Available online at www.sciencedirect.com

SciVerse ScienceDirect

Electronic Notes in Theoretical Computer Science 284 (2012) 3–22

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

Control Flow Analysis of Generalised Boolean Networks

Chiara Bodei[1], Linda Brodo[2], Davide Chiarugi[3]

[1] *Dipartimento di Informatica, Università di Pisa,* chiara@di.unipi.it

[2] *Dipartimento di Scienze dei Linguaggi, Università di Sassari,* brodo@uniss.it

[2] *Dipartimento di Scienze Matematiche e Informatiche, Università di Siena,* chiarugi3@unisi.it

Abstract

Generalised Boolean Networks are a well known qualitative model used to analyse the evolution of genetic networks as well as generic biological pathways. Despite the qualitative abstraction due to the few threshold concentration values considered for each biological element in the model, the complexity of the execution of a Generalised Boolean model could be non trivial. In this paper, we propose a tailored process algebra, called **Sim**-π_n, reminiscent of the π-calculus to model GBNs. We further apply the Control Flow Analysis methodology to the resulting computational model for making static (and therefore less computationally expensive) predictions on the dynamical evolution of the investigated networks.
The scope is twofold: helping in the setting up of the model, for checking its completeness, and checking the evolution of the model, in terms of the possibility to reach particular threshold values of the biological elements in the model, when varying the initial conditions.

1 Introduction

Regulatory networks play a crucial role in living organisms and understanding their connections and their whole dynamics is quite a challenging task. Executable models [13] of biological processes implied by these networks can provide useful insights. Nevertheless, building full detailed executable models of biological systems is often hampered by the lack of accurate, high-confidence parameters regarding, say, the kinetics of the chemical reactions or molecular concentrations. One possible solution consists in providing a qualitative model, able to grasp the essential features of the dynamic behavior. This is the approach followed in [33,30,31,29] where the logical Thomas' method, the Generalised Boolean Network (GBN), is presented. In this model, the state of each gene (seen as a regulatory entity) is represented by a concentration threshold value that varies on a limited number of values, e.g., *Low*, *Medium* or *High*. Values abstractly correspond to distinguished levels of

1571-0661 © 2012 Elsevier B.V. Open access under CC BY-NC-ND license.
doi:10.1016/j.entcs.2012.05.012

regulation. As a matter of fact, regulators usually act differently above certain threshold values. The logic abstraction is able to capture this non-linear nature of networks behavior, despite its simplicity. The (global) state evolves to the next one, due to a set of functions that define the next state of each gene starting from the current state of the genes which regulate it.

The GBN model allows one to infer biological models from incomplete biological data and to study the steady states and the feedbacks (positive or negative). Nevertheless their application generates an exponential growth of the states and there is not a large number of formal techniques and analysis tools available. Other executable languages like Petri Nets [26] or Process Algebras (e.g. [19,6]) can offer instead their well-founded theory and tool support, once rendered the logical regulatory models inside their complementary frameworks. Furthermore, GBNs can hardly handle incomplete or inconsistent data, i.e. cases in which there could be more than one next state, while, again, Petri Nets or Process Algebras can model these situations, by exploiting non determinism. Finally, both Petri Nets and Process Algebras can be easily extended in order to deal with quantitative information, like stochastic ones, e.g., associating rates with transitions.

The theory of Petri Nets, already exploited in Systems Biology (e.g. in [15,14]), has been employed to analyse the dynamics of regulatory networks in several works (see e.g. [7,27,28]), in which the logical models are translated in terms of Petri Nets, thus exploiting some of the above useful features. Process Algebras provide an alternative framework to analyse the dynamics of regulatory networks. They ([23,25,24,9,10,11,5,22,8], to cite only a few) have been fruitfully used to model several kinds of biological systems, relying on the idea that a biological system can be abstractly modelled as a concurrent system. Our approach aims at using process algebras for modelling and analysing GBNs. More in details, our purposes are

- to introduce an executable process algebraic model able to capture the *synchronous* behavior of GBNs, where each entity composing the network can change its concentration only when all the other interacting entities have reached their threshold values;
- to propose a static analysis technique that, once applied to the obtained model, is able to provide safe approximations of the behavior of the modelled entities. As a consequence, we can test the faithfulness of the model and also provide confident predictions on the dynamics, in case the model is sufficiently accurate.

Which Kind of Process Algebra?

We represent GBN entities as processes and the functions that compute their next states, given the threshold values of all the other regulative entities, as the result of communications between the corresponding processes. We have that an entity can send its new concentration value, only when all its regulators have sent their values.

Process Algebras offer different kinds of synchronisation mechanisms. What we need here is a multiple synchronisation among each entity and its regulators.

Also, we need that all the entities evolve at the same time. To model such a peculiar synchronisation pattern of GBNs, we chose to introduce a *new* process algebra called $\mathbf{Sim}$-π_n, reminiscent of the π-calculus [19]. In our model, each entity offers different combinations of inputs. Each combination of inputs corresponds to the reception of a combination of values coming from the regulative entities, combination that leads to the output of the new value. To this aim, we extend the π-calculus with the selective input prefix $\{a_1(x_1 \in X_1), ..., a_n(x_n \in X_n)\}.P$ that simultaneously receives the outputs $b_1, ..., b_n$ on the channels a_i and continues as P, provided that each received value x_i, with $i \in [1, n]$ is included in the selection sets X_i. Note that a similar joint input has been introduced in [17] to extend CCS [20], where it is proven that the synchronisation on n inputs is more expressive that the one on n-1 inputs. We also need that all the genes simultaneously update their states, after the reception of the values coming from the regulative entities. To reproduce this behavior, every entity, in parallel, sends its threshold value to all the processes waiting for it. In the standard π-calculus, sent values are consumed by their recipients, while in $\mathbf{Sim}$-π_n, a value emitted persists until all the recipients have received it. During this first phase every entity commits on one combination of inputs, and it is therefore ready to output, in the second phase, the value resulting from that combination. As a consequence, at each step the process produces a new tuple of outputs ready for a new synchronisation step. Channels used for outputs resemble signals as defined in synchronous π-calculus [2]. What we obtain is a special *many to many* commitment, in which there is a simultaneous communication on all the input channels with all the corresponding outputs.

Which Kind of Analysis?

Operational semantics allows us to reason in terms of steps and of causal relationships among steps. Being an executable model, it is suitable for testing and comparing different hypotheses. As a consequence, our process algebraic model can facilitate the analysis, both dynamic and static, of GBNs behavior and of the network properties related to the concentration levels of the involved elements. In particular, we exploit Control Flow Analysis (CFA), i.e. a static analysis technique based on Flow Logic [21], by applying it to our process algebraic specification of GBNs. Control Flow Analysis provides a *safe over-approximation* of the *exact* behavior of a system, in terms of the possible reachable configurations. The CFA results reflect the dynamics of the GBN model. The advantages derive from the fact that the CFA is in general not time-consuming, nor computationally expensive and this is a crucial point when the number of biological elements or the number of thresholds for each element increases.

We finally apply our framework to a to a case study represented by a real biological pathway, i.e. to a model of the regulatory network controlling the T-helper (Th) lymphocytes differentiation process.

G_1	G_2	$[G_1]$	$[G_2]$
0	0	1	1
0	1	1	2
0	2	0	2
1	0	1	0
1	1	1	0
1	2	0	1

Table 1
The Next State Functions of the Entities G_1 and G_2

2 Generalised Boolean Networks

Boolean Networks (BNs) [16] have been introduced in Biology to model Gene Regulatory Networks (GRN), i.e. those networks of interaction between genes that are at the basis of the protein synthesis. A Boolean Network is composed by a set of entities which regulate each other in a positive or negative way. The state of each gene (seen as a regulatory entity) is represented by a boolean value, i.e. active (1) or inactive (0). The (global) state of a boolean network, composed of n genes, is represented as a n-dimension vector of boolean variables, one for each gene. The evolution from a state to the next one is computed by a set of n boolean functions, each acting on each single variable, that define the next state starting from the current state of the genes which regulate it.

We are interested in a generalisation of the BN model (GBN) [32], where each variable x_i can take multiple values, that abstractly correspond to different levels of regulation. More formally, a *generalised boolean network* consists of a set of n nodes $G = \{G_1, ..., G_n\}$ that represent *regulatory entities*. Each entity G_i has an associated set of possible values B_i and its behavior is usually determined by a subset of involved entities, collected by what is called its *neighborhood* $N(G_i) \subseteq G$, with indexes ranged over by $I_N(G_i) = \{l \in [1,n] | G_l \in N(G_i)\}$. The dynamics of each entity G_i is given by a logical *next-state* function $[G_i]$, which given the states of the entities in $N(G_i)$ computes the next state $[G_i]$. The next-state functions can be given by using state transition tables (the truth tables in the Boolean Networks). For the sake of uniformity, we fix the number of inputs to the whole number n of entities and we reason in terms of n-tuples. Note that here we focus on the *synchronous* model, where the states of all the entities are updated simultaneously in one single step, by using the corresponding next-state functions. A global state of a generalised boolean network with n entities is represented by a tuple of states $(b_1, ..., b_n)$, where $b_i \in B_i$ represents the state of entity $G_i \in G$. The state space is therefore $(B_1 \times ... \times B_n)$. The sequence of global states in $(B_1 \times ... \times B_n)$ starting from some initial state is called a *trace*. The complete behavior of a GBN is described by the set of all the traces. Strongly connected components of the state transition graphs are called *attractors*. Among attractors, we can distinguish *stable* states, i.e. states that have no successors (apart from themselves), or states that are involved in cycles.

Running Example To illustrate the GBN model, we present a small system inspired by [28], composed by two entities G_1 and G_2, whose concentration values vary in

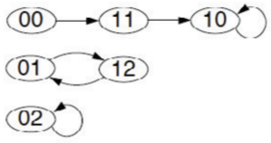

Fig. 1. State Transition Graph [28]

$B_1 = \{0, 1\}$ and in $B_2 = \{0, 1, 2\}$, respectively. The next-state functions are directly defined using the state transition tables in Table 1, where $[G_1]$ ($[G_2]$) represents the next state of G_1 (G_2, respectively). In this model only G_2 influences the concentration of G_1 (i.e. the next value only depends on the state of G_2) and both influence the concentration of G_2. More in detail, the entity G_1 inhibits G_2, while G_2 inhibits G_1 only when reaching the state 2. A state s is composed by a pair (b_1, b_2) with $b_i \in B_i$ with $i \in \{1, 2\}$. For instance, $s = (0, 0)$ is a state and, in one single step, we can pass from the state s to the state $s' = (1, 1)$ and therefore on the stable state $(1, 0)$, as shown by the graphical representation in Fig. 1.

BNs and its generalised version are traditionally applied to infer genetic regulatory networks starting from a partial biological knowledge (see [12] for a short review). Nevertheless, GBNs can also be used to describe metabolic networks, where it is possible to have a wider range of concentration values. In general, it is worthwhile investigating which conditions allow the molecular species of interest to reach threshold values.

3 The Process Algebra Sim-π_n

To model the synchronisation pattern of GBNs, we introduce a new process algebra called **Sim**-π_n, reminiscent of the π-calculus [19] (without summation and restriction), with restricted forms for processes and a special *many to many* synchronisation mechanism.

To obtain the required unisonous synchronisations, our processes P, that represent regulatory entities, are obtained by the parallel composition of special sub-processes S that appear in tailored form. Sub-processes S are structured in two parts: an initial guard, made by a set of inputs that must be all executed (first part), if any, before executing the only last output prefix (second part) in parallel with S. Alternatively a sub-process S can be the continuation $(\overline{a}\langle b\rangle \parallel S)$ or can be

$$\frac{(b_1, ..., b_n) \in (X_i^1 \times ... \times X_i^n) \text{ for all } i \in [1, n]}{\prod_{i\in[1,n]} \overline{a_i}\langle b_i\rangle \parallel \prod_{i\in[1,n]} s_i.(\overline{a_i}\langle b_i'\rangle \parallel S_i) \to \prod_{i\in[1,n]}(\overline{a_i}\langle b_i'\rangle \parallel S_i)}$$

$$\text{where } s_i = \{a_1(x_i^1 \in X_i^1), ..., a_n(x_i^n \in X_i^n)\}$$

$$\frac{P \to P'}{P \parallel Q \to P' \parallel Q} \qquad \frac{P \equiv P' \to Q' \equiv Q}{P \to Q}$$

Table 2
The Reduction Semantics of the Calculus

given by just an output. Formally, let $\mathcal{N}$ be a countable set of names, ranged over by $n, a, b, \ldots$, and $\mathcal{X}$ be a countable set of variables, ranged over by $x, y, z, \ldots$, then the processes are built according to the following syntax:

$$P ::= P \parallel S$$

$$S ::= \mathbf{0} \mid \{a_1(x_1 \in X_1), ..., a_n(x_n \in X_n)\}.(\overline{a}\langle b\rangle \parallel S) \mid (\overline{a}\langle b\rangle \parallel S) \mid \overline{a}\langle b\rangle$$

Like in the π-calculus, the term $\mathbf{0}$ denotes the empty process and the operator $\parallel$ denotes the parallel composition. As standard, we omit $\mathbf{0}$, when needed, and we use the shorthand $\prod_{i\in R} S_i$ for abbreviating the parallel composition of processes S_i for $i \in R$. The prefix $\overline{a}\langle b\rangle$ denotes the output of value b on the channel a. The multiple selective input prefix guard $\{a_1(x_1 \in X_1), ..., a_n(x_n \in X_n)\}.(\overline{a}\langle b'\rangle \parallel S)$ (see [4] for a similar construct) simultaneously gets the outputs $\overline{a_i}\langle b_i\rangle$ on all the channels $a_1, ..., a_n$ and continues as $(\overline{a}\langle b'\rangle \parallel S)$, provided that each b_i belongs to X_i for each $i \in [1, n]$, where the sets X_i do not include any bound name. In other words, a value received along the channel a_i is accepted only if it matches with one of the values included in the corresponding selection sets X_i. Note that inputs in **Sim**-π_n have exactly n items.

The reduction semantics of our calculus is given in Table 2. We use the standard notion of structural congruence $\equiv$ of π-calculus: in particular, processes form a commutative monoid with respect to the parallel composition. The communication is in parallel and in broadcast, i.e. each top-level output simultaneously synchronises with every corresponding input occurring in the combination of inputs, in the rest of the system, as explained below. Note that in the processes used to represent GBNs the values sent are not used for subsequent communications, but are passed for synchronisation purposes, unlike the π-calculus, and more in CCS [20] style. The other rules are standard.

The definition below allows the construction of a system of processes, given a Generalised Boolean Network, which shows a behavior equivalent to that of the original GBN.

Our idea consists in having a process for each regulative entity and a branch for each entry in the next-state function table, with the suitable selective joint input. Furthermore, to introduce the initial conditions, we use single outputs, not preceded by inputs. For the sake of simplicity, in the following we overload the symbols G_i and B_i, by using them both in the GBN notation and in the process algebraic one.

Definition 3.1 Given a GBN composed by n entities $G = \{G_1, ..., G_n\}$, where each entity G_i has associated its next-state function $[G_i] : B_1 \times ... \times B_n \to B_i$, the corresponding $\mathbf{Sim}$-π_n system of processes is specified as follows:

$G = G_1 \parallel ... \parallel G_n$

$G_i = \prod_{j\in[1,m]} G_{ij}$ with $m \leq |B_1| \times ... \times |B_n|$

$G_{ij} = \{a_1(x^1_{ij} \in X^1_{ij}), ..., a_n(x^n_{ij} \in X^n_{ij})\}.(\overline{a_i}\langle b^i_{ij}\rangle \parallel G_{ij})$,

where, for each gene G_i there is a channel a_i possibly sending values in $B_i \subseteq \mathcal{N}$, and where for all combinations $(b_1, ..., b_n)$ in $B_1 \times ... \times B_n$ such that $[G_i](b_1, ..., b_n) = b'_i$ there is a corresponding branch $G_{i\mathbf{j}}$, that includes the selective input $\{a_1(x^1_{i\mathbf{j}} \in X^1_{i\mathbf{j}}), ..., a_n(x^n_{i\mathbf{j}} \in X^n_{i\mathbf{j}})\}$, with $X^k_{i\mathbf{j}} = \{b_k\}$ for all $k \in [1, n]$.
The initial conditions are given in the form of a parallel composition of n outputs in $B_1 \times ... \times B_n$, as follows: $\prod_{i\in[1,n]} \overline{a_i}\langle b_i\rangle$.

In Section 5, we will exploit the full expressiveness of the selective inputs, by allowing not only singleton sets.

Our semantics works under the hypotheses that in each configuration there are exactly n branches (one for each entity) whose selective inputs match the given outputs. More precisely, given a configuration:

$$\prod_{i\in[1,n]} \overline{a_i}\langle b_i\rangle \parallel \prod_{i\in[1,n]} (\prod_{j\in[1,m]} s_{ij}.(\overline{a_i}\langle b^i_{ij}\rangle \parallel S_{ij}))$$

where $s_{ij} = \{a_1(x^1_{ij} \in X^1_{ij}), ..., a_n(x^n_{ij} \in X^n_{ij})\}$, by means of the commutative rule for parallel composition, we can always put in evidence the n branches that match with the given outputs, i.e. all the branches $S_{i\mathbf{j}}$ with indexes $\mathbf{j} \in [1, m]$, and where the corresponding selective inputs $\{a_1(x^1_{i\mathbf{j}} \in X^1_{i\mathbf{j}}), ..., a_n(x^n_{i\mathbf{j}} \in X^n_{i\mathbf{j}})\}$, are such that for all $i \in [1, n]$, $(b_1, ..., b_n) \in (X^1_{i\mathbf{j}} \times ... \times X^n_{i\mathbf{j}})$, thus obtaining

$$\prod_{i\in[1,n]} \overline{a_i}\langle b_i\rangle \parallel \prod_{i\in[1,n]} s_{i\mathbf{j}}.(\overline{a_i}\langle b^i_{i\mathbf{j}}\rangle \parallel S_{i\mathbf{j}}) \parallel \prod_{i\in[1,n]} (\prod_{k\in[1,m]\setminus\{\mathbf{j}\}} s_{ik}.(\overline{a_i}\langle b^i_{ik}\rangle \parallel S_{ik}))$$

The communication rule effect is that of consuming the initial tuple of outputs, producing the new tuple of outputs and restoring the whole process representing the entities, therefore leading to the new configuration:

$$\prod_{i\in[1,n]} \overline{a_i}\langle b^i_{i\mathbf{j}}\rangle \parallel \prod_{i\in[1,n]} (\prod_{j\in[1,m]} s_{ij}.(\overline{a_i}\langle b^i_{ij}\rangle \parallel S_{ij}))$$

Proposition 3.2 *Let G be a GBN composed by n entities $\{G_1, ..., G_n\}$, where each entity G_i has associated its next-state function $[G_i] : B_1 \times ... \times B_n \to B_i$, and G be the corresponding* $\mathbf{Sim}$-π_n *system of processes, that includes a parallel composition of outputs. In the state transition graph of G there exists a transition from the*

$$\begin{array}{lll|lll}
G_1 & = & G_{11} \parallel G_{12} & G_2 & = & G_{21} \parallel G_{22} \parallel G_{23} \parallel G_{24} \\
G_{11} & = & \{g_1(x_{11}^1 \in \{0\}), g_2(x_{11}^2 \in \{0\})\}.(\overline{g_1}\langle 1\rangle \parallel G_{11}) & G_{21} & = & \{g_1(y_{21}^1 \in \{0\}).g_2(y_{21}^2 \in \{0\})\}.(\overline{g_2}\langle 1\rangle \parallel G_{21}) \\
G_{12} & = & \{g_1(x_{11}^1 \in \{0\}), g_2(x_{11}^2 \in \{1\})\}.(\overline{g_1}\langle 1\rangle \parallel G_{12}) & G_{22} & = & \{g_1(y_{22}^1 \in \{0\}).g_2(y_{22}^2 \in \{1\})\}.(\overline{g_2}\langle 2\rangle \parallel G_{22}) \\
G_{13} & = & \{g_1(x_{13}^1 \in \{0\}), g_2(x_{13}^2 \in \{2\})\}.(\overline{g_1}\langle 0\rangle \parallel G_{13}) & G_{23} & = & \{g_1(y_{23}^1 \in \{0\}).g_2(y_{23}^2 \in \{2\})\}.(\overline{g_2}\langle 2\rangle \parallel G_{23}) \\
G_{14} & = & \{g_1(x_{14}^1 \in \{1\}), g_2(x_{14}^2 \in \{0\})\}.(\overline{g_1}\langle 1\rangle \parallel G_{14}) & G_{24} & = & \{g_1(y_{24}^1 \in \{1\}).g_2(y_{24}^2 \in \{0\})\}.(\overline{g_2}\langle 0\rangle \parallel G_{24}) \\
G_{15} & = & \{g_1(x_{15}^1 \in \{1\}), g_2(x_{15}^2 \in \{1\})\}.(\overline{g_1}\langle 1\rangle \parallel G_{15}) & G_{25} & = & \{g_1(y_{25}^1 \in \{1\}).g_2(y_{25}^2 \in \{1\})\}.(\overline{g_2}\langle 0\rangle \parallel G_{25}) \\
G_{16} & = & \{g_1(x_{16}^1 \in \{1\}), g_2(x_{16}^2 \in \{2\})\}.(\overline{g_1}\langle 0\rangle \parallel G_{16}) & G_{26} & = & \{g_1(y_{26}^1 \in \{1\}).g_2(y_{26}^2 \in \{2\})\}.(\overline{g_2}\langle 1\rangle \parallel G_{26})
\end{array}$$

Table 3
Our Running Example Specification

global state $s = (b_1, ..., b_n)$ to $s' = (b'_1, ..., b'_n)$ if and only if there is a transition on the corresponding system.

Proof Sketch By Def. 3.1, we have that for each reachable entry $(b_1, ..., b_n)$ in the state transition table for G, there exists a branch $G_{i\mathbf{j}}$ (with $\mathbf{j} \in [1, m]$ and $m \leq |B_1| \times \ldots \times |B_n|$) for each entity G_i in the form $\{a_1(x_{i\mathbf{j}}^1 \in X_{i\mathbf{j}}^1), ..., a_n(x_{i\mathbf{j}}^n \in X_{i\mathbf{j}}^n)\}.(\overline{a_i}\langle b_{i\mathbf{j}}^i\rangle | G_{i\mathbf{j}})$, such that $X_{i\mathbf{j}}^k = \{b_k\}$. These are exactly the premises of the communication rule leading to the desired transition. As a consequence, the operational semantics of **Sim**-π_n exactly reproduces the simultaneous synchronisations of the GBNs. □

Running Example (cont'd) The complete specification of our running example corresponding to the next-state functions in Table 1, is given in Table 3, where the channels are g_1 for the first component and g_2 for the second one.

Suppose to have the following initial condition process with two outputs: $I = \overline{g_1}\langle 0\rangle \parallel \overline{g_2}\langle 0\rangle$, corresponding to the state $(0, 0)$. The overall system is therefore $G_1 \parallel G_2 \parallel I$. The system evolves first by simultaneously synchronising on the values $0, 0$, then on the values $1, 1$ and finally on the values $1, 0$, as shown by the following transitions, that mimic the representation of the state trace (the sequence of states), leading the GBN from the state $(0, 0)$ to the state $(1, 1)$, and from $(1, 1)$ to $(1, 0)$. To emphasise this correspondence, we enrich the transition arrow with a label that records the corresponding GBN state. The last state is a stable one: indeed, from the last state $G_1 \parallel G_2 \parallel \overline{g_1^1}\langle 1\rangle \parallel \overline{g_2^0}\langle 0\rangle$ we can only reach the state itself. Also, we use the shorthand G'_1, G''_1 and G'''_1 to denote the process G_1 except for the first (fifth and fourth, respectively) parallel sub-process. Similarly, G'_2, G''_2 and G'''_2 denote the process G_2 except for the first (fifth and fourth, respectively) parallel sub-process.

$$\begin{array}{l}
\{g_1(x_{11}^1 \in \{0\}), g_2(x_{11}^2 \in \{0\})\}.(\overline{g_1}\langle 1\rangle \parallel G_{11}) \parallel G'_1 \parallel \{g_1(y_{21}^1 \in \{0\}).g_2(y_{21}^2 \in \{0\})\}.(\overline{g_2}\langle 1\rangle \parallel G_{21}) \parallel G'_2 \parallel I \\
\xrightarrow{00} \overline{g_1}\langle 1\rangle \parallel G_{11} \parallel G'_1 \parallel \overline{g_2}\langle 1\rangle \parallel G_{21} \parallel G'_2 \equiv G_1 \parallel G_2 \parallel \overline{g_1}\langle 1\rangle \parallel \overline{g_2}\langle 1\rangle \equiv \\
\{g_1(x_{15}^1 \in \{1\}), g_2(x_{15}^2 \in \{1\})\}.(\overline{g_1}\langle 1\rangle \parallel G_{15}) \parallel G''_1 \parallel \{g_1(y_{25}^1 \in \{1\}).g_2(y_{25}^2 \in \{1\})\}.(\overline{g_2}\langle 0\rangle \parallel G_{25}) \parallel G''_2 \\
\xrightarrow{11} \overline{g_1}\langle 1\rangle \parallel G_{15} \parallel G''_1 \parallel \overline{g_2}\langle 0\rangle \parallel G_{25} \parallel G''_2 \equiv G_1 \parallel G_2 \parallel \overline{g_1}\langle 1\rangle \parallel \overline{g_2}\langle 0\rangle \equiv \\
\{g_1(x_{14}^1 \in \{1\}), g_2(x_{14}^2 \in \{0\})\}.(\overline{g_1}\langle 1\rangle \parallel G_{14}) \parallel G'''_1 \parallel \{g_1(y_{24}^1 \in \{1\}).g_2(y_{24}^2 \in \{0\})\}.(\overline{g_2}\langle 0\rangle \parallel G_{24}) \parallel G'''_2 \\
\xrightarrow{10} \overline{g_1}\langle 1\rangle \parallel G_{14} \parallel G'''_1 \parallel \overline{g_2}\langle 0\rangle \parallel G_{24} \parallel G'''_2 \equiv G_1 \parallel G_2 \parallel \overline{g_1}\langle 1\rangle \parallel \overline{g_2}\langle 0\rangle
\end{array}$$

Similarly, we can model the other traces, one starting from $(0, 1)$ leading to $(1, 2)$ and back to $(0, 1)$, and the other starting from $(0, 2)$ and coming back to $(0, 2)$ (see Fig. 1).

Note that, in case of incomplete data, i.e. cases in which there could be more than one next state, the process algebraic framework offers us a way out, thanks to the possible introduction of the non deterministic choice operator $+$ in the syntax. We could indeed allow a process to branch on the same selective input, and to choose among different outputs, in a non deterministic way. Dynamically, only one output is chosen each time. In the GBN framework, this is not possible, because we deal with functions that, by definition, require a unique value for each input.

By summarising, **Sim**-π_n seems to offer a first encouraging answer on how rendering the GBN synchronous behavior in process algebraic terms.

4 Control Flow Analysis

The Control Flow Analysis (CFA) extends the one for the π-calculus in [3]. The CFA computes a safe over-approximation of all the possible values that the tuples of variables in the system may be bound to, and of the tuples of values that may simultaneously flow on channels. Furthermore, it can establish a causal relation between a configuration and the next one. In other words, it predicts all the possible communications, and consequently all the possible reachable configurations, in terms of concentration levels. More precisely, the analysis keeps track of the following information:

- An approximation $\rho : \mathcal{X} \times ... \times \mathcal{X} \to \wp(\mathcal{B}_1 \times ... \times \mathcal{B}_n)$ of name bindings (see [3]). If $(b_1, ..., b_n) \in \rho(x_1, ..., x_n)$ then each variable x_i can simultaneously assume the value b_i, for each $i \in [1, n]$.
- An approximation κ: $\mathcal{N} \times ... \times \mathcal{N} \to (\mathcal{B}_1 \times ... \times \mathcal{B}_n \to \wp(\mathcal{B}_1 \times ... \times \mathcal{B}_n))$ that, given a set of n channels and a a tuple of output values, provides all the possible tuples of output prefixes reached after the simultaneous synchronisation. If $(b_1, ..., b_n) \in \kappa(a_1, ..., a_n)(b'_1, ..., b'_n)$ the output tuple $(b'_1, ..., ' b_n)$ could be triggered by the previous inputs on the tuple $(b_1, ..., b_n)$. Furthermore, we introduce the tuple $(\epsilon, ..., \epsilon)$ to identify the ideal initial tuple. As a consequence, the tuples $(b_1, ..., b_n)$ in $\kappa(a_1, ..., a_n)(\epsilon, ..., \epsilon)$, are the possible tuples of simultaneously sent values.

The approximation of the behavior of a process P is represented by the pair ρ, κ, called *estimate* for P, whose correctness is validated against a set of clauses that operate upon judgments in the form $\rho, \kappa \models P$. Clauses, that amount to a structural traversal of process syntax, are given in Table 4. All the clauses dealing with a compound process require that the components are validated. The rule for the inactive process does not restrict the analysis result while the rules for parallel composition ensure that the analysis also holds for the immediate sub-processes. In particular, the analysis of the composition of the tuple of outputs and the parallel composition of the corresponding selective input branches amounts to separately analysing the part of outputs and all the selective input branches.

In analysing the parallel composition of n output prefixes, we have to check whether the corresponding sent values, collected in the tuple $(b_1, ..., b_n)$, belong to $\kappa(a_1, ..., a_n)(\epsilon, ..., \epsilon)$, i.e. it is a tuple of simultaneous outputs. Analysing instead the process $\{a_1(x_1 \in X_1^i), ..., a_n(x_n \in X_n^i)\}.(\overline{a_i}\langle b'_i\rangle \parallel S)$ requires more steps. Given an input combination, (i) we look in $\kappa(a_1, ..., a_n)(\epsilon, ..., \epsilon)$ for a tuple $(b_1, ..., b_n)$ of simultaneously sent values that match the corresponding selective inputs. For any matching combination, (ii) we check whether the tuple $(b_1, ..., b_n)$ composed by the values b_i possibly sent on the i-th channel is included in $\rho(x_i^1, ..., x_i^n)$ of names to which the tuples of variable $x_i^1, ..., x_i^n$ can evaluate. Furthermore, (iii) we check whether the resulting tuple of sent values $(b'_1, ..., b'_n)$ is included in the set of tuples that belong to $\kappa(a_1, ..., a_n)(b_1, ..., b_n)$ and to $\kappa(a_1, ..., a_n)(\epsilon, ..., \epsilon)$. In analysing a process, the tuple of initial sent values, that has no trigger, is recorded in $\kappa(a_1, ..., a_n)(\epsilon, ..., \epsilon)$, but also all the tuples that arise from the subsequent communications. This inclusion ensures the semantic correctness, since the estimate must be valid for all the derivatives of the initial system.

Intuitively, the estimate components take into account the possible dynamics of the process under consideration. In the clauses, the checks mimic the semantic evolution, by modelling the semantic preconditions and the consequences of the possible synchronisations. For the communication clause, e.g., it checks whether the precondition of a many to many synchronisation is satisfied, i.e. whether there is an output tuple in $\kappa(a_1, ..., a_n)(\epsilon, ..., \epsilon)$, matching the analysed input one. The conclusion imposes the additional requirements on the estimate components, necessary to give a valid prediction of the analysed synchronisation action, mainly that the variables x_k^i can be bound, and that the resulting outputs can be reached, due to the previous tuple, and collected in a new tuple.

Finally, the analysis of $S = a_1(x_i^1 \in X_i^1), ..., a_n(x_i^n \in X_i^n)\}.(\overline{a_i}\langle b'_i\rangle \parallel S)$ coincides with the one for $a_1(x_i^1 \in X_i^1), ..., a_n(x_i^n \in X_i^n)\}.\overline{a_i}\langle b'_i\rangle$, i.e. the CFA does not take into account infinite behavior and this is a source of approximation.

Note that, in case of incomplete data, i.e. when it is possible to have more than one output tuple after a given one, we have that $\kappa(a_1, ..., a_n)(b_1, ..., b_n)$ is not a singleton, but a set. This can clearly lead to an over-approximation of the possible pathways.

It is possible to prove that there always exists a least estimate (see [3] for a similar proof) and that the CFA is correct, i.e. it respects the semantic specification, as shown below.

Theorem 4.1 *If $P \to P'$ and $\rho, \kappa \models P$ then $\rho, \kappa \models P'$.*

Proof. By induction on the inference of $P \to P'$.
We only show the case of the *communication* rule, where $P = \prod_{i\in[1,n]} \overline{a_i}\langle b_i\rangle \parallel \prod_{i\in[1,n]} S_i$, with $S_i = \{a_1(x_i^1 \in X_i^1), ..., a_n(x_i^n \in X_i^n)\}.(\overline{a_i}\langle b'_i\rangle \parallel S_i)$ and $P' = \prod_{i\in[1,n]} \overline{a_i}\langle b'_i\rangle \parallel \prod_{i\in[1,n]} S_i$. Now, from $\rho, \kappa \models \prod_{i\in[1,n]} \overline{a_i}\langle b_i\rangle \parallel \prod_{i\in[1,n]} S_i$ iff
(1) $\rho, \kappa \models \prod_{i\in[1,n]} \overline{a_i}\langle b_i\rangle$ and
(2) $\rho, \kappa \models \prod_{i\in[1,n]} S_i$.

$$\frac{}{\rho, \kappa \models \mathbf{0}}$$

$$\frac{\rho, \kappa \models P_0 \wedge \rho, \kappa \models P_1}{\rho, \kappa \models P_0 \parallel P_1}$$

$$\frac{(b_1, ..., b_n) \in \kappa(a_1, ..., a_n)(\epsilon, ..., \epsilon)}{\rho, \kappa \models \prod_{i \in [1,n]} \overline{a_i}\langle b_i \rangle}$$

$$\frac{\begin{cases} \forall (b_1, ..., b_n) \in \kappa(a_1, ..., a_n)(\epsilon, ..., \epsilon) \text{ s.t. } (b_1, ...b_n) \in X_i^1 \times ... \times X_i^n \text{ for } i \in [1,n] : \\ (b_1, ..., b_n) \in \rho(x_i^1, ..., x_i^n) \text{ for all } i \in [1,n] \wedge \\ (b'_1, ..., b'_n) \in \kappa(a_1, ..., a_n)(b_1, ..., b_n) \wedge \\ (b'_1, ..., b'_n) \in \kappa(a_1, ..., a_n)(\epsilon, ..., \epsilon) \end{cases}}{\rho, \kappa \models \prod_{i \in [1,n]} \{a_1(x_i^1 \in X_i^1), ..., a_n(x_i^n \in X_i^n)\}.\overline{a_i}\langle b'_i \rangle}$$

$$\frac{\rho, \kappa \models \prod_{i \in [1,n]} \overline{a_i}\langle b_i \rangle \wedge \rho, \kappa \models \prod_{i \in [1,n]} \{a_1(x_i^1 \in X_i^1), ..., a_n(x_i^n \in X_i^n)\}.(\overline{a_i}\langle b'_i \rangle \parallel S_i)}{\rho, \kappa \models \prod_{i \in [1,n]} \overline{a_i}\langle b_i \rangle \parallel \prod_{i \in [1,n]} \{a_1(x_i^1 \in X_i^1), ..., a_n(x_i^n \in X_i^n)\}.(\overline{a_i}\langle b'_i \rangle \parallel S_i)}$$

$$\frac{\rho, \kappa \models \prod_{i \in [1,n]} \{a_1(x_i^1 \in X_i^1), ..., a_n(x_i^n \in X_i^n)\}.\overline{a_i}\langle b'_i \rangle}{\rho, \kappa \models \prod_{i \in [1,n]} \{a_1(x_i^1 \in X_i^1), ..., a_n(x_i^n \in X_i^n)\}.(\overline{a_i}\langle b'_i \rangle \parallel S_i)}$$

Table 4
CFA

The conjunct (2) in full is $\rho, \kappa \models \prod_{i \in [1,n]} \{a_1(x_i^1 \in X_i^1), ..., a_n(x_i^n \in X_i^n)\}.(\overline{a_i}\langle b'_i \rangle \parallel S_i)$, that for the CFA rules is equivalent to

$$\rho, \kappa \models \prod_{i \in [1,n]} \{a_1(x_i^1 \in X_i^1), ..., a_n(x_i^n \in X_i^n)\}.\overline{a_i}\langle b'_i \rangle.$$

We have that:

- from (1) $(b_1, ..., b_n) \in \kappa(a_1, ..., a_n)(\epsilon, ..., \epsilon)$;
- since $(b_1, ..., b_n) \in \kappa(a_1, ..., a_n)(\epsilon, ..., \epsilon)$ and $(b_1, ..., b_n) \in (X_i^1 \times ... \times X_i^n)$ for all $i \in [1,n]$, then $(b_1, ..., b_n) \in \rho_i(x_i^1, ..., x_i^n)$ and furthermore both $(b'_1, ..., b'_n) \in \kappa(a_1, ..., a_n)(\epsilon, ..., \epsilon)$ and $(b'_1, ..., b'_n) \in \kappa(a_1, ..., a_n)(b_1, ..., b_n)$ hold.

Since $(b'_1, ..., b'_n) \in \kappa(a_1, ..., a_n)(\epsilon, ..., \epsilon)$, we can deduce that $\rho, \kappa \models \prod_{i \in [1,n]} \overline{a_i}\langle b'_i \rangle$ and therefore $\rho, \kappa \models \prod_{i \in [1,n]} \overline{a_i}\langle b'_i \rangle \parallel \prod_{i \in [1,n]} S_i$, because $\rho, \kappa \models \prod_{i \in [1,n]} S_i$ holds by hypothesis. □

Running Example (cont'd) We can now apply our CFA to our running example, given the initial conditions $I = \overline{g_1}\langle 0 \rangle \parallel \overline{g_2}\langle 0 \rangle$, i.e. we analyse the system $G_1 \parallel G_2 \parallel I$.

$\kappa(g_1, g_2)(\epsilon, \epsilon) = \{(0,0), (1,1), (1,0)\}$		
$\kappa(g_1, g_2)(0,0) = \{(1,1)\}$	$\rho(x^1_{11}, x^2_{11}) \supseteq \{(0,0)\}$	$\rho(y^1_{21}, y^2_{21}) \supseteq \{(0,0)\}$
$\kappa(g_1, g_2)(1,1) = \{(0,1)\}$	$\rho(x^1_{15}, x^2_{15}) \supseteq \{(0,0)\}$	$\rho(y^1_{25}, y^2_{25}) \supseteq \{(1,1)\}$
$\kappa(g_1, g_2)(0,1) = \{(1,0)\}$	$\rho(x^1_{14}, x^2_{14}) \supseteq \{(0,0)\}$	$\rho(y^1_{24}, y^2_{24}) \supseteq \{(1,0)\}$

Table 5
Some of the CFA Results

Some of the results are reported in Table 5. To illustrate our analysis, we show some checks performed by the CFA.

- $\rho, \kappa \models G_1 \parallel G_2 \parallel I$ iff $\rho, \kappa \models G_1 \wedge \rho, \kappa \models G_2 \wedge \rho, \kappa \models I$ and therefore
  - · $\rho, \kappa \models G_1$ iff $\rho, \kappa \models G_{11} \wedge ... \wedge \rho, \kappa \models G_{16}$
  - · $\rho, \kappa \models G_2$ iff $\rho, \kappa \models G_{21} \wedge ... \wedge \rho, \kappa \models G_{26}$
- $\rho, \kappa \models I$ implies that
  - · $(0,0) \in \kappa(g_1, g_2)(\epsilon, \epsilon)$.
- Since $(0,0) \in \kappa(g_1, g_2)(\epsilon, \epsilon)$, $0 \in \{0,1\}$, and $0 \in \{0\}$, we have that $\rho, \kappa \models G_{11}$ and $\rho, \kappa \models G_{21}$ imply that
  - · $(0,0) \in \rho(x^1_{11}, x^2_{11}) \wedge (0,0) \in \rho_1(y^1_{21}, y^2_{21})$
  - · $(1,1) \in \kappa(g_1, g_2)(0,0)$, and $(1,1) \in \kappa(g_1, g_2)(\epsilon, \epsilon)$.

The analysis results reflect the dynamic behavior. We can obtain the possible traces of states, by inspecting the κ component. In this case, from the initial configuration of values $(0,0)$, the analysis tells us that a possible next configuration is $(1,1)$, from which it is possible to reach the configuration $(1,0)$. From $(1,0)$, we can only reach $(1,0)$. This is exactly one of the possible traces in Fig. 1. At the same time we can infer that starting from $(0,0)$ the configuration $(1,2)$ is not reachable. As this simple example shows, our method can be used for easily checking some properties of biological networks. Its usefulness becomes more relevant when very large samples are considered. In this case, methods based on model checking or on the analysis of simulation results can indeed be computationally expensive. Our analysis, that is a simple extension of the one in [3], operates indeed in low polynomial time in the size of the process algebraic specification of the system of entities.

5 Possible Optimisations

Slightly modifying our definition of processes derived by GBNs, we obtain a more compact encoding, that recall the more compact formulas, obtained by applying well-known logic minimisation techniques. Under this regard, we can exploit the full expressiveness of the sets in the selective inputs.

Note that if the entity G_i does not depend on the entity G_k, then we could put $X_k = B_k$ in the k-th position in all the branches of G_i. On the process algebraic side, this corresponds to have that the input on the rest of components synchronises, independently from the output on the k-th component. For instance, in our running example, we could have less branches in the specification of G_1, as follows:

$$\begin{aligned}
G_1 &= G_{11} \parallel G_{12} \parallel G_{13}\\
G_{11} &= \{g_1(x^1_{11} \in \{0,1\}), g_2(x^2_{11} \in \{0\})\}.(\overline{g_1}\langle 1\rangle \parallel G_{11})\\
G_{12} &= \{g_1(x^1_{12} \in \{0,1\}), g_2(x^2_{12} \in \{1\})\}.(\overline{g_1}\langle 1\rangle \parallel G_{12})\\
G_{13} &= \{g_1(x^1_{13} \in \{0,1\}), g_2(x^2_{13} \in \{2\})\}.(\overline{g_1}\langle 0\rangle \parallel G_{13})
\end{aligned}$$

Furthermore, we can merge in one single branch all the combinations that share one of the entries, thus leading to the same result. We can illustrate it still in our running example, where we have that the value of $[G_1]$ is 1 either if $G_2 = 0$ or $G_2 = 1$. Consequently, we can have a single branch for both cases, thus further reducing the whole number of branches, as in the following specification.

$$\begin{aligned}
G_1 &= G_{11} \parallel G_{12}\\
G_{11} &= \{g_1(x^1_{11} \in \{0,1\}), g_2(x^2_{11} \in \{0,1\})\}.(\overline{g_1}\langle 1\rangle \parallel G_{11})\\
G_{12} &= \{g_1(x^1_{12} \in \{0,1\}), g_2(x^2_{12} \in \{2\})\}.(\overline{g_1}\langle 0\rangle \parallel G_{12})
\end{aligned}$$

This operation recalls the corresponding minimisation on the multi-values disjunctive normal forms, obtained by combining product terms, made by literals in the form g_i^S where $S \subseteq B_i$. The two terms can be combined together because they differ in only the literal g_2.

$$g_1^1 = g_1^{0,1} \cdot g_2^0 + g_1^{0,1} \cdot g_2^1 = g_1^{0,1} \cdot (g_2^0 + g_2^1) = g_1^{0,1} \cdot g_2^{0,1}$$

Finally, when a more concise specification is needed, we could even omit the inputs on the variables x_{ik} that belong to $X_{ik} = B_k$ in the specification, as in:

$$\begin{aligned}
G_1 &= G_{11} \parallel G_{12} \parallel G_{13}\\
G_{11} &= \{g_2(x^2_{11} \in \{0,1\})\}.(\overline{g_1}\langle 1\rangle \parallel G_{11})\\
G_{12} &= \{g_2(x^2_{12} \in \{2\})\}.(\overline{g_1}\langle 0\rangle \parallel G_{12})
\end{aligned}$$

The CFA can be slightly modified accordingly. In the next section, we apply all the above optimisations to our case study.

6 Case Study: The T-helper Regulatory Network

In this section, we apply our framework to a case study represented by a real biological pathway, in order to understand the roles of the elements composing a molecular network in influencing the concentration of other species. In particular, we consider the regulatory network that controls the T-helper (Th) lymphocytes differentiation process, following the homologous Thomas' model taken from [18] and shown in Figure 2. Specifically, the model describes the network of interactions underlying the differentiation process of two cell lineages, namely lymphocytes T-helper 1 and 2 (Th1 and Th2, respectively) from the common precursor Th0. T-helpers play a crucial role in the context of mammals immune system and their differentiation is finely regulated by different molecules (e.g. Interleukines and Interferons) produced by Th themselves or by other cell types including monocytes and dendritic cells. For more details on T-helper lymphocytes and their role in the immune system we refer the reader to [1] or to [18].

First we show an example of computation according to our formalism illustrating also how to apply the CFA on it. Then we verify the consistency of our model comparing the results of our computation and analysis with those presented in [18]. Finally we show how our method can be used to gain insights on the roles played by the network elements.

We consider and specify the whole network described in [18] (see Table 1 for the complete list of GBN dynamical rules), composed by the following seventeen elements: (1) IL-12, (2) IL-18, (3) IFN-β, (4) IL-12R, (5) IL-18R, (6) IFN-βR, (7) $STAT$-4, (8) $IRAK$, (9) IFN-γ, (10) IFN-γR, (11) $STAT$-1, (12) IL-4, (13) IL-4R, (14) $STAT$-6, (15) $GATA$-3, (16) $SOCS$-1, (17) T-BET (with $GATA$-3 and T-BET without auto-activation). A sketch of the corresponding specification is presented in Tables 6 and 7.

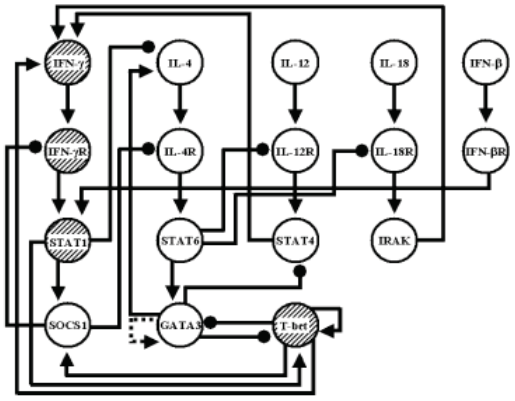

Fig. 2. The T-helper regulatory network. [Adapted from [18]]

Our whole system Sys is given by the parallel composition of the 17 processes represented in our specification. We choose the following tuple of channels: $G = (il12b, il18, fin\beta, il12r, il18r, fin\beta r, stat4, irak, fin\gamma, fin\gamma r, stat1, il4, il4r, stat6, gata3, socs1, Tibet)$ where the values that can pass on $ifn\gamma$, $ifn\gamma r$, $stat1$, and $tbet$ range over $\{L, M, H\}$, while the values that can pass on the other channels range over $\{L, H\}$, and L stands for Low, M for $Medium$ and H for $High$:

$IL\text{-}12 = IL\text{-}12_1 \Vert IL\text{-}12_2$	$IL\text{-}18 = IL\text{-}18_1 \Vert IL\text{-}18_2$
$IL\text{-}12_1 = \{il12(x^1 \in \{L\})\}.(\overline{il12}\langle L\rangle \Vert IL\text{-}12_1)$	$IL\text{-}18_1 = \{il18(x^2 \in \{L\})\}.(\overline{il18}\langle L\rangle \Vert IL\text{-}18_1)$
$IL\text{-}12_2 = \{il12(x^1 \in \{H\})\}.(\overline{il12}\langle H\rangle \Vert IL\text{-}12_2)$	$IL\text{-}18_2 = \{il18(x^2 \in \{H\})\}.(\overline{il18}\langle H\rangle \Vert IL\text{-}18_2)$
$IFN\text{-}\beta = IFN\text{-}\beta_1 \Vert IFN\text{-}\beta_2$	$STAT\text{-}6 = STAT\text{-}6_1 \Vert STAT\text{-}6_2$
$IFN\text{-}\beta_1 = \{ifn\beta(x^3 \in \{L\})\}.(\overline{ifn\beta}\langle L\rangle \Vert IFN\text{-}\beta_1)$	$STAT\text{-}6_1 = \{il4r(x^{13} \in \{L\})\}.(\overline{stat6}\langle L\rangle \Vert STAT\text{-}6_1)$
$IFN\text{-}\beta_2 = \{ifn\beta(x^3 \in \{H\})\}.(\overline{ifn\beta}\langle H\rangle \Vert IFN\text{-}\beta_2)$	$STAT\text{-}6_2 = \{il4r(x^{13} \in \{H\})\}.(\overline{stat6}\langle H\rangle \Vert STAT\text{-}6_2)$
$IFN\beta\text{-}R = IFN\beta\text{-}R_1 \Vert IFN\beta\text{-}R_2$	$IRAK\text{-} = IRAK_1 \Vert IRAK_2$
$IIFN\beta\text{-}R_1 = \{ifn\beta(x^3 \in \{L\})\}.(\overline{ifn\beta r}\langle L\rangle \Vert IFN\text{-}\beta_1)$	$IRAK_1 = \{il18r(x^5 \in \{L\})\}.(\overline{irak}\langle L\rangle \Vert IRAK_1)$
$IFN\beta\text{-}R_2 = \{ifn\beta(x^3 \in \{H\})\}.(\overline{ifn\beta r}\langle H\rangle \Vert IFN\text{-}\beta_2)$	$IRAK_2 = \{il18r(x^5 \in \{H\})\}.(\overline{irak}\langle H\rangle \Vert IRAK_2)$
$IL\text{-}12R = IL\text{-}12R_1 \Vert IL\text{-}12R_2 \Vert IL\text{-}12R_3$	
$IL\text{-}12R_1 = \{il12(x^1 \in \{L\}), stat6(x^{14} \in \{L,H\})\}.(\overline{il12r}\langle L\rangle \Vert IL\text{-}12R_1)$	
$IL\text{-}12R_2 = \{il12(x^1 \in \{H\}), stat6(x^{14} \in \{H\})\}.(\overline{il12r}\langle L\rangle \Vert IL\text{-}12R_2)$	
$IL\text{-}12R_3 = \{il12(x^1 \in \{H\}), stat6(x^{14} \in \{L\})\}.(\overline{il12r}\langle H\rangle \Vert IL\text{-}12R_3)$	
$IL\text{-}18R = IL\text{-}18R_1 \Vert IL\text{-}18R_2 \Vert IL\text{-}18R_3$	
$IL\text{-}18R_1 = \{il18(x^1 \in \{L\}), stat6(x^{14} \in \{L,H\})\}.(\overline{il12r}\langle L\rangle \Vert IL\text{-}18R_1)$	
$IL\text{-}18R_2 = \{il18(x^1 \in \{H\}), stat6(x^{14} \in \{H\})\}.(\overline{il12r}\langle L\rangle \Vert IL\text{-}18R_2)$	
$IL\text{-}18R_3 = \{il18(x^1 \in \{H\}), stat6(x^{14} \in \{L\})\}.(\overline{il12r}\langle H\rangle \Vert IL\text{-}18R_3)$	
$STAT\text{-}4 = STAT\text{-}4_1 \Vert STAT\text{-}4_2 \Vert STAT\text{-}4_2$	
$STAT\text{-}4_1 = \{il12r(x^4 \in \{H\}), gata3(x^{15} \in \{L,H\})\}.(\overline{il12r}\langle L\rangle \Vert STAT\text{-}4_1)$	
$STAT\text{-}4_2 = \{il12r(x^1 \in \{H\}), gata3(x^{15} \in \{H\})\}.(\overline{il12r}\langle L\rangle \Vert STAT\text{-}4_2)$	
$STAT\text{-}4_3 = \{il12r(x^1 \in \{H\}), gata3(x^{15} \in \{L\})\}.(\overline{il12r}\langle H\rangle \Vert STAT\text{-}4_3)$	
$IFN\text{-}\gamma = IFN\text{-}\gamma_1 \Vert IFN\text{-}\gamma_2 \Vert IFN\text{-}\gamma_3 \Vert IFN\text{-}\gamma_4$	
$IFN\text{-}\gamma_1 = \{stat4(x^7 \in \{L,M\}).irak(x^8 \in \{L,H\}), tbet(x^{17} \in \{L\})\}.(\overline{ifn\gamma}\langle L\rangle \Vert IFN\text{-}\gamma_1)$	
$IFN\text{-}\gamma_2 = \{stat4(x^7 \in \{L,M\}).irak(x^8 \in \{L,H\}), tbet(x^{17} \in \{M\})\}.(\overline{ifn\gamma}\langle M\rangle \Vert IFN\text{-}\gamma_2)$	
$IFN\text{-}\gamma_3 = \{stat4(x^7 \in \{L,M,H\}).irak(x^8 \in \{L,H\}), tbet(x^{17} \in \{H\})\}.(\overline{ifn\gamma}\langle H\rangle \Vert IFN\text{-}\gamma_3)$	
$IFN\gamma\text{-}R = IFN\gamma\text{-}R_1 \Vert IFN\gamma\text{-}R_2 \Vert IFN\gamma\text{-}R_3 \Vert IFN\gamma\text{-}R_4$	
$IFN\gamma\text{-}R_1 = \{ifn\gamma r(x^{10} \in \{L\}), socs1(x^{16} \in \{L,H\})\}.(\overline{ifn\gamma r}\langle L\rangle \Vert IFN\gamma\text{-}R_1$	
$IFN\gamma\text{-}R_2 = \{ifn\gamma r(x^{10} \in \{M\}), socs1(x^{16} \in \{L\})\}.(\overline{ifn\gamma r}\langle M\rangle \Vert IFN\gamma\text{-}R_2$	
$IFN\gamma\text{-}R_3 = \{ifn\gamma r(x^{10} \in \{H,M\}), socs1(x^{16} \in \{H\})\}.(\overline{ifn\gamma r}\langle M\rangle \Vert IFN\gamma\text{-}R_3$	
$IFN\gamma\text{-}R_4 = \{ifn\gamma r(x^{10} \in \{H\}), socs1(x^{16} \in \{L\})\}.(\overline{ifn\gamma r}\langle H\rangle \Vert IFN\gamma\text{-}R_2$	
$STAT\text{-}1 = STAT\text{-}1_1 \Vert STAT\text{-}1_2 \Vert STAT\text{-}1_3 \Vert STAT\text{-}1_4$	
$STAT\text{-}1_1 = \{ifn\beta r(x^6 \in \{L\}), ifn\gamma r(x^{10} \in \{L\})\}.(\overline{stat1}\langle L\rangle \Vert STAT\text{-}1_1)$	
$STAT\text{-}1_2 = \{ifn\beta r(x^6 \in \{H\}), ifn\gamma r(x^{10} \in \{L,M\})\}.(\overline{stat1}\langle M\rangle \Vert STAT\text{-}1_2)$	
$STAT\text{-}1_3 = \{ifn\beta r(x^6 \in \{L\}), ifn\gamma r(x^{10} \in \{M\})\}.(\overline{stat1}\langle M\rangle \Vert STAT\text{-}1_3)$	
$STAT\text{-}1_4 = \{ifn\beta r(x^6 \in \{L,H\}), ifn\gamma r(x^{10} \in \{H\})\}.(\overline{stat1}\langle H\rangle \Vert STAT\text{-}1_4)$	

Table 6
Case Study Specification

Computing the model and applying our CFA: an example

Suppose to have the following initial condition process I with 17 outputs, corresponding to the state $(L, L, H, L, L, L, L, L, L, L, L, L, L, L, L, L, L)$

$$\overline{il12}\langle L\rangle \parallel \overline{il18}\langle L\rangle \parallel \overline{ifn\beta}\langle H\rangle \parallel \overline{il12r}\langle L\rangle \parallel \overline{il18r}\langle L\rangle \parallel \overline{ifn\beta r}\langle L\rangle \parallel \overline{stat4}\langle L\rangle \parallel \overline{irak}\langle L\rangle \parallel$$
$$\overline{ifn\gamma}\langle L\rangle \parallel \overline{ifn\gamma r}\langle L\rangle \parallel \overline{stat1}\langle L\rangle \parallel \overline{il4}\langle L\rangle \parallel \overline{il4r}\langle L\rangle \parallel \overline{stat6}\langle L\rangle \parallel \overline{gata3}\langle L\rangle \parallel \overline{socs1}\langle L\rangle \parallel \overline{tbet}\langle L\rangle$$

We can have the following computation, where we use the shorthand Sys' to denote Sys except for $IFN\text{-}\beta_2$ and $IFN\text{-}\beta R_2$:

$$\begin{aligned} & Sys \parallel I \\ \equiv\ & \{ifn\beta(x^3 \in \{H\})\}.(\overline{ifn\beta}\langle H\rangle \parallel IFN\text{-}\beta_3) \parallel \{ifn\beta(x^3 \in \{H\})\}.(\overline{ifn\beta r}\langle H\rangle \parallel IFN\text{-}\beta R_2) \parallel Sys' \parallel I \\ \equiv\ & Sys \parallel I' \end{aligned}$$

$$\begin{aligned}
&IL\text{-}4 = IL\text{-}4_1 \parallel IL\text{-}4_2 \parallel IL\text{-}4_3\\
&IL\text{-}4_1 = \{stat1(x^{11} \in \{L, M, H\}), gata3 \in \{L\}\}.(\overline{il4}\langle L\rangle \parallel IL\text{-}4_1)\\
&IL\text{-}4_2 = \{stat1(x^{11} \in \{M, H\}), gata3 \in \{H\}\}.(\overline{il4}\langle L\rangle \parallel IL\text{-}4_2)\\
&IL\text{-}4_3 = \{stat1(x^{11} \in \{L\}), gata3 \in \{H\}\}.(\overline{il4}\langle H\rangle \parallel IL\text{-}4_3)
\end{aligned}$$

$$\begin{aligned}
&IL\text{-}4R = IL\text{-}4R_1 \parallel IL\text{-}4R_2 \parallel IL\text{-}4R_3 \parallel IL\text{-}4R_4\\
&IL\text{-}4R_1 = \{il4(x^{12} \in \{L\}), socs1(x^{16} \in \{L, H\})\}.(\overline{il4r}\langle L\rangle \parallel IL\text{-}4R_1)\\
&IL\text{-}4R_2 = \{il4(x^{12} \in \{H\}), socs1(x^{16} \in \{L\})\}.(\overline{il4r}\langle H\rangle \parallel IL\text{-}4R_2)\\
&IL\text{-}4R_3 = \{il4r(x^{13} \in \{H\}), socs1(x^{16} \in \{H\})\}.(\overline{il4r}\langle L\rangle \parallel IL\text{-}4R_3)
\end{aligned}$$

$$\begin{aligned}
&GATA\text{-}3 = GATA\text{-}3_1 \parallel GATA\text{-}3_2 \parallel GATA\text{-}3_3\\
&GATA\text{-}3_1 = \{stat6(x^{14} \in \{L\}), tbet(x^{17} \in \{L\}\}.(\overline{gata3}\langle L\rangle \parallel GATA\text{-}3_1)\\
&GATA\text{-}3_2 = \{stat6(x^{14} \in \{L, H\}), tbet(x^{17} \in \{M, H\}\}.(\overline{gata3}\langle L\rangle \parallel GATA\text{-}3_2)\\
&GATA\text{-}3_3 = \{stat6(x^{14} \in \{H\}), tbet(x^{17} \in \{L\}\}.(\overline{gata3}\langle H\rangle \parallel GATA\text{-}3_3)
\end{aligned}$$

$$\begin{aligned}
&SOCS\text{-}1 = SOCS\text{-}1_1 \parallel SOCS\text{-}1_2 \parallel SOCS\text{-}1_3 \parallel SOCS\text{-}1_4\\
&SOCS\text{-}1_1 = \{stat1(x^{11} \in \{L\}), tbet(x^{17} \in \{L\})\}.(\overline{socs1}\langle L\rangle \parallel SOCS\text{-}1_1)\\
&SOCS\text{-}1_2 = \{stat1(x^{11} \in \{L\}), tbet(x^{17} \in \{M, H\})\}.(\overline{socs1}\langle H\rangle \parallel SOCS\text{-}1_2)\\
&SOCS\text{-}1_3 = \{stat1(x^{11} \in \{M, H\}), tbet(x^{17} \in \{L, M, H\})\}.(\overline{socs1}\langle H\rangle \parallel SOCS\text{-}1_3)
\end{aligned}$$

$$\begin{aligned}
&T\text{-}BET = T\text{-}BET_1 \parallel ... \parallel T\text{-}BET_7\\
&T\text{-}BET_1 = \{stat1(x^{11} \in \{L, M, H\}), gata3(x^{15} \in \{L, H\}), tbet(x^{17} \in \{L\})\}.(\overline{tbet}\langle L\rangle \parallel T\text{-}BET_1\\
&T\text{-}BET_2 = \{stat1(x^{11} \in \{L, H\}), gata3(x^{15} \in \{H\}), tbet(x^{17} \in \{M, H\})\}.(\overline{tbet}\langle L\rangle \parallel T\text{-}BET_2\\
&T\text{-}BET_3 = \{stat1(x^{11} \in \{L\}), gata3(x^{15} \in \{L\}), tbet(x^{17} \in \{M\})\}.(\overline{tbet}\langle M\rangle \parallel T\text{-}BET_3\\
&T\text{-}BET_4 = \{stat1(x^{11} \in \{M\}), gata3(x^{15} \in \{L, H\}), tbet(x^{17} \in \{M\})\}.(\overline{tbet}\langle M\rangle \parallel T\text{-}BET_4\\
&T\text{-}BET_5 = \{stat1(x^{11} \in \{L\}), gata3(x^{15} \in \{L\}), tbet(x^{17} \in \{H\})\}.(\overline{tbet}\langle H\rangle \parallel T\text{-}BET_5\\
&T\text{-}BET_6 = \{stat1(x^{11} \in \{M\}), gata3(x^{15} \in \{L, H\}), tbet(x^{17} \in \{H\})\}.(\overline{tbet}\langle H\rangle \parallel T\text{-}BET_6\\
&T\text{-}BET_7 = \{stat1(x^{11} \in \{H\}), gata3(x^{15} \in \{L\}), tbet(x^{17} \in \{M, H\})\}.(\overline{tbet}\langle H\rangle \parallel T\text{-}BET_7
\end{aligned}$$

Table 7
Case Study Specification (Continued)

where I', that corresponds to the state $(L, L, H, L, L, H, L, L, L, L, L, L, L, L, L, L, L)$, is the new output tuple and it is specified in **Sim**-π_n as follows

$$\begin{aligned}
&\overline{il12}\langle L\rangle \parallel \overline{il18}\langle L\rangle \parallel \overline{ifn\beta}\langle H\rangle \parallel \overline{il12r}\langle L\rangle \parallel \overline{il18r}\langle L\rangle \parallel \overline{ifn\beta r}\langle L\rangle \parallel \overline{stat4}\langle L\rangle \parallel \overline{irak}\langle L\rangle \parallel\\
&\overline{ifn\gamma}\langle L\rangle \parallel \overline{ifn\gamma r}\langle L\rangle \parallel \overline{stat1}\langle L\rangle \parallel \overline{il4}\langle L\rangle \parallel \overline{il4r}\langle L\rangle \parallel \overline{stat6}\langle L\rangle \parallel \overline{gata3}\langle L\rangle \parallel \overline{socs1}\langle L\rangle \parallel \overline{tbet}\langle L\rangle
\end{aligned}$$

Now, we can have the following computation, where Sys'' stands for Sys except for $STAT\text{-}1_2$:

$Sys \parallel I' \equiv \{ifn\beta r(x^6 \in \{H\}), ifn\gamma r(x^{10} \in \{L, M\})\}.(\overline{stat1}\langle M\rangle \parallel STAT\text{-}1_2) \parallel Sys'' \parallel I' \equiv Sys \parallel I''$
where I'', that corresponds to the state $(L, L, H, L, L, H, L, L, L, L, M, L, L, L, L, L, L)$, is the new output tuple and it is specified as follows:

$$\begin{aligned}
&\overline{il12}\langle L\rangle \parallel \overline{il18}\langle L\rangle \parallel \overline{ifn\beta}\langle H\rangle \parallel \overline{il12r}\langle L\rangle \parallel \overline{il18r}\langle L\rangle \parallel \overline{ifn\beta r}\langle H\rangle \parallel \overline{stat4}\langle L\rangle \parallel \overline{irak}\langle L\rangle \parallel\\
&\overline{ifn\gamma}\langle L\rangle \parallel \overline{ifn\gamma r}\langle L\rangle \parallel \overline{stat1}\langle M\rangle \parallel \overline{il4}\langle L\rangle \parallel \overline{il4r}\langle L\rangle \parallel \overline{stat6}\langle L\rangle \parallel \overline{gata3}\langle L\rangle \parallel \overline{socs1}\langle L\rangle \parallel \overline{tbet}\langle L\rangle
\end{aligned}$$

Similarly, we can start with the initial condition process $\hat{I}$, that corresponds to the state $(H, H, H, L, L, L, L, L, L, L, L, L)$ and that it is specified as above. The computation will lead us to have the output tuple, corresponding to the state $(H, H, H, H, H, H, L, L, L, L, L)$ and, in turn, to the sequence of tuples that correspond to the following sequence of

$$\begin{aligned}
\kappa(G)(\epsilon, ..., \epsilon) \supseteq \{ &(L, L, H, L, L, L, L, L, L, L, L, L, L, L, L, L, L), (L, L, H, L, L, H, L, L, L, L, L, L, L, L, L, L, L),\\
&(L, L, H, L, L, H, L, L, L, L, M, L, L, L, L, L, L)\}\\
\kappa(G)(L, L, H, L, L, L, L, L, L, L, L, L, L, L, L, L, L) &\supseteq \{(L, L, H, L, L, H, L, L, L, L, L, L, L, L, L, L, L)\}\\
\kappa(G)(L, L, H, L, L, H, L, L, L, L, L, L, L, L, L, L, L) &\supseteq \{(L, L, H, L, L, H, L, L, L, L, M, L, L, L, L, L, L)\}
\end{aligned}$$

Table 8
Some of the CFA Results of the Case Study

tuples: $(H, H, H, H, H, H, H, H, L, L, M)$, $(H, H, H, H, H, H, H, H, H, L, M)$, $(H, H, H, H, H, H, H, H, H, H, M)$ and finally to the tuple $(H, H, H, H, H, H, H, H, H, H, H)$. Now we apply the CFA to the first composed system $Sys \parallel I$. Some of the results are reported in Table 8.

Verifying model consistency

We have performed a set of *in silico* experiments to verify whether our model is consistent with the corresponding specification presented in [18]. To this aim, we have tested the capability of our model of reproducing peculiar features of the described network, such as the characterisation of attractors, in particular of stable states. In [18] the authors identified four stable states and associated to each of them a biological counterpart. In our framework these states can be represented as

$$\begin{aligned}
&(1)\ (L, L, L, L, L, L, L, L, L, L, L, L, L, L, L, L, L)\\
&(2)\ (L, L, L, L, L, L, L, L, H, M, M, L, L, L, L, H, H)\\
&(3)\ (L, L, L, L, L, L, L, L, M, M, M, L, L, L, L, H, M)\\
&(4)\ (L, L, L, L, L, L, L, L, L, L, L, H, H, H, H, L, L)
\end{aligned}$$

where (1) corresponds to the observed state of the Th0 cell, both (2) and (3) correspond to the observed state of the Th1 cell, and (4) corresponds to the observed state of the Th2 cell.

For all the four cases depicted above, we applied our analysis assuming each stable state as an initial state. We found that no other state can be reached, consistently with [18]. To further verify the viability of our model, we repeated this kind of experimentation on the attractors identified in [18] when models of mutants Th1/Th2 lymphocytes are analysed. These mutant cells exhibit different combinations of null mutations or over activation of some genes resulting in the hypo or hyper production of the corresponding proteins. In our specifications (following [18]) these two kind of mutations are rendered setting to H or L respectively the levels of the proteins codified by the mutated genes. We tested all the 35 mutants considered in [18] and we found that our model is able to describe all the 110 stable states identified for mutants.

Gaining Insights on the studied network

We are now interested in clarifying the role played by the combined effect of the Interleukins IL-12 and IL-18 and of the Interferon-β (IFN-β) on the intracellu-

IL-12	IL-18	IFN-β	$STAT$-1
L	L	L	L
L	L	H	M
H	L	L	M
H	L	H	M
H	H	L	H
H	H	H	H

Table 9
Dependence Table of $STAT$-1 on the Initial Values of IL-12 IL-18 IFN-β

lar molecule $STAT$-1, which is crucial in the intracellular network involved in the Th differentiation process. Monitoring $STAT$-1 levels allows us to obtain information on the global activation state of the pathway. We performed a set of *in silico* experiments starting from different initial conditions, corresponding to the following states: $(L, L, H, L, L, L, L, L, L, L, L, L, L, L, L, L, L)$, $(H, L, L, L, L, L, L, L, L, L, L, L, L, L, L, L, L)$, $(H, H, L, L, L, L, L, L, L, L, L, L, L, L, L, L, L)$, and also $(H, L, H, L, L, L, L, L, L, L, L, L, L, L, L, L, L)$, $(H, H, H, L, L, L, L, L, L, L, L, L, L, L, L, L, L)$.
From a biological point of view, setting these "initial states" corresponds to investigate the behavior of the network, when different experimental conditions are applied. The initial conditions processes specified above, correspond to experiments in which the concentrations of IL-12, IL-18 and IFN-β are initially set to the values *Low* or *High* in different combinations.
For the sake of space, we will not show here all the traces of the computations. We just report a summary table in Table 9, representing how $STAT$-1 depends on the combinations of IL-12 IL-18 IFN-β values. Applying our CFA to the different obtained systems, we can observe that there are pathways that allow $STAT$-1 to reach either *Medium* or *High* concentration values. From a closer observation of the pathways verifying this property, it turns out an important insight on the relative role of IFN-β w.r.t. IL-12 or IL-18. In particular it can be noticed that the initial concentration value of IFN-β is important in determining the final concentration level of $STAT$-1, only when IL-12 or IL-18 are both *Low*. Instead, when the concentrations of IL-12 and IL-18 are *High*, IFN-β can be either *Low* or *High*, without changing the value of $STAT$-1.

7 Conclusions

Using Generalised Boolean Networks to model regulatory networks presents some disadvantages, mainly due to the low number of analysis tools and to the difficulty in handling incomplete or inconsistent data. To overcome these limitations, we have relied on the process algebraic framework. We have indeed translated the logical models in terms of **Sim**-π_n, a novel process algebra. As a result, we have obtained process models which show the same behavior of the original GBNs and that are ready to be analysed with the usual process algebraic tools. In particular, we have applied Control Flow Analysis to the process algebraic specifications, therefore gaining insights on the studied biological system, while paying a low computational cost. We have showed these features through a case study represented by the regulatory network underlying Th lymphocytes differentiations. In particular, we have investigated the role played by the genes composing the network in determining the final state of the system under different experimental conditions. This example have showed how our toolkit can be fruitfully exploited to assess interesting properties of biological systems. Our method could be particularly useful in the analysis of large GBN models (e.g. whole cell models). Even when partial biological knowledge is available, exploiting both **Sim**-π_n and CFA allows us to design and study realistic models of biological systems i.e. models which are consistent with the observed behavior of their biological counterparts. We are planning to use our framework for describing and analysing large-scale models of biological networks, such as cross-talking signalling pathways. Furthermore, we intend to analyse the expressive power of **Sim**-π_n, as done in [17].

References

[1] H. K. Abbas, A. K. Lichtman, S. Pillai. *Cellular and molecular immunology*, 7th edition Elsevier-Saunders, 2009.

[2] R. M. Amadio, F. Dabrowski. *Feasible reactivity in a synchronous Pi-calculus.* Proc. of the 9th International ACM SIGPLAN Conference on Principles and Practice of Declarative Programming (PPDP'07), ACM, 2007.

[3] C. Bodei, P. Degano, F. Nielson, H. Riis Nielson. *Static analysis for the π-calculus with applications to security.* Information and Computation 168(1): 68–92, 2001.

[4] C. Bodei, P. Degano, C. Priami. *Checking security policies through an enhanced Control Flow Analysis* Journal of Computer Security 13(1): 49-85, 2005.

[5] L. Cardelli. Brane Calculi - Interactions of Biological Membranes. *Proc. of Computational Methods in Systems Biology (CMSB'04).* Lecture Notes in Computer Science 3082, pp. 257-278, Springer, 2005.

[6] L. Cardelli and A. D. Gordon. Mobile Ambients. *Theoretical Computer Science* 240(1): 177-213 (2000).

[7] C. Chaouiya, A. Naldi, E. Remy, D. Thieffry. *Petri net representation of multi-valued logical regulatory graphs* Natural Computing 10(2): 727-750, 2011.

[8] D. Chiarugi, P. Degano, R. Marangoni. *A Computational Approach to the Functional Screening of Genomes.* PLoS Computational Biology 3(9), 2007.

[9] V. Danos, Jean Krivine. Transactions in RCCS. *Proc. of Conference on Concurrency Theory (CONCUR'05).* Lecture Notes in Computer Science 3653, pp. 398-412, Springer 2005.

[10] V. Danos, C. Laneve. Graphs for Core Molecular Biology. *Proc. of Computational Methods in Systems Biology (CMSB'03).* Lecture Notes in Computer Science 2602, pp. 34-46, Springer 2003.

[11] V. Danos, J. Feret, W. Fontana, R. Harmer, J. Krivine. *Rule-based modelling of cellular signalling.* Proc. of Conference on Concurrency Theory (CONCUR'07). Lecture Notes in Computer Science, Springer, 2007.

[12] H. de Jong. *Modeling and simulation of genetic regulatory Systems : A literature review.* Journal of Computational Biology, 9(1):69-105, 2002.

[13] J. Fisher, T. A. Henzinger. *Executable cell biology.* Nature Biotechnology 25(11), 2005.

[14] P. Goss, J. Pecked. *Quantitative modeling of stochastic systems in molecular biology by using stochastic Petri Nets.* Proc. Natl Acad. Sci. USA 95, 1998.

[15] R. Hofestadt, S. Helen. *Quantitative modeling of biochemical networks.* In Silico Biology 1, 1998.

[16] S.A. Kauffman. *The Origins of Order: Self-Organization and Selection in Evolution.* Oxford University Press, New York, 1993.

[17] C. Laneve, A. Vitale. *The Expressive Power of Synchronizations.* Proc. of Logic in Computer Science (LICS 2010), pag. 382-391.

[18] L. Mendoza. *A network model for the control of the differentiation process in Th cells.* BioSystems 84:101-114, 2005

[19] R. Milner. *Communicating and mobile systems: the π-calculus.* Cambridge University Press, 1999.

[20] R. Milner. *Communication and Concurrency.* Prentice-Hall, 1989.

[21] H. R. Nielson and F. Nielson. Flow logic: A multi-paradigmatic approach to static analysis. In *The Essence of Computation*, Lecture Notes in Computer Science 2566, pp. 223–244. Springer, 2002.

[22] C. Priami and P. Quail. Beta-binders for Biological Interactions. *Proc. of Computational Methods in Systems Biology (CMSB'04)*, Lecture Notes in Computer Science 3380, pp. 20–33, Springer, 2005.

[23] C. Priami, A. Regev, E. Y. Shapiro, W. Silverman. Application of a stochastic name-passing calculus to representation and simulation of molecular processes. *Inf. Process. Lett.* 80(1): 25-31 (2001).

[24] A. Regev, E. M. Panina, W. Silverman, L. Cardelli, and E. Y. Shapiro. BioAmbients: An abstraction for biological compartments. *Theoretical Computer Science* 325(1): 141-167. Elsevier, 2004.

[25] A. Regev, W. Silverman, E. Y. Shapiro. Representation and Simulation of Biochemical Processes Using the Pi-Calculus Process Algebra. *Pacific Symposium on Biocomputing*, pp. 459-470, 2001.

[26] W. Reisig. *Petri Nets: an Introduction.* Springer-Verlag, 1985.

[27] L. J. Steggles, R. Banks and A. Wipeout. *Modelling and Analysing Genetic Networks: From Boolean Networks to Petri Nets.* Proc. of Conference on Computational Methods in Systems Biology (CMSB'06), Lecture Notes in Computer Science 4210, Springer, 2006.

[28] L. J. Steggles, R. Banks and A. Wipeout. *An Abstraction Theory for Qualitative Models of Biological Systems.* Proc. of Membrane Computing and Biologically Inspired Process Calculi (MeCBIC'10), Electronic Proceedings in Theoretical Computer Science 40: 23-38, 2010.

[29] D. Thieffry, R. Thomas. *Qualitative Analysis of Gene Networks.* Pacific Symposium on Biocomputing, 3: 77-88, 1998.

[30] R. Thomas. *Regulatory networks seen as asynchronous automata: A logical description.* Journal of Theoretical Biology, 153:1-23,1991.

[31] R. Thomas. *The Role of feedback circuits: positive feedback circuits are a necessary condition for positive real eigenvalues in the jacobian matrix.* Physical Chemistry, 98 (1148), 1994.

[32] R. Thomas, D. Thieffry, and M. Kaufman. *Dynamical behavior of biological regulatory networks: I. Biological role of feedback loops and practical use of the concept of the loop-characteristic state.* Bull. Math. Biol. 57(2), 247-276, 1995.

[33] R. Thomas and R. D'Ari. *Biological Feedback.* CRC Press, Boca Raton, Florida, 1990.