \neq z_i$ to make it bounce to $\upsilon_{\varsigma'}$ where $\varsigma'[z] = z_i$ and $\varsigma'[a] = \varsigma[a], \forall a \in \Sigma, a \neq z$. We denote by $\mathcal{H}_\sigma$ this set of actions (equation (4.1)).

Thus, the process c_j is no longer independently hit by cooperating processes of sorts in σ ($\mathcal{H}_{rm}$, equation (4.2)), but by processes of the cooperative sort υ which

represent the states in $\top$ ($\mathcal{H}_{coop}$, equation (4.3)).

$$\mathcal{H}_\sigma = \{z_i \to \upsilon_\varsigma \Rsh \upsilon_{\varsigma'} \mid z \in \sigma \wedge z_i \in L_z \wedge \varsigma \in S \wedge \varsigma[z] \neq z_i \wedge \varsigma'[z] = z_i \wedge (\varsigma'[a] = \varsigma[a], \forall a \in \Sigma, a \neq z)\} \tag{4.1}$$

$$\mathcal{H}_{rm} = \{z_i \to c_j \Rsh c_k \in \mathcal{H} \mid z \in \sigma\} \tag{4.2}$$

$$\mathcal{H}_{coop} = \{\upsilon_\varsigma \to c_j \Rsh c_k \mid \varsigma \in \top\} \ . \tag{4.3}$$

The new Process Hitting refined by the cooperation is defined by equation (4.4).

$$(\Sigma \cup \{\upsilon\}, L \times L_\upsilon, (\mathcal{H} \setminus \mathcal{H}_{rm}) \cup \mathcal{H}_\sigma \cup \mathcal{H}_{coop}) \ . \tag{4.4}$$

Such a construction can be easily extended to the interpretation of fully defined logical functions of the form $f : L_{a^1} \times \cdots \times L_{a^n} \mapsto L_c$ (such a function maps a state of the cooperating sorts $\sigma = \{a^1, \ldots, a^n\}$ to a bounce process c_k). The sets $\mathcal{H}_{rm}$ (equation (4.2)) and $\mathcal{H}_{coop}$ (equation (4.3)) can be respectively replaced with $\mathcal{H}^f_{rm}$ (equation (4.5)) and $\mathcal{H}^f_{coop}$ (equation (4.6)).

$$\mathcal{H}^f_{rm} = \{z_i \to c_j \Rsh c_k \in \mathcal{H} \mid z \in \sigma, z_i \in L_z, c_j, c_k \in L_c\} \tag{4.5}$$

$$\mathcal{H}^f_{coop} = \{\upsilon_\varsigma \to c_j \Rsh c_k \mid c_j \in L_c \wedge f(\varsigma) = c_k\} \ . \tag{4.6}$$

The complexity of these encodings is exponential with the number of cooperating sorts: the cooperative sort υ gathers $|L_{a^1}| \cdot \ldots \cdot |L_{a^n}|$ processes. Nevertheless, the size of υ can be drastically reduced by splitting the encoded function into several cooperative sorts. Such a step is equivalent in putting parentheses within the logical function and having one sort per pair of parenthesis. This is illustrated in figure 4.5.

It is very important to note that the proposed construction introduces a temporal shift in the application of the cooperativity. It results in potential spurious transitions when the state of the cooperative sort is incoherent with the actual state of the cooperating sorts. This behavior is somehow similar to a biological complex: the complex $A - B$ can be present while individuals A and B are absent. In addition, such spurious behaviors can be eliminated by adding the notion of priority to Process Hitting to make the update of a cooperative sort always happen before any other action. This is discussed in the last section of the chapter.

Overall, applied to a Process Hitting of the generalized dynamics of an IG, refinement using cooperativity allows the encoding of additional knowledge into the logical functions between regulators. The resulting model dynamics is smaller (in term of transitions) than the generalized one. We remark that the presented constructions allow us to automatically encode a Boolean or multi-valued network into Process Hitting.

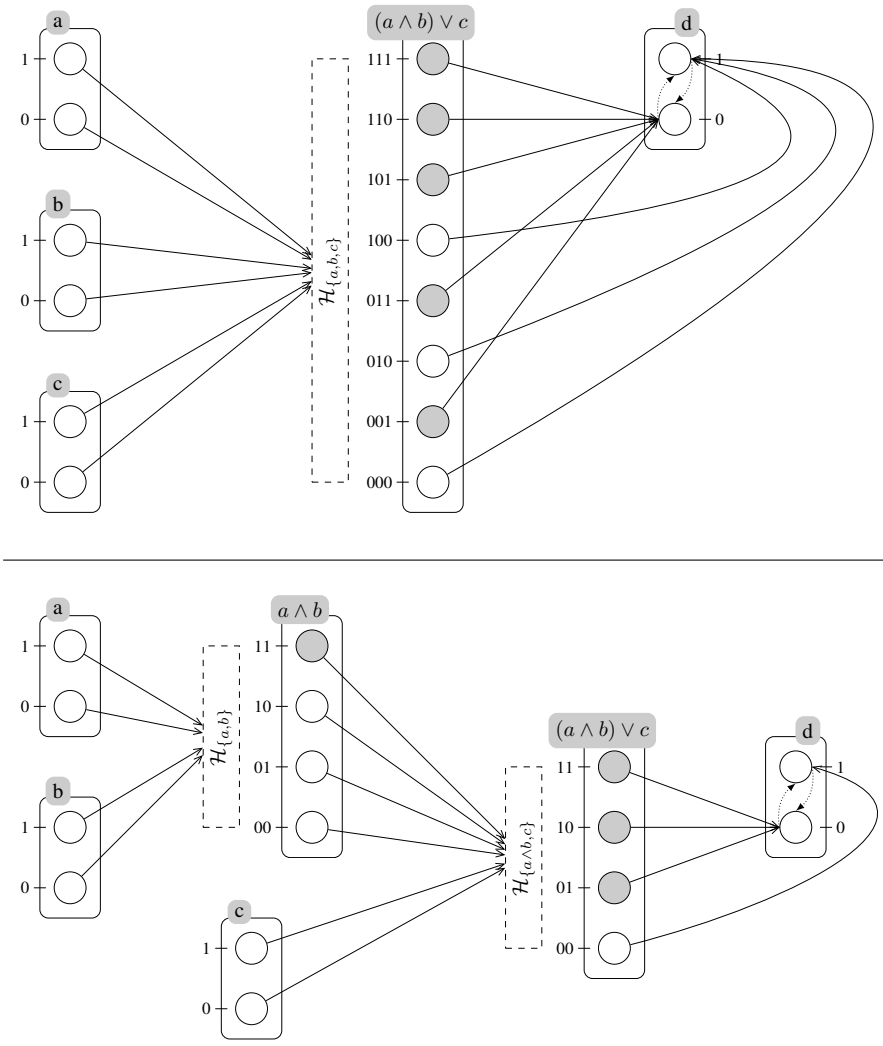


Figure 4.5: Encoding of the Boolean function $f^d(a, b, c) = (a_1 \wedge b_1) \vee c_1$: (top) a single cooperative sort is used with 8 processes (bottom) splitting of the cooperativity: two sorts are used with 4 processes in each. Grayed processes are those satisfying the encoded part of the Boolean expression. For the sake of clarity, actions on cooperative sorts ($\mathcal{H}_\sigma$, equation (4.1)) are not drawn, but represented by a dashed rectangle.

Refined Dynamics of the Incoherent Feed-forward Loop

Returning to our example of the incoherent feed-forward loop, we may know that a and b cooperate for c as such: c is active if and only if a is absent and b is present. Therefore, we refine our generalized dynamics using a cooperative sort that encodes the Boolean function $\neg a \wedge b$, as shown in figure 4.6.

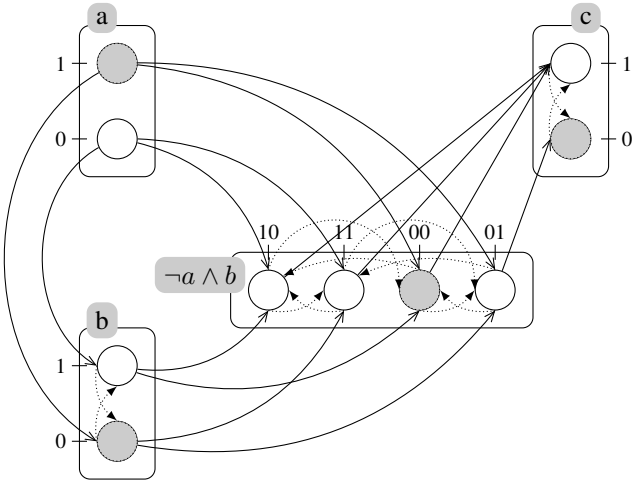


Figure 4.6: Refined Process Hitting encoding the dynamics of the Incoherent Feed-forward Loop where the activation of c needs both a inactive and b active. The process 01 of the sort $\neg a \wedge b$ mirrors the (only) state where c_1 should be present. Grayed processes indicate the state where figure 4.7 starts.

In the generalized dynamics, due to the undefined cooperation between a and b when both are present, c oscillated. In our refined dynamics, it is no longer the case: c converges to c_0 as a is active. Part of the transition graph is shown in Figure 4.7 when starting in the state $\langle a_1, b_0, ab_{00}, c_0\rangle$. It ends on the fixed point $\langle a_1, b_1, ab_{11}, c_0\rangle$. The initial process of the cooperative sort (named ab) has been intentionally choosen incoherent with the state of a and b.

4.2.5. *Relationship with Boolean/multi-valued Networks*

The modeling scheme we propose in this chapter is the following: starting from an Interaction Graph (IG) and under the extreme values assumption (section §4.2.3), we can model in Process Hitting a transition system that includes the dynamics (transitions) of any Boolean network sharing the same IG. Then, with additional knowledge on the cooperations between the regulators, we can modify the Process Hitting to refine the dynamics, i.e., remove transitions that are impossible with respect to the

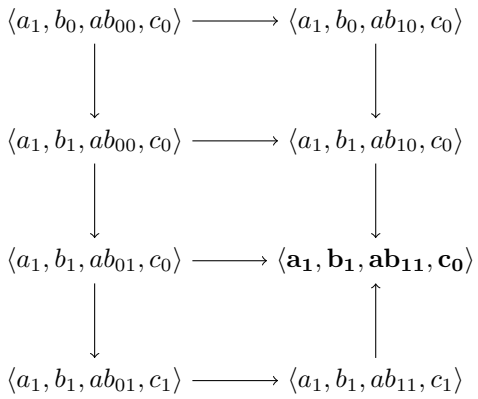


Figure 4.7: Transition graph of the Process Hitting in figure 4.6 from the state represented by grayed processes.

specified cooperations (logical functions). The effect of this refining procedure on dynamics is illustrated by figure 4.8.

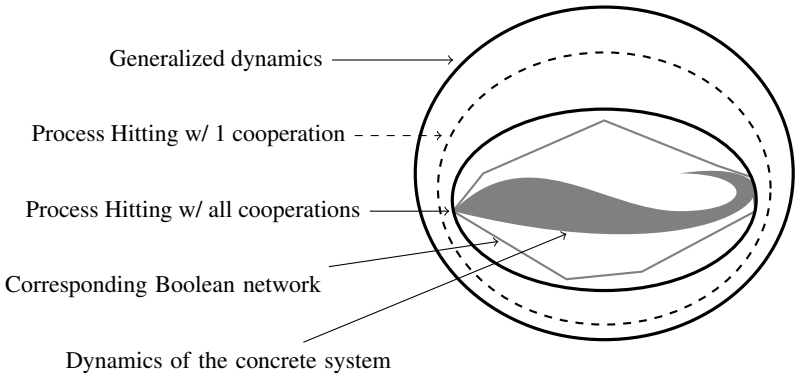


Figure 4.8: Illustration of the inclusion of dynamics (transitions) between several refinement steps of Process Hitting models of generalized dynamics.

This "modeling by over-approximation" approach has multiple advantages: one can obtain a (single) dynamical model without a complete knowledge on the cooperations (discrete parameters); if a behavior is impossible in the over-approximation, it is impossible for any concrete model having its dynamics included in the Process Hitting model. This allows us to reason on multiple dynamical models at once.

We first detail the link between the generalized dynamics of an IG and the compatible Boolean or multi-valued networks. Then, we discuss the link with Process Hitting models refined using cooperativity.

Generalized Dynamics

We state the theorem on the generalized dynamics of an IG $\mathcal{G}$ in Process Hitting (section §4.2.3) which includes the dynamics of any asynchronous multi-valued network F whose IG $\mathbf{IG}(F) \subseteq \mathcal{G}$ (for a possible definition of $\mathbf{IG}(F)$ see [RIC 10]). The main argument is the following: if a regulated component of F increases (resp. decreases), then from the extreme values assumption, there exists a regulator that effectively activates (resp. inhibits) the component: in such a case, the generalized dynamics necessarily include the same transition. In the proof, $\mathcal{H}_a^b$ and $\mathcal{H}_a^a$ refer to the sets defined in section §4.2.3.

THEOREM.– *Given any Interaction Graph $\mathcal{G} = (\Gamma, E_+, E_-)$, if $\mathbf{PH}(\mathcal{G}) = (\Gamma, L, \mathcal{H})$, for any asynchronous multi-valued network F with extreme values assumption such that $\mathbf{IG}(F) \subseteq \mathcal{G}$, for any state $s, s' \in L$, $s \neq s'$, $F(s) = s' \Rightarrow \exists h \in \mathcal{H} : s \cdot h = s'$.*

PROOF.– *If $F(s) = s'$ with $s \neq s'$, there exists a unique $a \in \Gamma$ such that $s'[a] - s[a] = \pm 1$ and $\forall b \in \Gamma, b \neq a, s[b] = s'[b]$. If $\exists c \in \Gamma : c \to a \in \mathcal{G}$, from extreme values assumption, if $s'[a] - s[a] = 1$ (resp. -1), there exists $b \in \Gamma$ such that $s[b] \in \mathsf{levels}_+(b \to a)$ (resp. $\mathsf{levels}_-(b \to a)$), hence $s[a] \to s[a] \Rsh s'[a] \in \mathcal{H}_a^b$. Otherwise, $\nexists c \in \Gamma : c \to a \in \mathcal{G}$ (a has no regulator), hence $s[a] \to s[a] \Rsh s'[a] \in \mathcal{H}_a^a$.*

Refined Dynamics

Refining a Process Hitting model of the generalized dynamics of an IG may imply the deletion of some action and the definition of cooperativities. Intuitively, these refinement steps remove transitions and put constraints on the compatible logical functions having dynamics included in the refined Process Hitting. In [FOL 12], we give a complete and automatic method to derive all Boolean/multi-valued network whose dynamics is included in a Process Hitting. An implementation of the inference algorithm is distributed with PINT (section §4.5.1).

Discussion

The Process Hitting can be defined as a restriction of general asynchronous automata networks, where the arity of synchronizations between automata is at most two and each synchronization changes the state of only one automata. This restriction prevents the construction of bisimular Boolean networks or Petri nets, as those formalisms allow the specification of synchronization with arbitrary arities. One extension of the Process Hitting attaches to an action a priority level: an action with priority p can be applied only if no action of priority lower than p is applicable. With prioritized actions, it is possible to weakly bisimulate any asynchronous Boolean/multi-valued networks or safe Petri nets with Process Hitting [FOL 13, PAU 12a].

On the other hand, as it is depicted in the next section, the specific restrictions imposed by the Process Hitting allow the derivation of static analysis for dynamical properties which are highly scalable. Moreover, the spurious transitions due to cooperativity in classical Process Hitting (without priority) do not prevent any reasoning on dynamics by over-approximation: if a behavior (sequence of transitions) is impossible in the Process Hitting, it is impossible in any Boolean/multi-valued networks encoded by the Process Hitting. The Process Hitting approach makes the formal analysis of dynamics of very large interaction networks tractable and allows us to reason on multiple Boolean/multi-valued networks at once.

4.3. Static Analysis of Discrete Dynamics

4.3.1. *Motivation*

Biological regulatory networks can be studied using their state graph, which contains the complete dynamics of the system. However, the state graph grows exponentially with each added component, quickly making the problem become intractable. For this reason, alternative approaches have emerged which rely only on the interaction graph (IG). For example, starting from conjectures formulated by René Thomas, research has shown that multi-stationarity systems require the presence of at least one positive circuit and that oscillatory behavior requires the presence of a negative circuit. Although these conjectures were proven [RIC 10] and provide some useful insight to the inner workings of a system, they do not provide enough detail of the dynamic properties to conduct a full analysis. The structure of Process Hitting, however, lends it to other analysis methods which can be performed without computing the state graph, such as static analysis by abstract interpretation [COU 77]. In this method, a systems behavior is described via several abstractions giving over- and under-approximations of a particular property without running any simulation. In this section we will focus on the potential of static analysis as it applies to Process Hitting systems. In section §4.3.2, we will demonstrate the computation of fixed points, also known as steady states. Section §4.3.3 introduces an efficient reachability analysis based on abstract structures called graphs of local causality which are computed statically. Finally, using these very graphs of local causality, we are also able to compute cut sets, that is, the sets of processes necessary to demonstrate a given behavior, as shown in section §4.3.4.

4.3.2. *Fixed points*

The expression of fixed points in a given regulatory network is potentially one of the most crucial behaviors to include in a model and is often the aspect of greatest interest for a biologist. For this reason, a great amount of research has gone toward the topic of fixed point analysis. We have already mentioned that a positive circuit is

a proven necessary condition for multiple fixed points, as shown in [RIC 10], but this does not speak to the exact number of fixed points present in the system. In the field of Boolean networks, analysis of the interaction graph can give an upper bound of the possible number of fixed points [ARA 08] and the topological fixed points independent of logical functions [PAU 10], but these still do not give a complete enumeration. In [NAL 07], the authors propose an efficient method for enumerating all the fixed point by relying on the encoding of logical functions using decision diagrams.

In this section, we show that the fixed points of a Process Hitting model can be completely derived from its structure. More precisely, we show that this enumeration is equivalent to listing the n-cliques of a n-partite graph directly derived from the Process Hitting. This method provides an efficient alternative to listing all the fixed point of an interaction network.

The principal idea of this method is relatively simple: if two processes are linked by an action (that is, one hits the other), they cannot belong to the same fixed point. By constructing a complementary representation of a Process Hitting called the *hitless graph*, we can exploit this idea to find all fixed points of the system. In a hitless graph, an edge is drawn between two processes if there exists no action between them. Logically, two processes of the same sort are never linked and no process which possesses a self-hit is included in the graph. The hitless graph of a Process Hitting is n-partite, where each partition corresponds to exactly one sort of the original Process Hitting and n is the number of sorts with at least one process not hitting itself; therefore: $n \leq |\Sigma|$.

DEFINITION.– *Given a Process Hitting $\mathcal{PH} = (\Sigma, L, \mathcal{H})$, its* hitless graph *is the undirected graph (V, E) defined by:*

$$V \triangleq \bigcup_{a \in \Sigma} \{a_i \in L_a \mid \forall a_k \in L_a, \nexists a_i \to a_i \Rsh a_k \in \mathcal{H}\}$$

$$E \triangleq \{\{a_i, b_j\} \in V \times V \mid a \neq b, \forall b_k \in L_b, \nexists a_i \to b_j \Rsh b_k \in \mathcal{H} \wedge \forall a_l \in L_a, \nexists b_j \to a_i \Rsh a_l \in \mathcal{H}\}$$

DEFINITION.– *If $n \in \mathbb{N}$, a graph (V, E) is* n-partite *if and only if:*

– $V = \bigcup_{k \in [\![1;n]\!]} V_k$,
– $\forall k, k' \in [\![1; n]\!], k \neq k' \Rightarrow V_k \cap V_{k'} = \emptyset$,
– $\forall \{a_i, b_j\} \in E, \exists k, k' \in [\![1; n]\!], k \neq k' \wedge a_i \in V_k \wedge b_j \in V_{k'}$.

A *clique* is a collection of connected processes in the hitless graph, with a n-clique being a clique with n elements. If $n = |\Sigma|$, a n-clique of the hitless graph corresponds

to a fixed point of the Process Hitting since no playable action exists amongst the processes to move the system to a new state. The search for n-cliques in a n-partite graph can be efficiently encoded with constraint programming on integers, which avoids an explicit computation of the hitless graph and makes the search for fixed points in Proces Hitting efficient in practice.

DEFINITION.– *Given a graph (V, E), a subset of the vertices $C \subseteq V$ is a $|C|$-clique if and only if $\forall (a_i, b_j) \in C \times C, \{a_i, b_j\} \in E$.*

THEOREM.– *The fixed points of a Process Hitting $\mathcal{PH} = (\Sigma, L, \mathcal{H})$ are exactly the $|\Sigma|$-cliques of its hitless graph.*

The search of fixed points by using n-cliques has been implemented in Pint using SAT techniques. It successfully computes on models with hundreds of components in less than a second. For the example of figure 4.1, this search is illustrated by the hitless graph of figure 4.9 and the following fixed points are found:

$$\langle a_0, b_2, c_1, d_2\rangle \qquad \langle a_1, b_0, c_0, d_2\rangle \qquad \langle a_1, b_2, c_0, d_1\rangle$$
$$\langle a_1, b_2, c_0, d_2\rangle \qquad \langle a_1, b_2, c_1, d_2\rangle$$

4.3.3. *Abstract Interpretation using Graphs of Local Causality*

An efficient way of checking a *reachability property*, described below, was developed in [PAU 12b]: with this methodology, the analysis of models with hundreds or thousands of components becomes tractable. In this section, we present a general overview and the main results of this method, along with an insight of the five developed approximations. For further detail, we would like to refer the reader to the original publication which contains all properties and proofs.

An *objective*, defined below, is a pair of processes which describes the eventual activation of a process from another process. For instance, $d_0 \Rsh^* d_2$ depicts the reachability of process d_2 from a state where d_0 is active. We call *reachability property* a property of the form $\mathcal{P} = \mathsf{reach}(\varsigma, \omega)$ where ς is an initial *context* (a generalization of a state) and $\omega = P_1 :: P_2 :: \ldots :: P_{|\omega|}$ is an *objective sequence*. Objectives are useful in describing a reachability property, since the reachability of a process a_i from a state s is denoted by the objective: $s[a] \Rsh^* a_i$. For example, the property $\mathcal{P} = \mathsf{reach}(\langle a_0, b_0, b_1, c_0, d_0\rangle, b_1 \Rsh^* b_0 :: d_0 \Rsh^* d_2)$ on the Process Hitting of figure 4.1 means that starting from state $\langle a_0, b_0, c_0, d_0\rangle$ or $\langle a_0, b_1, c_0, d_0\rangle$, we can play several actions in order to reach a state s_1 s.t. $b_0 \in s_1$; then, starting from this state, s_1, we can play some other actions to reach a new state s_2 such that $d_2 \in s_2$.

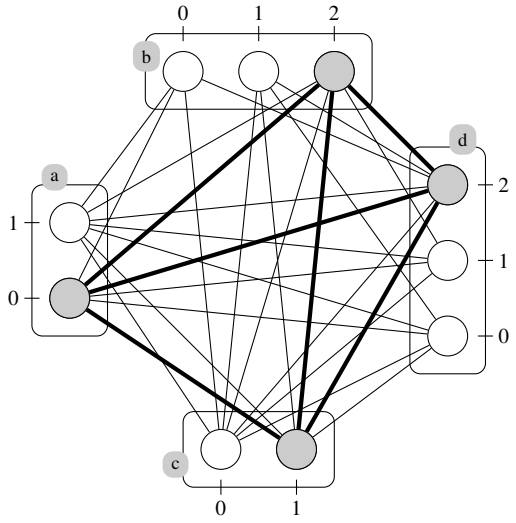


Figure 4.9: The hitless graph of the Process Hitting example of figure 4.1. Edges link two processes of different sorts if they are not involved in the same action. Given that it consists of a 4-clique, as stated by the thick edges, the highlighted state $\langle a_0, b_2, c_1, d_2 \rangle$ is an example of stable state.

DEFINITION.– *A* context ς *associates with each sort in* Σ *a non-empty subset of its processes:* $\forall a \in \Sigma, \varsigma[a] \subseteq L_a \wedge \varsigma[a] \neq \emptyset$. *For a given context* ς*, we write:* $a_i \in \varsigma \triangleq a_i \in \varsigma[a]$*, and for any state* $s \in L$*:* $s \subseteq \varsigma \triangleq \forall a \in \Sigma, s[a] \in \varsigma[a]$.

DEFINITION.– *An* objective *is a pair of processes denoted* $a_i \restriction^* a_j$ *that depict the bounce from a process* a_i *to another process* a_j *of the same sort after some actions. We note:* $\mathsf{target}(a_i \restriction^* a_j) \triangleq a_i$, $\mathsf{bounce}(a_i \restriction^* a_j) \triangleq a_j$, *and the set of all objectives:* $\mathbf{Obj} \triangleq \{a_i \restriction^* a_j \mid a \in \Sigma \wedge (a_i, a_j) \in L_a \times L_a\}$. *An* objective sequence $P_1 :: P_2 :: \ldots$ *is a sequence of objectives verifying that the target of each objective is equal to the bounce of the previous objective on the same sort in the sequence, if it exists.*

Checking for the exact reachability property, $\mathcal{P}$, would be too computationally difficult as the required computing time grows exponentially with the number of sorts. However, an under-approximation, $\mathcal{Q}$, and an over-approximation, $\mathcal{R}$, were developed, that can be checked in time polynomial with the number of sorts, and exponential with the number of hits on a single sort. Rather than using the Process Hitting model to check $\mathcal{P}$, we use $\mathcal{Q}$ in order to ensure that reachability is possible and $\mathcal{R}$ to refute the given reachability property. These approximations are based primarily on two abstractions of scenarios called *bounce sequences* (**BS**), and *abstract bounce*

sequences ($\mathbf{BS}^\wedge$). A bounce sequence consists of a sequence of actions hitting the same sort in order to solve an objective. For example, $b_0 \to d_0 \Rsh d_1 :: b_1 \to d_1 \Rsh d_2$ is a bounce sequence of the objective $d_0 \Rsh^* d_2$ and abstracts any scenario containing these actions in the same order with possible intermediate actions. A bounce sequence leads to the definition of an *abstract bounce sequence* which consists of the set of hitters, thus focusing on the required processes to play a bounce sequence while abstracting the order of the involved actions. The abstract bounce sequence for the previous example would be $\{b_0, b_1\}$. We note that there are two particular cases of abstract bounce sequences for an objective P: $\mathbf{BS}^\wedge(P) = \emptyset$ if P cannot be solved, and $\mathbf{BS}^\wedge(P) = \{\emptyset\}$ if P can be solved without involving any other sort. Provided the particular structure of the PH, these abstractions can be statically computed on the Process Hitting model.

DEFINITION.– *A* bounce sequence ζ *is a sequence of actions such that* $\forall n \in \mathbb{I}^\zeta, n < |\zeta|, \mathsf{bounce}(\zeta_n) = \mathsf{target}(\zeta_{n+1})$. *We will write* $\mathbf{BS}$ *to denote the set of bounce sequences, and* $\mathbf{BS}(P)$ *to denote the set of bounce sequences resolving the objective* P*:* $\mathbf{BS}(a_i \Rsh^* a_j) \triangleq \{\zeta \in \mathbf{BS} \mid \mathsf{target}(\zeta_1) = a_i \wedge \mathsf{bounce}(\zeta_{|\zeta|}) = a_j \wedge \forall m, n \in \mathbb{I}^\zeta, n > m, \mathsf{bounce}(\zeta_n) \neq target(\zeta_m)\}$.

DEFINITION.– *The set of* abstract bounce sequences *of an objective* $P \in \mathbf{Obj}$ *is the set:* $\mathbf{BS}^\wedge(P) \triangleq \{\zeta^\wedge \mid \zeta \in \mathbf{BS}(P) \wedge \nexists \zeta' \in \mathbf{BS}(P), \zeta'^\wedge \subsetneq \zeta^\wedge\}$ *where* $\zeta^\wedge \triangleq \{\mathsf{hitter}(\zeta_n) \mid n \in \mathbb{I}^\zeta \wedge \mathsf{sort}(\mathsf{hitter}(\zeta_n)) \neq \mathsf{sort}(P)\}$.

Computing the approximations $\mathcal{Q}$ and $\mathcal{R}$ depends on the construction of *graphs of local causality* (GLC). These directed graphs use the previous abstractions to give conclusions about the successive reachability of an objective sequence from an initial state. The nodes of a GLC fall into three categories:

– an objective node (in $\mathbf{Obj}$) depicts an objective to be met;

– a solution node (in $\mathbf{Sol} \triangleq \wp(\mathbf{Proc})$) stands for one of the abstract bounce sequences which meets the requirements of an objective;

– a process node (in $\mathbf{Proc}$) is a process required by a solution in the graph.

Edges in a GLC link the different nodes to their requirements. The GLC $\mathcal{A}^\omega_\varsigma$, used to compute the over-approximation $\mathcal{R}$ is recursively defined as follows: an objective P is linked to all of its related solutions in $\mathbf{BS}^\wedge(P)$ (equation (4.8)), a solution to all the processes it contains (equation (4.9)), and a process a_i to all the objectives of the form $a_j \Rsh^* a_i$ where a_j is in the initial context (equation (4.10)). Finally, equation (4.7) includes the objectives of ω in the GLC. The GLC $\lceil \mathcal{B}^\omega_\varsigma \rceil$ used to compute the under-approximation $\mathcal{Q}$ is built in a similar fashion, except for two key differences. First, an objective $a_j \Rsh^* a_i$ may link to another objective $a_k \Rsh^* a_i$ if the process a_k is required to solve the former (equation (4.12)). Secondly, the structure is saturated: for any process a_i mentioned, an objective starting from a_i is added (equation (4.11)).

EXAMPLE.– Figure 4.10 gives an example of the two GLCs computed on the Process Hitting model of Figure 4.1 for two different reachability properties.

DEFINITION.– *The graph of local causality* $\mathcal{A}^\omega_\varsigma \triangleq (V^\omega_\varsigma, E^\omega_\varsigma)$ *is the smallest graph with* $V^\omega_\varsigma \subseteq \mathbf{Proc} \cup \mathbf{Obj} \cup \mathbf{Sol}$ *and* $E^\omega_\varsigma \subseteq V^\omega_\varsigma \times V^\omega_\varsigma$ *so that:*

$$\omega \subseteq V^\omega_\varsigma \tag{4.7}$$

$$P \in V^\omega_\varsigma \cap \mathbf{Obj} \wedge ps \in \mathbf{BS}(P) \Rightarrow (P, ps) \in E^\omega_\varsigma \tag{4.8}$$

$$ps \in V^\omega_\varsigma \cap \mathbf{Sol} \wedge a_i \in ps \Rightarrow (ps, a_i) \in E^\omega_\varsigma \tag{4.9}$$

$$a_i \in V^\omega_\varsigma \cap \mathbf{Proc} \wedge a_j \in \varsigma \Rightarrow (a_i, a_j \Rsh^* a_i) \in E^\omega_\varsigma \tag{4.10}$$

DEFINITION.– *The graph of local causality* $\lceil \mathcal{B}^\omega_\varsigma \rceil$ *is defined as:*

$$\lceil \mathcal{B}^\omega_\varsigma \rceil \triangleq \mathbf{lfp}\{\mathcal{B}^\omega_\varsigma\} \left(\mathcal{B}^\omega_\varsigma \mapsto \mathcal{B}^\omega_{\varsigma \Cap \mathsf{procs}(\mathcal{B}^\omega_\varsigma)} \right) \tag{4.11}$$

with $\mathcal{B}^\omega_\varsigma = (V^\omega_\varsigma, E^\omega_\varsigma)$ *the smallest graph so that* $V^\omega_\varsigma \subseteq \mathbf{Proc} \cup \mathbf{Obj} \cup \mathbf{Sol}$ *and* $E^\omega_\varsigma \subseteq V^\omega_\varsigma \times V^\omega_\varsigma$*, that respects equations* (4.7)*,* (4.8)*,* (4.9) *and* (4.10)*, and:*

$$P \in V^\omega_\varsigma \cap \mathbf{Obj} \wedge q \in \mathsf{maxCont}_\varsigma(\mathsf{sort}(P), P) \Rightarrow (P, q \Rsh^* \mathsf{bounce}(P)) \in E^\omega_\varsigma \tag{4.12}$$

where for all GLC (V, E)*:*

$$\mathsf{procs}(V, E) \triangleq (V \cap \mathbf{Proc}) \cup \{\mathsf{target}(P), \mathsf{bounce}(P) \mid P \in V \cap \mathbf{Obj}\}$$

and for all context ς *and set of processes* ps*:*

$$\forall a \in \Sigma, (\varsigma \Cap ps)[a] \triangleq \begin{cases} \{p \in ps \mid \mathsf{sort}(p) = a\} & \textit{if } ps \cap L_a \neq \emptyset \\ \varsigma[a] & \textit{otherwise} \end{cases}$$

and for all context ς*, process* a *and objective* P*:*

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

The approximations of $\mathcal{P} = \mathsf{reach}(\varsigma, \omega)$ take the form of several properties on these GLCs: the over-approximation $\mathcal{R} = \mathcal{R}_1 \wedge \mathcal{R}_2 \wedge \mathcal{R}_3$ is a conjunction of three properties on $\mathcal{A}^\omega_\varsigma$, while the under-approximation $\mathcal{Q} = \mathcal{Q}_1 \vee \mathcal{Q}_2$ is a disjunction of two properties on $\lceil \mathcal{B}^\omega_\varsigma \rceil$. In practice, this means that only one of the sub-properties of $\mathcal{R}$ must be invalidated in order to invalidate $\mathcal{P}$, and if at least one of the sub-properties of $\mathcal{Q}$ is true, then the reachability $\mathcal{P}$ is also true. Properties $\mathcal{R}_1$ and $\mathcal{Q}_1$ are unordered approximations since the order of the requested reachabilities in $\mathcal{P}$ is ignored:

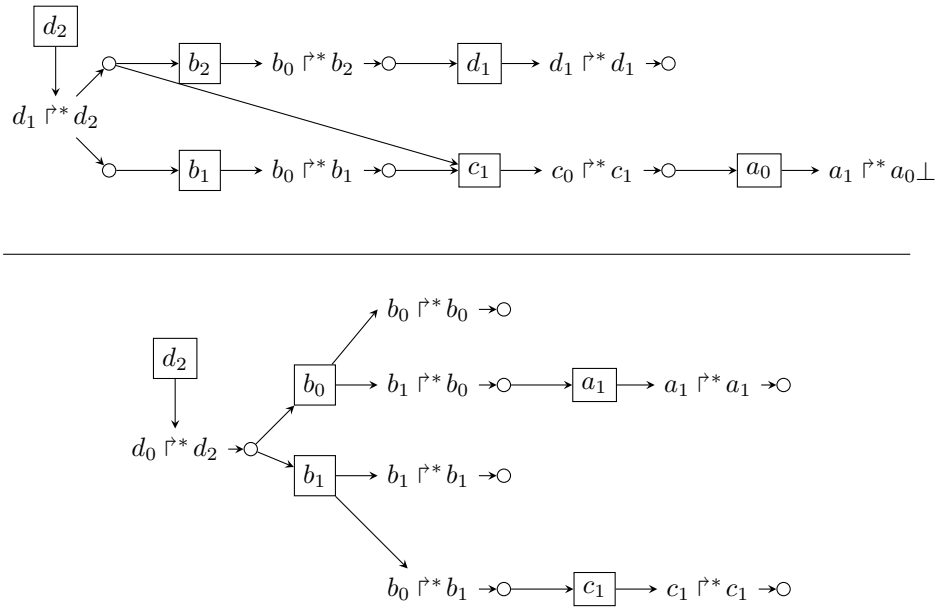


Figure 4.10: (top) GLC $\mathcal{A}^{\omega}_{\varsigma}$ computed on the Process Hitting model of figure 4.1 for $\mathsf{reach}(\langle a_1, b_0, c_0, d_1\rangle, d_1 \Rsh^* d_2)$. (bottom) GLC $\lceil \mathcal{B}^{\omega}_{\varsigma} \rceil$ computed on the same Process Hitting model for $\mathsf{reach}(\langle a_1, b_1, c_1, d_0\rangle, d_0 \Rsh^* d_2)$. Square nodes are processes, round nodes are solutions and nodes with no border are objectives. The $\bot$ symbol denotes an objective with no successor solution.

– $\mathcal{R}_1$ states that, starting from each objective in ω, there exists a path of $\mathcal{A}^{\omega}_{\varsigma}$ with no loop such that, from an objective node, at least one linked solution node is traversed and, from any other node, all linked nodes are traversed.

– $\mathcal{Q}_1$ states that $\lceil \mathcal{B}^{\omega}_{\varsigma} \rceil$ has no cycle and all its leaves are solution nodes.

Based on these unordered approximations, one can recursively build the ordered approximation $\mathcal{R}_2$ (resp. $\mathcal{Q}_2$) by using the property $\mathcal{R}_1$ (resp. $\mathcal{Q}_1$) to check $\mathsf{reach}(\varsigma, (\omega_1))$, and then, by iterating, check $\mathsf{reach}(\varsigma', (\omega_2, \ldots, \omega_{|\omega|}))$, where ς' is an update of ς given the results of the first step which is not detailed here. The last over-approximation $\mathcal{R}_3$ exploits more a precise ordering of constraints between occurrences of processes that can be statically extracted from the model but which we will not detail here.

THEOREM.– $\mathcal{Q}_1 \vee \mathcal{Q}_2 \Rightarrow \mathsf{reach}(\varsigma, \omega) \Rightarrow \mathcal{R}_1 \wedge \mathcal{R}_2 \wedge \mathcal{R}_3$.

EXAMPLE.– The over-approximation property $\mathcal{R}_1$ is false on the structure of figure 4.10(top): indeed, the two possible paths in the graph visit the objective node

$a_1 \Rsh^* a_0$ which has no successor. However, the under-approximation property $\mathcal{Q}_1$ is true on the structure of figure 4.10(bottom), as all the leaves of this acyclic graph are solutions. We can therefore conclude that property $\mathsf{reach}(\langle a_1, b_0, c_0, d_1\rangle, d_1 \Rsh^* d_2)$ is false, and property $\mathsf{reach}(\langle a_1, b_1, c_1, d_0\rangle, d_0 \Rsh^* d_2)$ is true.

Overall, this method is considered efficient since each step can be performed in polynomial time. Building both GLCs is polynomial with respect to the number of sorts and exponential with respect to the number of actions hitting a single sort. However, assuming that the number of processes inside each sort is very small (typically 4 or below), this step is typically very efficient. Approximation properties can be checked in polynomial time with respect to the size of $\mathcal{A}^\omega_\varsigma$ and $\lceil \mathcal{B}^\omega_\varsigma \rceil$. In the case of the under-approximation, $\mathcal{Q}_1$, a more conclusive checking can be achieved by removing a subset of solutions at the cost of being exponential with respect to the number of solutions for a single objective. These new analyses can, therefore, also be considered efficient, although they come at a risk of a non conclusive response. Most cases, however, do reach conclusions, as shown in section §4.5.

4.3.4. *Cut sets*

More information about the system can be derived from the GLCs than reachability properties as presented in the previous section. It was shown in [PAU 13] that analyzing the structure $\mathcal{A}^\omega_\varsigma$ also gives an approximation of the requirements which ensure a reachability property. In particular, it is possible to derive *cut sets*, that is, sets of necessary processes which, should they be *disabled*, would prevent the considered reachability. To disable a process p is to remove any action involving p. Such cut sets can be especially of interest for therapeutic applications by preventing the expression of an uncontrollable gene by knocking in/out other observable genes, that is, prevent the activation of a particular process by disabling other, controllable, processes. Cut sets are also very useful to refute a model: if a cut set computed from the model does not prevent the reachability in the concrete (modeled) system, then it is a proof that there exists concrete behaviors that are not reproducible by the model.

Given a set of processes, $\mathcal{O}bs \subseteq \mathbf{Proc}$, we introduce an algorithm computing on $\mathcal{A}^\omega_\varsigma$ the set $\mathbb{V}(a_i)$ of minimal cut N-sets of processes in $\mathcal{O}bs$ that are necessary for the independent reachability of each process $a_i \in \mathbf{Proc} \cap V^\omega_\varsigma$. Assuming a first *valuation* $\mathbb{V}$ associating each node with its cut N-sets, this valuation can be refined on node n by using $\mathsf{update}(\mathbb{V}, n)$ defined below. In the following, if A is a set and $B^1, \ldots, B^n \in \wp(\wp(A))$ are sets of sets, we note the *sets of sets product*: $\widetilde{\prod}_{i\in[1;n]} B^i \triangleq B^1 \widetilde{\times} \cdots \widetilde{\times} B^n \in \wp(\wp(A))$ where:

$$\{e_1, \ldots, e_n\} \widetilde{\times} \{e'_1, \ldots, e'_m\} \triangleq \{e_i \cup e'_j \mid i \in [1; n] \wedge j \in [1; m]\} \ .$$

The product $\widetilde{\times}$ is commutative, and we have in particular: $\emptyset \widetilde{\times} B^i = \emptyset$ and $\widetilde{\prod}_{\emptyset} \triangleq \{\emptyset\}$. If $M : A \to B$ is a mapping from elements in A to elements in B, $M\{a \mapsto b\}$ is the mapping M where $a \in A$ now maps to $b \in B$.

DEFINITION.– *A* valuation $\mathbb{V} : V_\varsigma^\omega \mapsto \wp(\wp^{\leq N}(\mathcal{O}bs))$ *is a mapping from each node of* $\mathcal{A}_\varsigma^\omega$ *to a set of* N*-sets of local states.* $\mathbf{Val}$ *is the set of all valuations.* $\mathbb{V}_0 \in \mathbf{Val}$ *refers to the valuation such that* $\forall n \in V_\varsigma^\omega, \mathbb{V}_0(n) = \emptyset$.

DEFINITION.– *For any* $\mathbb{V} \in \mathbf{Val}$ *and* $n \in V_\varsigma^\omega$, $\mathsf{update} : \mathbf{Val} \times V_\varsigma^\omega \to \mathbf{Val}$ *with*

$$\mathsf{update}(\mathbb{V}, n) \triangleq \begin{cases} \mathbb{V}\{n \mapsto \zeta^N(\bigcup_{m \in \mathsf{children}(n)} \mathbb{V}(m))\} & \text{if } n \in \mathbf{Sol} \\ \mathbb{V}\{n \mapsto \zeta^N(\widetilde{\prod}_{m \in \mathsf{children}(n)} \mathbb{V}(m))\} & \text{if } n \in \mathbf{Obj} \\ \mathbb{V}\{n \mapsto \zeta^N(\widetilde{\prod}_{m \in \mathsf{children}(n)} \mathbb{V}(m))\} & \text{if } n \in \mathbf{Proc} \setminus \mathcal{O}bs \\ \mathbb{V}\{n \mapsto \zeta^N(\{\{a_i\}\} \cup \widetilde{\prod}_{m \in \mathsf{children}(n)} \mathbb{V}(m))\} & \text{if } n \in \mathbf{Proc} \cap \mathcal{O}bs \end{cases}$$

where $\zeta^N(\{e_1, \ldots, e_n\}) \triangleq \{e_i \mid i \in [1; n] \wedge |e_i| \leq N \wedge \nexists j \in [1; n], j \neq i, e_j \subset e_i\}$, e_i *being sets,* $\forall i \in [1; n]$.

Starting with $\mathbb{V}_0$, one can repeatedly apply update on each node of $\mathcal{A}_\varsigma^\omega$ to refine its valuation. Only nodes where one of their children values has been modified should be considered for updating. Hence, the order of nodes updates should follow the topological order of the GLC, where children have a lower rank than their parents (i.e., children are treated before their parents). If the graph is truly acyclic, then it is sufficient to update the value of each node only once. In the general case, the nodes of a strongly connected component (which all have the same rank) have to be iteratively updated until convergence of their valuation.

Figure 4.11 formalizes this procedure where $\mathsf{rank}(n)$ refers to the topological rank of n. The node $n \in V_\varsigma^\omega$ to be updated is selected as being the one having the least rank amongst the nodes to update, delimited by $\mathcal{M}$ (line 4). In the case where several nodes with the same lowest rank are in $\mathcal{M}$, they can be either arbitrarily or randomly picked. Once picked, the value of n is updated (line 6). If the new valuation of n is different from the previous one, the parents of n are added to the list of nodes to update (line 8).

Given the particular construction of $\mathcal{A}_\varsigma^\omega$, the theorem given below allows for the derivation of minimal cut sets for a given reachability property. Such a result is possible due to the fact that all abstract bounce sequences computed for $\mathcal{A}_\varsigma^\omega$ are sound: for any objective $P \in V_\varsigma^\omega \cap \mathbf{Obj}$, if a process of each set in $\mathbf{BS}^\wedge(P)$ is disabled, then P becomes unreachable from any state $s \in L$ so that $\mathsf{target}(P) \in s$. Furthermore, the algorithm is guaranteed to terminate. Finally, we note that since the GLC abstracts several dynamical constraints in the underlying Process Hitting model such as the ordering of transitions, the computed cut sets under-approximate the complete cut sets

1: $\mathcal{M} \leftarrow V_\varsigma^\omega$
2: $\mathbb{V} \leftarrow \mathbb{V}_0$
3: **while** $\mathcal{M} \neq \emptyset$ **do**
4: $\quad n \leftarrow \arg\min_{m \in \mathcal{M}} \{\mathsf{rank}(m)\}$
5: $\quad \mathcal{M} \leftarrow \mathcal{M} \setminus \{n\}$
6: $\quad \mathbb{V}' \leftarrow \mathsf{update}(\mathbb{V}, n)$
7: $\quad$ **if** $\mathbb{V}'(n) \neq \mathbb{V}(n)$ **then**
8: $\quad\quad \mathcal{M} \leftarrow \mathcal{M} \cup \mathsf{parents}(n)$
9: $\quad$ **end if**
10: $\quad \mathbb{V} \leftarrow \mathbb{V}'$
11: **end while**
12: **return** $\mathbb{V}$

Figure 4.11: $\mathcal{A}_\varsigma^\omega$-MINIMAL-CUT-NSETS algorithm.

of the model: some cut sets may be missed, and some returned may be non-minimal for the concrete model.

THEOREM.– *The valuation* $\mathbb{V}$ *computed by* $\mathcal{A}_\varsigma^\omega$-MINIMAL-CUT-NSETS *on the GLC* $\mathcal{A}_\varsigma^\omega = (V_\varsigma^\omega, E_\varsigma^\omega)$ *verifies:* $\forall a_i \in \mathbf{Proc} \cap V_\varsigma^\omega, \forall kls \in \mathbb{V}(a_i) \setminus \{\{a_i\}\}$, a_j *is not reachable from* ς *if all processes of* kls *are disabled.*

EXAMPLE.– If an action $a_1 \rightarrow a_1 \Rsh a_0$ is added to the Process Hitting model of figure 4.1, then a trivial solution (i.e., with no successor) is added in the GLC of figure 4.10(top) as a successor of the objective $a_1 \Rsh^* a_0$. Table 4.1 then details the result of algorithm $\mathcal{A}_\varsigma^\omega$-MINIMAL-CUT-NSETS on this GLC. As this GLC contains no cycle, each node is visited once. The result for d_2 consists in five cut sets: three of them are singletons and two are couples of processes.

4.4. Towards a Stochastic Semantic

Up to this point, we have considered Process Hitting in the context of its own syntax: that is to say, in terms of processes interacting with one another via actions, hits and bounces. As we have seen, analysis on this level can answer some of the most interesting questions about the constructed model, including steady state behavior or whether or not a particular state is reachable. However, this analysis is not complete. To answer deeper, more nuanced questions of the model, we must change its context to that of stochasticity. Rather than working with specific pathways (a single state moving into another via playable actions) we will adopt a more global point of view. From this perspective, we consider the fact that a Process Hitting model constructs a problem with a unique solution, a surface which defines the probability of existing at

Node in $\mathcal{A}^{\omega}_{\varsigma}$	Type	Valuation ($\mathbb{V}$)
$\mathbf{BS}^{\wedge}(d_1 \Rsh^* d_1) \ni \emptyset$	**Sol**	$\emptyset$
d_1	**Proc**	$\{\{d_1\}\}$
b_2	**Proc**	$\{\{b_2\},\{d_1\}\}$
$\mathbf{BS}^{\wedge}(a_1 \Rsh^* a_0) \ni \emptyset$	**Sol**	$\emptyset$
a_0	**Proc**	$\{\{a_0\}\}$
c_1	**Proc**	$\{\{a_0\},\{c_1\}\}$
b_1	**Proc**	$\{\{a_0\},\{b_1\},\{c_1\}\}$
$\mathbf{BS}^{\wedge}(d_1 \Rsh^* d_2) \ni \{b_2, c_1\}$	**Sol**	$\{\{a_0\},\{b_2\},\{c_1\},\{d_1\}\}$
$d_1 \Rsh^* d_2$	**Obj**	$\{\{a_0\},\{b_1,b_2\},\{b_1,d_1\},\{c_1\}\}$
d_2	**Proc**	$\{\{a_0\},\{b_1,b_2\},\{b_1,d_1\},\{c_1\},\{d_2\}\}$

Table 4.1: Result of the execution of algorithm $\mathcal{A}^{\omega}_{\varsigma}$-MINIMAL-CUT-NSETS on the Process Hitting model in Figure 4.1, with the addition of the action: $a_1 \to a_1 \Rsh a_0$. Nodes of the form $\mathbf{BS}^{\wedge}(P) \ni ps$ represent one solution of the objective P. The nodes that are not mentioned in this table have the same valuation than their only successor.

any state at any given time. From this distribution we can extract all that there is to know about the system as it has been defined. In this way, the solution is "complete". In this section, we will investigate how to move a Process Hitting model to a more general form without loss of information, what methods are commonly used to solve the system, what kind of questions can be posed of this solution that were not previously accessible and, finally, how to enrich the Process Hitting framework thanks to its newfound stochastic nature.

A Process Hitting action, in its most basic interpretation, moves the system from one state z to another with a given propensity which depends on the state and time, $a(z,t)$. Each action corresponds to a Markov equation which tallies changes in the probability of existing at a given state and time, or $\Phi(z,t)$. Some actions move from the current state, z_c, to another, z_o, causing an outflow in probability at that state, while other actions move into said state, causing inflow in the probability. If we consider all j actions which concern a particular state, we can write the Markov equations for each state in terms of net change:

$$\frac{\partial \Phi(z_c,t)}{\partial t} = \sum_j a_j(z_o,t)\Phi(z_o,t) - \sum_k a_k(z_c,t)\Phi(z_c,t)$$

These equations are linear, time dependent, Partial Differential Equations (PDE). Like any other PDE, they can be solved given an initial condition and boundary conditions. So, unlike static analysis in which one can pose questions based on sufficient conditions of one or two variables, stochastic analysis requires a full initial condition to construct a properly defined problem. Boundary conditions, however, are the same for all Process Hitting models: probability cannot exist in any state not contained by

the sorts, thus is zero everywhere else. The solution to this problem is a sort of multivariate Bernoulli distribution in which exactly one of the K outcomes is successful, or 1-in-K. This is also referred to as the categorical distribution.

Access to this underlying solution gives new avenues of analysis for Process Hitting: for instance, moments such as mean and covariance are quickly derived from a probability distribution and be used to develop hierarchical models. In addition, by observing the probability distribution, one can make informed qualitative statements about the overall behavior of the system and have access to the temporal nature of the problem in addition to the steady state projection.

4.4.1. *Numerical Techniques*

4.4.1.1. *Direct Solution of the Partial Differential Equation*

Once we assemble the system of equations formed by the Markov translations of Process Hitting actions, we can start looking its solution. Ideally, as a system of PDEs with defined initial condition and boundary conditions, we would obtain a direct, analytical solution. This would be exact according to the model as it is defined, without any associated error. However, as is common in the world of PDEs, this is almost never an option outside toy problems. In addition, realistic models in gene regulation, even when given in their most simplistic form, contain many interconnected species. The very enumeration of the possible states of the resulting system create a combinatorial explosion: twenty sorts with two processes each yield 2^{20} states, over one million states. Realistic models are much larger: when considering 300 sorts, the number of states skyrockets to 10^{90}, keeping in mind that 10^{80} is the presumed number of elementary particles in the universe! This is a frequent obstacle in the field of computer science and has been dubbed the "curse of dimensionality".

In order to circumvent this problem, we turn to the many numerical techniques in the field of partial differential equations which can give approximations of the true solution. Each of these techniques come with their own set of tools, including uncertainty quantification to examine the fitness of the resulting approximation, canonical implementations in multiple platforms, and a plethora of literature examining the strengths and weaknesses of each methodology, some of which may be particularly suited to gene regulatory networks. For example, decompositional methods such as Proper Generalized Decomposition (PGD) [CHI 10, CHA 13] can overcome the curse of dimensionality without a priori knowledge of the state space and can incorporate unknown parameters at the cost of an additional dimension. Another particularly adept algorithm is known as Finite State Projection (FSP). In this method, the state-explosion is controlled by recognizing that it is most probable that only a finite number of states will be visited and, thus, truncates the system while retaining enough information to satisfy the modeler's needs. Many adaptations of this algorithm have

been explored over the years, as illustrated in [MUN 08]. As it stands today, no one method satisfies all of the requirements of scalability, computational ease and biological realism, though research in this field is making clear advances.

4.4.1.2. *Simulation Techniques*

So far we have investigated the direct treatment of the Markov equations based on their qualities as partial differential equations. However, some of the most famous and broadly used techniques for obtaining a probability distribution solution for a model such as Process Hitting are simulation-based. In this approach, we let the system begin at a given initial condition and evolve according to the Markov jump equations, keeping track of its pathway until a given stop time. This represents a single trial: by averaging over many trials, we approximate the underlying probability distribution. As the number of trials increase and, eventually, go to infinity, the approximation converges to the true solution. This approach offers many advantages: not only is it very straightforward to understand, but conditions can be formally obtained for determining the number of trials needed to guarantee a "good" fit. As a categorical distribution, the conjugate prior of the solution is a Dirichlet distribution. This allows all information from previous trials can be effortlessly used as a prior for the current run. Of greatest interest in the field of bioinformatics, however, is that no enumeration of the state space is needed. Only visited states are tallied, which is always a much more restricted space when working with networks of large size (recall the curse of dimensionality mentioned in the previous section).

There are instances in which simulation fails: for complex systems or systems with important states exhibiting very low probability, massive numbers of trials are needed to achieve good approximations of the solution. This can be prohibitive in regards to computational run-time and available memory. Even here, approximative methodologies have been developed to overcome some of these difficulties. Under the umbrella of Stochastic Simulation Algorithms (SSA), we find not only Gillespie's algorithm, an exact solver though computationally expensive, but also its adaptations. Two overarching trends exist: methods which attempt to leap forward in time, such as τ-leaping, and methods which separate fast and slow reactions, treating part of the system as though it is deterministic in nature. Although they do attempt to alleviate some of the biggest problems of simulation, even these algorithms can be prohibited by the sheer number of random number generations or trials required.

The construction of a Process Hitting system sets up a problem, the solution of which is a probability distribution. In this section, we have explored two ways of approaching the generated Markov equations: direct treatment via their properties as PDEs or simulation-based techniques. Each of these methods have their advantages and disadvantages. It is up to the modeler to determine which is appropriate for a given regulatory network. In practice, it is common to use static analysis for the initial exploration of a model. Important behaviors are tested against biological data to

validate the basic structure of the system. Later on, simulation or solution techniques are used to perform deeper validation and to obtain an accurate model to export for long term use. Simulation remains the gold standard, although numerical treatment is growing in prevalence as state of the art methodologies from other computational fields make their way into the field of bioinformatics.

4.4.2. *Rates and Stochastic Absorption*

Only the most general dynamics of a system can be derived from the basic interaction graph, that is, the weighted, directed graph indicating the inhibition and activation interactions between genes. Our model is not complete without an in-depth understanding of the reaction rates, until now, only vaguely referenced by the propensity function $a(z,t)$. The correspondence of each Process Hitting model to a stochastic π-calculus allows us to increase the expressiveness with regards to stochastic and temporal features via conditions placed on the channels defined within the calculus. Once the foundation is laid in the Process Hitting language, a model can be further refined through the addition of these features, a theme fore fully explored in [PAU 11b].

The use rate of an action controls the duration and probability of a reaction (a communication along a channel) and corresponds to its own probability distribution along the time axis. The most simplistic and commonly used distribution for a reaction is the exponential distribution with random variable rate, r: $a(z,t) = [z_c = z]\, 1 - e^{-rt}$. The exponential distribution has many advantages that suggest that it should, in fact, be our first choice when modeling relatively unknown reactions. First, the exponential distribution has the greatest entropy of all distributions with mean μ contained by $[0, \infty]$, meaning that it contains the least amount of prior information possible. It also has well-defined moments, including $\mu = r^{-1}$ and $var = r^{-2}$. However, perhaps the most beneficial trait is the fact that the exponential distribution exhibits memorylessness, making simulation algorithms such as Gillespie very efficient.

Any distribution can be used for the use rate, although, as we have demonstrated, some are more adept than others. So what if we would like to control more precisely the channel, say, by modifying the typical amount of wait time without changing the average duration of an action? To this end, we propose the use of another special distribution via the introduction of a new parameter which we call stochasticity absorption, sa. Here, the classic exponential distribution is replaced by the sum of sa random variables, each of which is an exponential distribution with parameter $r \times sa$. This is known as the Erlang distribution, a particular case of the gamma distribution.The mean of this distribution remains the same since $sa\,(r \times sa)^{-1} = r^{-1}$, however, the variance is divided by sa: $sa\,(r \times sa)^{-2} = r^{-2} sa^{-1}$. Thus, the rate becomes more tightly bounded and resembles more a wait time than a stochastic feature, hence the name "stochasticity absorption".

Thanks to this combination of rate and stochasticity absorption, we have shown in [PAU 11b] that any time interval has a corresponding couple of parameters (r, sa) such that the resulting distribution of firing time has the matching confidence interval (given a confidence coefficient). Figure 4.12(left) illustrates the effect of increasing the stochasticity absorption while preserving the rate: the confidence interval of the action duration shrinks around the mean duration fully characterized by the rate. Hence, it is possible to model stochastic dynamics with much more precise timing constraints. This is particularly relevant for qualitative modeling: some logical transitions may hide hundreds or thousands biochemical reactions and should thus exhibit rather low variance. Figure 4.12(right) gives a simple example of two concurrent actions $a_0 \to b_0 \nearrow b_1$ and $a_0 \to a_0 \nearrow a_1$. Due to their stochastic parameters, the resulting probability of observing the reachability of b_1 from the state $\langle a_0, b_0 \rangle$ is very close to 0.

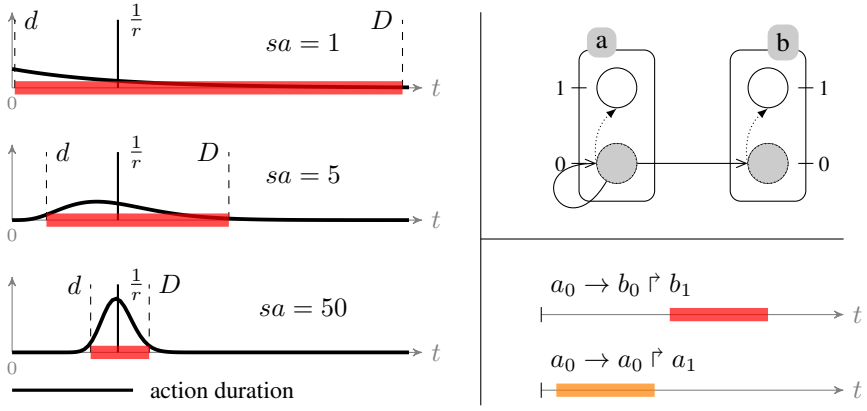


Figure 4.12: (left) Probability distribution of the action duration with a fixed rate r and a varying stochasticity absorption sa. The confidence interval $[d; D]$ at 95% is also indicated. (right) Simple Process Hitting (top) with two actions parametrized using rate and stochasticity absorption resulting in the given the confidence interval (bottom).

Such a framework allows us to model stochastic dynamics with controlled variances. Process hitting may therefore take advantage of both stochastic and timed modeling paradigms.

4.5. Biological Applications

4.5.1. *The Tool* PINT

PINT[1] is a set of command-line programs that implement the various analysis of Process Hitting, including those presented in this chapter. It is distributed freely under the CeCILL license. Graphical interfaces are also available[2], and a web interface[3] as well.

PINT comes with a textual language for describing Process Hitting models. A Process Hitting can be described by the flat list of its actions, but also with macros that ease the writing of the model, in particular for biological interaction networks. Figure 4.13 shows the source file of the refined Process Hitting for the Incoherent Feed-forward Loop described in section §4.2. This file makes use of two macros: `BRN` which computes the generalized dynamics of the given interaction graph; and `COOPERATIVITY` which does the refining according to section §4.2.4. The flat list of actions can be obtained by using the `phc` utility (`phc -l dump -i source.ph`).

```
(* declaration of sorts *)
process a 1
process b 1
process c 1

(* generalized dynamics *)
BRN([a 1 -> + b; a 1 -> - c; b 1 -> + c])

(* cooperativity: c becomes 1 only when a=0 and b=1 *)
COOPERATIVITY([a;b] -> c 0 1, [[0;1]])
```

Figure 4.13: Source file for the Incoherent Feed-forward Loop model described in section §4.2.

Among the features of PINT, one can find: a stochastic simulator that supports stochasticity absorption (non-Markovian simulation); an implementation of the fixed point listing; a static analyzer for reachability properties and cut sets computation; the inference of Boolean/multi-valued networks compatible with the Process Hitting model; importation from/exportation to various formats for biological networks. Further documentation can be found on the PINT website mentioned above.

1. `http://loicpauleve.name/pint`
2. `http://loicpauleve.name/pint/gui`
3. `http://mobyle.biotempo.univ-nantes.fr/cgi-bin/portal.py`

4.5.2. *Biological Examples*

Now that we have established how to construct, refine and analyze a Process Hitting model, we will tackle a more realistic biological example in order to illustrate how the framework can be applied. In particular, we detail an investigation on a medium-scale model of the EGF receptor in order to emphasize the complementarity between our modeling approach through refinement of a generalized dynamics, and our static methods for reachability and cut sets analysis. Finally, we briefly present benchmarks of PINT for the analysis of reachability and cut sets on large scale networks, up to 100 and approximatively 10,000 components.

4.5.2.1. *Investigating the dynamics of EGF receptor*

ErbB is the genetic family of four structurally similar receptor tyrosine kinases, of which the epidermal growth factor receptor (EGFR or ErbB-1) is most widely studied in the field of oncology. In healthy cells, production of ErbB leads to signaling pathway which regulates the cells transition from the G1 to S life phase, a checkpoint which determines whether a cell should divide, delay division or enter a quiescent state. Over expression of ErbB is associated with many kinds of cancer, and drugs which target it and its receptor are common treatments for breast, lung and colon cancers. In the case of breast cancer, drug resistance is present in roughly 30% of patients, leading to a call for alternative targets along the same molecular pathway. Innovative therapies which exploit the structure of the regulatory network have the potential to advance medical interventions on a very real clinical level. Here, we begin our investigation by considering a simplified model of twenty sorts which does not contain all sub reactions, but does capture the most important dynamics of the ErbB signaling process. The directed graph for this network was taken from [SAH 09]. This medium-scale model contains two main proteins of interest: EGF can be considered as the input protein as it is the only one without predecessor, and pRB can be considered as the only output, having no successor.

In order to illustrate reasoning on model dynamics using the Process Hitting and the approach of refining the generalized dynamics of the interaction graph, we apply several analysis on three different Process Hitting models:

– Model (1) contains the generalized dynamics of the IG and therefore encompasses no cooperative sorts.

– In Model (2), the dynamics are refined by the addition of 14 cooperative sorts. While reviewing the literature, components which were noted as being particularly important to the chain of reactions were selected. Indeed, the knockdown experiments conducted in [SAH 09] showed that eliminating these components led to a significant decrease of the production of the output protein pRB. Therefore, all Boolean functions involving these components were included in this model via cooperative sorts. In theory, less cooperative sorts would have sufficed, but we used the splitting method presented in section §4.2.4 to reduce their size.

– Finally, model (3) was built using all of the Boolean functions provided in [SAH 09], taking the form of 22 cooperative sorts.

Table 4.2 sums up several static analysis results on these models from the methods developed in section §4.3, that we briefly comment below.

Model	**Fixed points**	$\mathcal{P}_{\mathrm{EGF}_1}$	$\mathcal{P}_{\mathrm{EGF}_0}$	**Cut sets**
(1)	0	True	True	30
(2)	0	True	False	12
(3)	3	True	False	11

Table 4.2: Results for several analyses on the three models of the EGFR/ErbB regulation with 20 components. In the first column, we list the models being used. The second column gives the number of fixed points enumerated by the static analysis of section §4.3.2. The third column gives the result of the static analysis described in section §4.3.3 for property $\mathsf{reach}(\varsigma, \mathrm{pRB}_0 \Rsh^* \mathrm{pRB}_1)$, where ς is the initial state where all components are at level 0 and the input protein EGF is at level 1. This experiment was repeated for EGF at level 0 and is listed in column four.

Fixed points Model (3) is the only model that contains fixed points. Amongst the three steady states found are two states that correspond to a complete propagation of the input signal, that is, in the case where EGF is active and in the case where it is not. The two other models contain no fixed point because some cooperations are not fully defined, leading to oscillations as a consequence of nondeterministic behavior.

Reachability of the output The main reachability property of interest here is $\mathrm{pRB}_0 \Rsh^* \mathrm{pRB}_1$, given that pRB is the only output of all three models. If we suppose that all components are at first inactive, there are two reachability properties of interest:

– $\mathcal{P}_{\mathrm{EGF}_1} = \mathsf{reach}(\varsigma \Cap \{\mathrm{EGF}_1\}, \mathrm{pRB}_0 \Rsh^* \mathrm{pRB}_1)$,

– $\mathcal{P}_{\mathrm{EGF}_0} = \mathsf{reach}(\varsigma, \mathrm{pRB}_0 \Rsh^* \mathrm{pRB}_1)$,

where ς is the initial context where all sorts are at level 0. In other words, $\mathcal{P}_{\mathrm{EGF}_1}$ means that pRB can be activated in normal conditions (when the input EGF is activated), and $\mathcal{P}_{\mathrm{EGF}_0}$ means that pRB can be activated despite EGF being at 0, a condition which can be used to invalidate a faulty model. We note that $\mathcal{P}_{\mathrm{EGF}_1}$ is, of course, true for all models, and $\mathcal{P}_{\mathrm{EGF}_0}$ is only true for Model (1): the generalized dynamics are too permissive for this system. We also note that the fact that $\mathcal{P}_{\mathrm{EGF}_0}$ is false for the Model (2) is sufficient to prove that the reachability is also impossible for the Model (3), because of the abstraction relationship between the two models.

Cut sets Table 4.3 details the distribution of cut sets on the three models. As detailed in section §4.3.4, a cut set is a set of processes that, if all disabled, makes the

reachability property impossible. From the abstraction relationship between the models, all cut sets of Model (3) are subsets of cut sets of Model (2), and all cut sets of Model (2) are subsets of cut sets of Model (1).

	Model (1)		Model (2)		Model (3)	
Size	**Number**	**Total**	**Number**	**Total**	**Number**	**Total**
1	1 (pRB)	1	6	6	6	6
2	0	1	2	8	2	8
3	2	3	0	8	3	11
4	7	10	1	9		
5	2	12	3	12		
6	4	16				
7	6	22				
8	2	24				
9	6	30				

Table 4.3: Distribution of the cut sets for the three models.

This investigation highlights the advantage of using (partial) refinements of the generalized dynamics of the network using Process Hitting: without a complete knowledge on the precise logical functions between regulators, it is already possible to derive formal dynamical properties of the system and draw conclusions on any subsequently refined models.

4.5.2.2. *Performances on large-scale networks*

The static analysis of Process Hitting models using abstract interpretation presented in section §4.3 has been proved to have a low complexity compared to exact model-checking. Hence, we expect our method to outperform classical model-checkers when applied to large-scale networks. However, our methods may not give conclusive results (necessary conditions are satisfied but not the sufficient conditions). Table 4.4 lists some execution times obtained on various Process Hitting models of signaling pathways: one medium-scale (20 components) and large-scale (104 components) model of EGF receptor [SAH 09, SAM 09], and one medium-scale (40 components) and large-scale (94 components) model of T-Cell receptor [KLA 06, SAE 07]. The number of sorts is typically greater than the number of components as some sorts are used to encode logical functions. These Process Hitting models are available on the PINT website mentioned previously. For each of these models, a broad range of reachability properties have been checked from many initial conditions. Verification has been conducted using exact symbolic model-checkers Biocham [CAL 06] and libddd [HAM 09] in addition to PINT. Whereas PINT makes tractable the formal analysis of the dynamics of those models (thanks to the low complexity of our static analyzes), we emphasize that *all the analyzes are conclusive*.

Model	Nb sorts	Nb procs	Nb actions	Nb states	Biocham	libddd	PINT
egfr20	35	196	670	2^{64}	[3s-out]	[1s-150s]	0.007s
tcrsig40	54	156	301	2^{73}	[1s-out]	[0.6s-out]	0.004s
tcrsig94	133	448	1124	2^{194}	out	out	0.030s
egfr104	193	748	2356	2^{320}	out	out	0.050s

Table 4.4: Comparison of execution times for various reachability properties on several models between Biocham (using NuSMV), libddd, and PINT. An out-of-time/memory is noted "out". When execution times vary significantly depending on the reachability properties, minimum and maximum durations are given.

In order to demonstrate once again the scalability of our approach, we apply our algorithm of under-approximating cut sets for reachability to a model consisting of approximately 10,000 components. This model is a dynamical interpretation of the full PID database [SCH 09] referencing various influences (complex formation, inductions (activations) and inhibitions, transcriptional regulation, etc.) between more than 9,000 biological components (proteins, genes, ions, etc.). Amongt the numerous biological components, the activation of some are known to control key mechanisms of cell dynamics. Those activations are the consequence of intertwining signaling pathways and depend on the environment of the cell (represented by the presence of certain *entry-point* molecules). Uncovering the environmental and intermediate components which play a major role in these signaling dynamics is of great biological interest.

Using this model, we focus on the independent reachability of active SNAIL transcription factor, involved in the epithelial to mesenchymal transition [MOU 07] and of active p15INK4b cyclin-dependent kinase inhibitors involved in cell cycle regulation [DRA 12]. For this analysis, we considered a very large number of different initial conditions: the cut sets are computed from an initial context encoding for about 2^{3000} different initial states. Cut sets would then indicate combinations of knock-outs/ins that should prevent the activations of those components.

The studied Process Hitting[4] gathers more than 21,000 sorts, either biological or logical, containing between 2 and 4 local states. Such a system generates 2^{33874} states. The Graph of Local Causality relates 20,045 nodes, including 5,671 processes (biological or logical). Table 4.5 shows execution time of PINT for the computation of cut N-sets, that are cut sets whose cardinality is at most N, up to $N = 6$.

According to section §4.3.4, all of the returned cut sets are exact: if all the processes of a cut set are disabled, the (transient) reachability of the related process becomes impossible. Also according to section §4.3.4, it may occur that some cut sets

4. Models and scripts available at `http://loicpauleve.name/cutsets.tbz2`

N	Visited nodes in the GLC	Exec. time	Nb N-sets	
			$SNAIL_1$	$p15INK4b_1$
1	29,022	0.9s	1	1
2	36,602	1.6s	6	6
3	44,174	5.4s	0	92
4	54,322	39s	30	60
5	68,214	8.3m	90	80
6	90,902	2.6h	930	208

Table 4.5: Results for the computation of cut N-sets for 2 processes. For each N, only the number of additional N-sets is displayed.

have been missed, or are non-minimal for the concrete Process Hitting model. This is due to the fact that the computation is done on the Graph of Local Causality which over-approximates the dynamics, resulting in an under-approximation of the cut sets.

To our knowledge, PINT is currently the only tool able to perform (transient) reachability and cut sets analysis for dynamics of networks at such a scale, ranging from hundreds to several thousands of components.

4.6. Conclusion

Assessment

In this chapter, we presented an important new approach to the discrete modeling of biological regulatory systems. The fundamental objective of our work is to address very large networks: in doing so, it then becomes possible to work on more realistic models which are of greater interest to biologists.

The ultimate goal of this modeling framework is to allow us to analyze the operational behavior of biological systems, that is, to understand not only what can possibly occur and what will occur ineluctably, but also to understand how to prevent some events from occurring at all, as they may bring upon pathological states. At the very least, we can already make some situations less likely to occur.

Regulatory interactions (activations and inhibitions) are generally characterized by the crossing of some threshold concentration level or expression level. While each of these reactions is, alone, very simple, the overall model complexity arises from the number of interacting agents and the states in which they can be. The sheer number of possible combinations of these states leads to exponential growth of the size of the resulting model. Unfortunately, the huge dimensionality of this model prevents us from analyzing a large number of real systems using classical approaches. As we have shown in this chapter, it is imperative to design methods which make it possible

to smartly abstract pieces of information. This subtle abstraction does not create a loss of knowledge, but only keeps it aside, as though placed on layers of tracing paper. These papers are never all simultaneously superimposed but, rather, taken one by one in accordance with the stage of analysis which one wants to perform. The needs of the modeler may therefore dictate which of the sheets of tracing paper will be taken from a stack which contains a sum of knowledge that cannot be studied as a whole.

It is exactly this decomposition of data which can be realized thanks to our new approach. Initially inspired by the π-calculus, but from which it has since departed, Process Hitting is founded on simple principles:

– Rather than focusing on the concept of global state, our method starts by gathering in a fundamental entity (the so-called *sort*) the various possible states of each component of the system. Based on the principle that, at any moment, this component can be only in one of these states (the current *process*), one avoids the direct enumeration of the state space, a very desirable property in that the space is of huge dimension and much of the global states are unreachable or unexpressed.

– To study the operation of the system, we begin at an initial state (a tuple composed of one process from each sort) and monitor all the states in which each component can be after an interaction between two of the components in the tuple. Only these two components are taken into account in the reaction and only the one that undergoes said reaction changes state. The information to be kept for these steps is reduced to the so-called *process hit*. A complete list of hits allows us to easily reconstruct any situation which can occur according to the system. Furthermore, it is also possible that, after some initial evolution, some events will become unable to occur. Some knowledge of the precedence relationships (immediate precedence or other kinds of precedence, etc. ...) must therefore, at times, be reintroduced to the set of data to be treated.

Process Hitting, therefore, is very well suited to analysis by processing superimposed layers of data. Therein is the great contribution and uniqueness of this framework.

It is to be noted that there exists a complete semantics for Process Hitting, described in part by this chapter, which associate static analysis methods to it. These methods, also described within the chapter, are well suited to the study of discrete, or qualitative, behaviors. Moreover, a projection towards stochastic semantics has been included to illustrate the function and depth of Process Hitting in a concrete way, including ways to exploit this semantic in order to increase model expressivity. One section of this chapter was devoted to a demonstration of the types of biological applications which we were already able to tackle while obtaining some spectacular results on very large, real systems.

Future work

These first very encouraging results motivate us to push onward and to consider future work which seems both interesting and promising. Among those is the automatic inference of biological regulatory networks, a work which has already been launched and relies on the use of Answer Set Programming (ASP).

– In an initial approach, [FOL 12] presents a work which establishes a link between a Process Hitting model and the corresponding Interaction Graph, then to the corresponding modeling according to René Thomas's approach [THO 91]. This is done by developing a method of enumerating compatible parameterizations efficiently.

– A work a little more on the technical than theoretical side is also currently being done so that Process Hitting models may be automatically derived from knowledge established in databases. This approach seems well adapted and specifically useful for the treatment of timed data.

Other forecasted works touch on a smattering of ideas:

– A greatly interesting extension consists of the introduction of *priorities*, as presented in [FOL 13]. The development of priorities lends itself naturally to as of yet open prospects of deepening the description of time-lag behaviors and *urgent actions*.

– In a slightly different direction, as cooperative sorts were presented in section §4.2.4 in order to express actions that have to be processed whenever a logical combination of incoming events have occurred, there is another forthcoming idea of designing multiple actions which are simultaneously provoked by some event. These *multi-actions* are made possible as an improvement of the above so-called urgent actions.

– In addition to the work already done on static analysis as it pertains to reachability properties, we will also try to deal with more intricate properties such as conditional reachability. For example, it could be of interest to know if some state is reachable as long as some actions on certain states are avoided or on the condition that we do visit some given list of states.

– A greater pursuit of the stochastic semantic and investigation of novel numerical techniques specifically designed for such a qualitative modeling scheme may open doors to greater depths of analysis and, perhaps, the inclusion of more stochastic/temporal features to Process Hitting.

This is a non exhaustive list of the work we intend to be involved in for the following years. Many beautiful results were thus already obtained, but several interesting prospects still remain to be explored which enrich this framework for the analysis of biological systems.

4.7. Bibliography

[AKU 08] AKUTSU T., HAYASHIDA M., TAMURA T., "Algorithms for inference, analysis and control of Boolean networks", *Algebraic Biology*, p. 1–15, Springer, 2008.

[ARA 08] ARACENA J., "Maximum number of fixed points in regulatory boolean networks", *Bulletin of Mathematical Biology*, vol. 70, num. 5, p. 1398–1409, Springer, 2008.

[ARA 09] ARACENA J., CH. E. G., MOREIRA A., SALINAS L., "On the robustness of update schedules in Boolean networks.", *Biosystems*, vol. 97, num. 1, p. 1-8, 2009.

[BER 92] BERNARDINELLO L., DE CINDIO F., "A survey of basic net models and modular net classes", ROZENBERG G., Ed., *Advances in Petri Nets 1992*, vol. 609 of *Lecture Notes in Computer Science*, p. 304-351, Springer Berlin / Heidelberg, 1992.

[BER 08] BERNOT G., COMET J.-P., KHALIS Z., "Gene regulatory networks with multiplexes", *European Simulation and Modelling Conference Proceedings*, p. 423–432, 2008.

[CAL 06] CALZONE L., FAGES F., SOLIMAN S., "BIOCHAM: an environment for modeling biological systems and formalizing experimental knowledge", *Bioinformatics*, vol. 22, num. 14, p. 1805-1807, 2006.

[CHA 13] CHANCELLOR C., AMMAR A., CHINESTA F., MAGNIN M., ROUX O., "Linking Discrete and Stochastic Models: The Chemical Master Equation as a Bridge between Process Hitting and Proper Generalized Decomposition", *Computational Methods in Systems Biology*, Springer, p. 50–63, 2013.

[CHI 10] CHINESTA F., AMMAR A., CUETO E., "On the use of proper generalized decompositions for solving the multidimensional chemical master equation", *European Journal of Computational Mechanics/Revue Européenne de Mécanique Numérique*, vol. 19, num. 1-3, p. 53–64, Taylor & Francis, 2010.

[COU 77] COUSOT P., COUSOT R., "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*, ACM, p. 238–252, 1977.

[DRA 12] DRABSCH Y., TEN DIJKE P., "TGF-β signalling and its role in cancer progression and metastasis", *Cancer Metastasis Rev.*, vol. 31, num. 3-4, p. 553–568, Dec 2012.

[FOL 12] FOLSCHETTE M., PAULEVÉ L., INOUE K., MAGNIN M., ROUX O., "Concretizing the Process Hitting into Biological Regulatory Networks", GILBERT D., HEINER M., Eds., *Computational Methods in Systems Biology*, Lecture Notes in Computer Science, p. 166–186, Springer Berlin Heidelberg, 2012.

[FOL 13] FOLSCHETTE M., PAULEVÉ L., MAGNIN M., ROUX O., "Under-approximation of reachability in multivalued asynchronous networks", *Electronic Notes in Theoretical Computer Science*, vol. 299, p. 33 - 51, 2013, 4th International Workshop on Interactions between Computer Science and Biology (CS2Bio'13).

[HAM 09] HAMEZ A., THIERRY-MIEG Y., KORDON F., "Building efficient model checkers using hierarchical set decision diagrams and automatic saturation", *Fundamenta Informaticae*, vol. 94, num. 3, p. 413–437, IOS Press, 2009.

[IDE 01] IDEKER T., GALITSKI T., HOOD L., "A new approach to decoding life: systems biology", *Annual review of genomics and human genetics*, vol. 2, num. 1, p. 343–372, Annual Reviews 4139 El Camino Way, PO Box 10139, Palo Alto, CA 94303-0139, USA, 2001.

[INO 11] INOUE K., "Logic Programming for Boolean Networks", *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Two*, IJCAI'11, AAAI Press, p. 924–930, 2011.

[KAU 69] KAUFFMAN S. A., "Metabolic stability and epigenesis in randomly constructed genetic nets", *Journal of theoretical biology*, vol. 22, num. 3, p. 437–467, Elsevier, 1969.

[KLA 06] KLAMT S., SAEZ-RODRIGUEZ J., LINDQUIST J., SIMEONI L., GILLES E., "A methodology for the structural and functional analysis of signaling and regulatory networks", *BMC Bioinformatics*, vol. 7, num. 1, Page 56, 2006.

[MAN 03] MANGAN S., ALON U., "Structure and function of the feed-forward loop network motif.", *PNAS*, vol. 100, num. 21, p. 11980–11985, Oct 2003.

[MOU 07] MOUSTAKAS A., HELDIN C. H., "Signaling networks guiding epithelial-mesenchymal transitions during embryogenesis and cancer progression", *Cancer Sci.*, vol. 98, num. 10, p. 1512–1520, Oct 2007.

[MUN 08] MUNSKY B. E., *The finite state projection approach for the solution of the master equation and its applications to stochastic gene regulatory networks*, ProQuest, 2008.

[NAL 07] NALDI A., THIEFFRY D., CHAOUIYA C., "Decision diagrams for the representation and analysis of logical models of genetic networks", *Computational Methods in Systems Biology*, Springer, p. 233–247, 2007.

[NOU 11a] NOUAL M., "Synchronism vs Asynchronism in Boolean networks", *CoRR*, vol. abs/1104.4039, 2011.

[NOU 11b] NOUAL M., REGNAULT D., SENÉ S., "Non-monotony and Boolean automata networks", *CoRR*, vol. abs/1111.4552, 2011.

[PAU 10] PAULEVÉ L., RICHARD A., "Topological Fixed Points in Boolean Networks", *Comptes Rendus de l'Académie des Sciences - Series I - Mathematics*, vol. 348, num. 15-16, p. 825–828, 2010.

[PAU 11a] PAULEVÉ L., MAGNIN M., ROUX O., "Refining Dynamics of Gene Regulatory Networks in a Stochastic π-Calculus Framework", *Transactions on Computational Systems Biology XIII*, p. 171-191, Springer, 2011.

[PAU 11b] PAULEVÉ L., MAGNIN M., ROUX O., "Tuning Temporal Features within the Stochastic π-Calculus", *Software Engineering, IEEE Transactions on*, vol. 37, num. 6, p. 858–871, IEEE, 2011.

[PAU 11c] PAULEVÉ L., RICHARD A., "Static Analysis of Boolean Networks Based on Interaction Graphs: a Survey", *Electronic Notes in Theoretical Computer Science*, vol. 284, p. 93 - 104, 2011, Proceedings of The Second International Workshop on Static Analysis and Systems Biology (SASB 2011).

[PAU 12a] PAULEVÉ L., MAGNIN M., ROUX O., From the Process Hitting to Petri Nets and Back, Technical Report num. hal-00744807, October 2012.

[PAU 12b] PAULEVÉ L., MAGNIN M., ROUX O., "Static Analysis of Biological Regulatory Networks Dynamics using Abstract Interpretation", *Mathematical Structures in Computer Science*, vol. 22, num. 04, p. 651-685, 2012.

[PAU 13] PAULEVÉ L., ANDRIEUX G., KOEPPL H., "Under-Approximating Cut Sets for Reachability in Large Scale Automata Networks", SHARYGINA N., VEITH H., Eds., *Computer Aided Verification*, vol. 8044 of *Lecture Notes in Computer Science*, p. 69-84, Springer Berlin Heidelberg, 2013.

[RIC 06a] RICHARD A., COMET J.-P., BERNOT G., "Formal methods for modeling biological regulatory networks", *Modern Formal Methods and Applications*, p. 83–122, Springer, 2006.

[RIC 06b] RICHARD A., COMET J.-P., BERNOT G., "*Modern Formal Methods and Applications*", Chapter Formal Methods for Modeling Biological Regulatory Networks, p. 83–122, 2006.

[RIC 10] RICHARD A., "Negative circuits and sustained oscillations in asynchronous automata networks", *Advances in Applied Mathematics*, vol. 44, num. 4, p. 378–392, 2010.

[SAE 07] SAEZ-RODRIGUEZ J., SIMEONI L., LINDQUIST J. A., HEMENWAY R., BOMMHARDT U., ARNDT B., HAUS U.-U., WEISMANTEL R., GILLES E. D., KLAMT S. et al., "A Logical Model Provides Insights into T Cell Receptor Signaling", *PLoS Computational Biology*, vol. 3, num. 8, Pagee163, Public Library of Science, 2007.

[SAH 09] SAHIN O., FROHLICH H., LOBKE C., KORF U., BURMESTER S., MAJETY M., MATTERN J., SCHUPP I., CHAOUIYA C., THIEFFRY D., POUSTKA A., WIEMANN S., BEISSBARTH T., ARLT D., "Modeling ERBB receptor-regulated G1/S transition to find novel targets for de novo trastuzumab resistance", *BMC Systems Biology*, vol. 3, num. 1, 2009.

[SAM 09] SAMAGA R., SAEZ-RODRIGUEZ J., ALEXOPOULOS L. G., SORGER P. K., KLAMT S., "The Logic of EGFR/ErbB Signaling: Theoretical Properties and Analysis of High-Throughput Data", *PLoS Computational Biology*, vol. 5, num. 8, Pagee1000438, Public Library of Science, 2009.

[SCH 09] SCHAEFER C. F., ANTHONY K., KRUPA S., BUCHOFF J., DAY M., HANNAY T., BUETOW K. H., "PID: The Pathway Interaction Database", *Nucleic Acids Res.*, vol. 37, p. D674-9, 2009.

[THO 91] THOMAS R., "Regulatory networks seen as asynchronous automata: a logical description", *Journal of Theoretical Biology*, vol. 153, num. 1, p. 1–23, Elsevier, 1991.

[WIE 48] WIENER N., *Cybernetics: Control and communication in the animal and the machine*, Wiley New York, 1948.