Attractor and Slicing Analysis of a T Cell Differentiation Model Based on Reaction Systems*

Linda Brodo[1], Roberto Bruni[2], Moreno Falaschi[3], Roberta Gori[2], and Paolo Milazzo[2]

[1] Dipartimento di Scienze Economiche e Aziendali, University of Sassari
Via Muroni 25, 07100 Sassari, Italy. Email: brodo@uniss.it
[2] Department of Computer Science, University of Pisa
Largo B. Pontecorvo 3, 56127 Pisa, Italy
Email: {roberto.bruni,roberta.gori,paolo.milazzo}@unipi.it
[3] Department of Information Engineering and Mathematics, University of Siena
Via Roma 56, 53100 Siena, Italy. Email: moreno.falaschi@unisi.it

Abstract. Reaction Systems (RSs) were introduced in the field of natural computing as a qualitative model inspired by biological systems. In a RS, each reaction comprises a set of reactants that generate products unless inhibited by reaction inhibitors. We propose a method for analyzing the *attractors* of a RS model (the states to which the system converges) and for identifying the entities responsible for their attainment. Our approach builds on (i) the construction of a Labeled Transition System (LTS) based on a formal SOS semantics of RSs, and (ii) the application of a slicing method to the trajectories within the LTS. Our model analysis provides new insights with respect to previous studies. We illustrate our methodology on a case study that demonstrates its capability to identify stimulus combinations that lead to specific phenotypes, but also to elucidate the involvement of proteins within T cells in each scenario. These findings allow for a better understanding of the phenomenon and for the identification of potential drug targets for diseases.

Keywords: Systems Biology · LTS · Slicing · Attractors.

1 Introduction

Reaction Systems (RSs) [17,18] are a successful computational framework inspired by biochemical systems. They offer a broad and flexible platform for constructing qualitative models, abstracting away the specific count of modeled

* Supported by project "MEDICA" (2022RNTYWZ) funded by MUR-PRIN_2022, by PNRR ECS00000017 - "THE - Tuscany Health Ecosystem" - Spokes 3 and 6 - CUP I53C22000780001 funded by the EC under the NextGeneration EU programme, by the Italian MUR PRIN 2022 PNRR project *Resource Awareness in Programming: Algebra, Rewriting, and Analysis* (P2022HXNSC), and by the INdAM–GNCS project CUP_E53C22001930001

entities (e.g., molecules). A RS comprises a finite set of reactions, where each reaction is a triple (R, I, P), with R, I, and P being finite, non-empty sets of entities defined over a fixed finite set S called the *background set*. A computation in a RS begins with an initial state (a finite set of entities) and, at each step, repeatedly generates new entities by applying all enabled reactions to the current state. Each reaction (R, I, P) contributes to the new state only if all entities in R (the reactants) and no element in I (the inhibitors) are present in the current state. If these two conditions are met, the set P (the products) is included in the next state. The union of the products of all reactions determines the new state.

RSs are an interactive model of computation that can be fed with an ordered sequence of states (sets of entities) called *context sequence*. Then, the i-th state in a computation will be determined by the result of the application of the reactions to the previous state, together with the set of entities in the i-th set of the context sequence.

In this paper, we present a methodology for analyzing the attractors of a RS model, which represent the set of states towards which the system converges, and for identifying biological entities responsible for their attainment. Exploiting the operational semantics defined in [11], our method starts by computing a Labeled Transition System (LTS) describing all the possible behaviours of the RS model. Subsequently, we employ an automated methodology, known as slicing (see [12]), to simplify the trajectories described by the LTS.

To demonstrate the effectiveness of our methodology, we apply it to a case study, focusing on the analysis of a model representing T cell differentiation in the immune system. This model encompasses reactions enabling T cells to manifest various phenotypes in response to environmental stimuli. Our analysis unveils novel insights compared to previous studies, revealing the combinations of stimuli associated with each phenotype and identifying the specific proteins within T cells implicated in each scenario.

Several tools find fruitful application in our study. The first is BioReSolve [5], a freely available tool that we have previously developed: it implements RSs in SWI-Prolog, along with a slicing analysis method. In order to develop the analysis of attractors, we have designed and implemented a new Python script leveraging the `networkx` package. Our script interacts with BioReSolve through the Python-to-Prolog binding facilitated by the `swiplserver` Python package.

Roughly, our methodology consists of the following steps: firstly, we utilise BioReSolve to run our RS model, obtained through automatic translation from the Boolean network to the corresponding RS. Subsequently, we add the contexts which are necessary to start a computation in BioReSolve by considering a non-deterministic context which provides all combinations of entities/stimuli as different initial alternatives. The computation proceeds along each path by repeating the same initial context of that path until reaching a loop in the computation graph. BioReSolve returns a representation of the LTS as a graph in dot format, which can be loaded by our Python script. Our Python script conducts an analysis of the attractors, considering different *targets* corresponding to the different combinations of phenotypes that a T cell can exhibit. The slicing

tool, which performs a form of causality analysis, is used to simplify the computation traces, retaining only the information pertinent to producing a *target* of our analysis. This enables us to uncover significant facts, such as identifying the genes necessary for the expression of a target transcription factor.

Our methodology is general, and hence it could be applied to many other case studies, such as the ones in the public data base *Cell Collective* [28], which contains our current case study. Moreover, our methodology introduces an original analysis of attractors combined with a slicing algorithm that provides a deeper understanding of the evolution of the system and the entities involved. This can have a significant impact in drug design, since understanding which genes are necessary or, more generally, contribute to the expression of some other genes or transcription factors involved in a disease can be helpful for the identification of potential drugs or vaccine targets.

Related work. This paper contributes to the research in computational modelling and analysis of gene regulatory networks [22,4]. In particular, the attractors and slicing analyses, combined together, constitute a new methodology for the investigation of causality properties in such networks. Causality properties can be studied by running simulations [24,29], to investigate the role of initial configurations of active genes and identify attractors. Moreover, formal causality approaches have been investigated for biological applications in [16,7], and in particular for RSs in [1,2]. Compared to these approaches, ours provides information about the role of both the environmental stimuli (or the "input" nodes of the networks) and the intermediate genes involved in the achievement of a given target, computed in an integrated way and with scalable performances.

Structure of the paper. in Section 2, we present the basic framework of RSs and the slicing algorithm. Then, Section 3 presents the implementation of our methodology. In Section 4 we define the RS model of T cell differentiation whose analysis is described in Section 5. We draw some concluding remarks and sketch future work in Section 6.

2 Reaction Systems and Slicing Analysis

In this section we recall background notions about the Reaction Systems, as well as the definition of a SOS semantics and of a slicing analysis method.

2.1 Reaction Systems

The theory of Reaction Systems (RSs) [9] was born in the field of natural computing to model the qualitative behavior of biochemical reactions in living cells. We recall here the main concepts and we use the term *entities* to denote generic molecular substances (e.g., atoms, ions, molecules) that may be present in the states of a biochemical system.

Let S be a finite set of entities, a reaction in S is a triple $r = (R, I, P)$, where $R, I, P \subseteq S$ are non empty sets and $R \cap I = \emptyset$. The sets R, I, P are the *reactants*, *inhibitors*, and *products*, respectively. All reactants have to be present in the current state for the reaction to take place. The presence of any of the inhibitors blocks the reaction. Products are the outcome of the reaction, to be released in the next state. Given a state $W \subseteq S$, the result of a reaction $r = (R, I, P)$ on W, denoted by $res_r(W)$, is defined as follows, where $en_r(W)$ is called the *enabling predicate*:

$$res_r(W) \triangleq \begin{cases} P & \text{if } en_r(W) \\ \emptyset & \text{otherwise} \end{cases} \qquad en_r(W) \triangleq R \subseteq W \ \wedge \ I \cap W = \emptyset$$

A Reaction System is a pair $\mathcal{A} = (S, A)$ where S is the set of entities, and A is a finite set of reactions over S. Given $W \subseteq S$, the result of the reactions A on W, denoted $res_A(W)$, is $res_A(W) \triangleq \bigcup_{r \in A} res_r(W)$.

Since living cells can react to environmental stimuli, the behavior of a RS $\mathcal{A} = (S, A)$ is formalized as an *interactive process*. An n-step *interactive process* in $\mathcal{A}$, with $n \geq 0$, is a pair $\pi = (\gamma, \delta)$ s.t. $\gamma = \{C_i\}_{i \in [0,n]}$ is the *context sequence* and $\delta = \{D_i\}_{i \in [0,n]}$ is the *result sequence*, where $C_i, D_i \subseteq S$ for $i \in [0, n]$, $D_0 = \emptyset$, and $D_{i+1} = res_A(D_i \cup C_i)$ for $i \in [0, n-1]$. The context sequence γ represents the environment, while the result sequence δ is determined by γ and A. We call $\tau = W_0, \ldots, W_n$ the *state sequence*, with $W_i \triangleq C_i \cup D_i$ for $i \in [0, n]$.

Example 1. We consider a toy RS defined as $\mathcal{A} \triangleq (S, A)$ where $S \triangleq \{\mathsf{a}, \mathsf{b}, \mathsf{c}\}$, and the set $A \triangleq \{r_1\}$ only contains the reaction $r_1 \triangleq (\{\mathsf{a}, \mathsf{b}\}, \{\mathsf{c}\}, \{\mathsf{b}\})$ (written as $(\mathsf{ab}, \mathsf{c}, \mathsf{b})$ for short). Then, we consider a 4-steps interactive process $\pi \triangleq (\gamma, \delta)$, where $\gamma \triangleq \{C_0, C_1, C_2, C_3\}$, with $C_0 \triangleq \{\mathsf{a}, \mathsf{b}\}$, $C_1 \triangleq \{\mathsf{a}\}$, $C_2 \triangleq \{\mathsf{c}\}$, and $C_3 \triangleq \{\mathsf{c}\}$; and $\delta \triangleq \{D_0, D_1, D_2, D_3\}$, with $D_0 \triangleq \emptyset$, $D_1 \triangleq \{\mathsf{b}\}$, $D_2 \triangleq \{\mathsf{b}\}$, and $D_3 \triangleq \emptyset$. Then, the resulting state sequence is $\tau = W_0, W_1, W_2, W_3 = \{\mathsf{a}, \mathsf{b}\}, \{\mathsf{a}, \mathsf{b}\}, \{\mathsf{b}, \mathsf{c}\}, \{\mathsf{c}\}$. In fact, it is easy to check that, e.g., $W_0 = C_0$, $D_1 = res_A(W_0) = res_A(\{\mathsf{a}, \mathsf{b}\}) = \{\mathsf{b}\}$ because $en_{r_1}(W_0)$, and $W_1 = C_1 \cup D_1 = \{\mathsf{a}\} \cup \{\mathsf{b}\} = \{\mathsf{a}, \mathsf{b}\}$.

2.2 SOS Rules for Reaction Systems

Our methodology exploits the algebraic syntax and the formal semantics for RSs introduced in [11]. As in process algebras such as CCS [27], SOS inference rules define the behavior of each operator. This induces a LTS semantics for RSs, where states are terms of the algebra, transitions correspond to steps of the RS, and their labels retain information on the entities needed to perform each step.

Definition 1 (RS processes). *Let S be a finite set of entities. A* RS process P *is any term defined by the following grammar:*

$$\mathsf{P} := [\mathsf{M}] \qquad \mathsf{M} := (R, I, P) \mid D \mid \mathsf{K} \mid \mathsf{M} | \mathsf{M} \qquad \mathsf{K} ::= \mathbf{0} \mid X \mid C.\mathsf{K} \mid \mathsf{K} + \mathsf{K} \mid \mathsf{rec}\ X.\ \mathsf{K}$$

where $R, I, P, C, D \subseteq S$, $R, I, P \neq \emptyset$, and X belongs to the set of process variables.

A RS process P embeds a *mixture* process M obtained as the parallel composition of some reactions (R, I, P), some set of current entities D (possibly the empty set), and some *context* K. We write $\prod_{j\in J} \mathsf{M}_j$ for the parallel composition of all M_j with $j \in J$. A context process K is a possibly non-deterministic and recursive system: the nil context $\mathbf{0}$ stops the computation; the prefixed context $C.\mathsf{K}$ makes the entities C available to the reactions, and then leaves K be the context offered at the next step; the non-deterministic choice $\mathsf{K}_1 + \mathsf{K}_2$ allows the context to behave as either K_1 or K_2; X is a process variable, and $\mathsf{rec}\ X.\ \mathsf{K}$ is the usual recursive operator of process algebras.

We say that P and P' are structurally equivalent, written $\mathsf{P} \equiv \mathsf{P}'$, when they denote the same term up to the laws of commutative monoids (unit, associativity and commutativity) for parallel composition $\cdot|\cdot$, with $\emptyset$ as the unit, and the laws of idempotent and commutative monoids for choice $\cdot + \cdot$, with $\mathbf{0}$ as the unit. We also assume $D_1|D_2 \equiv D_1 \cup D_2$ for any $D_1, D_2 \subseteq S$.

Definition 2 (RSs as RS processes). *Let $\mathcal{A} = (S, A)$ be a RS, and $\pi = (\gamma, \delta)$ an n-step interactive process in $\mathcal{A}$, with $\gamma = \{C_i\}_{i\in[0,n]}$ and $\delta = \{D_i\}_{i\in[0,n]}$. For any step $i \in [0, n]$, the corresponding RS process $[\![\mathcal{A}, \pi]\!]_i$ is defined as follows:*

$$[\![\mathcal{A}, \pi]\!]_i \triangleq \left[\prod_{r\in A} r \mid D_i \mid \mathsf{K}_{\gamma^i}\right]$$

where $\mathsf{K}_{\gamma^i} \triangleq C_i.C_{i+1}.\cdots.C_n.\mathbf{0}$ is the serialization of the entities offered by γ^i (the shifting of γ at the i-th step). We write $[\![\mathcal{A}, \pi]\!]$ as a shorthand for $[\![\mathcal{A}, \pi]\!]_0$.

Example 2. The encoding of the RS $\mathcal{A} = (S, A)$, in Example 1, is as follows:

$$\mathsf{P} \triangleq [\![\mathcal{A}, \pi]\!] = [\![(\{\mathsf{a}, \mathsf{b}, \mathsf{c}\}, \{(\mathsf{ab}, \mathsf{c}, \mathsf{b})\}), \pi]\!] = [(\mathsf{ab}, \mathsf{c}, \mathsf{b}) \mid \emptyset \mid \mathsf{K}_\gamma] \equiv [(\mathsf{ab}, \mathsf{c}, \mathsf{b}) \mid \mathsf{K}_\gamma]$$

where $\mathsf{K}_\gamma = \{\mathsf{a}, \mathsf{b}\}.\{\mathsf{a}\}.\{\mathsf{c}\}.\{\mathsf{c}\}.\mathbf{0}$ (written as $\mathsf{ab.a.c.c.}\mathbf{0}$ for short), and $D_0 = \emptyset$ can be discarded thanks to structural congruence.

A transition label ℓ is a tuple $\langle (D, C) \rhd R, I, P\rangle$ with $D, C, R, I, P \subseteq S$. The sets D, C record the entities currently in the system; R records entities whose presence is assumed (acting as reactants or as inhibitors); I records entities whose absence is assumed (acting as inhibitors or as missing reactants); P records the products of enabled reactions.

The operational semantics of RS processes is defined by the SOS rules in Fig. 1. Process $\mathbf{0}$ does nothing. Rule (*Ent*) makes available the entities in the (possibly empty) set D, then reduces to $\emptyset$. In rule (*Cxt*), a prefixed context process $C.\mathsf{K}$ makes available the entities in C and then reduces to K. (*Rec*) is the classical rule for recursion. Rules (*Suml*) and (*Sumr*) select a move of either the left or the right component, resp., discarding the other one. Rule (*Pro*) executes the reaction (R, I, P) (its reactants, inhibitors, and products are recorded in the label), and remains available at the next step together with P. Rule (*Inh*) applies if the reaction (R, I, P) can not be executed; the label records the causes that disable the reaction: some inhibitors ($J \subseteq I$) are present or some reactants ($Q \subseteq R$) are missing, with $J \cup Q \neq \emptyset$. The rule (*Par*) puts two processes in

$$\frac{}{D \xrightarrow{\langle (D,\emptyset)\rhd\emptyset,\emptyset,\emptyset\rangle} \emptyset}\ (Ent) \qquad \frac{}{C.\mathsf{K} \xrightarrow{\langle (\emptyset,C)\rhd\emptyset,\emptyset,\emptyset\rangle} \mathsf{K}}\ (Cxt)$$

$$\frac{\mathsf{K}_1 \xrightarrow{\ell} \mathsf{K}_1'}{\mathsf{K}_1 + \mathsf{K}_2 \xrightarrow{\ell} \mathsf{K}_1'}\ (Suml) \qquad \frac{\mathsf{K}_2 \xrightarrow{\ell} \mathsf{K}_2'}{\mathsf{K}_1 + \mathsf{K}_2 \xrightarrow{\ell} \mathsf{K}_2'}\ (Sumr) \qquad \frac{\mathsf{K}[^{\mathsf{rec}\ X.\mathsf{K}}/_X] \xrightarrow{\ell} \mathsf{K}'}{\mathsf{rec}\ X.\ \mathsf{K} \xrightarrow{\ell} \mathsf{K}'}\ (Rec)$$

$$\frac{}{(R,I,P) \xrightarrow{\langle (\emptyset,\emptyset)\rhd R,I,P\rangle} (R,I,P) \mid P}\ (Pro) \qquad \frac{J \subseteq I \quad Q \subseteq R \quad J \cup Q \neq \emptyset}{(R,I,P) \xrightarrow{\langle (\emptyset,\emptyset)\rhd J,Q,\emptyset\rangle} (R,I,P)}\ (Inh)$$

$$\frac{\mathsf{M}_1 \xrightarrow{\ell_1} \mathsf{M}_1' \quad \mathsf{M}_2 \xrightarrow{\ell_2} \mathsf{M}_2' \quad \ell_1 \frown \ell_2}{\mathsf{M}_1 \mid \mathsf{M}_2 \xrightarrow{\ell_1 \cup \ell_2} \mathsf{M}_1' \mid \mathsf{M}_2'}\ (Par) \qquad \frac{\mathsf{M} \xrightarrow{\langle (D,C)\rhd R,I,P\rangle} \mathsf{M}' \quad R \subseteq D \cup C}{[\mathsf{M}] \xrightarrow{\langle (D,C)\rhd R,I,P\rangle} [\mathsf{M}']}\ (Sys)$$

Fig. 1: SOS semantics of the RS processes.

parallel by joining all the set components of the labels. The check $\ell_1 \frown \ell_2$ guarantees that reactants and inhibitors are consistent (we let $W_i \triangleq D_i \cup C_i$):

$$\langle (D_1,C_1)\rhd R_1,I_1,P_1\rangle \frown \langle (D_2,C_2)\rhd R_2,I_2,P_2\rangle \triangleq (W_1 \cup W_2 \cup R_1 \cup R_2) \cap (I_1 \cup I_2) = \emptyset$$

In the conclusion of rule (Par) $\ell_1 \cup \ell_2$ stands for the component-wise union of labels (see definition below, where the notation $X_{1,2} \triangleq X_1 \cup X_2$ is used):

$$\langle (D_1,C_1)\rhd R_1,I_1,P_1\rangle \cup \langle (D_2,C_2)\rhd R_2,I_2,P_2\rangle \triangleq \langle (D_{1,2},C_{1,2})\rhd R_{1,2},I_{1,2},P_{1,2}\rangle$$

Rule (Sys) requires that all the processes of the system have been considered, and checks that all the needed reactants are available in the system ($R \subseteq D \cup C$).

Example 3. The RS process $\mathsf{P}_0 \triangleq [(\mathsf{ab},\mathsf{c},\mathsf{b}) \mid \mathsf{ab.a.c.c.}\mathbf{0}]$ from Example 2, and its next state P_1 have a unique outgoing transition:

$$[(\mathsf{ab},\mathsf{c},\mathsf{b}) \mid \mathsf{ab.a.c.c.}\mathbf{0}] \xrightarrow{\langle (\emptyset,\mathsf{ab})\rhd \mathsf{ab},\mathsf{c},\mathsf{b}\rangle} [(\mathsf{ab},\mathsf{c},\mathsf{b}) \mid \mathsf{b} \mid \mathsf{a.c.c.}\mathbf{0}] \xrightarrow{\langle \mathsf{b},\mathsf{a}\rhd \mathsf{ab},\mathsf{c},\mathsf{b}\rangle} [(\mathsf{ab},\mathsf{c},\mathsf{b}) \mid \mathsf{b} \mid \mathsf{c.c.}\mathbf{0}]$$

From $\mathsf{P}_2 = [(\mathsf{ab},\mathsf{c},\mathsf{b}) \mid \mathsf{b} \mid \mathsf{c.c.}\mathbf{0}]$, three transitions lead to the same state $\mathsf{P}_3 \triangleq [(\mathsf{ab},\mathsf{c},\mathsf{b}) \mid \mathsf{c.}\mathbf{0}]$, each with a different cause why reaction $(\mathsf{ab},\mathsf{c},\mathsf{b})$ is not enabled:

1. $\mathsf{P}_2 \xrightarrow{\langle \mathsf{b},\mathsf{c}\rhd \mathsf{c},\emptyset,\emptyset\rangle} \mathsf{P}_3$ the presence of c inhibits the reaction;
2. $\mathsf{P}_2 \xrightarrow{\langle \mathsf{b},\mathsf{c}\rhd \emptyset,\mathsf{a},\emptyset\rangle} \mathsf{P}_3$ the absence of a inhibits the reaction;
3. $\mathsf{P}_2 \xrightarrow{\langle \mathsf{b},\mathsf{c}\rhd \mathsf{c},\mathsf{a},\emptyset\rangle} \mathsf{P}_3$ this label summarizes the two previous ones.

From now on, we assume transitions $\mathsf{P} \xrightarrow{\langle (D,C)\rhd R,I,P\rangle} \mathsf{P}'$ to maximize the sets J and Q in applying rule (Inh) (see [11]). In [11], Theorem 19, we have shown that the set-theoretic dynamics of a RS matches the SOS semantics of its RS process.

2.3 Attractors

In biological systems, attractors depict an equilibrium dynamics where a finite set of states repeats infinitely in an ordered manner. Thus, in RSs an attractor is defined as a cyclic set of states, whose behaviour is repeated identically forever or until an external condition changes and differently stimulates the system. The number of states composing an attractor defines its size.

2.4 SOS Rules for a Slicing Computation

A slicing computation, presented in the next subsection, needs a slightly different state configuration that includes the whole past state sequence $W_i = C_i \cup D_i$. To do that, we add as a prefix to the state configuration $[\mathsf{M}]$ the list of the previous result and context sets, written $(\mathsf{D}, \mathsf{C})[\mathsf{M}]$, where (D, C) stands for the list $(D_0, C_0), \ldots, (D_n, C_n)$. Definition 1 is thus updated to carry on such a history.

Definition 3 (RS processes with history). *Let $\mathcal{A} = (S, A)$ be a RS, and $\pi = (\gamma, \delta)$ an n-step interactive process in $\mathcal{A}$, with $\gamma = \{C_i\}_{i\in[0,n]}$ and $\delta = \{D_i\}_{i\in[0,n]}$. For any step $i \in [0, n]$, the new process configuration $[\![\mathcal{A}, \pi]\!]_i$ is defined as:*

$$[\![\mathcal{A}, \pi]\!]_i \triangleq (\mathsf{D}, \mathsf{C})[\mathsf{M}]$$

where $(\mathsf{D}, \mathsf{C}) \equiv (D_0, C_0), \ldots, (D_{i-1}, C_{i-1})$, and, $[\mathsf{M}] = \left[\prod_{r\in A} r \mid D_i \mid \mathsf{K}_{\gamma^i}\right]$.

The next step consists in enriching the operational semantics to deal with the history. We only have to modify the (Sys) inference rule in Figure 1 as follows:

$$\frac{\mathsf{M} \xrightarrow{\langle (D,C) \triangleright R, I, P\rangle} \mathsf{M}' \quad R \subseteq D \cup C}{(\mathsf{D}, \mathsf{C})[\mathsf{M}] \xrightarrow{\langle (D,C) \triangleright R, I, P\rangle} (\mathsf{D}, \mathsf{C}) :: (D, C)[\mathsf{M}']} \ (HistSys)$$

where, given the history $(\mathsf{D}, \mathsf{C}) = (D_0, C_0), \ldots, (D_{i-1}, C_{i-1})$, we let the notation $(\mathsf{D}, \mathsf{C}) :: (D, C)$ stand for the history $(D_0, C_0), \ldots, (D_{i-1}, C_{i-1}), (D, C)$.

2.5 Slicing RS Computations

In the context of programming, dynamic slicing is a technique that helps a user to debug a program by simplifying a partial execution trace, by pruning parts which are irrelevant and highlighting parts of the program which have been wrongly ignored during execution. Our slicing technique for RSs is defined in [12]. We resume it here. The technique consists of three steps.

Enriched Semantics (Step S1). The slicing process requires some extra information from the execution of the processes. More precisely, (1) at each operational step we need to highlight the reactions that have been applied; and (2) we need to determine the part of the context which adds to the previous state the entities which are necessary to produce the marked entities in the following state. For solving (1) and (2), in Section 2.4 we have introduced an enriched semantics that records computation sequences. We need to keep track of the state sequence of the computation for the slicing process, by keeping separated the produced entities D_i in a computation step from the context C_i.

Marking the state (Step S2). Let the final configuration in a partial computation be (D_m, C_m). The user (or a logical formula expressed by a monitor [12]) selects a subset $D_{sliced} \subseteq D_m$ that may explain the behavior of the program.

Trace Slice (Step S3). Starting from the pair (D_{sliced}, C_m) denoting the user's (or monitor) marking, we get a slicing step deleting from the execution trace all information not related to D_{sliced}.

Input: - a trace $(D_0, C_0) \xrightarrow{N_1} \cdots \xrightarrow{N_m} (D_m, C_m)$
- a marking $D_{sliced} \subseteq D_m$

Output: a sliced trace $(D'_0, C'_0) \xrightarrow{N'_1} \cdots \xrightarrow{N'_m} (D_{sliced}, C_m)$

begin
  let $D'_m = D_{sliced}$
  for $i = m$ *to* 1 **do**
    let $D'_{i-1} = \emptyset \wedge C'_{i-1} = \emptyset \wedge N'_i = \emptyset$
    for $j \in N_i$ *where* $r_j = (R_j, I_j, P_j)$, *such that* $(D'_i \cap P_j \neq \emptyset)$ **do**
      let $N'_i = N'_i \cup \{j\}$
      if $\neg en_{r_j}(D_{i-1})$, **then**
      $C'_{i-1} = C'_{i-1} \cup (R_j \backslash D_{i-1}) \wedge D'_{i-1} = D'_{i-1} \cup \{R_j \cap D_{i-1}\}$
      else $D'_{i-1} = D'_{i-1} \cup R_j$
    end
  end
end

Algorithm 1: Trace Slicer

Marking algorithm. Let us now explain how the slicing Algorithm 1 works. We assume that the reactions are numbered consecutively by positive integer numbers, and denote the j-th reaction in the RS by the notation r_j. Please notice also that each history $(D_0, C_0) :: \cdots :: (D_m, C_m)$ computed in m steps by Definition 3, defines a *trace* $(D_0, C_0) \xrightarrow{N_1} \cdots \xrightarrow{N_m} (D_m, C_m)$ on which we perform the slicing computation, where N_i is the set of reactions applied in the i-th computation step. Here, each reaction is represented by its numeric position in the list of reactions, i.e., $N_i = \{j \mid en_{r_j}(D_{i-1} \cup C_{i-1})\}$ for any $i \in [1, m]$. Abusing the notation, we write $r_j \in N$ whenever $j \in N$.

Our algorithm returns a sliced trace which contains only the (usually small) subset of the entities which are necessary for deriving the marked entities. Let us consider the more complex case of context dependent computations. First of all, the user (or a logic formula expressed by a monitor [12]) has to indicate the subset D_{sliced} of the entities in the last state of the computation D_m that she wants to mark. Then, the backward slicing process can start. Now, let us consider the iteration i of the slicer. Marking the relevant information in previous state (D_{i-1}, C_{i-1}) requires analyzing the rules which have been applied at step $i-1$. So, if $r_j \in N_{i-1}$ and $r_j = (R_j, I_j, P_j)$, we need to check if r_j produces at least one entity which is marked in the next state. If this is the case, j is added to the set of marked rules. Then, it is necessary to check if the context C_{i-1} was essential for applying rule r_j, or if all necessary entities were already included in D_{i-1}. Thus, it is necessary to compute the entities in C_{i-1} which are missing in D_{i-1} in order for rule r_j to be enabled, and those entities are marked in (added to) context C'_{i-1}. The elements in R_j are added to the marked entities in D'_{i-1}.

3 Implementation of the Analyzer

Our experimentation exploits BioReSolve [5], a tool implementing the SOS semantics of RSs and the slicing analysis methods described in the previous section.

It has been developed in SWI-Prolog [31] by some of the authors of this paper, according to the specifications given in [10] and [12].

To apply our method in the analysis of gene regulatory networks, we interface with BioReSolve through a Python script. This script makes it possible (i) to exploit Python packages such as `networkx` to process the LTS generated by BioReSolve for identifying attractors, and (ii) to automate the execution of the slicing analysis method for multiple gene targets and LTS states. This interaction is made possibile by the usage of the `swiplserver` Python package, allowing to run SWI-Prolog code via Python function invocations.

More precisely, the Python script performs the following steps:

1. it loads an LTS generated by BioReSolve in dot format and uses it to create a `networkx DiGraph` object;
2. it initializes the list of targets of the analysis as a list of objects of a class `Target` with two fields, `present` and `absent`, denoting the genes that have to be expressed and non expressed, respectively, in the states of attractors that have to be considered as target;
3. it defines a few functions that will be used during the attractor analysis:
   - function `check_node(node)` checks whether the LTS state `node` belongs to an attractor of the LTS (i.e., it is in a cycle);
   - function `compute_attractor(node)` returns the list of genes that are present in states of the attractor reachable from state `node`[4];
   - function `target_computations(target)` selects the LTS traces that lead to attractors in target `target`, and for each of them it provides a description of the corresponding context sequences;
4. it executes the main loop that, for every target to be investigated, invokes the `target_computation` function and then for every state of every attractor of such a target, asks BioReSolve to compute the sliced computation leading to that state. Entities mentioned in a sliced computation are then collected in a set representing the entities that actually played a role for the presence of the target genes in the attractor. Finally, the set of entities obtained for each target are used to compute two results: the union of all of them (representing those entities that *may* play a role to achieve the target) and the intersection of all of them (representing those entities that *must* play a role).

The Python script we developed, along with the BioReSolve specification of the RS model to be analysed in this paper, are freely available at [19].

4 RS Model of T Cell Differentiation

In this section we define the RS model of T cell differentiation on which we demonstrate our attractor and slicing analyses. The model is based on the Boolean network model proposed in [29].

[4] We will show that the LTS for the analysis described in this paper are such that every state, apart from the initial one, leads to exactly one attractor.

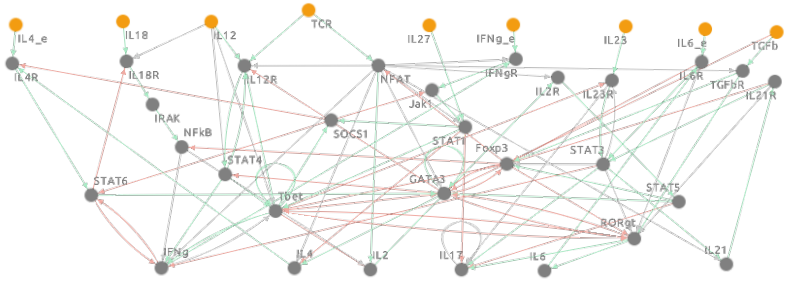

Fig. 2: Graphical representation of the Boolean network model of T Cell differentiation from [29]. The model is available in electronic format at [28].

4.1 The Boolean network model

T cell differentiation is a widely studied biological phenomenon for which several Boolean network models have been proposed [30,32,29]. We considered the model investigated in [29] and available on CellCollective [20]. The analysis performed in [29] is based on the simulation method provided by such a platform.

We choose this model since it describes a realistic regulation system that is involved in many diseases [23,21,26] and it has been investigated under both the viewpoints of causality (effect of environmental conditions) and of reachability (reachable phenotypes). The Boolean network model is graphically represented as shown in Fig. 2 and is fully specified by the Boolean update formulas shown in Fig. 3. It includes 9 *input* nodes (orange), that correspond to stimuli that the T cell can receive by the external environment. Then, it contains 23 nodes representing relevant T cell genes, and 6 nodes representing secreted cytokines.

T cells can differentiate by expressing four different phenotypes, denoted Th1, Th2, Th17 and iTreg. Such phenotypes correspond to the expression of four different trancription factors included in the model: Tbet (for Th1), GATA3 (for Th2), RORgt (for Th17) and Foxp3 (for iTreg). There exists experimental evidence that a T cell can express more than one phenotype (hence, more than one of the associated transcription factors) [25]. Moreover, through the computational analysis performed in [29], the authors hypothesize that T cells may express also other combinations of phenotypes, that might include three or even four of them. Finally, the case in which none of the four relevant transcription factors is expressed (i.e. no phenotype) is denoted as Th0.

4.2 The RS Model

Translating a Boolean network model into an equivalent RS can be done by turning every Boolean formula of the network into disjunctive normal form. Then, every term of the disjunction (which is a conjunction of atoms, possibly negated) is translated into a reaction in which (i) reactants are the positive

```
 Jak1 = IFNgR and not SOCS1
 IL21 = STAT3 and NFAT
IL18R = IL18 and IL12 and not STAT6
SOCS1 = STAT1 or Tbet
  IL6 = RORgt
STAT5 = IL2R
 IL17 = (RORgt and not STAT1)
        or (STAT3 and IL17 and IL23R and not STAT1 and not STAT5)
STAT4 = IL12R and IL12 and not GATA3
IFNgR = (IFNg_e and NFAT) or (IFNg and NFAT)
STAT6 = IL4R and not IFNg and not SOCS1
GATA3 = (STAT6 and NFAT and not TGFb and not RORgt and not Foxp3
        and not Tbet) or (GATA3 and not Tbet)
        or (STAT5 and not TGFb and not RORgt and not Foxp3 and not Tbet)
  IL4 = GATA3 and NFAT and not STAT1
 NFkB = IRAK and not Foxp3
  IL2 = NFAT and NFkB and not Tbet
IL23R = (IL23 and STAT3 and not Tbet) or STAT3
 Tbet = (STAT4 and not RORgt and not Foxp3)
        or (STAT1 and not RORgt and not Foxp3)
        or (Tbet and not IL12 and not IFNg and not RORgt and not Foxp3)
TGFbR = TGFb and NFAT
RORgt = TGFbR and ((STAT3 and IL21R) or (STAT3 and IL6R)) and not Tbet
        and not GATA3 and not Foxp3
 IL6R = IL6 or IL6_e
IL21R = IL21
Foxp3 = (TGFbR and not (IL6R and STAT3) and not IL21R and not GATA3)
        or (STAT5 and not (IL6R and STAT3) and not IL21R and not GATA3)
 IRAK = IL18R
IL12R = (IL12 and NFAT) or (STAT4 and not GATA3) or Tbet or (TCR and not GATA3)
 IL2R = IL2 and NFAT
STAT3 = IL21R or IL23R or IL6R
 IFNg = NFkB or (STAT4 and NFkB and NFAT and not STAT3 and not STAT6)
        or (Tbet and not STAT3)
 NFAT = TCR and not Foxp3
STAT1 = (IL27 and NFAT) or Jak1
 IL4R = (IL4 and not SOCS1) or IL4_e
```

Fig. 3: Update rules of the Boolean network model of T Cell differentiation from [29]. Available in electronic format at [28].

atoms, (ii) inhibitors are the negated atoms and (iii) the (only) product is the Boolean variable that is updated through the considered formula. For example, the Boolean formula for IL12R, namely:

```
IL12R = (IL12 and NFAT) or (STAT4 and not GATA3) or Tbet or (TCR and not GATA3)
```

is translated into the following four reactions:

$$(\{\text{IL12}, \text{NFAT}\}, \emptyset, \{\text{IL12R}\}) \quad (\{\text{STAT4}\}, \{\text{GATA3}\}, \{\text{IL2R}\})$$
$$(\{\text{Tbet}\}, \emptyset, \{\text{IL12R}\}) \quad (\{\text{TCR}\}, \{\text{GATA3}\}, \{\text{IL12R}\})$$

that, in BioReSolve syntax, are coded as follows:

```
react([il12,nfat],[void],[il12r]),  react([stat4],[gata3],[il2r]),
react([tbet],[void],[il12r]),       react([tcr],[gata3],[il12r])
```

Even more complex forms of Boolean networks (such as Threshold Boolean Networks [8]) can be translated into RSs, as we demonstrated in [3].

The RS model of T cell differentiation obtained from the translation of the Boolean model is given in Fig. 4.

```
react([il4r],[socs1,ifng],[stat6]),          react([tgfb,nfat],[void],[tgfbr]),
react([tbet],[void],[il12r]),                react([stat4],[gata3],[il2r]),
react([tcr],[gata3],[il12r]),                react([il12,nfat],[void],[il12r]),
react([il2r],[void],[stat5]),                react([gata3],[tbet],[gata3]),
react([stat5],[tgfb,rorgt,foxp3,tbet],[gata3]),
react([stat6,nfat],[tgfb,rorgt,foxp3,tbet],[gata3]),
react([stat3],[void],[il23r]),               react([il23,stat3],[tbet],[il23r]),
react([tbet],[stat3],[ifng]),                react([nfkb],[void],[ifng]),
react([stat4,nfkb,nfat],[stat6,stat3],[ifng]),
react([tgfbr,stat3,il6r],[tbet,gata3,foxp3],[rorgt]),
react([tgfbr,stat3,il21r],[tbet,gata3,foxp3],[rorgt]),
react([il21],[void],[il21r]),                react([il18,il12],[stat6],[il18r]),
react([il6e],[void],[il6r]),                 react([il6],[void],[il6r]),
react([il18r],[void],[irak]),                react([il12r,il12],[gata3],[stat4]),
react([tbet],[void],[socs1]),                react([stat1],[void],[socs1]),
react([gata3,nfat],[stat1],[il4]),           react([stat3,nfat],[void],[il21]),
react([rorgt],[void],[il6]),                 react([stat5],[il21r,il6r,gata3],[foxp3]),
react([stat5],[il21r,stat3,gata3],[foxp3]),  react([tgfbr],[il21r,il6r,gata3],[foxp3]),
react([tgfbr],[il21r,stat3,gata3],[foxp3]),  react([tbet],[ifng,il12,rorgt,foxp3],[tbet]),
react([stat4],[rorgt,foxp3],[tbet]),         react([stat1],[rorgt,foxp3],[tbet]),
react([il27,nfat],[void],[stat1]),           react([jak1],[void],[stat1]),
react([il21r],[void],[stat3]),               react([il23r],[void],[stat3]),
react([il6r],[void],[stat3]),                react([ifngr],[socs1],[jak1]),
react([il2,nfat],[void],[il2r]),             react([il4e],[void],[il4r]),
react([il4],[socs1],[il4r]),                 react([irak],[foxp3],[nfkb]),
react([tcr],[foxp3],[nfat]),                 react([stat3,il17,il23r],[stat1,stat5],[il17]),
react([rorgt],[stat1],[il17]),               react([nfat,nfkb],[tbet],[il2]),
react([ifng,nfat],[void],[ifngr]),           react([ifnge,nfat],[void],[ifngr])
```

Fig. 4: Reactions of the RS model in BioReSolve syntax.

5 Attractors and Slicing Analyses

In this section we describe the analyses we performed on the RS model of T Cell differentiation. We describe both the methodology we applied and the results we obtained. We start from the generation of the LTS obtained by the execution of BioReSolve. Then, we perform an analysis aimed at identifying attractors and characterizing environmental configurations (i.e., RS contexts) leading to them. Finally, we perform the slicing analysis in order to identify relevant and crucial internal regulators for the expression of the different phenotypes.

5.1 Generation of the LTS

The LTS of the RS model is generated by invoking the `main_do(digraph)` directive of BioReSolve after providing model and context specifications in the expected syntax. In order to test all of the possibile configurations of the environment, the RS context in the BioReSolve specification is defined as a process that, at the first step, chooses in a non-deterministic way the set of environmental stimuli that have to be present. Then, the chosen stimuli will persist in the context. According to the syntax of RS processes given in Definition 1, this is expressed by the following parallel composition of context processes:

$$\mathsf{rec\ x11.TGFb.x11} + \mathsf{rec\ x0.}\emptyset\mathsf{.x0} \mid \ldots \mid \mathsf{rec\ x91.TCR.x91} + \mathsf{rec\ x0.}\emptyset\mathsf{.x0}$$

which, in the BioReSolve model specification, is expressed as follows:

Fig. 5: Overall representation of the LTS as a graph without labels. Enlargments of this image are available in Figures 6 and 7.

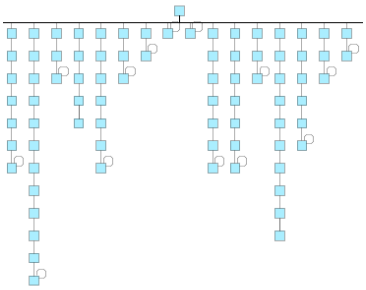

Fig. 6: Central part of the LTS depicted in Figure 5. The node on top is the initial state, from which several independent branches start.

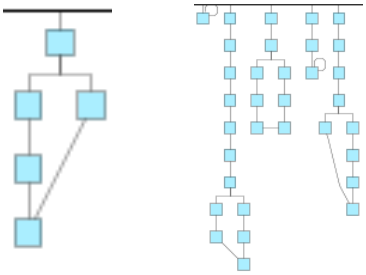

Fig. 7: Two fragments of the LTS with a trace leading to an attractor of size four (left), and a few traces leading to attractors of different sizes (right).

```
myenvironment('[ x1 = ({tgfb}.x11 + {}.x0), x2 = ({il23}.x21 + {}.x0),
                 x3 = ({il12}.x31 + {}.x0), x4 = ({il18}.x41 + {}.x0),
                 x5 = ({il4e}.x51 + {}.x0), x6 = ({il27}.x61 + {}.x0),
                 x7 = ({il6e}.x71 + {}.x0), x8 = ({ifnge}.x81 + {}.x0),
                 x9 = ({tcr}.x91 + {}.x0),
                 x11 = {tgfb}.x11, x21 = {il23}.x21, x31 = {il12}.x31,
                 x41 = {il18}.x41, x51 = {il4e}.x51, x61 = {il27}.x61,
                 x71 = {il6e}.x71, x81 = {ifnge}.x81, x91 = {tcr}.x91,
                 x0 = {}.x0 ]').
mycontext("[x1,x2,x3,x4,x5,x6,x7,x8,x9]").
```

Since we have 9 input entities representing the possible stimuli, this definition of the context process leads to an LTS with an initial branching into 2^9 different states, each being the start of a deterministic computation (i.e. a linear trace).

To give an idea of the structure of the LTS, in Fig. 5 we show its representation as a graph (without labels). Moreover, in Fig. 6 and Fig. 7 we zoom on some portions of the graph. In particular, Fig. 6 shows the initial state of the LTS and a few alternative traces all ending with a self-loop in the final state that corresponds to an attractor of size 1 (a steady state). On the other hand, Fig. 7 shows some examples of traces ending with attractors of different sizes.

5.2 Attractors analysis

BioReSolve returns the LTS as a graph in dot format, which is then loaded by our Python script. Then, attractors related with a target of interest are identified by looking for cycles in the LTS. Due to the LTS structure, this is particularly simple, and it is done in the `target_computations` function defined in our Python script by invoking repeatedly the `find_cycle` function provided by `networkx` package.

The analysis is performed by considering different *targets* corresponding to the different combinations of phenotypes that a T cell can express. As we described in Section 4.1, T cells can exhibit four phenotypes, Th1, Th2, Th17 and iTreg, represented by the expression of the four transcription factors Tbet, GATA3, RORgt and Foxp3, respectively. Since we are interested also in combinations of phenotypes, we consider all of the 2^4 combinations of such transcription factors: each of them will be a target of our analysis (hence, 16 targets overall).

For each target, we identify the attractors that include states in which the target transcription factors are expressed (not all necessarily in the same state), and in which the other transcription factors are not expressed. For example, for the target containing the combination of transcription factors {Tbet,GATA3}, we select the attractors (i.e., the cycles) that include at least one state in which Tbet is present, at least one state (possibly the same) in which GATA3 is present, and no state in which either RORgt or Foxp3 are present.

The filtering procedure just described gives as a result the subset of the attractors that correspond to the achievement of the considered target. Each of these attractors is reachable by making a different choice of environmental stimuli in the first step. The collection of all of those choices allows us to determine, for each target, which combinations of stimuli lead to it.

In Table 1 we report the number of different contexts (choices of environmental stimuli) that lead to each target. The table shows only targets that can be reached, and they result to be those in which only one transcription factor is expressed, and those in which Tbet is expressed together with another transcription factor. For each target, the contexts leading to it are summarized in the table by a Boolean formula, while in Figure 8 they are depicted graphically.

Analysis results show that 248 contexts (out of 512) lead to the expression of at least one of the four transcription factors under study. Tbet, corresponding to phenotype Th1, is expressed in the majority of the cases (152 out of 248), half of the times in combination with another transcription factor (in 76 cases out of 152). Also Foxp3 is often expressed (in 96 cases out of 248), while RORgt and GATA3 are less frequent (64 and 12 cases, respectively).

Fig. 8 allows us to make the following main observations about the role of the environment in the T cell differentiation process:

- TCR has to be present in order to express any of the phenotypes
- IL23R and IL18R do not contribute to determine any of the phenotypes
- The presence of TGFb mostly discriminates between the expression of either Foxp3/RORgt or tbet/GATA3.

Target	**Contexts**	**Formula**
Tbet	76	(not X11) and (X91) and (((not X31) and (X61)) or ((not X31) and (not X61) and (X71) and (X81)) or ((not X31) and (not X51) and (not X61) and (not X71) and (X81)) or ((X31) and (X71)))
GATA3	8	(not X11) and (not X31) and (X51) and (not X61) and (not X81) and (X91)
Foxp3	72	(not X71) and (not X91) and (((X11) and (not X61) and (not X81)) or (X31))
RORgt	16	(X11) and (not X31) and (not X61) and (X71) and (X91)
Tbet,GATA3	4	(not X11) and (not X31) and (X51) and (not X61) and (not X71) and (X81) and (X91)
Tbet,Foxp3	24	(X11) and (not X31) and (not X71) and (X91) and ((X61) or (X81))
Tbet,RORgt	48	(X11) and (X71) and (X91) and ((X31) or (X61))
TOTAL	**248**	

Table 1: Summary of contexts leading to each reachable target.

Moreover, the detailed characterization of the contexts leading to the expression of each phenotype could allow several more specific observations to be done.

The results of the analysis we conducted partially agree with those of a similar analysis done in [29]. However, there are also some significant differences between the results of the two studies. First of all, in [29] the authors show that it is also possibile to reach configurations in which three or four (i.e., all) phenotypes are expressed. Morevoer, they show that a configuration in which only RORgt is expressed cannot be reached. The Boolean network we started from is exactly the same as the one considered in [29]. The main difference between the two analysis approaches lies in the semantics of the environment: in our case the environment is modelled by a context process that remains the same after the initial non-deterministic choice, while in [29] the analysis is performed by applying a simulation method that allows activity levels and noise to be taken into account for the input species. Hence, the method applied in [29] can lead to a larger range of hypotheses about the modelled system behaviours (such as the possibility for T cell to express more than two phenotypes), while ours is more conservative and consistent with the standard simulation approaches.

5.3 Slicing analysis

For each target, slicing analysis is conducted by invoking the `main_do(slice,S)` directive of BioReSolve for each state of each attractor of the target. This is obtained by implementing suitable nested loops in the Python script that, at each iteration, execute BioReSolve through the Python-to-Prolog binding provided by the `swiplserver` package.

Slicing analysis in BioReSolve requires, in addition to the RS reactions, to provide the specification of a monitor (i.e., a logic formula) to collect the proper information during the model execution (see [12] for details). Monitored entities are only the transcription factors that the target requires to be present.

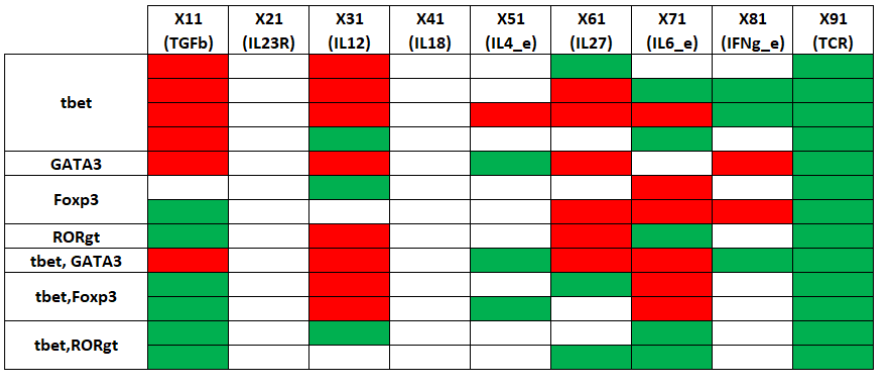

Fig. 8: Graphical representation of the Boolean formula given in Table 1 and characterizing contexts leading to each reachable target. Green cells denote that the corresponding environmental stimulus has to be present, red cells denote absences, and white cells denote that the stimulus is irrelevant.

Moreover, the context specification now does not contain the non deterministic choice of the general RS model: it is immediately initialized with a specific configuration of environmental stimuli. Here we show an example of specification of both the context and the monitor for the slicing analysis of one of the states of the target { Tbet, RORgt } reachable when the context does not provide IL12, IL4_e and INFg_e (denoted as `x31` `x51` and `x81`, respectively):

```
mycontext("[x11,x21,x0,x41,x0,x61,x71,x0,x91]").
mymonitor("[ m0 ]").
mymondef("[ m0 = ([{il12r,il21,il21r,il23r,il6r,nfat,rorgt,
                    socs1,stat1,stat3,tbet,tgfbr} inW].no({tbet,rorgt})
                + [-({il12r,il21,il21r,il23r,il6r,nfat,rorgt,
                    socs1,stat1,stat3,tbet,tgfbr} inW)].m0) ]").
```

Results of slicing analysis are summarized in Figure 9, where it is possibile to see, for each target, which internal genes are *strictly necessary* and which are somehow *relevant* for the expression of the transcription factor of the target. We remark that this is a form of causality analysis: for example, one gene that results to be necessary may cause the execution of a chain of reactions leading after some steps to the expression of one of the target transcription factors.

Once necessary and relevant genes for one target are identified, we can look for them in the results of the analysis conducted for the other targets. This leads to a notion of *specificity* that is very important. For example, a gene that is necessary for the expression of a phenotype and that is not relevant for the expression of the other phenotypes (i.e., it is highly specific) is a perfect candidate as a target for a drug aimed at inhibiting the expression of its phenotype.

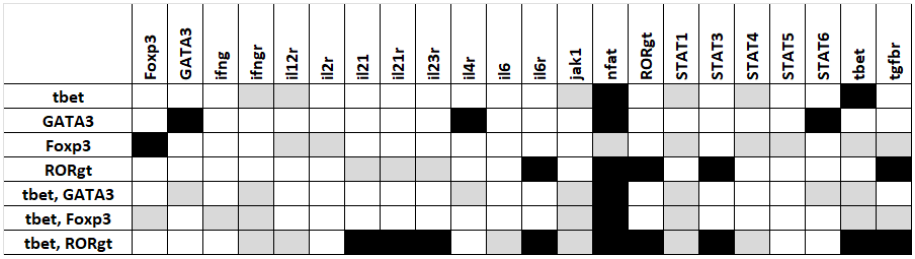

Fig. 9: Results of slicing analysis. Black cells identify genes (in columns) that are strictly necessary for the achievement of each target (in rows). Gray cells identify relevant genes. White cells identify genes that are not relevant.

The table in Figure 9 allows us to assess necessity, relevance and specificity of each internal gene for each phenotype. In particular, for the four phenotypes characterized by a single transcription factor we can observe that:

- for target Tbet, the analysis suggests that IFNgR and Jak1 are two relevant genes with high specificity, although they are not strictly necessary for the achievement of the target (i.e., their inhibition or knock-out does not guarantee that the target will not be reached);
- for target GATA3, the analysis suggests that IL4R and STAT6 are strictly necessary and highly specific;
- for target Foxp3, the analysis suggests that IL2R and STAT5 are relevant and highly specific, although not strictly necessary; and
- for target RORgt, the analysis suggests that IL6R and STAT3 are strictly necessary and highly specific, but also IL21, IL21R and IL23 result to be particularly important and specific (as also pointed out in [25]).

6 Conclusions and future work

In this paper we have presented a methodology for analyzing the attractors of a RS model and for identifying the biological entities that are responsibile for their achievement. Our methodology considers an operational semantics of RSs which allows us to construct Labeled Transition Systems (LTSs) to describe the behaviours of the model. Then, we apply a slicing algorithm to identify the minimal entities which describe a behaviour that we analyse.

Our methodology can be applied to Boolean network models of gene regulation, by translating them into RSs. We remark that such a translation emphasizes the different causes for the activation of each gene (representing them as different reactions). Moreover, it enables the application of analysis tools already developed for RSs.

We have applied our analysis to a model of T cell differentiation in the immune system. The analysis of the model allowed us to determine which combinations of stimuli lead to each phenotype, and also which proteins inside T

cells are involved in each case. This analysis contributes to the understanding of the behaviour of the model, to correct it in case some mistakes are identified and to use it for identifying drug targets. We discussed the relation with other approaches in the literature.

Our methodology is general and can be applied to the (large number of) case studies in the public database on the CellCollective platform [20], as well as to other Boolean network models.

As a future work we plan to extend our methodology in several ways, for instance by deriving information on the number of paths in which important molecules for a given target are used, by considering subsets of molecules and by counting the paths in which a (set of) molecule(s) is present. Our method might also be combined with other analyses (e.g. [11,13,14,6]) which are based on quantitative extensions of RSs, in order to define models which can express directly properties which require to count the number of entities in the system, or to express the time and speed of a reaction. The slicing might also be improved by considering the role of inhibitors as it was done in [15], where we defined a predictor analysis of causality in RSs based on static analyses.

References

1. Barbuti, R., Gori, R., Levi, F., Milazzo, P.: Investigating dynamic causalities in reaction systems. Theor. Comput. Sci. **623**, 114–145 (2016). https://doi.org/10.1016/j.tcs.2015.11.041
2. Barbuti, R., Gori, R., Levi, F., Milazzo, P.: Generalized contexts for reaction systems: definition and study of dynamic causalities. Acta Informatica **55**, 227–267 (2018)
3. Barbuti, R., Gori, R., Milazzo, P.: Encoding Boolean networks into reaction systems for investigating causal dependencies in gene regulation. Theoretical Computer Science **881**, 3–24 (2021). https://doi.org/10.1016/j.tcs.2020.07.031
4. Barbuti, R., Gori, R., Milazzo, P., Nasti, L.: A survey of gene regulatory networks modelling methods: from differential equations, to Boolean and qualitative bioinspired models. Journal of Membrane Computing **2**, 207–226 (2020). https://doi.org/10.1007/s41965-020-00046-y
5. BioReSolve web page, a prolog interpreter for Reaction Systems analysis, http://pages.di.unipi.it/bruni/LTSRS/, accessed: 18 March 2024
6. Bodei, C., Brodo, L., Gori, R., Levi, F., Bernini, A., Hermith, D.: A static analysis for brane calculi providing global occurrence counting information. Theor. Comput. Sci. **696**, 11–51 (2017). https://doi.org/10.1016/J.TCS.2017.07.008
7. Bodei, C., Gori, R., Levi, F.: Causal static analysis for brane calculi. Theoretical Computer Science **587**, 73–103 (2015)
8. Bornholdt, S.: Boolean network models of cellular regulation: prospects and limitations. Journal of the Royal Society Interface **5**(suppl_1), S85–S94 (2008). https://doi.org/10.1098/rsif.2008.0132.focus
9. Brijder, R., Ehrenfeucht, A., Main, M., Rozenberg, G.: A tour of reaction systems. Int. J. Found. Comput. Sci. **22**(07), 1499–1517 (2011). https://doi.org/10.1142/S0129054111008842

10. Brodo, L., Bruni, R., Falaschi, M.: A process algebraic approach to reaction systems. Theoretical Computer Science **881**, 62–82 (2021). https://doi.org/https://doi.org/10.1016/j.tcs.2020.09.001
11. Brodo, L., Bruni, R., Falaschi, M.: A logical and graphical framework for reaction systems. Theoretical Computer Science **875**, 1–27 (2021). https://doi.org/10.1016/j.tcs.2021.03.024
12. Brodo, L., Bruni, R., Falaschi, M.: Dynamic slicing of reaction systems based on assertions and monitors. In: Hanus, M., Inclezan, D. (eds.) Proc. of Practical Aspects of Declarative Languages - 25th Int. Symp., PADL 2023. Lecture Notes in Computer Science, vol. 13880, pp. 107–124. Springer (2023). https://doi.org/10.1007/978-3-031-24841-2_8
13. Brodo, L., Bruni, R., Falaschi, M., Gori, R., Levi, F., Milazzo, P.: Exploiting modularity of SOS semantics to define quantitative extensions of reaction systems. In: Aranha, C., Martín-Vide, C., Vega-Rodríguez, M.A. (eds.) Proc. of TPNC 2021. Lecture Notes in Computer Science, vol. 13082, pp. 15–32. Springer (2021). https://doi.org/10.1007/978-3-030-90425-8_2
14. Brodo, L., Bruni, R., Falaschi, M., Gori, R., Levi, F., Milazzo, P.: Quantitative extensions of reaction systems based on SOS semantics. Neural Comput. Appl. **35**(9), 6335–6359 (2023). https://doi.org/10.1007/s00521-022-07935-6, https://doi.org/10.1007/s00521-022-07935-6
15. Brodo, L., Bruni, R., Falaschi, M., Gori, R., Milazzo, P., Montagna, V., Pulieri, P.: Causal Analysis of Positive Reaction Systems. International Journal on Software Tools for Technology Transfer (2024). https://doi.org/10.1007/s10009-024-00757-y
16. Clark, A., Galpin, V., Gilmore, S., Guerriero, M.L., Hillston, J.: Formal methods for checking the consistency of biological models. In: Advances in Systems Biology. pp. 461–475. Springer (2012). https://doi.org/10.1007/978-1-4419-7210-1_27
17. Ehrenfeucht, A., Rozenberg, G.: Reaction systems. Fundamenta Informaticae **76**, 1–18 (2006), https://content.iospress.com/articles/fundamenta-informaticae/fi75-1-4-15
18. Ehrenfeucht, A., Rozenberg, G.: Reaction systems: a formal framework for processes based on biochemical interactions. Electronic Communications of the EASST **26**, 1–10 (2010). https://doi.org/10.1007/978-3-642-02424-5_3
19. Github repository with the python script developed for this paper, https://github.com/Unipisa/TCell-AttractorsSlicingAnalyses, accessed: 18 March 2024
20. Helikar, T., Kowal, B., McClenathan, S., Bruckner, M., Rowley, T., Madrahimov, A., Wicks, B., Shrestha, M., Limbu, K., Rogers, J.A.: The cell collective: toward an open and collaborative approach to systems biology. BMC systems biology **6**(1), 1–14 (2012). https://doi.org/10.1186/1752-0509-6-96
21. Hirahara, K., Nakayama, T.: CD4+ T-cell subsets in inflammatory diseases: beyond the T h 1/T h 2 paradigm. International immunology **28**(4), 163–171 (2016). https://doi.org/10.1093/intimm/dxw006
22. Karlebach, G., Shamir, R.: Modelling and analysis of gene regulatory networks. Nature reviews Molecular cell biology **9**(10), 770–780 (2008). https://doi.org/10.1038/nrm2503
23. Lafaille, J.J.: The role of helper T cell subsets in autoimmune diseases. Cytokine & growth factor reviews **9**(2), 139–151 (1998). https://doi.org/10.1016/s1359-6101(98)00009-4
24. Li, F., Long, T., Lu, Y., Ouyang, Q., Tang, C.: The yeast cell-cycle network is robustly designed. Proceedings of the National Academy of Sciences **101**(14), 4781–4786 (2004). https://doi.org/10.1073/pnas.0305937101

25. Luckheeram, R.V., Zhou, R., Verma, A.D., Xia, B.: CD4+ T cells: differentiation and functions. Clinical and developmental immunology **2012** (2012). https://doi.org/10.1155/2012/925135
26. Meng, X., Yang, J., Dong, M., Zhang, K., Tu, E., Gao, Q., Chen, W., Zhang, C., Zhang, Y.: Regulatory T cells in cardiovascular diseases. Nature Reviews Cardiology **13**(3), 167–179 (2016). https://doi.org/10.1038/nrcardio.2015.169
27. Milner, R.: A Calculus of Communicating Systems, LNCS, vol. 92. Springer (1980). https://doi.org/10.1007/3-540-10235-3
28. Puniya, B.L.: CD4+ T cell differentiation model webpage on the CellCollective platform, https://research.cellcollective.org/?dashboard=true#module/6678:1/cd4-t-cell-differentiation/1, accessed: 18 March 2024
29. Puniya, B.L., Todd, R.G., Mohammed, A., Brown, D.M., Barberis, M., Helikar, T.: A mechanistic computational model reveals that plasticity of CD4+ T cell differentiation is a function of cytokine composition and dosage. Frontiers in physiology **9**, 878 (2018). https://doi.org/10.3389/fphys.2018.00878
30. Saez-Rodriguez, J., Simeoni, L., Lindquist, J.A., Hemenway, R., Bommhardt, U., Arndt, B., Haus, U.U., Weismantel, R., Gilles, E.D., Klamt, S., et al.: A logical model provides insights into T cell receptor signaling. PLoS computational biology **3**(8), e163 (2007). https://doi.org/10.1371/journal.pcbi.0030163
31. SWI-Prolog home page, https://www.swi-prolog.org/, accessed: 18 March 2024
32. Thakar, J., Albert, R.: Boolean models of within-host immune interactions. Current opinion in microbiology **13**(3), 377–381 (2010). https://doi.org/10.1016/j.mib.2010.04.003