Refining Dynamics of Gene Regulatory Networks in a Stochastic π-Calculus Framework

Loïc Paulevé, Morgan Magnin, and Olivier Roux

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

Abstract. In this paper, we introduce a framework allowing to model and analyse efficiently Gene Regulatory Networks (GRNs) in their temporal and stochastic aspects. The analysis of stable states and inference of René Thomas' discrete parameters derives from this logical formalism. We offer a compositional approach which comes with a natural translation to the Stochastic π-Calculus. The method we propose consists in successive refinements of generalised dynamics of GRNs. We illustrate the merits and scalability of our framework on the control of the differentiation in a GRN generalising metazoan segmentation processes, and on the analysis of stable states within a large GRN studied in the scope of breast cancer researches.

1 Introduction

Modelling, analysis and numerical or stochastic simulations are a usual means to predict the behaviour of complex living systems such as interacting genes.

Regulations between genes (activation or inhibition) are generally represented by Gene Regulatory Network (GRN) graphs. However, a GRN graph is not enough to describe dynamics. In continuous frameworks such as ordinary differential equations, parameters for differential equations are needed. In logical (or qualitative) frameworks such as boolean or discrete networks, dynamics are driven by René Thomas' parameters or equivalent [1].

Hybrid modelling brings quantitative aspects — such as temporal or stochastic parameters — to logical modelling. In the field of formal languages, κ language [2] or Stochastic π-Calculus [3,4,5] bring theoretical Computer Science frameworks for biological modelling. In the field of formal verifications of biological systems, frameworks like Time or Stochastic Petri Nets [6,7], Biocham [8], Timed Automata [9], Charon (Hybrid Modelling) [10] and Linear Hybrid Modelling [11] bring the first bricks for verifying and controlling dynamics of such systems.

Inference of temporal and stochastic parameters is still challenging as the domain of parameters is continuous and its volume generally grows exponentially with the number of genes. Compositional approaches, inherent to process algebras, aspire at reducing this complexity by allowing a local reasoning.

Our aim is temporal parameters synthesis for verifying formal properties on hybrid models.

Our contribution consists in the introduction of both temporal and stochastic parameters into process algebra models of GRNs through a new Stochastic π-Calculus framework: the Process Hitting framework.

Starting from a GRN without any other parameters, its largest dynamics are expressed in Process Hitting and then are refined to match the expected behaviour. Such a refinement is achieved by constructing cooperativity between genes and by creating stable states. As we will show, detecting stable states of a Process Hitting is straightforward, as well as infering the René Thomas' discrete parameters K.

Moreover, the Stochastic π-Calculus naturally brings time and stochasticity into our Process Hitting framework. We introduce a *stochasticity absorption factor* to highlight either the temporal or stochastic aspect of reactions. The direct translation of Process Hitting to the Stochastic π-Calculus allows simulations of such models by softwares like BioSpi [3] or SPiM [12].

Several points motivate the choice of the Stochastic π-Calculus framework for introducing the Process Hitting. The Stochastic π-Calculus can be considered as a "low-level" process algebra: there are very few operators compared, for instance, to the Beta Workbench [13] or Bio-PEPA [14]. This makes the presentation of the Process Hitting as a Stochastic π-Calculus framework more generic. Moreover, the Stochastic π-Calculus comes with a bunch of established tools as previously cited and translations into various framework have already been formalized, such as to PRISM, a probabilistic model checking tool [15,16].

This paper is structured as follows. Section 2 introduces our framework and how it is used to build the generalised dynamics of a GRN. Section 3 presents dynamics refinement techniques and Section 4 shows how infering the René Thomas' parameters leading to such dynamics. Introduction of both temporal and stochastic parameters within Process Hitting models is addressed in Section 5. The overall approach is illustrated by two applications in Section 6. The first applies the refinement method to a toy GRN involved in biological segmentations phenomena. The temporal and stochastic parameters are then infered to bring a particular behaviour to the system. The second application shows the scalability of the Process Hitting framework by modelling a large GRN composed of 20 genes.

Notations. Given a set S, $\overbrace{S \times \cdots \times S}^{n}$, will be abbreviated as S^n. If S is finite and countable, we note $|S|$ its cardinality. Given a n-tuple C, $C[x/y]$ refers to the n-tuple C within the element y has been substituted by x. Belonging and cartesian product for n-tuples are defined similarly to sets. $[x_i; x_{i+n}]$ refers to the interval $\{x_i, x_{i+1}, \ldots, x_{i+n}\}$. '$\wedge$' stands for the logical *and* connector.

2 Generalised Dynamics for Gene Regulatory Networks

First, we recall the basis of the René Thomas' discrete modelling framework from which we designed our refinement approach. This method is described in subsections 2.2 and 2.3. This leads to a straightforward translation into the π-Calculus which makes it possible to express generalised dynamics of GRNs.

2.1 Gene Regulatory Networks

GRNs are often described by interaction graphs where nodes are genes with activation and inhibition relations respectively represented by positive and negative edges [1,17].

In the discrete framework of René Thomas, each gene has at least two qualitative levels. The influence of an activating (resp. inhibiting) gene on its target depends on a threshold value: when the level of the gene is greater or equal than the threshold, the gene holds a positive (resp. negative) effect; when the level of the gene is lower than the threshold, the gene holds a negative (resp. positive) effect [1].

Definition 1 (Gene Regulatory Network Graph). *A* Gene Regulatory Network graph *is a triple* (Γ, E_+, E_-) *where* Γ *is the finite set of genes and* $a \xrightarrow{t} b \in E_+$ *(resp.* E_-*),* t *positive integer, if and only if the gene* a *above level* t *is an activator (resp. inhibitor) for* b*. We note* a_i *the level* i *of the gene* a*.*

Given a GRN graph (Γ, E_+, E_-), the maximum qualitative level for gene $a \in \Gamma$ is noted a_{l_a} where l_a is the highest threshold involved in its regulations:

$$l_a = max(\{t \mid \exists b \in \Gamma, a \xrightarrow{t} b \in E_+ \cup E_-\}) \ . \tag{1}$$

We denote $levels_+(a, b)$ (resp. $levels_-(a, b)$) the set of levels of a where a effectively activates (resp. inhibits) b.

Definition 2 (Effective Levels). *If* $a \xrightarrow{t} b \in E_+$*,* $levels_+(a, b) = [a_t; a_{l_a}]$ *and* $levels_-(a, b) = [a_0; a_{t-1}]$*. If* $a \xrightarrow{t} b \in E_-$*,* $levels_+(a, b) = [a_0; a_{t-1}]$ *and* $levels_-(a, b) = [a_t; a_{l_a}]$*. Else* $levels_+(a, b) = levels_-(a, b) = \emptyset$*.*

2.2 The Process Hitting Framework

We want to describe the action of a gene at a given level on another one. If the gene a at a given level i is an activator for b, it has a positive action on b, meaning the level of b will tend to increase. Conversely if a at a level i' is an inhibitor for b, it has a negative action on b and then the level of b will tend to decrease.

The action is "a at level i making b at level j increase (or decrease) to level k". We say a_i *hits* b_j to make it *bounce* to b_k and note this action $a_i \rightarrow b_j \Rsh b_k$. In the process hitting framework, a_i, b_j, b_k are refered as *processes* and a, b as *sorts*. Sorts can represent genes, but also logical entities, as described in further sections.

Definition 3 (Action). *An* action *is noted* $a_i \rightarrow b_j \Rsh b_k$ *where* a_i *is a process of sort* a *and* $b_j \neq b_k$ *two processes of sort* b*.* $a_i \rightarrow b_j$ *is the* hit *part, and* $b_j \Rsh b_k$ *the* bounce *part. When* $a_i = b_j$*, such an action is refered as a* self-action *and* a_i *is called a* self-hitting process*.*

In this paper, one and only one process of each sort is present at any instant. Hence, hits between different processes of a same sort are prohibited. The set of these living processes gives the state of the Process Hitting.

Definition 4 (Process Hitting). *A* Process Hitting $\mathcal{PH}$ *is a triple* $(\Sigma, L, \mathcal{H})$*:*

- $\Sigma = \{a, b, \dots\}$ *is the finite countable set of sorts,*
- $L = \prod_{a\in\Sigma} L_a$ *is the set of states for* $\mathcal{PH}$*, with* $L_a = \{a_0 \dots a_{l_a}\}$ *the finite and countable set of processes of sort* $a \in \Sigma$ *and* l_a *a positive integer,* $a \neq b \Rightarrow a_i \neq b_j \; \forall (a_i, b_j) \in L_a \times L_b$*,*
- $\mathcal{H} = \{a_i \rightarrow b_j \upharpoonright b_k, \cdots \mid (a, b) \in \Sigma^2,\; (a_i, b_j, b_k) \in L_a \times L_b \times L_b, b_j \neq b_k, a = b \Rightarrow a_i = b_j\}$*, is the finite set of actions.*

At a given state $s \in L$, an action $a_i \rightarrow b_j \upharpoonright b_k$ is playable if both processes a_i and b_j are present in s. When this action is played, the process b_k replaces b_j.

Definition 5 (Next States). *Let* $(\Sigma, L, \mathcal{H})$ *be a Process Hitting and* $s \in L$ *be one of its states. The set of the next possible states for* s *are computed as follows:*

$$next(s) = \{s[b_k/b_j] \mid \exists (a_i, b_j) \in s^2, \exists b_k \in L_b, a_i \rightarrow b_j \upharpoonright b_k \in \mathcal{H}\}\ .$$

Definition 6 (Stable state). *Let* $\mathcal{PH} = (\Sigma, L, \mathcal{H})$ *be a Process Hitting and* $s \in L$ *be a state,* s *is a* stable state *for* $\mathcal{PH}$ *if and only if* $next(s) = \emptyset$*.*

2.3 Graphical Representations of a Process Hitting

We set up two complementary graphical representations of a Process Hitting. The first one exhibits the actions between process levels, the second one points out the absence of hits between them. We finally define the State Graph of a Process Hitting. Figure 1 shows an instance for each of these three representations.

Given a Process Hitting $(\Sigma, L, \mathcal{H})$, its *Hypergraph* represents each action $a_i \rightarrow b_j \upharpoonright b_k \in \mathcal{H}$ by a directed hyperedge from a_i to b_k passing by b_j. The hit part (a_i to b_j) is drawn as a plain edge and the bounce part (b_j to b_k) as a dotted edge.

Definition 7 (Process Hitting Hypergraph). *The* Hypergraph *of a Process Hitting* $(\Sigma, L, \mathcal{H})$ *is a couple* (P, A) *where* $P = \bigcup_{a\in\Sigma} L_a$ *are the vertices and* $A \subseteq P^3$ *the directed hyperedges given by* $A = \{(a_i, b_j, b_k) \mid a_i \rightarrow b_j \upharpoonright b_k \in \mathcal{H}\}$*.*

In the following we introduce a complementary representation we call *Hitless Graph.* It will allow us to obtain extra results such as the stable states of a Process Hitting (Section 3.2). The Hitless Graph of a Process Hitting $(\Sigma, L, \mathcal{H})$ relates two processes of different sorts if and only if they hit neither each other nor themselves. Vertices of a Hitless Graph may be split into $n \leq |\Sigma|$ partitions having no element inside related to each other: a partition is, for any sort, a subset of its processes without self-actions. Such a graph is called n-partite.

Definition 8 (n-Partite Graph). *A graph $G = (V, E)$ is* n-partite *if and only if $V = \bigcup_{k=1}^{n} V_k,\ V_k \neq \emptyset,\ \forall 1 \leq k, k' \leq n, V_k \cap V_{k'} = \emptyset$ and $(a_i, b_j) \in E \Rightarrow$ $\exists 1 \leq k \neq k' \leq n,\ a_i \in V_k \wedge b_j \in V_{k'}$.*

Definition 9 (Hitless Graph). *Given a Process Hitting $\mathcal{PH} = (\Sigma, L, \mathcal{H})$, its Hitless Graph $\overline{\mathcal{PH}} = (V, E)$ is defined as a non-directed graph where the vertices V and edges E are computed as follows:*

$$V = \bigcup_{a \in \Sigma} \{a_i \in L_a \mid \forall a_{i'} \in L_a \nexists a_i \rightarrow a_i \upharpoonright a_{i'} \in \mathcal{H}\}$$

$$E = \{(a_i, b_j) \in V^2 \mid \forall b_{j'} \in L_b \nexists a_i \rightarrow b_j \upharpoonright b_{j'} \in \mathcal{H} \wedge \forall a_{i'} \in L_a \nexists b_j \rightarrow a_i \upharpoonright a_{i'} \in \mathcal{H}\} \ .$$

Property 1. By construction of V and E, $\overline{\mathcal{PH}}$ is a n-partite graph, $n \leq |\Sigma|$, where each partition is a subset of processes for one and only one sort and to each sort corresponds at most one partition.

We also define the n-cliques of a graph which are subsets of n vertices such that each element is related to each other.

Definition 10 (n-Clique). *Given a graph $G = (V, E)$, $C \subseteq V$ is a* $|C|$-clique *of G if and only if $\forall (a_i, b_j) \in C^2, \{a_i, b_j\} \in E$.*

Property 2. n-cliques of a n-partite graph have one and only one vertex in each partition.

Finally, the *State Graph* of a Process Hitting represents the possible transitions between each couple of its states.

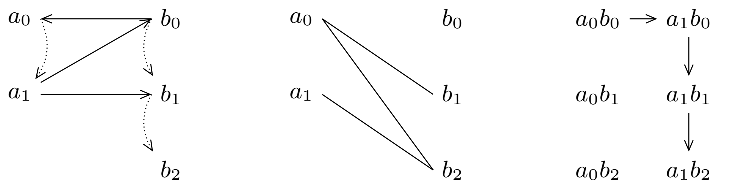
Definition 11 (State Graph). *Given a Process Hitting $(\Sigma, L, \mathcal{H})$, its* State Graph *is a directed graph $\mathcal{S} = (L, E \subseteq L^2)$ with $(s, s') \in E \Leftrightarrow s' \in next(s)$.*

2.4 From Process Hitting to the π-Calculus

A main advantage of our approach is its natural translation to the π-Calculus. In this subsection we propose a method to translate any Process Hitting into a π-Calculus expression.

We briefly present the fragment of the π-Calculus which is sufficient for translating a Process Hitting. The full syntax for π-Calculus and examples can be found in [5,18]. π-Calculus expressions compose two kinds of objects: independently defined processes and channels shared by some processes. A process P has the capability to output (resp. input) on a channel γ and then become P', noted $!\gamma.P'$ (resp. $?\gamma.P'$). Output and input are synchronized operations, i.e. an outputting process is blocked until another process inputs on the same channel. A process can also execute an internal action (τ), nil operation ($\mathbf{0}$) or one amongst several ($P' + P''$).

Let $\mathcal{PH} = (\Sigma, L, \mathcal{H})$ be a Process Hitting. For each process a_i of $\mathcal{PH}$, a π-Calculus process A_i is defined as follows. For each action $a_i \rightarrow b_j \upharpoonright b_k \in \mathcal{H}$ where



(a) Hypergraph (b) Hitless Graph (c) State Graph

Fig. 1. Graphical representations for the Process Hitting $\mathcal{PH} = (\{a, b\}, \{a_0, a_1\} \times \{b_0, b_1, b_2\}, \mathcal{H})$ with $\mathcal{H} = \{b_0 \rightarrow a_0 \Rsh a_1, a_1 \rightarrow b_0 \Rsh b_1, a_1 \rightarrow b_1 \Rsh b_2\}$.

$a \neq b$, a new channel γ_α is created. The π-Calculus process A_i has the ability to output on this channel and the π-Calculus process B_j has the ability to input on this channel so as to become B_k (2). For each self-action $a_i \rightarrow a_i \Rsh a_j \in \mathcal{H}$, A_i has the ability to become A_j after performing an internal action τ_α (3).

$$A_i ::= \sum_{\substack{\alpha = a_i \rightarrow b_j \Rsh b_k \in \mathcal{H} \\ a \neq b}} !\gamma_\alpha . A_i + \sum_{\substack{\alpha = b_j \rightarrow a_i \Rsh a_k \in \mathcal{H} \\ a \neq b}} ?\gamma_\alpha . A_k \tag{2}$$

$$+ \sum_{\alpha = a_i \rightarrow a_i \Rsh a_k \in \mathcal{H}} \tau_\alpha . A_k \tag{3}$$

2.5 Generalised Dynamics for Gene Regulatory Networks

Our method to analyse GRNs takes benefit from the use of refinement techniques. Starting from the largest set of possible dynamics for the GRN, we gradually take into account only the specified behaviours and exclude the other ones, thus leading to a restrictive process.

We call this largest set of dynamics the *generalised dynamics* for the GRN graph. It is described by the following rules: the level of a gene increases (resp. decreases) if and only if at least one of its activators (resp. inhibitors) is present. The absence of activators is equivalent to the presence of one inhibitor.

Let $\mathcal{G} = (\Gamma, E_+, E_-)$ be a GRN graph. For all $(a, b) \in \Gamma^2$, we build the set of actions $\mathcal{H}_a^b$ from a to b reflecting the rules above:

– If $a \xrightarrow{t} b \in E_+$, all processes of sort a below the threshold t hit all processes of sort b but b_0 to make them decrease to the level below. Moreover, all processes of sort a above the threshold t hit all processes of sort b but b_{l_b} to make them increase to the level above:

$$\begin{aligned} \mathcal{H}_a^b = \{&a_i \rightarrow b_j \Rsh b_{j-1} \mid 0 \leq i < t, 1 \leq j \leq l_b\} \\ \cup \{&a_{i'} \rightarrow b_{j'} \Rsh b_{j'+1} \mid t \leq i' \leq l_a, 0 \leq j' < l_b\} \ . \end{aligned}$$

– If $a \xrightarrow{t} b \in E_-$, the actions are defined similarly to the previous case except for the bounce directions which are reversed:

$$\begin{aligned}\mathcal{H}_a^b = &\{a_i \rightarrow b_j \upharpoonright b_{j+1} \mid 0 \le i < t, 0 \le j < l_b\} \\ &\cup \{a_{i'} \rightarrow b_{j'} \upharpoonright b_{j'-1} \mid t \le i' \le l_a, 1 \le j' \le l_b\} \ .\end{aligned}$$

– If $b = a$ and $\nexists c \in \Gamma,\ c \xrightarrow{t} b \in E_- \cup E_+$, gene b lives in absence of activators: all processes of sort b but b_0 hit themselves to decrease to the level below.

$$\mathcal{H}_b^b = \{b_i \rightarrow b_i \upharpoonright b_{i-1} \mid 1 \le i \le l_b\} \ .$$

– Obviously, if $a \xrightarrow{t} b \notin E_- \cup E_+$ for any t and the previous case does not hold, we define $\mathcal{H}_a^b = \emptyset$.

The Process Hitting for the generalised dynamics of $\mathcal{G}$ is given by

$$\mathcal{PH} \ = \ (\Gamma,\ \prod_{a \in \Gamma} \{a_0, \dots, a_{l_a}\},\ \bigcup_{(a,b) \in \Gamma^2} \mathcal{H}_a^b) \ .$$

3 Refining Dynamics of Gene Regulatory Networks

We present two methods which aim at narrowing the set of dynamics of a Process Hitting for a GRN: the first one is based on cooperativity between genes, the other one deals with the knowledge of the stable states.

3.1 Cooperative Hits

Given two genes c and f regulating a gene a, the action of c on a may depend on the level of f: there exists a cooperativity between c and f on a. In discrete frameworks, the cooperativity is often described by a boolean function between genes levels [1,19]. We show how to build cooperativity within Process Hitting.

Let $(\Sigma, L, \mathcal{H})$ be a Process Hitting and $\sigma \subset \Sigma$ be a set of sorts cooperating on a given process a_k to make it bounce to $a_{k'}$. The set of all states of the cooperating sorts is denoted by $S = \prod_{z \in \sigma} L_z$. The subset of states where the cooperativity is effective is defined by $\top \subset S$.

For applying this cooperation, a new sort is added to the Process Hitting. This sort is called a *cooperative sort* and is refered as υ. The set of processes of sort υ is defined by $L_\upsilon = \{\upsilon_\varsigma, \forall \varsigma \in S\}$. Each process z_i of sort $z \in \sigma$ hits processes υ_ς of the cooperative sort υ where $z_i \notin \varsigma$ to make it bounce to $\upsilon_{\varsigma[z_i/z_j]}, z_j \in \varsigma$. We denote $\mathcal{H}_\sigma$ the set of such actions (4). In this way, the process of sort υ reflects the current state of its representatives.

The cooperativity between υ processes is added into the Process Hitting by replacing hits $\mathcal{H}_{coop}$ from processes of sorts present in σ to c_k (5) by hits $\mathcal{H}'_{coop}$ from processes of the cooperative sort υ selected in $\top$ (6).

$$\mathcal{H}_\sigma = \{z_i \rightarrow \upsilon_\varsigma \upharpoonright \upsilon_{\varsigma[z_i/z_j]} \mid \forall z \in \sigma, \forall (z_i, z_j) \in L_z^2, \forall \upsilon_\varsigma \in L_\upsilon, z_j \in \varsigma\} \quad (4)$$

$$\mathcal{H}_{coop} = \{z_i \rightarrow a_k \upharpoonright a_{k'} \in \mathcal{H} \mid \forall z \in \sigma\} \quad (5)$$

$$\mathcal{H}'_{coop} = \{\upsilon_\varsigma \rightarrow a_k \upharpoonright a_{k'} \mid \forall \varsigma \in \top\} \ . \quad (6)$$

The resulting Process Hitting is $(\Sigma \cup \{\upsilon\}, L \times L_\upsilon, (\mathcal{H} \setminus \mathcal{H}_{coop}) \cup \mathcal{H}_\sigma \cup \mathcal{H}'_{coop})$.

Example 1. Let $(\{f,c,a\}, \{f_0,f_1\} \times \{c_0,c_1\} \times \{a_0,a_1\}, \mathcal{H})$ be a Process Hitting where $\{f_1 \to a_0 \Rsh a_1, c_0 \to a_0 \Rsh a_1\} \subset \mathcal{H}$. The creation of a cooperativity between f_1 and c_0 on a_0 ($\sigma = \{f,c\}$, $\top = \{f_1 c_0\}$) is illustrated by Figure 2.

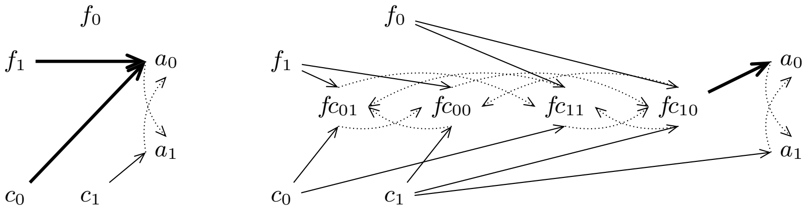


Fig. 2. Construction of a cooperative hit between f_1 and c_0 on a_0 (thick lines): $\sigma = \{f,c\}$, $\top = \{f_1 c_0\}$, $\upsilon = fc$. fc_{01} stands for the process corresponding to the state $f_0 c_1$ of the cooperating processes σ.

3.2 Stable State Pattern

Given a Process Hitting $(\Sigma, L, \mathcal{H})$, we prove the $|\Sigma|$-cliques of its Hitless Graph are exactly its stable states. Thus, stable states may be created by removing from the Process Hitting the very hits that make such patterns appear.

Theorem 1. *Let $\mathcal{PH} = (\Sigma, L, \mathcal{H})$ be a Process Hitting and $\overline{\mathcal{PH}}$ its Hitless Graph. A state $s \in L$ is stable if and only if s is a $|\Sigma|$-clique for $\overline{\mathcal{PH}}$.*

Proof. By definition, $\mathit{next}(s) = \emptyset$ if and only if there is no hit between any couple of processes present in s. This is equivalent to have s a clique of $\overline{\mathcal{PH}}$.

Figure 3 shows an instance of Process Hitting having one stable state.

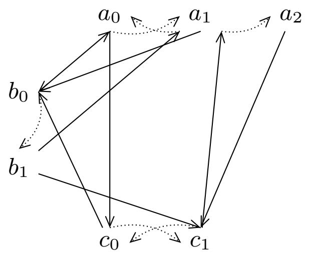
We outline an algorithm for finding the n-cliques of a Hitless Graph $\overline{\mathcal{PH}} = (V, E)$ where $n = |\Sigma|$.

Thanks to Prop. 1, we split V into n partitions corresponding to each process: $V = \cup_{a \in \Sigma} V_a$, $V_a \subseteq L_a$. If one of this partition is empty, there can not be n-cliques as it requires to have at least one vertex in each partition. We will assume $V_a \neq \emptyset$, $\forall a \in \Sigma$.

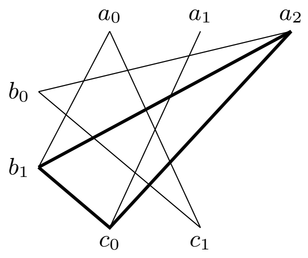
For each partition $a \in \Sigma$ and each vertex $a_i \in V_a$, we define $E^b_{a_i} = \{b_j \in V_b \mid (a_i, b_j) \in E\}$ for each other partition $b \in \Sigma, b \neq a$, the set of vertices in V_b related to a_i. If there exists $b \in \Sigma$ such that $E^b_{a_i} = \emptyset$, the vertex a_i is removed from candidates as it can not belong to a n-clique. Finally, we set $E^a_{a_i} = \{a_i\}$.

Once this pruning is performed, we enumerate potential n-cliques. To reduce this enumeration, we choose the partition a sharing the least number of edges. For each vertex $a_i \in V_a$ we test for all $s \in \prod_{b \in \Sigma} E^b_{a_i}$ if s is a clique of $\overline{\mathcal{PH}}$.

For instance in Figure 3(b), a_1 is removed from the Hitless Graph ($E^b_{a_1} = \emptyset$), the partition associated to c is chosen (involved into only 4 edges), two states are tested: $a_0 b_0 c_1$ and $a_2 b_1 c_0$ and the latter reveals to be the only 3-clique.



(a) Hypergraph



(b) Hitless Graph

Fig. 3. A Process Hitting represented by its Hypergraph (a) and its Hitless Graph (b). The Hitless Graph contains only one 3-clique between a_2, b_1 and c_0 (thick lines): this is the only stable state of this system.

4 From Process Hitting to René Thomas' Parameters

A René Thomas' discrete parameter gives the attractor levels for a gene when its regulators are in a given configuration. Many frameworks and tools dedicated to the study of GRNs take the full set of René Thomas' parameters as essential input [1,9,11]. In this section, we give a formal method to infer René Thomas' parameters for a GRN modelled in the Process Hitting framework.

Let (Γ, E_+, E_-) be a GRN graph. A René Thomas' parameter $K_{a,A,B}$, $a \in \Gamma, A \cup B \subseteq \Gamma,\ A \cap B = \emptyset$, gives the interval of attracting levels for a when genes in A are activating a and genes in B are inhibiting a. In this configuration, if the level of a is in $K_{a,A,B}$, then it will never change; otherwise the level of a will tend to a level in $K_{a,A,B}$.

Let $(\Sigma, L, \mathcal{H})$ be a Process Hitting where sorts are standing either for genes or for cooperative sorts, i.e. $\Sigma = \Gamma \cup \{\sigma^1, \dots, \sigma^u\}$ with $\forall 1 \leq v \leq u,\ \sigma^v \subset \Gamma$. Let $K_{a,A,B}$ be the René Thomas' parameter to infer. For each sorts $b \in A \cup B$, its context $C^b_{a,A,B}$ is defined as the subset of processes L_b imposed by the René Thomas' parameter: if $b \in A$ (resp. B) only processes corresponding to positive (resp. negative) effective levels (Def. 2) are allowed. For each process $b \in \Gamma$ not regulating a (i.e. $b \notin A \cup B$), its context $C^b_{a,A,B}$ is simply L_b (7). The context $C^\sigma_{a,A,B}$ for cooperative sorts $\sigma \in \{\sigma^1, \dots, \sigma^u\}$ is the set of states of its representatives in their context (8).

$$\forall b \in \Gamma,\ C^b_{a,A,B} = \begin{cases} levels_+(b,a) & \text{if } b \in A, \\ levels_-(b,a) & \text{if } b \in B, \\ L_b & \text{otherwise.} \end{cases} \tag{7}$$

$$\forall \sigma \in \{\sigma^1, \dots, \sigma^u\},\ C^\sigma_{a,A,B} = \{\sigma_\varsigma \mid \forall \varsigma \in \prod_{b \in \sigma} C^b_{a,A,B}\}\ . \tag{8}$$

We denote $\mathcal{H}_{a,A,B}$ the subset of the set of actions $\mathcal{H}$ on a that may be performed by processes of the context of any sort (9). A process of sort a is reachable if it belongs to the context of a or is the result of any action in $\mathcal{H}_{a,A,B}$. The set of such processes is noted $L^?_{a,A,B}$ (10). The set of reachable processes of sort a not hit by any other processes is noted $L^*_{a,A,B}$ (11). Thus, as long as present processes are in the context of their sort, if a process of sort a is in $L^*_{a,A,B}$, it will not be bounced. In this way, $L^*_{a,A,B}$ is called the set of *focal processes* of a.

$$\mathcal{H}_{a,A,B} = \{b_i \rightarrow a_j \Rsh a_k \in \mathcal{H} \mid b_i \in C^b_{a,A,B} \wedge a_j \in C^a_{a,A,B}\} \tag{9}$$

$$L^?_{a,A,B} = C^a_{a,A,B} \cup \{a_k \mid \forall b_i \rightarrow a_j \Rsh a_k \in \mathcal{H}_{a,A,B}\} \tag{10}$$

$$L^*_{a,A,B} = L^?_{a,A,B} \setminus \{a_j \mid \forall b_i \rightarrow a_j \Rsh a_k \in \mathcal{H}_{a,A,B}\} \ . \tag{11}$$

Finally, we check that the focal processes are attractors, i.e. all actions $\mathcal{H}_{a,A,B}$ make processes of sort a bounce in the direction of the focal processes. If such a condition is satisfied, the focal processes correspond to the value of the requested René Thomas' parameter. We point up that all these operations are linear with the number of actions in the Process Hitting.

Condition 1 (Focal processes are attractors)
$\forall b_i \rightarrow a_j \Rsh a_k \in \mathcal{H}_{a,A,B},\ \forall a_f \in L^*_{a,A,B},\ |f-k| < |f-j|$.

Property 3 If $L^*_{a,A,B}$ satisfies Cond. 1, $L^*_{a,A,B}$ is an interval.

Proof If $L^*_{a,A,B} = \{a_f, \dots, a_{f'}\}$ is not an interval, there exists $b_i \rightarrow a_j \Rsh a_k \in \mathcal{H}_{a,A,B}$ such that $f < j < f'$. If Cond. 1 applies, we have $|f-k| < |f-j| \Rightarrow k < j \Rightarrow |f'-k| > |f'-j|$ which contradicts Cond. 1.

Theorem 2. *If $L^*_{a,A,B} \neq \emptyset$ and Cond. 1 holds, then $K_{a,A,B} = L^*_{a,A,B}$.*

Proof. By construction of $L^*_{a,A,B}$ and application of Cond. 1 and Prop. 3, it immediately appears that if $L^*_{a,A,B} \neq \emptyset$, it is the set of attracting levels for a.

Consequently, there might exist configurations without any correspondence with René Thomas' parameters. First, $L^*_{a,A,B} = \emptyset$ means the gene a is unstable in the fixed context, i.e. its level is changing forever. Second, Cond. 1 is violated when there exists opposite focal processes, i.e. the fate of a is not deterministic.

One of the main reasons for non-determinism of Process Hitting is the absence of cooperativity between hits to a same target which may then independently be bounced to both higher and lower processes. We leave as an open question the problem to know whether such unstable and/or non-deterministic dynamics are biologically relevant.

5 Temporal and Stochastic Parameters

Further dynamics refinements may be achieved by taking into account the temporal and stochastic dimensions of biological reactions. On the one hand, we may consider the probability of a reaction to occur at a given state. By introducing stochastic parameters into discrete models, we aim at computing the probability of observing an expected behaviour. On the other hand, because they are faster, some reactions always apply before others. By introducing temporal parameters into discrete models, we aim at reducing their dynamics to match such behaviours.

5.1 From Process Hitting to the Stochastic π-Calculus

The Stochastic π-Calculus [20] adds the capability to attach *use rates* to channels and internal actions of the π-Calculus. This gives a natural introduction for temporal and stochastic aspects in our Process Hitting framework.

A use rate controls both the duration and the probability of a reaction (communication on a channel or internal action). It is associated to a probability distribution for firing reaction along the time. The usual probability distribution is the exponential one, allowing efficient simulations through a Gillespie-like algorithm [12,21]. This is the one we consider for the rest of this paper.

The probability along time t of firing a reaction with use rate r is given by $F(t) = 1 - e^{-rt}$. The average duration of this reaction is r^{-1} with a variance of r^{-2}. When x reactions are possible having use rates of $r_1, \ldots, r_x$ respectively, the probability that the y^{th} reaction is fired is given by $\frac{r_y}{r_1+\cdots+r_x}$.

The translation of Process Hitting $(\Sigma, L, \mathcal{H})$ into the Stochastic π-Calculus is the same as the one presented in Section 2.4. Additionally, to each channel γ_α, or internal action τ_α, a use rate r_α is attached.

5.2 Stochasticity Absorption

Use rates are both temporal and stochastic parameters. Nonetheless, these two aspects are closely tied: the lower a use rate is, the higher the variance around its mean duration is. We introduce a *stochasticity absorption factor* to control this variance to favour either the stochastic or the temporal behaviour of an action.

We propose to replace the exponential distribution of a reaction with a rate r by the distribution of the sum of sa random variables each having an exponential distribution of parameter $r.sa$. The resulting probability distribution is also known as the *Erlang distribution*. The average duration is unchanged: $(r.sa)^{-1}sa = r^{-1}$, but the variance is divided by sa: $(r.sa)^{-2}sa = r^{-2}sa^{-1}$. sa stands for the stochasticity absorption factor. Based on the previously presented translation from the Process Hitting to the Stochastic π-Calculus, we supply a simple method to achieve this stochasticity absorption factor which do not require to adapt simulation algorithms based on the memoryless property of the exponential law [22].

Basically, to each channel γ_α, or internal action τ_α, a use rate r_α and a stochasticity absorption factor sa_α is attached. To each component α of the sum defined by the π-Calculus process A_i in the expressions (2),(3), a counter c_α is attached, initially, $c_\alpha = 1$. This counter is given as a parameter for A_i. As long as this counter is not equal to sa_α, A_i is restarted and the counter is incremented by one. When the counter reaches the stochasticity absorption factor value, the next process replaces A_i, having all its counters reset to 1. Let $(\Sigma, L, \mathcal{H})$ be a Process Hitting, for each process a_i of $\mathcal{PH}$, a π-Calculus process A_i is defined as follows.

$$\begin{aligned} A_i(\tilde{c}) ::= & \sum_{\substack{\alpha = a_i \to b_j \,\upharpoonright\, b_k \in \mathcal{H} \\ a \neq b}} !\gamma_\alpha . A_i(\tilde{c}) \\ & + \sum_{\substack{\alpha = b_j \to a_i \,\upharpoonright\, a_k \in \mathcal{H} \\ a \neq b}} [c_\alpha < sa_\alpha]?\gamma_\alpha . A_i(\tilde{c}[c_\alpha + 1]) + [c_\alpha = sa_\alpha]?\gamma_\alpha . A_k(\tilde{1}) \\ & + \sum_{\alpha = a_i \to a_i \,\upharpoonright\, a_k \in \mathcal{H}} [c_\alpha < sa_\alpha]\tau_\alpha . A_i(\tilde{c}[c_\alpha + 1]) + [c_\alpha = sa_\alpha]\tau_\alpha . A_k(\tilde{1}) \end{aligned}$$

where $\hat{c} = c_1, \ldots, c_n$ with $n = |\{b_j \to a_i \upharpoonright a_k \in \mathcal{H}\}|$. $\hat{c}[c_\alpha + 1] = c_1, \ldots, c_\alpha + 1, \ldots, c_n$. $A_k(\tilde{1})$ is an abbreviation for the recursive call to A_k with all parameters set to 1. $[cond]\pi.P$ stands for an action π enabled only when *cond* is satisfied.

6 Applications

6.1 Metazoan Segmentation

In this section, we illustrate our method and its benefits on a case study in which our aim is to control the final state of the corresponding GRN. The GRN we chose has been established *in silico* by François et al. [23] but in a differential equations framework. It aims at generalizing a common motif present in biological segmentation networks such as the *Drosophila*.

The GRN (Figure 4(a)) is composed of three genes. A wavefront gene f activates the gap-gene a whose products are responsible or stripes. Gene f also activates a gene c whose products repress the gene a. The auto-inhibition of c generalizes a chain of repressors on a. The apparition of stripes has to be regular. We attach to each gene two processes representing their qualitative levels (missing or present) — for instance c_0 (absence) and c_1 (presence) are processes for c. When f switches off, c goes to process c_0 and a has two fates, ending either at process a_0 or a_1. We are interested in controlling the final process for a.

The Process Hitting for generalised dynamics of the GRN (Section 2) is computed first. Figure 4(b) shows its hypergraph. The specification of dynamics implies two cooperative hits in the Process Hitting: first, c_0 needs products of f to bounce to process c_1; second, expression of a only increases if both f activates it (i.e. process f_1 is present) and c does not inhibit it (i.e. c_0 is present). Consequently, we create a cooperative sort fc reflecting the state of f and c (Section 3.1) and replace the independent hits from c_0 and f_1 to c_0 and a_0 by hits from fc_{10}. The resulting Process Hitting is represented in Figure 5(a).

By looking at the Hitless Graph of the Process Hitting (Figure 5(b)), only one stable state is present: $f_0 c_0 fc_{00} a_0$. The stability of the state $f_0 c_0 fc_{00} a_1$ is controlled by the absence of inhibition by f_0 on a_1. By removing the action

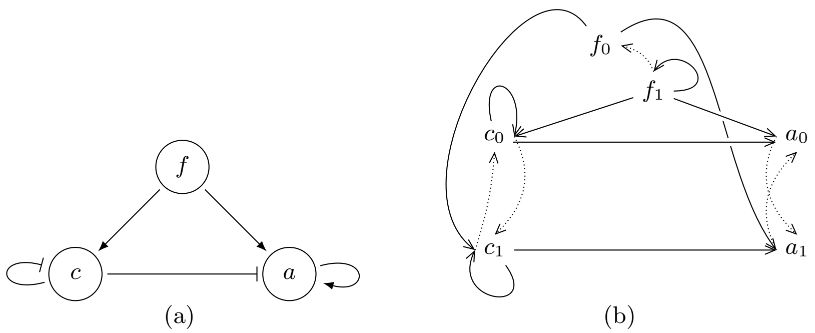


Fig. 4. The starting Gene Regulatory Network graph (a), arrow-ended edges represent the positive regulations, and bar-ended edges the negative ones. All regulation thresholds are 1. The Process Hitting (b) for its generalized dynamics. Cooperativity between f_1 and c_0 on a_0 and c_0 will be applied in the same way as in example. 1.

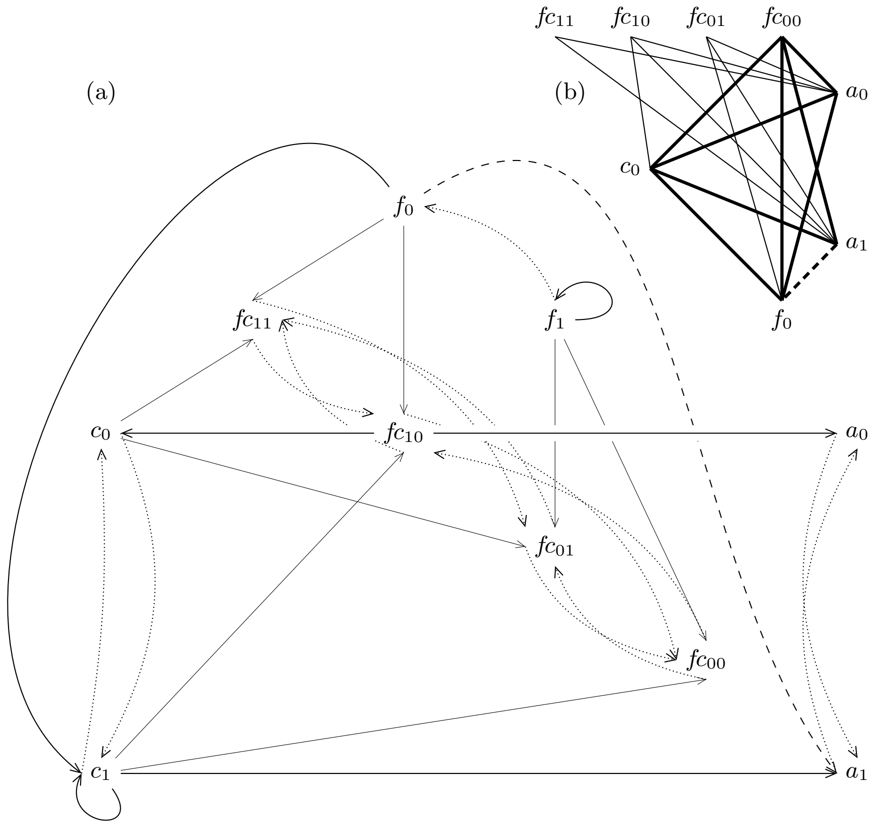


Fig. 5. The final Process Hitting (a) resulting from the refinement of the generalized dynamics depicted on Fig. 4(b). Cooperativity between f_1 and c_0 on a_0 and c_0 has been built in the same way as in example 1. Absence of hit from f_0 to a_1 (dashed lines) controls the presence of the relation between f_0 and a_1 in the Hitless Graph (b). If such a relation exists, two 4-cliques are presents: $c_0 f_0 fc_{00} a_0$ and $c_0 f_0 fc_{00} a_1$ (thick lines).

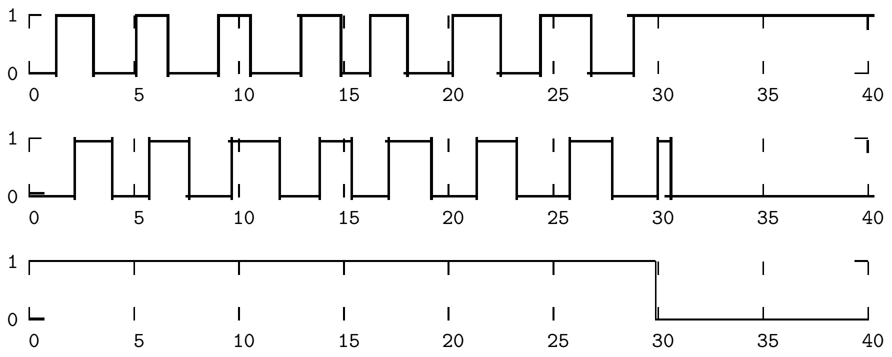


Fig. 6. Simulation of the Process Hitting for segmentation: evolution of the expressions of the gap-gene a (top), the autonomous clock c (middle) and the wavefront f

$f_0 \rightarrow a_1 \upharpoonright a_0$ from the Process Hitting, we make the state $f_0 c_0 fc_{00} a_1$ stable. The full set of corresponding René Thomas' parameters for the genes a and c is inferred by applying the method depicted in Section 4. We get:

$$\begin{array}{lll}
K_{a,\emptyset,\{a,c,f\}} = 0 & K_{a,\{a,c\},\{f\}} = 1 & K_{c,\emptyset,\{c,f\}} = 0 \\
K_{a,\{a\},\{c,f\}} = 0 & K_{a,\{a,f\},\{c\}} = 0 & K_{c,\{c\},\{f\}} = 0 \\
K_{a,\{c\},\{a,f\}} = 0 & K_{a,\{c,f\},\{a\}} = 1 & K_{c,\{f\},\{c\}} = 0 \\
K_{a,\{f\},\{a,c\}} = 0 & K_{a,\{a,c,f\},\emptyset} = 1 & K_{c,\{c,f\},\emptyset} = 1 \ .
\end{array}$$

We are interested in controlling the final process of sort a — either a_0 or a_1 — when f goes down to f_0. Looking at the Process Hitting hypergraph on Figure 5(a) and considering f_0 is present, we deduce that the more c_1 is present, the more a_1 may be hit by c_1 to bounce to a_0; similarly, the more fc_{10} is present, the more a_0 may be hit by it to bounce to a_1. We tune actions only triggered by f_0: we reduce the presence of c_1 by increasing the rate of the action $f_0 \rightarrow c_1 \upharpoonright c_0$ and extend the presence of fc_{10} by reducing the rate of $f_0 \rightarrow fc_{10} \upharpoonright fc_{00}$. This leads to an increase of the probability for a to bounce to process a_1.

Finally, to obtain regular stripes, we set a high stochasticity absorption factor to actions responsible of the bounces of processes of sort c and a when f_1 is present. Figure 6 plots the evolution of the genes a,c and f during a simulation under SPiM [24] of the Process Hitting illustrated by Figure 5(a) with initial state $f_1 c_0 fc_{10} a_0$ and a fast rate for the action $f_0 \rightarrow c_1 \upharpoonright c_0$ compared to the rate of $c_1 \rightarrow a_1 \upharpoonright a_0$. The rate values have been arbitrarily choosen and respect the infered relations between them. Appendix B.1 details the Process Hitting used for the simulation.

From the obtained simulation trace, we observe that f_0 hits c_1 before c_1 had time to hit a_1: the final state is then $f_0 c_0 fc_{00} a_1$.

Thanks to the Process Hitting framework, it has been easy to build a qualitative model of the biological system by refining the generalised dynamics of the GRN. Using a simple reasoning on the Process Hitting structure, relation between regulation delays to favour a final state have been infered. These results are coherent with those obtained using differential equations as done in [23].

6.2 ERBB Receptor-Regulated G1/S Transition

The aim of this section is to demonstrate the scalability of the refining approach on Process Hittings modelling large GRNs.

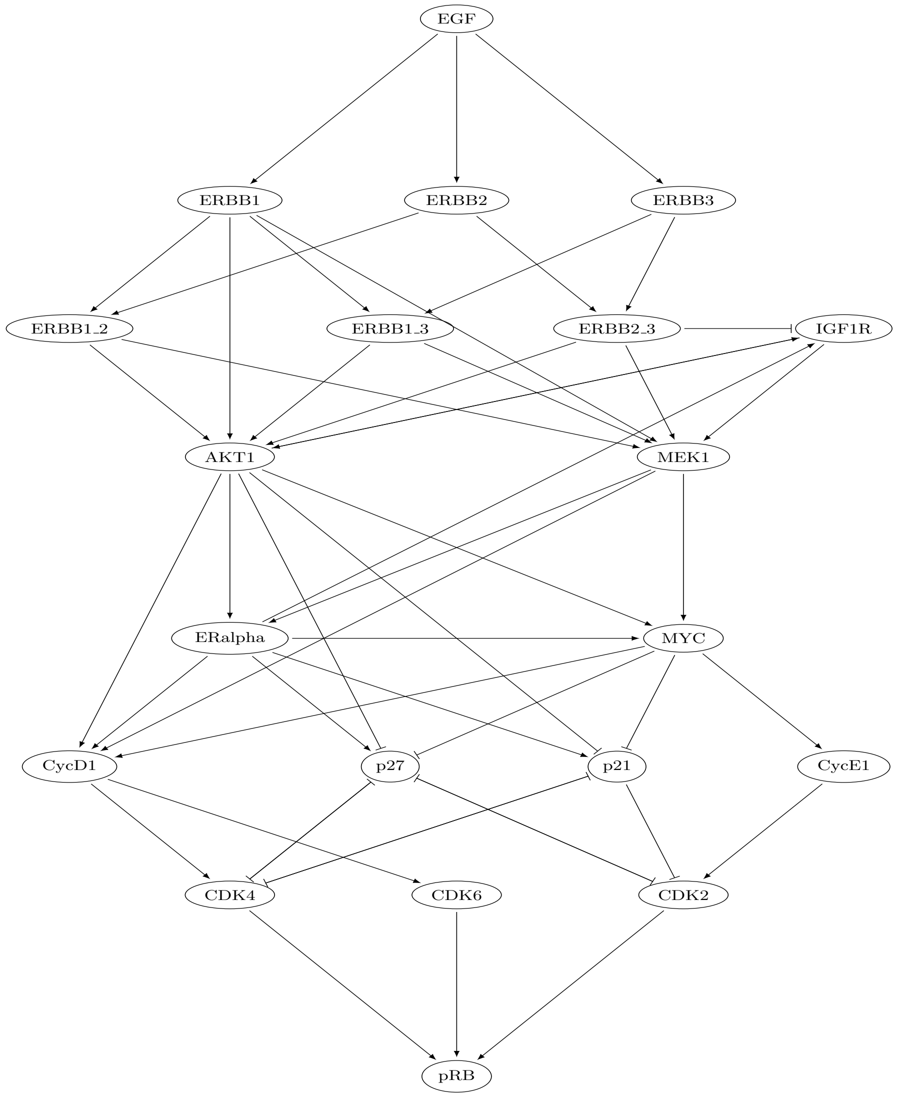


Fig. 7. ERBB receptor-regulated G1/S transition GRN reproduced from [25, Figure 3]

The selected GRN relates regulations between 20 genes. This GRN models the *ERBB receptor-regulated G1/S transition* involved in the breast cancer. It has been extracted from published data by Sahin et al. [25] and is reproduced in Figure 7. This network acts as an activation cascade for the gene *pRB* which controls the G1/S transition involved in cell divisions. The gene *EGF* then acts as an input of this cascade: when expressed, *pRB* will be potentially activated. Based on the literature, Sahin et al. have also established a set of logical rules controlling the activation of the various genes present in the network.

Starting from the GRN, its generalised dynamics expressed in Process Hitting is first computed. Then, cooperations between the different sorts are built from the logical rules. The Process Hitting obtained contains 670 actions and stands for 2^{64} ($\approx 2.10^{19}$) states that has hopefully not be built. This model is fully detailled in Appendix B.2.

The computation of all the stable states present in the dynamics is done using the algorithm sketched by Section 3.2. It results in 5 stables states (also detailled in the appendix) that are computed in less than one second.

It is worth noticing that no assumption is made on the initial state of the system. *All* the stable states of the model are computed. This is a major difference with the approach presented in [25] where only dynamics starting from a fixed state can be studied.

7 Conclusion

We introduced the Process Hitting framework for modelling qualitative dynamics of GRNs with temporal and stochastic features. Temporal and stochastic parameters determine probabilities, durations and temporal variance of reactions in the model. We exhibited a direct translation from Process Hitting to the Stochastic π-Calculus. Detection of stable states and inference of René Thomas' parameters for dynamics derive from this framework. The methods we offered work by successive refinements of generalised dynamics for GRNs, by specifying both the cooperativity between genes and the expected stable states. We illustrated this method by inferring temporal parameters for the dynamics of a GRN generalizing metazoan segmentation processes (with the aim of controlling its final state). The scalability of the presented approach has been experimented on a Process Hitting modelling a GRN composed of 20 genes and computing its stable states.

The Process Hitting brings a formal framework for progressively adding knowledge of the dynamics of a GRN by refining an abstracted behaviour. The compositionality of the framework and the presence of particular structure patterns lead to scalable methods for dynamics analyses (stable states, René Thomas' parameters, etc.). Mainly, thanks to these Process Hitting patterns, it becomes possible to perform a local analysis, which has the major advantage to prevent us from exploring the full state and parameter space.

In future works, we aim at identifying more Process Hitting patterns leading to the emergence of particular behaviours (e.g. oscillations) and especially hybrid patterns coupling both discrete structure and continuous temporal and stochastic parameters. The verification of Process Hittings could be performed by using a translation into Petri Nets or into PRISM. Translating Process Hittings into more sophisticated process aglebras (Beta Workbench, Bio-PEPA, etc.) may also be of interest. Finally, techniques have to be developed to infer automatically temporal and stochastic parameters of Process Hittings modelling GRNs.

Supplementary Material

The Process Hitting compiler to SPiM, a stable states computer and presented models are available at the following URL:
http://www.irccyn.ec-nantes.fr/˜pauleve/processhitting-refining.tar.gz

References

1. Richard, A., Comet, J.P., Bernot, G.: Formal Methods for Modeling Biological Regulatory Networks. In: Modern Formal Methods and Applications, pp. 83–122 (2006)
2. Danos, V., Feret, J., Fontana, W., Harmer, R., Krivine, J.: Rule-Based Modelling of Cellular Signalling. In: Caires, L., Li, L. (eds.) CONCUR 2007. LNCS, vol. 4703, pp. 17–41. Springer, Heidelberg (2007)
3. Priami, C., Regev, A., Shapiro, E., Silverman, W.: Application of a stochastic name-passing calculus to representation and simulation of molecular processes. Inf. Process. Lett. 80(1), 25–31 (2001)
4. Kuttler, C., Niehren, J.: Gene regulation in the pi calculus: Simulating cooperativity at the lambda switch. In: Priami, C., Ingólfsdóttir, A., Mishra, B., Riis Nielson, H. (eds.) Transactions on Computational Systems Biology VII. LNCS (LNBI), vol. 4230, pp. 24–55. Springer, Heidelberg (2006)
5. Blossey, R., Cardelli, L., Phillips, A.: A compositional approach to the stochastic dynamics of gene networks. In: Priami, C., Cardelli, L., Emmott, S. (eds.) Transactions on Computational Systems Biology IV. LNCS (LNBI), vol. 3939, pp. 99–122. Springer, Heidelberg (2006)
6. Popova-Zeugmann, L., Heiner, M., Koch, I.: Time petri nets for modelling and analysis of biochemical networks. Fundamenta Informaticae 67(1), 149–162 (2005)
7. Heiner, M., Gilbert, D., Donaldson, R.: Petri Nets for Systems and Synthetic Biology. In: Formal Methods for Computational Systems Biology, pp. 215–264 (2008)
8. Rizk, A., Batt, G., Fages, F., Soliman, S.: On a Continuous Degree of Satisfaction of Temporal Logic Formulae with Applications to Systems Biology. In: Computational Methods in Systems Biology, pp. 251–268 (2008)
9. Siebert, H., Bockmayr, A.: Incorporating Time Delays into the Logical Analysis of Gene Regulatory Networks. In: Computational Methods in Systems Biology, pp. 169–183 (2006)
10. Alur, R., Belta, C., Kumar, V., Mintz, M., Pappas, G.J., Rubin, H., Schug, J.: Modeling and analyzing biomolecular networks. Computing in Science and Engineering 4(1), 20–31 (2002)
11. Ahmad, J., Bernot, G., Comet, J.P., Lime, D., Roux, O.: Hybrid modelling and dynamical analysis of gene regulatory networks with delays. Complexus 3(4), 231–251 (2006)
12. Phillips, A., Cardelli, L.: Efficient, correct simulation of biological processes in the stochastic pi-calculus. In: Calder, M., Gilmore, S. (eds.) CMSB 2007. LNCS (LNBI), vol. 4695, pp. 184–199. Springer, Heidelberg (2007)

13. Dematte, L., Priami, C., Romanel, A.: The Beta Workbench: a computational tool to study the dynamics of biological systems. Brief Bioinform., bbn023 (2008)
14. Ciocchetta, F., Hillston, J.: Bio-pepa: A framework for the modelling and analysis of biological systems. Theoretical Computer Science 410(33-34), 3065–3084 (2009)
15. Hinton, A., Kwiatkowska, M., Norman, G., Parker, D.: PRISM: A tool for automatic verification of probabilistic systems. In: Hermanns, H. (ed.) TACAS 2006. LNCS, vol. 3920, pp. 441–444. Springer, Heidelberg (2006)
16. Norman, G., Palamidessi, C., Parker, D., Wu, P.: Model checking probabilistic and stochastic extensions of the π-calculus. IEEE Transactions on Software Engineering 35(2), 209–223 (2009)
17. Bernot, G., Cassez, F., Comet, J.P., Delaplace, F., Müller, C., Roux, O.: Semantics of biological regulatory networks. Electronic Notes in Theoretical Computer Science 180(3), 3–14 (2007)
18. Milner, R.: A calculus of mobile processes, parts. I and II. Information and Computation 100, 1–77 (1992)
19. Bernot, G., Comet, J.P., Khalis, Z.: Gene regulatory networks with multiplexes. In: European Simulation and Modelling Conference Proceedings, pp. 423–432 (October 2008)
20. Priami, C.: Stochastic π-Calculus. The Computer Journal 38(7), 578–589 (1995)
21. Gillespie, D.T.: Exact stochastic simulation of coupled chemical reactions. The Journal of Physical Chemistry 81(25), 2340–2361 (1977)
22. Priami, C.: Stochastic π-calculus with general distributions. In: Proc. of the 4th Workshop on Process Algebras and Performance Modelling, CLUT, pp. 41–57 (1996)
23. Francois, P., Hakim, V., Siggia, E.D.: Deriving structure from evolution: metazoan segmentation. Mol. Syst. Biol. 3 (2007)
24. Phillips, A.: SPiM, `http://research.microsoft.com/~aphillip/spim`
25. 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 3(1) (2009)

A Process Hitting Related Tools

This appendix briefly presents currently available implementations of tools manipulating Process Hittings. They are available at the following URL:

http://www.irccyn.ec-nantes.fr/˜pauleve/processhitting-refining.tar.gz

Implemented in the OCAML language, these tools have a command-line user interline.

A.1 Process Hitting Specification

A basic language has been setup to specify Process Hitting using a text file. Main features are presented here, more details can be found in the provided archive. Full examples of Process Hitting specifications are given in Appendix B.

Sort definition. A sort is declared by giving the process with the highest rank (i.e. a_{l_a} for the sort a, with the notations used in Section 2).

```
process a X
```

Action specification. An action $a_i \to b_j \upharpoonright b_k$ is added by the following instruction:

```
a i -> b j k @ rate ~ absorption
```

Generalised dynamics of GRNs. The **GRN** macro computes the generalised dynamics of the specified GRN according to Section 2.5. An activating (resp. inhibiting) edge $a \xrightarrow{X} b$ from gene a to gene b active with a threshold X is noted as `a X -> + b` (resp. `a X -> - b`).

Hereafter is an instance of the use of the **GRN** macro for the GRN having activating edges $a \xrightarrow{1} b$ and $a \xrightarrow{2} a$, and inhibiting edge $b \xrightarrow{1} b$.

```
GRN([a 1 -> + b; b 1 -> - a; a 2 -> + a ])
```

Refinement: cooperations. The **COOPERATIVITY**`([a1;...;an] -> b j k, [s1;...;sp])` macro creates a cooperative sort $\sigma = \{a^1, \ldots, a^n\}$ for the bounce $b_j \upharpoonright b_k$. This cooperation is effective for every state $s_1, \ldots, s_p$.

The following instruction creates a cooperativity between sorts a and b to bounce process c_0 to c_1 only if $a_1 b_0$ or $a_0 b_1$ are present.

```
COOPERATIVITY([a;b] -> c 0 1, [[1;0];[0;1]])
```

Refinement: action removing An action $a_i \to b_j \upharpoonright b_k$ can be deleted by using the macro **RM**:

```
RM({a i -> b j k})
```

A.2 Compiler to SPiM

This tool translates a Process Hitting specification into a SPiM model according to Section 5.2.

```
phc -spim <model.ph> <output.spi>
```

A.3 Stable States Listing

The stable states of a Process Hitting are determined used an implementation of the algorithm sketched in Section 3.2. This algorithm computes the n-cliques of the hitless graph for the given Process Hitting, where n is the number of sorts. The efficiency of this computation heavily relies on the order of sorts when building cliques. Currently, basic heuristics are used to select the order of the sorts. More sophisticated analyses may conduct to dramatically improve the efficiency of this algorithm.

This tool is compiled into the executable `ph-stable-states` and takes as argument the filename of the Process Hitting specification :

```
ph-stable-states <model.ph>
```

B Process Hitting Examples

This appendix details the Process Hittings used in Section 6. They are specified in the language presented in the previous appendix.

B.1 Metazoan Segmentation

The following Process Hitting models the metazoan segmentation presented in Section. 6.1. Actions are specified separately and rates have been assigned to values matching the relations infered in the application case study. The `directive sample` and **initial_state** instructions are of use for SPiM only. A result of the execution of this Process Hitting translated into SPiM is given by Figure 6.

```
directive sample 40.

process a 1 process c 1 process f 1
process fc 3 (* cooperative sort {f,c} *)
c 1 -> fc 0 1 @5.
c 1 -> fc 2 3 @5.
c 0 -> fc 1 0 @10.
c 0 -> fc 3 2 @5.
f 1 -> fc 0 2 @10.
f 1 -> fc 1 3 @10.
f 0 -> fc 2 0 @0.1
f 0 -> fc 3 1 @0.1

(* actions on c *)
fc 2 -> c 0 1 @0.5~50 (* only if (f1,c0) *)
c 1 -> c 1 0 @0.5~50
(* actions on a *)
fc 2 -> a 0 1 @1.~50 (* only if (f1,c0) *)
c 1 -> a 1 0 @1.~50
(* actions on f *)
f 1 -> f 1 0 @0.034~100 (* auto-off *)
f 0 -> c 1 0 @0.1

initial_state f 1, c 0, a 0
```

B.2 ERBB Receptor-Regulated G1/S Transition

The following Process Hitting results from the case study presented in Section 6.2. It starts by specifying the GRN depicted by Figure 7 to compute its generalised dynamics. The logical rules setup by Sahin et al. [25] are then applied by using sorts cooperativity.

This Process Hitting contains 670 actions and 2^{64} ($\approx 2.10^{19}$) states. Only 5 stable states exist and are determined in less than a second using the tool presented in the previous appendix.

Below is the list of stable states present in the Process Hitting. For each stable state, only genes at level 1 are written.

- AKT1, CDK2, CDK4, CDK6, CycD1, CycE1, EGF, ERBB1, ERBB1_2, ERBB1_3, ERBB2, ERBB2_3, ERBB3, ERalpha, MEK1, MYC, pRB.
- AKT1, CDK2, CDK4, CDK6, CycD1, CycE1, ERalpha, IGF1R, MEK1, MYC, pRB.
- AKT1, CDK2, CycE1, EGF, ERBB1, ERBB1_2, ERBB1_3, ERBB2, ERBB2_3, ERBB3, ERalpha, MEK1, MYC.
- AKT1, CDK2, CycE1, ERalpha, IGF1R, MEK1, MYC.
- $\emptyset$ (all genes have level 0).

```
process AKT1 1
process CDK2 1 process CDK4 1 process CDK6 1
process CycD1 1 process CycE1 1
process EGF 1 process ERalpha 1
process ERBB1 1 process ERBB1_2 1 process ERBB1_3 1
process ERBB2 1 process ERBB2_3 1 process ERBB3 1
process IGF1R 1 process MEK1 1 process MYC 1
process p21 1 process p27 1
process pRB 1

GRN([
    ERBB2_3 1 -> + AKT1; ERBB2_3 1 -> + MEK1; ERBB2_3 1 -> - IGF1R;
    ERBB2 1 -> + ERBB2_3; ERBB2 1 -> + ERBB1_2; ERBB3 1 -> + ERBB2_3;
    ERBB3 1 -> + ERBB1_3;
    CycE1 1 -> + CDK2;
    MEK1 1 -> + CycD1; MEK1 1 -> + ERalpha; MEK1 1 -> + MYC;
    CDK4 1 -> + pRB; CDK4 1 -> - p21; CDK4 1 -> - p27;
    ERalpha 1 -> + CycD1; ERalpha 1 -> + IGF1R; ERalpha 1 -> + p21;
    ERalpha 1 -> + MYC; ERalpha 1 -> + p27;
    MYC 1 -> + CycE1; MYC 1 -> - p21; MYC 1 -> + CycD1; MYC 1 -> - p27;
    CDK6 1 -> + pRB;
    ERBB1 1 -> + ERBB1_2; ERBB1 1 -> + ERBB1_3; ERBB1 1 -> + AKT1;
    ERBB1 1 -> + MEK1;
    IGF1R 1 -> + AKT1; IGF1R 1 -> + MEK1;
    ERBB1_3 1 -> + AKT1; ERBB1_3 1 -> + MEK1;
    p27 1 -> - CDK2; p27 1 -> - CDK4;
    CDK2 1 -> - p27; CDK2 1 -> + pRB;
    p21 1 -> - CDK2; p21 1 -> - CDK4;
    CycD1 1 -> + CDK4; CycD1 1 -> + CDK6;
    EGF 1 -> + ERBB1; EGF 1 -> + ERBB2; EGF 1 -> + ERBB3;
    AKT1 1 -> + CycD1; AKT1 1 -> + MYC; AKT1 1 -> - p27; AKT1 1 -> + ERalpha;
    AKT1 1 -> + IGF1R; AKT1 1 -> - p21;
    ERBB1_2 1 -> + AKT1; ERBB1_2 1 -> + MEK1;
])

COOPERATIVITY([ERBB1;ERBB2] -> ERBB1_2 0 1, [[1;1]])
COOPERATIVITY([ERBB1;ERBB3] -> ERBB1_3 0 1, [[1;1]])
COOPERATIVITY([ERBB2;ERBB3] -> ERBB2_3 0 1, [[1;1]])

COOPERATIVITY([ERBB2_3;AKT1] -> IGF1R 0 1, [[0;1]])
COOPERATIVITY([ERBB2_3;ERalpha] -> IGF1R 0 1, [[0;1]])
COOPERATIVITY([AKT1;ERalpha] -> IGF1R 1 0, [[0;0]])

COOPERATIVITY([AKT1;MEK1] -> ERalpha 1 0, [[0;0]])
COOPERATIVITY([AKT1;MEK1;ERalpha] -> MYC 1 0, [[0;0;0]])
COOPERATIVITY([ERBB1;ERBB1_2;ERBB1_3;ERBB2_3;IGF1R] -> AKT1 1 0,
     [[0;0;0;0;0]])
COOPERATIVITY([ERBB1;ERBB1_2;ERBB1_3;ERBB2_3;IGF1R] -> MEK1 1 0,
     [[0;0;0;0;0]])
COOPERATIVITY([CycE1;p21;p27] -> CDK2 0 1, [[1;0;0]])
COOPERATIVITY([CycD1;p21;p27] -> CDK4 0 1, [[1;0;0]])
COOPERATIVITY([ERalpha;MYC;AKT1;MEK1] -> CycD1 0 1, [[1;1;1;0];[1;1;0;1]])
COOPERATIVITY([AKT1;MEK1] -> CycD1 1 0, [[0;0]])
COOPERATIVITY([ERalpha;AKT1;MYC;CDK4] -> p21 0 1, [[1;0;0;0]])
COOPERATIVITY([ERalpha;CDK4;CDK2;AKT1;MYC] -> p27 0 1, [[1;0;0;0;0]])
COOPERATIVITY([CDK2;CDK4;CDK6] -> pRB 0 1, [[0;1;1];[1;1;1]])
RM({CDK2 0 -> pRB 1 0})
RM({EGF 1 -> EGF 1 0}) (* prevent self-degradation (input) *)
```