Fundamenta Informaticae XX (2019) 1–20
DOI 10.3233/FI-2015-0000
IOS Press

Encoding Threshold Boolean Networks into Reaction Systems for the Analysis of Gene Regulatory Networks

Roberto Barbuti
Dipartimento di Informatica,
Università di Pisa, Italy
roberto.barbuti@unipi.it

Pasquale Bove
Dipartimento di Informatica,
Università di Pisa, Italy
pasquale.bove@unipi.it

Roberta Gori
Dipartimento di Informatica,
Università di Pisa, Italy
roberta.gori@unipi.it

Damas Gruska
Department of Applied Informatics,
Comenius University in Bratislava, Slovak Republic
damas.gruska@gmail.com

Francesca Levi
Dipartimento di Informatica,
Università di Pisa, Italy
francesca.levi@unipi.it

Paolo Milazzo
Dipartimento di Informatica,
Università di Pisa, Italy
paolo.milazzo@unipi.it

Abstract. Gene Regulatory Networks represent the interactions among genes regulating the activation of specific cell functionalities and they have been successfully modeled using threshold Boolean networks. In this paper we propose a systematic translation of threshold Boolean networks into Reaction Systems. Our translation produces a non redundant set of rules with the minimal number of objects. This translation allows us to simulate the behavior of a Boolean network simply by executing the (closed) Reaction System we obtain. This can be very useful for investigating the role of different genes simply by "playing" with the rules. We developed a tool able to systematically translate a threshold Boolean network into a Reaction System. We use our tool to translate two well known Boolean networks modelling biological systems, the Yeast-Cell Cycle and the SOS response in Escherichia coli. The resulting reaction systems can be used for investigating dynamic causalities among genes.

Address for correspondence: Roberta Gori, Dipartimento di Informatica, Università di Pisa, Italy roberta.gori@unipi.it.

1. Introduction

In the context of molecular biology of cells, *Gene Regulatory Networks* (GRNs) represent the interactions among genes regulating the activation of specific cell functionalities. More specifically, genes in a GRN can be either active (i.e. the corresponding protein is expressed) or not, and each active gene can either stimulate or inhibit the activation of a number of other genes. Moreover, the activation of some genes is also usually influenced by other factors such as the availability of some substances in the environment, or the reception of a signal form neighbor cells. As a consequence, gene regulatory networks can be seen as the mechanism that allow a cell to react to external stimuli. When a stimulus is received, it causes a change in the activation state of a few genes, that, in turn, influence other genes, allowing a new configuration of active genes (corresponding to a new set of active cell functionalities) to be reached.

Several approaches have been proposed to model and analyze GRNs (see [31] for a survey on this topic). Modeling techniques can either deal only with the qualitative aspects of such networks (treating them essentially as logic circuits), or can describe also the quantitative aspects, such as the rates of the interactions. The latter approach is for sure more precise, but requires many additional details of the network dynamics to be taken into account, such as the rates of transcription and translation of genes' DNA into proteins, and the rates of protein-protein and protein-DNA interactions. Qualitative models are often sufficient to reason on the behavior of the regulatory networks, although with some degree of approximation.

In the qualitative modeling setting, one of the most successful modeling frameworks for gene regulatory networks are *Boolean networks*. In this setting, a particular simple form of Boolean networks, the so called *threshold Boolean networks* [18, 25, 28], have been widely used to model the dynamics of quite complex regulatory networks. In threshold Boolean networks, the Boolean function of each node depends on the sum of its input signals only. This variant of Boolean networks can be easily implemented and, at the same time, it is well suited for representing gene regulatory networks.

Boolean networks allow dynamical properties of GRNs to be investigated. Starting from an initial configuration of active genes, the dynamics of a GRNs is expressed as a sequence of steps in which such a configuration is updated according to the influences among the genes described by the Boolean network. Dynamical properties can be investigated either by performing simulations, or by constructing the (finite) graph representing all possible dynamical evolutions. Example of properties that are often studied on these models are reachability and stability of configurations, and confluence of evolutions started from different initial configurations into stable configurations (attractors).

Analysis of dynamical properties may become computationally very expensive. In order to reduce the state space to be analyzed, minimization techniques can be applied. The Boolean function represented by a Boolean network can be synthesized in any framework of logic minimization. The classical approach to logic minimization that produces sum of products two level formulas can be used (see e.g., Espresso [29]).

Other analysis methods for GRNs could be applied by changing the representation of the Boolean networks describing them. In this paper we propose a systematic translation of threshold Boolean networks into *Reaction Systems* [19, 12]. Reaction systems were introduced by Ehrenfeucht and Rozenberg as a novel model for the description of biochemical processes driven by the interaction among reactions in living cells. Reaction systems are based on two opposite mechanisms, namely *facilitation* and *inhibition*. Facilitation means that a reaction can occur only if all its reactants are present, while inhibition means that the reaction cannot occur if any of its inhibitors is present. The state of a Reaction System

consists of a finite set of objects which can evolve by means of application of reactions. The presence of an object in a state expresses the fact that the corresponding biological entity, in the real system being modeled, is present. Quantities (or concentrations) of the entities are not described: Reaction Systems are hence a *qualitative* modeling formalism.

In this setting, the dynamic run of the Reaction System simulates the evolution of the Boolean network. This correspondence allows us to "play" with the rules of the Reaction System related to different genes in order to detect dynamic causality dependencies between genes activation/deactivation. Moreover, we believe that this correspondence will allow us to apply to Boolean networks well-known techniques to detect causality relationships between objects in biological systems. The understanding of causality relationships among the events happening in a biological (or bio-inspired) system is an issue investigated in the context both of systems biology (see e.g. [24, 10, 11, 9]) and of natural computing (see e.g. [16]).

In [13], Brijder, Ehrenfeucht and Rozenberg initiate an investigation of *causalities* in Reaction Systems [19, 12]. Causalities deal with the ways entities of a Reaction System influence each other. In [13], both static/structural causalities and dynamic causalities are discussed, introducing the idea of *predictor*. In [4, 3, 5, 7, 6, 8], the idea of predictors was enhanced by defining the notions of *formula based predictor* and *specialized formula based predictor*. These new concepts allow us to study all causal dependencies of one object from all the others. The Reaction System encoding a Boolean network could be investigated by computing the specialized formula based predictor of a particular activation/deactivation gene configuration. This would allow us to obtain a logic formula characterizing all alternative activation/deactivation gene configurations that lead to the requested configuration in a bounded number of steps. This could be very useful to understand which genes are *necessary* for reaching a requested configuration. Moreover, related causality analyses consist in determining whether a configuration of a Reaction System is a k*-ancestor* [17, 23] of a given (observed) configuration (i.e. the latter can be obtained from the former in k steps), counting the number of k-ancestors, and determining whether a k-ancestor exists at all for a given k. Some of these problems can also be re-formulated in terms of information flow properties [21], for instance by exploiting the notion of *opacity* [15, 14], which has been introduced for Reaction Systems in [23, 22].

The translation we propose in this paper produces a *non redundant* set of rules with the *minimal number* of objects. We developed a tool able to systematically translate a threshold Boolean network into a Reaction System. We use our tool to translate two well known Boolean networks modelling biological systems: the Yeast-Cell Cycle and the SOS response in Escherichia coli.

The paper is organized as follows. Section 2 introduces the main concepts of (Closed) Reaction Systems. In Section 3 we describe how Boolean networks are defined and how they work. Section 4 presents our encoding of threshold Boolean networks into Reaction Systems. The tool realizing the proposed translation is presented in Section 5. Finally, in Section 6 we apply our tool to translate, simulate and study the Yeast-Cell Cycle Boolean Network, while in Section 7 our tool is applied to the regulatory network describing the SOS response in Escherichia coli. A description of future works in Section 8 concludes the paper.

2. Closed Reaction Systems

In this section we recall the basic definition of Reaction Systems [19, 12]. Let S be a finite set of symbols, called objects. A *reaction* is formally a triple (R, I, P) with $R, I, P \subseteq S$, composed of *reactants* R, *inhibitors* I, and *products* P. Reactants and inhibitors $R \cup I$ of a reaction are collectively called *resources* of such a reaction, and we assume them to be disjoint ($R \cap I = \emptyset$), otherwise the reaction would never be applicable. The set of all possible reactions over a set S is denoted by $\mathrm{rac}(S)$. Finally, a *Reaction System* is a pair $\mathcal{A} = (S, A)$, where S is a finite support set, and $A \subseteq \mathrm{rac}(S)$ is a set of reactions.

The state of a Reaction System is described by a set of objects. Let $a = (R_a, I_a, P_a)$ be a reaction and T a set of objects. The result $\mathrm{res}_a(T)$ of the application of a to T is either P_a, if T separates R_a from I_a (i.e. $R_a \subseteq T$ and $I_a \cap T = \emptyset$), or the empty set $\emptyset$ otherwise. The application of multiple reactions at the same time occurs without any competition for the used reactants (*t*hreshold supply assumption). Therefore, each reaction which is not inhibited can be applied, and the result of the application of multiple reactions is cumulative. Formally, given a Reaction System $\mathcal{A} = (S, A)$, the result of application of $\mathcal{A}$ to a set $T \subseteq S$ is defined as $\mathrm{res}_{\mathcal{A}}(T) = \mathrm{res}_A(T) = \bigcup_{a \in A} \mathrm{res}_a(T)$.

An important feature of Reaction System is the assumption about the *non-permanency* of objects: the objects carried over to the next step are only those produced by reactions. All the other objects vanish, even if they are not involved in any reaction.

The dynamics of a Reaction System is generally driven by the *contextual* objects, namely the objects supplied to the system by the external environment at each step. *Closed* Reaction Systems are the subset of general Reaction Systems where the external environment provides objects at the first step only.

This allows us to simplify the dynamics of a (closed) Reaction System $\mathcal{A} = (S, A)$. Indeed, given the initial set D_0 the semantics can be simply defined as the *result sequence*, $\delta = D_1, \ldots, D_n$ where each set D_i, for $i \geq 1$, is obtained from the application of reactions $\mathcal{A}$ to the state obtained at the previous step D_{i-1} ; formally $D_i = res_{\mathcal{A}}(D_{i-1})$ for all $1 \leq i < n$. For the sake of simplicity, we write $D_{i-1} \rightarrow_{\mathcal{A}} D_i$ as a shorthand for $D_i = res_{\mathcal{A}}(D_{i-1})$. In this case the sequence of states of the Reaction System coincides with the result sequence $\delta = D_1, \ldots, D_n$.

3. Boolean Networks

We present a formal definition of threshold Boolean networks [28] considering a set M of n elements, $S_1, S_2, \ldots S_n$ to be *nodes* of a network. We assign to each element, at each time instant t, a value $S_i(t) \in \{0, 1\}$ denoting if the element S_i is present at that instant or not. The interactions among elements are given by the set of *edges* of the network called E. An edge from element S_j to element S_i is denoted a_{ij} (where $i \neq j$ given that a element cannot activate/inhibit itself). Each edge in E can be either *activating* or *inhibiting*. This is represented by a value associated to the edge: an activating edge has value 1 while an inhibiting edge has value -1. Elements M can be partitioned in two sets M_{sa} and M_{nsa} of *self-activating* and *non-self-activating* elements, respectively, with $M = M_{sa} \cup M_{nsa}$. A self-activating element, present at time t and not inhibited, will be present also at time $t + 1$, while a non-self-activating element will not. Moreover, we assume that each element S_i has associated a value $\theta_i \in \Theta$ (we assume $\Theta = \mathbb{N}$) which is called the *threshold parameter* of S_i. The pair (M, E) is called a *threshold Boolean network*.

The states of the nodes in the network are updated in parallel at discrete time steps. The rules for

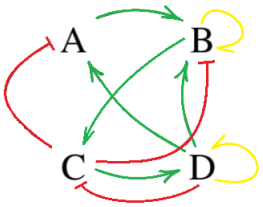

Figure 1. An example of gene regulatory network.

updating the values of nodes are the following, for $i \in \{1, \ldots, n\}$

$$S_i(t+1) = \begin{cases} 1 & \text{if} \quad \sum_j a_{ij} S_j(t) > \theta_i \\ 0 & \text{if} \quad \sum_j a_{ij} S_j(t) < \theta_i \\ S_i(t) & \text{if} \quad S_i \in M_{sa} \wedge \sum_j a_{ij} S_j(t) = \theta_i \\ 0 & \text{if} \quad S_i \in M_{nsa} \wedge \sum_j a_{ij} S_j(t) = \theta_i \end{cases}$$

where the value θ_i is the threshold parameter associated to the element S_i.

Typically, the threshold parameter θ_i associated to S_i is equal to 0 so that the switch is inactive if there is no activators, and it switches on when one or more activators and no inibithors are present. Note that the effect of activators and inhibitors is addictive: the number of activators has to be greater than the number of inhibitors in order for the element to be activated, and self-activation makes the element itself to be counted among activators. A node which needs more than one activators to be switched on can be represented in the model by setting θ_i to a value greater than 0.

There is a natural representation of Boolean networks in terms of a graph where the nodes represent the elements and the edges represent the interactions between the elements; an *activating edge* is indicated by a (green) $\rightarrow$ while an *inhibiting* one is indicated by a (red) $\dashv$. Non self-activating elements are represented by nodes with (yellow) half-arrow ($\rightharpoonup$) self loops.

Starting from an initial condition, the network produces a dynamical sequence of states, and it can reach either a fixed point or a periodic attractor. We introduce an example to illustrate threshold Boolean networks and their dynamic evolution.

Example 3.1. Let us consider the threshold Boolean network (M, E) with elements $M = \{A, B, C, D\}$ such that $M_{sa} = \{A, B\}$ and $M_{nsa} = \{C, D\}$ and with the edges depicted in Fig. 1. Thus, the elements A and C are self-activating, while B and D are not. We also assume that the threshold parameter for each element is 0.

We describe the temporal evolution of the network by considering an initial state in which only element D is present. We have

Step	A	B	C	D
1	0	0	0	1
2	1	1	0	0
3	1	1	1	0
4	0	0	1	1
5	0	0	0	1

Initially, element D stimulates the activation of both elements A and B because element C, their inhibitor, is not present. Note that at the second step the element D is inactive because it is non self-activating. Then, at step 3, the element C is active because it is activated by B, while A and B are still active because they are self-activating and at step 2 C was inactive. At step 4, element C activates D and inhibits A and B, which become inactive. Finally, at step 5 also C becomes inactive because inhibited by D. The last state coincides with the first one, so we reached the end of a cycle. Different evolutions can be obtained starting from different configurations. Consider, for example, an initial configuration where all the elements except B are active.

Step	A	B	C	D
1	1	0	1	1
2	1	1	1	1
3	1	1	1	1

After one step the system reaches a stationary state, that is a stable state from which no different configuration can be reached.

4. Encoding Threshold Boolean Networks into Reaction Systems

We present an encoding of threshold Boolean networks into closed Reaction Systems. Given a Boolean network (M, E) with $M = \{S_1, S_2, \dots S_n\}$ we define, for $S_i \in M$,

$$Act(S_i) = \{S_j \mid j \in [1, n] \ \wedge \ a_{ij} = 1\} \qquad In(S_i) = \{S_j \mid j \in [1, n] \ \wedge \ a_{ij} = -1\}$$

We recall that a_{ij} denotes an edge from element S_j to element S_i. Hence, $Act(S_i)$ reports the elements S_j which *activates* S_i and analogously $In(S_i)$ reports the elements S_j which *inhibits* it.

Definition 4.1. Let (M, E) be a *threshold Boolean network* with elements $M = \{S_1, S_2, \dots S_n\}$ and threshold parameters $\Theta = \{\theta_1, \theta_2, \dots \theta_n\}$. We define its translation as the closed *Reaction System* $\mathcal{RS}((M, E)) = (M, A)$, where reactions in A are constructed according to the following inference rules:

$$1) \ \frac{P_i \subseteq Act(S_i) \ \ I_i \subseteq In(S_i) \qquad \#P_i - \#(In(S_i) \setminus I_i) = \theta_i + 1}{(P_i, I_i, \{S_i\}) \in A} \qquad 2) \ \frac{S_i \in M_{sa} \ \ P_i \subseteq Act(S_i) \ \ I_i \subseteq In(S_i) \qquad \#P_i - \#(In(S_i) \setminus I_i) = \theta_i}{(P_i \cup \{S_i\}, I_i, \{S_i\}) \in A}$$

The closed Reaction System $\mathcal{RS}((M,E))$ simulates the threshold Boolean network (M,E) using reactions obtained by applying either the inference rule 1) or the inference rule 2). Rule 1) defines the reactions which simulate the production of an element S_i at time $t+1$ whenever at time t the number of the elements which activate S_i minus the number of the elements which inhibit S_i is greater than θ_i (according to the rule given in Section 3). This behavior is simulated by a reaction which has as *product* S_i, as *reactants* P_i and as *inhibitors* I_i where P_i is a subset of the elements which activates S_i and I_i is a subset of the elements which inhibits it. Note that this reaction can be applied if in the Reaction System state none of the elements in I_i is present. As a consequence, the set of the elements which are inhibitors of S_i and which may be present is given by $In(S_i)\backslash I_i$. Therefore, we require that the cardinality of P_i minus that of $In(S_i)\backslash I_i$ is greater than θ_i.

It is worth noting that our aim is to built a non redundant set of reactions, that is, for each pair of reactions $a_1 = (R_a^1, I_a^1, P_a^1)$ and $a_2 = (R_a^2, I_a^2, P_a^2)$ such that $P_a^1 \cap P_a^2 \neq \emptyset$, we have that either $R_a^2 \not\supseteq R_a^1$ or $I_a^2 \not\supseteq I_a^1$. This condition ensures that each reaction is necessary, namely there exist two configurations in which each of the two reactions is applicable, while the other is not. To achive this goal, we require that the difference between the cardinality of P_i and the cardinality of $In(S_i)\backslash I_i$ is exactly equal to θ_i+1.

Rule 2 applies only in case of self activating nodes by adding a new rule that model the self activation. Indeed, in this case, activating elements remain active at time $t+1$ if they are present, according to the rule given in Section 3. In Reaction Systems, due to the non-permanency of objects, the objects carried over to the next step are only those produced by reactions. Therefore, in this case, the reaction which simulates the behavior has S_i as *reactant* and also as *product*. Similarly as in the case of Rule 1, P_i is a subset of the elements which activate S_i, and I_i is a corresponding subset of the elements which inhibits S_i. In this case, however, we require that the cardinality of P_i minus the number of inhibitors which might be present (i.e. $(In(S_i)\backslash I_i)$) is exactly θ_i.

Example 4.2. We give the translation of the threshold Boolean network (M,E) presented in Example 3.1. Fig. 1) illustrates the interactions between the elements of the network $M = \{A,B,C,D\}$ such that $M_{sa} = \{A,C\}$ and $M_{nsa} = \{B,D\}$.

By assuming again that the threshold parameter for each element is 0 we obtain the closed Reaction System $\mathcal{RS}((M,E)) = (M,A)$ with reactions A defined as follows:

$$
\begin{array}{lllll}
(\{D\},\{C\},\{A\}) & (\{C\},\emptyset,\{D\}) & (\{A,D\},\emptyset,\{B\}) & (\{A\},\{C\},\{A\}) & (\{C\},\{D\},\{C\}) \\
(\{B\},\{D\},\{C\}) & (\{A\},\{C\},\{B\}) & (\{D\},\{C\},\{B\}) & (\{A,D\},\{\},\{A\}) & (\{B,C\},\{\},\{C\})
\end{array}
$$

The reactions on the first three columns on the left are obtained by applying Rule 1, while those in the two columns on the right by applying Rule 2. The three columns on the left contain one or more reactions for each element to be produced.

For the production of A, C and D there is exactly one reaction each. The reaction producing A has D as reactant and C as inhibitor, while C can be produced from B if inhibitor D is not present. Element D can be produced from reactant C without any other condition.

Element B can be activated by two elements, A and D, and it is inhibited by C. Hence, there are three different reactions corresponding to the possible combinations of elements which can activate B. Note that the requirements of Rule 1 guarantee that only minimal combinations of reactants and inhibitors are considered. For instance a reaction such as $a_2 = (\{A,D\},\{C\},\{B\})$ is not present in the set of reactions because it is subsumed by $a_1 = (\{A,D\},\emptyset,\{B\})$ since the latter reaction can be

applied regardless the presence of C. This implies that there cannot exists any set in which reaction a_2 is applicable while a_1 is not, and, indeed, $R_a^2 \not\supseteq R_a^1$ or $I_a^2 \not\supseteq I_a^1$ does not hold in this case.

The two columns on the right shows the additional reactions for self-activating elements, A and C, obtained by Rule 2. Similarly as in the previous case there can be one or more reactions for each self-activating element, and these reaction have the self-activating element both as a reactant and as a product.

We can now prove the soundness of the translation of threshold Boolean networks into closed Reaction Systems. To relate the state of a threshold Boolean network with the configuration of the associated Reaction System we introduce the following definition.

Definition 4.3. Given a *threshold Boolean network* (M, E) with $M = \{S_1, \dots S_n\}$ and a state at time t, $S(t) = \{S_1(t), S_2(t), \dots S_n(t)\}$, the translation of the state $S(t)$ into a corresponding Reaction System configuration is given by $\mathcal{RS}(S(t))$, defined as follows: $\mathcal{RS}(S(t)) = \{S_i \mid S_i(t) = 1, i \in [1, n]\}$.

Theorem 4.4. Let (M, E) be a *threshold Boolean network* with elements $M = \{S_1, S_2, \dots S_n\}$ and threshold parameters $\Theta = \{\theta_1, \theta_2, \dots \theta_n\}$. Given a state at time t, $S(t) = \{S_1(t), S_2(t), \dots S_n(t)\}$ we have that

$$\mathcal{RS}(S(t+1)) \;\; = \;\; res_{\mathcal{A}}(\mathcal{RS}(S(t)))$$

where $\mathcal{A} = \mathcal{RS}((M, E)) = (M, A)$ is the Reaction System obtained by the translation.

Proof:
We first show that $\mathcal{RS}(S(t+1)) \;\; \supseteq \;\; res_{\mathcal{A}}(\mathcal{RS}(S(t)))$.

Suppose that $S_i \in res_{\mathcal{A}}(\mathcal{RS}(S(t)))$, then there are two cases:

- There exists in A a rule $(P_i, I_i, \{S_i\})$ such that $P_i \subseteq \mathcal{RS}(S(t))$ and $I_i \not\subseteq \mathcal{RS}(S(t))$. For construction of the rule, we know that $P_i \subseteq Act(S_i)$, $I_i \subseteq In(S_i)$, $\#P_i - \#(In(S_i) \backslash I_i) = \theta_i - 1$. Because the rule is applicable, we know that the number of promoters in $\mathcal{RS}(S(t))$ is greater than the number of inhibitors, thus $\sum_j a_{ij} S_j(t) \; > \; \theta_i$ and $S_i(t+1) = 1$. By definition, $S_i \in \mathcal{RS}(S(t+1))$.

- $S_i \in M_{sa}$ and there exists in A a rule $(P_i \cup \{S_i\}, I_i, \{S_i\})$ such that $P_i \subseteq \mathcal{RS}(S(t))$ and $I_i \not\subseteq \mathcal{RS}(S(t))$. By the construction of the rule, we know that $P_i \subseteq Act(S_i)$, $I_i \subseteq In(S_i)$, $\#P_i - \#(In(S_i) \backslash I_i) = \theta_i$. Since the rule is applicable, we know that the number of promoters in $\mathcal{RS}(S(t))$ is greater than or equal to the one of inhibitors, thus $\sum_j a_{ij} S_j(t) \; \geq \; \theta_i$ and $S_i(t+1) = 1$.

  Therefore, by definition, $S_i \in \mathcal{RS}(S(t+1))$.

We now prove that $RS(S(t+1)) \;\; \subseteq \;\; res_{\mathcal{A}}(RS(S(t)))$.
Assume that $S_i \in \mathcal{RS}(S(t+1))$, then, by definition, $S(t+1) = 1$. There are two cases:

- $\sum_j a_{ij} S_j(t) \; > \; \theta_i$, then we have $\#PT_i > \#\overline{TI}_i - \theta_i$,
  where $PT_i = \{S_j | S_j \in Act(S_i) \; \wedge S_j(t) = 1\}$ and $\overline{TI}_i = \{S_j | S_j \in In(S_i) \; \wedge S_j(t) = 1\}$. By construction, $PT_i \subseteq \mathcal{RS}(S(t))$ and $\overline{TI}_i \subseteq \mathcal{RS}(S(t))$. By definition, we have a rule in A, $(P_i, I_i, \{S_i\})$, such that $P_i \subseteq PT_i$, $I_i \subseteq I, \overline{TI}_i \subseteq (In(S_i) \backslash I_i)$, and $\#P_i - \#(In(S_i) \backslash I_i) = \theta_i + 1$. Since $I_i \cap \overline{TI}_i = \emptyset$, the rule is applicable and $S_i \in res_{\mathcal{A}}(\mathcal{RS}(S(t)))$.

- $S_i \in M_{sa}$, $S_i(t) = 1$, and $\sum_j a_{ij} S_j(t) > \theta_i$, then we have $\#PT_i > \#\overline{TI}_i - \theta_i$,
  where $PT_i = \{S_j | S_j \in Act(S_i) \wedge S_j(t) = 1\}$ and $\overline{TI}_i = \{S_j | S_j \in In(S_i) \wedge S_j(t) = 1\}$. By construction, $PT_i \subseteq \mathcal{RS}(S(t))$ and $\overline{TI}_i \subseteq \mathcal{RS}(S(t))$. By definition we have a rule in A, $(P_i \cup \{S_i\}, I_i, \{S_i\})$, such that $P_i \subseteq PT_i$, $I_i \subseteq I, \overline{TI}_i \subseteq (In(S_i) \backslash I_i)$, and $\#P_i - \#(In(S_i) \backslash I_i) = \theta_i$. Since $I_i \cap \overline{TI}_i = \emptyset$, the rule is applicable and $S_i \in res_{\mathcal{A}}(\mathcal{RS}(S(t)))$.

This concludes the proof. □

Due to Theorem 4.4, a threshold Boolean network (M, E) with a state $S(0)$ at time 0 can be simulated by the corresponding closed Reaction System $\mathcal{RS}((M, E))$ by considering the initial state $\mathcal{RS}(S(0))$.

At this point, it is important to count the number reactions of the closed Reaction System necessary to simulate a threshold Boolean network (M, E), since this would give us a measure of the space complexity of the encoding. Such a number depends on the number of the nodes M and of the edges of the network E, and also on the threshold parameters Θ. For each $S_i \in M$ the number of the reactions which have S_i as a product depends on the cardinalities of $Act(S_i)$ and $In(S_i)$, and on θ_i. Indeed, $Act(S_i)$ and $In(S_i)$ represents the number of the incoming edges which *activates* and *inhibits* S_i respectively.

Proposition 4.5. Given a *threshold Boolean network* (M, E) with elements $M = \{S_1, S_2, \ldots S_n\}$ and threshold parameters $\Theta = \{\theta_1, \theta_2, \ldots \theta_n\}$, let $\mathcal{RS}((M, E)) = (S, A)$ be the corresponding closed Reaction System. The following two properties hold:

- For each $i \in \{1, \ldots, n\}$, the number of the reactions which produce the element S_i, $N(S_i) = \#(\{(R, I, P) \in A \mid S_i \in P\})$, can be computed as follows:

$$N(S_i) = \begin{cases} \sum_{k=1+\theta_i}^{min(m_i,(l_i+1+\theta_i))} \binom{m_i}{k} \times \binom{l_i}{k-1-\theta_i}, & \text{if } S_i \in M_{nsa}; \\ \sum_{k=1+\theta_i}^{min(m_i,(l_j+1+\theta_i))} \binom{m_i}{k} \times \binom{l_i}{k-1-\theta_i} + & \\ \sum_{h=\theta_i}^{min(m_i,(l_j+\theta_i))} \binom{m_i}{h} \times \binom{l_i}{k-\theta_i}, & \text{if } S_i \in M_{sa}. \end{cases}$$

  where $m_i = \#(Act(S_i))$ and $l_i = \#(IN(S_i))$.

- We have that $\#(A) = \sum_{i=1}^{n} N(S_i)$.

The previous result is a direct consequence of Definition 4.1.

Example 4.6. Let us consider the closed Reaction System $\mathcal{RS}(M, E) = (M, A)$, presented in the Example 4.2, which is the translation of the threshold Boolean network (M, E) of Example 3.1. The reaction system has 12 reactions. Indeed, since A, B and C belong to M_{sa} while D belongs to M_{nsa} and

all threshold parameters are 0, by applying Proposition 4.5, we obtain:

$$N(A) = \sum_{i=1}^{min(1,2)} \binom{1}{i} \times \binom{1}{i-1} + \sum_{i=0}^{min(1,1)} \binom{1}{i} \times \binom{1}{i} = 1 + (1+1) = 3$$

$$N(B) = \sum_{i=1}^{min(2,2)} \binom{2}{i} \times \binom{1}{i-1} + \sum_{i=0}^{min(2,1)} \binom{2}{i} \times \binom{1}{i} = (2+1) + (1+2) = 6$$

$$N(C) = \sum_{i=1}^{min(1,1)} \binom{1}{i} \times \binom{0}{i-1} + \sum_{i=0}^{min(1,0)} \binom{1}{i} \times \binom{0}{i} = 1 + 1 = 2$$

$$N(D) = \sum_{i=1}^{min(1,2)} \binom{1}{i} \times \binom{0}{i-1} = 1$$

5. The Translation Tool

We developed a software tool, written in C, which takes a description of a Boolean network and returns the set of rules of the Reaction System simulating the network. The description of the Boolean network is given in a text file named "genes.txt" as follows. The first line of the file must contain the number of nodes of the Boolean network. The subsequent lines describe each single node of the network. Each line contains the name of the node, followed by a boolean value which specifies whether the node is self-activating, 1, or not, 0, and, finally, by the value of the related threshold θ. After the description of the nodes, the subsequent lines describe each single edge of the network. First of all we have a line with the number of edges. Then, we have a line describing each edge in which we have the ending node, S_j, the starting node, S_i, and a value a_{ij} (either -1 or $+1$). Several controls have been implemented in our tool in order to be sure that file "genes.txt" describes a consistent boolean network.

As an example, the input file corresponding to the network described in the Example 3.1 should be the following.

```
4
A 1 0
B 0 0
C 1 0
D 0 0
8
A C -1
A D 1
B A 1
B C -1
B D 1
C B 1
C D -1
D C 1
```

Observe that the self-loop of the non-self-activating nodes B and D are not described in the list of edges because they are indicated (with a 1) in the description of the nodes.

The output is produced in the file "rules.txt". In this case the rules of the reaction system $\mathcal{A}_1$ produced by the tool are the following.

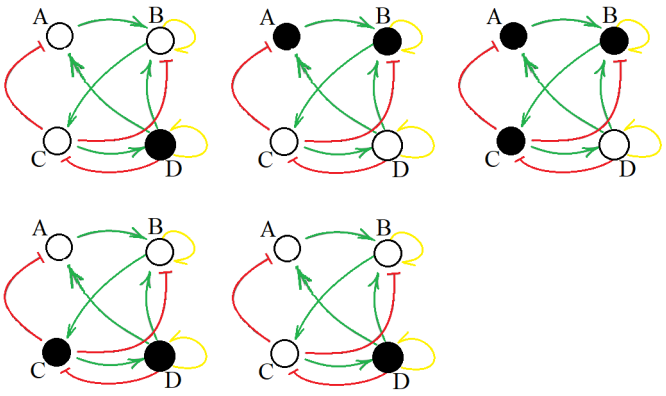

Figure 2. The evolution of the gene regulatory network of Fig. 1

```
({D},{C},{A})
({A,D},{},{A})
({A},{C},{A})
({A},{C},{B})
({D},{C},{B})
({A,D},{},{B})
({B},{D},{C})
({C},{D},{C})
({B,C},{},{C})
({C},{},{D})
```

The rules in file "rules.txt" can now be used to execute the reaction system that simulate the behaviour of the Boolean network. Thus, the following evolution can be observed.

$$\{\mathtt{D}\} \rightarrow_{\mathcal{A}_1} \{\mathtt{A},\mathtt{B}\} \rightarrow_{\mathcal{A}_1} \{\mathtt{A},\mathtt{B},\mathtt{C}\} \rightarrow_{\mathcal{A}_1} \{\mathtt{C},\mathtt{D}\} \rightarrow_{\mathcal{A}_1} \{\mathtt{D}\}$$

As expected, the steps of the execution of the reaction system (depicted in Fig. 2, where dark circles indicates the activation of the corresponding genes) "mimic" the evolution of the Boolean network as described in Example 3.1.

The tool is available at the address `http://www.di.unipi.it/msvbio/software/TBN2RS.html`.

6. Simulating the Yeast-Cell Cycle Boolean Network

The cell-cycle process by which a cell goes and divides into two cells is a vital process the regulation of which is conserved among the eukaryotes [27]. The process mainly consists in four phases depicted in Figure 3.

In phase G1 the cell grows and, under appropriate conditions, commits to division. In phase S the DNA is synthesized and chromosomes replicated, G2 is the phase where the cell checks the duplicated

chromosomes, and finally in the M (Mitosis) phase the cell is divided into two. After the Mitosis phase, the cell enters the G1 phase, hence completing a "cycle". There are about 800 genes involved in the cell-cycle process of the budding yeast [33]. However, the number of key regulators that are responsible for the control and regulation of this complex process is much smaller. Based on extensive literature studies, the authors in [26] constructed a network of key regulators involving 11 genes. The relations between genes are described by the boolean network (M_{Cell}, E_{Cell}) depicted in Figure 4, where the threshold parameter θ is always 0. The boolean network was used to study the time evolution of the protein states. Starting from the $2^{11} = 2048$ possible initial states describing a configuration for gene activation, they discovered that all of them flow into one of seven attractor stationary states. In particular, among the seven fixed points there is one big attractor that attracts 1764 initial states. We translated the Boolean network (M_{Cell}, E_{Cell}) of Figure 4 into a Reaction System $\mathcal{A}_{cell} = \mathcal{RS}(M_{Cell}, E_{Cell}) = (M_{Cell}, A_{Cell})$ using the tool described in Section 5. The Reaction System obtained as result in file "rules.txt" has 50 reactions. For the sake of simplicity, we show a manipulated version of it where each reaction has a identification name. The 50 reactions are listed in Fig. 5.

Let us focus on the translation of reactions describing the production (activation) of a single node of the Boolean network. Consider the central node named `Sic1` in the Boolean network of Figure 4. It has 2 *activating* incoming arcs and 3 *inhibiting* incoming arcs. Since $\texttt{Sic1} \in M_{sa}$, by Proposition 4.5 there will be $\sum_{i=1}^{min(2,4)} \binom{2}{i} \times \binom{3}{i-1} + \sum_{i=0}^{min(2,3)} \binom{2}{i} \times \binom{3}{i} = (2+3) + (1+6+3) = 15$ reactions producing `Sic1`. Indeed, by applying our translation we find the rules $a_7 - a_{22}$ in Fig. 5 for the production of `Sic1`.

Note that the behavior of the reactions producing `Sic1` faithfully model the activation of gene `Sic1` in the Boolean network. Consider the case where genes `Cdc20`, `Swi5` and `Clb5, 6` are all active according to the Boolean network of Figure 4 after one step `Sic1` becomes active. The previous state is represented in the Reaction System as the set of activated genes $D_0 = \{\texttt{Cdc20}, \texttt{Swi5}, \texttt{Clb5}, \texttt{6}\}$. Now starting from D_0, we can apply rule $(\{\texttt{Cdc20}, \texttt{Swi5}\}, \{\texttt{Clb1}, \texttt{2}, \texttt{Cln1}, \texttt{2}\}, \{\texttt{Sic1}\})$ to obtain the production(activation) of gene `Sic1`, therefore $\texttt{Sic1} \in \texttt{D}_1$ where $D_0 \rightarrow_{\mathcal{A}_{Cell}} D_1$. Note that if `Cln1, 2` was also active in the initial state then gene `Sic1` could not be activated according to the Boolean network. This is modeled in the Reaction System by the fact that none of the 15 rules producing `Sic1` could be applied to the set $D_0 = \{\texttt{Cdc20}, \texttt{Swi5}, \texttt{Clb5}, \texttt{6}, \texttt{Cln1}, \texttt{2}\}$. Once we have obtained the complete Reaction System $\mathcal{A}_{cell}$ that simulates the entire Boolean network we can run it with any initial state D_0 in order to study the behaviour of the cell when some genes are activated. As a first experiment we executed the Reaction System with the initial state D_0 that it was observed in nature triggers the cell-cycle. Indeed, usually

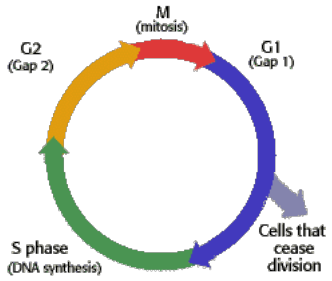

Figure 3. The complete cell-cycle

the cell stays in a stationary state where just genes Sic1 and Cdh1 are active. When the cell grows, the external cell size signal Cell size arrives and activates Cln3. This "excites" the cell from its stationary state and triggers the cycle. We can observe the different states of activation/deactivation of genes during the cell cycle by executing the Reaction System with an initial state where Sic1, Cdh1 and Cln3 are active.
Thus, the following evolution can be observed.

$$\{\texttt{Sic1}, \texttt{Cdh1}, \texttt{Cln3}\} \rightarrow_{\mathcal{A}_{\texttt{Cell}}} \{\texttt{SBF}, \texttt{MBF}, \texttt{Sic1}\} \rightarrow_{\mathcal{A}_{\texttt{Cell}}}$$
$$\{\texttt{SBF}, \texttt{MBF}, \texttt{Sic1}, \texttt{Cln1}, 2\} \rightarrow_{\mathcal{A}_{\texttt{Cell}}} \{\texttt{SBF}, \texttt{MBF}, \texttt{Cln1}, 2\} \rightarrow_{\mathcal{A}_{\texttt{Cell}}}$$
$$\{\texttt{SBF}, \texttt{MBF}, \texttt{Cln1}, 2, \texttt{Clb5}, 6\} \rightarrow_{\mathcal{A}_{\texttt{Cell}}}$$
$$\{\texttt{SBF}, \texttt{MBF}, \texttt{Cln1}, 2, \texttt{Clb5}, 6, \texttt{Clb1}, 2, \texttt{Mcm1}\} \rightarrow_{\mathcal{A}_{\texttt{Cell}}}$$
$$\{\texttt{Cln1}, 2, \texttt{Clb5}, 6, \texttt{Clb1}, 2, \texttt{Mcm1}, \texttt{Cdc20}\} \rightarrow_{\mathcal{A}_{\texttt{Cell}}} \{\texttt{Clb1}, 2, \texttt{Mcm1}, \texttt{Cdc20}, \texttt{Swi5}\} \rightarrow_{\mathcal{A}_{\texttt{Cell}}}$$
$$\{\texttt{Clb1}, 2, \texttt{Mcm1}, \texttt{Cdc20}, \texttt{Swi5}, \texttt{Sic1}\} \rightarrow_{\mathcal{A}_{\texttt{Cell}}} \{\texttt{Mcm1}, \texttt{Cdc20}, \texttt{Swi5}, \texttt{Sic1}\} \rightarrow_{\mathcal{A}_{\texttt{Cell}}}$$
$$\{\texttt{Cdc20}, \texttt{Swi5}, \texttt{Sic1}, \texttt{Cdh1}\} \rightarrow_{\mathcal{A}_{\texttt{Cell}}} \{\texttt{Swi5}, \texttt{Sic1}, \texttt{Cdh1}\} \rightarrow_{\mathcal{A}_{\texttt{Cell}}}$$
$$\{\texttt{Sic1}, \texttt{Cdh1}\} \rightarrow_{\mathcal{A}_{\texttt{Cell}}} \{\texttt{Sic1}, \texttt{Cdh1}\}$$

At this point the evolution reaches the stationary state $\{\texttt{Sic1}, \texttt{Cdh1}\}$ and the cell waits for another external stimulus to arrive, that is an external new cell size signal that activates gene Cln3 and triggers a new cycle. The evolution of the Reaction System represents the evolution of the Boolean network depicted in Figure 6 that describes the entire cell cycle. The Reaction System $\mathcal{A}_{Cell}$ can now be used for studying the influence that each gene has in the cell cycle. Each gene can be silenced in turn simply by deleting the rules that produces such gene. Note that this corresponds to simulate the Boolean network where we canceled the node representing the gene together with all his arcs. As a second example consider the case where gene SBF was silenced. To this aim, let the Reaction System $\mathcal{A}_{Cell} = (M_{Cell}, A_{Cell})$, we consider the Reaction System $\mathcal{A}_{Cell-SBF} = (M_{Cell}, A_{Cell}/\{(R_a, P_a, \{\texttt{SBF}\}) |\ a \in A_{Cell}\})$. In this case, starting from the stationary state $\{\texttt{Sic1}, \texttt{Cdh1}, \texttt{Cln3}\}$ the following evolution can be observed.

$$\{\texttt{Sic1}, \texttt{Cdh1}, \texttt{Cln3}\} \rightarrow_{\mathcal{A}_{\texttt{Cell}-\texttt{SBF}}} \{\texttt{Cdh1}, \texttt{MBF}, \texttt{Sic1}\} \rightarrow_{\mathcal{A}_{\texttt{Cell}-\texttt{SBF}}} \{\texttt{Cdh1}, \texttt{MBF}, \texttt{Sic1}\}$$

The corresponding evolution on the Boolean network is depicted in Figure 7.

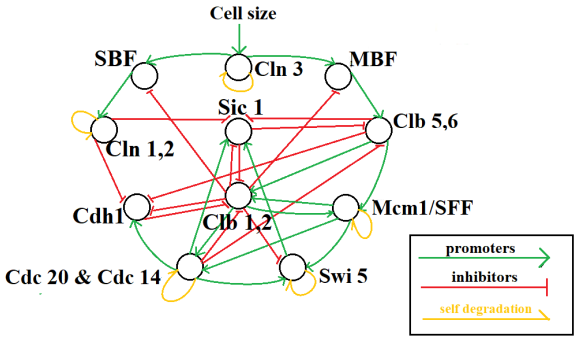

Figure 4. The Boolean network (M_{Cell}, E_{Cell}).

$$
\begin{array}{lcl}
a_1 & = & (\{Cln3\}, \{Clb1, 2\}, \{SBF, MBF\}) \\
a_2 & = & (\{SBF\}, \{Clb1, 2\}, \{SBF\}) \\
a_3 & = & (\{SBF, Cln3\}, \{\}, \{SBF\}) \\
a_4 & = & (\{MBF\}, \{Clb1, 2\}, \{MBF\}) \\
a_5 & = & (\{MBF, Cln3\}, \{\}, \{MBF\}) \\
a_6 & = & (\{SBF\}, \{\}, \{Cln1, 2\}) \\
a_7 & = & (\{Cdc20\}, \{Clb5, 6, Clb1, 2, Cln1, 2\}, \{Sic1\}) \\
a_8 & = & (\{Swi5\}, \{Clb5, 6, Clb1, 2, Cln1, 2\}, \{Sic1\}) \\
a_9 & = & (\{Sic1\}, \{Clb5, 6, Clb1, 2, Cln1, 2\}, \{Sic1\}) \\
a_{10} & = & (\{Cdc20, Swi5\}, \{Clb1, 2, Cln1, 2\}, \{Sic1\}) \\
a_{11} & = & (\{Cdc20, Swi5\}, \{Clb5, 6, Cln1, 2\}, \{Sic1\}) \\
a_{12} & = & (\{Cdc20, Swi5\}, \{Clb5, 6, Clb1, 2\}, \{Sic1\}) \\
a_{13} & = & (\{Sic1, Cdc20, Swi5\}, \{Cln1, 2\}, \{Sic1\}) \\
a_{14} & = & (\{Sic1, Cdc20, Swi5\}, \{Clb5, 6\}, \{Sic1\}) \\
a_{15} & = & (\{Sic1, Cdc20, Swi5\}, \{Clb1, 2\}, \{Sic1\}) \\
a_{16} & = & (\{Sic1, Cdc20, Swi5\}, \{Clb1, 2\}, \{Sic1\}) \\
a_{17} & = & (\{Sic1, Cdc20\}, \{Clb1, 2, Cln1, 2\}, \{Sic1\}) \\
a_{18} & = & (\{Sic1, Cdc20\}, \{Clb5, 6, Cln1, 2\}, \{Sic1\}) \\
a_{19} & = & (\{Sic1, Cdc20\}, \{Clb5, 6, Clb1, 2\}, \{Sic1\}) \\
a_{20} & = & (\{Sic1, Swi5\}, \{Clb1, 2, Cln1, 2\}, \{Sic1\}) \\
a_{21} & = & (\{Sic1, Swi5\}, \{Clb5, 6, Cln1, 2\}, \{Sic1\}) \\
a_{22} & = & (\{Sic1, Swi5\}, \{Clb5, 6, Clb1, 2\}, \{Sic1\}) \\
a_{23} & = & (\{Cdc20\}, \{Clb5, 6, Cln1, 2, Clb1, 2\}, \{Cdh1\}) \\
a_{24} & = & (\{Cdh1\}, \{Clb5, 6, Cln1, 2, Clb1, 2\}, \{Cdh1\}) \\
a_{25} & = & (\{Cdh1Cdc20\}, \{Clb5, 6, Clb1, 2\}, \{Cdh1\}) \\
a_{26} & = & (\{Cdh1Cdc20\}, \{Cln1, 2, Clb1, 2\}, \{Cdh1\}) \\
a_{27} & = & (\{Cdh1Cdc20\}, \{Clb5, 6, Cln1, 2\}, \{Cdh1\}) \\
a_{28} & = & (\{MBF\}, \{Sic1, Cdc20\}, \{Clb5, 6\}) \\
a_{29} & = & (\{MBF, Clb5, 6\}, \{Cdc20\}, \{Clb5, 6\}) \\
a_{30} & = & (\{MBF, Clb5, 6\}, \{Sic1\}, \{Clb5, 6\}) \\
a_{28} & = & (\{Clb5, 6\}, \{Sic1, Cdc20\}, \{Clb5, 6\}) \\
a_{29} & = & (\{Clb5, 6\}, \{\}, \{Mcm1\}) \\
a_{30} & = & (\{Clb1, 2\}, \{\}, \{Mcm1\}) \\
a_{31} & = & (\{Clb1, 2\}, \{\}, \{Cdc20\}) \\
a_{32} & = & (\{Mcm1\}, \{\}, \{Cdc20\}) \\
a_{33} & = & (\{Cdc20\}, \{Clb12\}, \{Swi5\}) \\
a_{34} & = & (\{Mcm1\}, \{Clb12\}, \{Swi5\}) \\
a_{35} & = & (\{Cdc20, Mcm1\}, \{\}, \{Swi5\}) \\
a_{36} & = & (\{Mcm1\}, \{Sic1, Cdh1, Cdc20\}, \{Clb1, 2\}) \\
a_{37} & = & (\{Clb5, 6\}, \{Sic1, Cdh1, Cdc20\}, \{Clb1, 2\}) \\
a_{38} & = & (\{Clb1, 2\}, \{Sic1, Cdh1, Cdc20\}, \{Clb1, 2\}) \\
a_{39} & = & (\{Mcm1, Clb5, 6\}, \{Cdh1, Cdc20\}, \{Clb1, 2\}) \\
a_{40} & = & (\{Mcm1, Clb5, 6\}, \{Cdh1, Sic1\}, \{Clb1, 2\}) \\
a_{41} & = & (\{Mcm1, Clb5, 6\}, \{Sic1, Cdc20\}, \{Clb1, 2\}) \\
a_{42} & = & (\{Mcm1, Clb1, 2\}, \{Cdh1, Cdc20\}, \{Clb1, 2\}) \\
a_{43} & = & (\{Mcm1, Clb1, 2\}, \{Cdh1, Sic1\}, \{Clb1, 2\}) \\
a_{44} & = & (\{Mcm1, Clb1, 2\}, \{Sic1, Cdc20\}, \{Clb1, 2\}) \\
a_{45} & = & (\{Clb1, 2, Clb5, 6\}, \{Cdh1, Cdc20\}, \{Clb1, 2\}) \\
a_{46} & = & (\{Clb1, 2, Clb5, 6\}, \{Cdh1, Sic1\}, \{Clb1, 2\}) \\
a_{47} & = & (\{Clb1, 2, Clb5, 6\}, \{Sic1, Cdc20\}, \{Clb1, 2\}) \\
a_{48} & = & (\{Clb1, 2, Clb5, 6, Mcm1\}, \{Cdh1\}, \{Clb1, 2\}) \\
a_{49} & = & (\{Clb1, 2, Clb5, 6, Mcm1\}, \{Sic1\}, \{Clb1, 2\}) \\
a_{50} & = & (\{Clb1, 2, Clb5, 6, Mcm1\}, \{Cdc20\}, \{Clb1, 2\})
\end{array}
$$

Figure 5. The reactions of $\mathcal{A}_{cell}$.

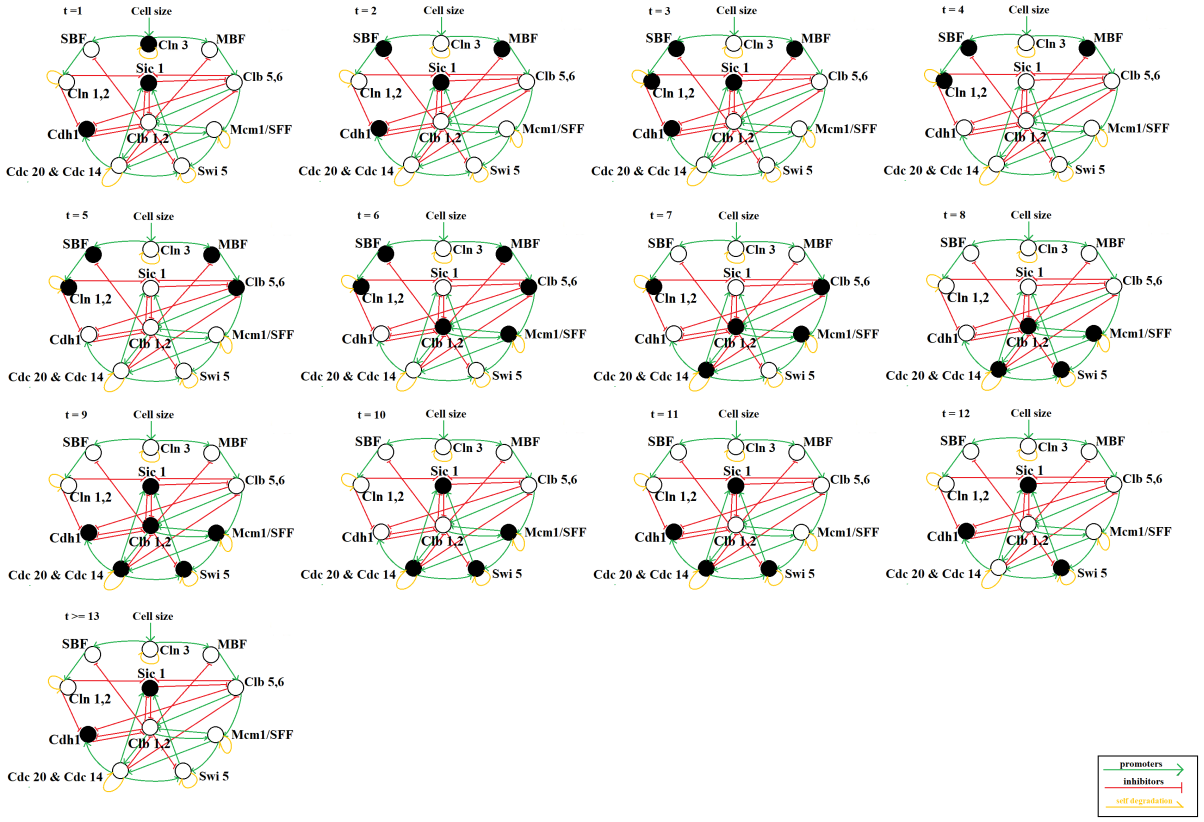

Figure 6. The cell cycle evolution

It can be observed that if gene SBF was silenced, the cell could not perform the cycle because after one step it reaches a new stationary state. This shows that the gene SBF was, in some way, necessary for the cycle to be performed. As a last example consider the case where gene Mcm1 was silenced. The evolution we obtain by considering the Reaction System $\mathcal{A}_{Cell-Mcm1} = (M_{Cell}, A_{Cell}/\{(R_a, P_a, \{\texttt{Mcm1}\}) | \ a \in A_{Cell}\})$ starting with the stationary state $\{\texttt{Sic1}, \texttt{Cdh1}, \texttt{Cln3}\}$ is directly depicted in Figure 8.

In this case we obtain a very different result from the previous one. Indeed, it can be observed that even if gene Mcm1 was silenced the cell could perform most of its cycle and go back to the initial stationary state. This suggests that the gene Mcm1 was not necessary for the cycle to be performed. Indeed, the cell can recover even if, for some reasons, gene Mcm1 could not be activated.

7. The SOS response in Escherichia coli

DNA damage in Escherichia coli evokes a response mechanism called the SOS response. The genetic circuit of this mechanism includes the genes RecA and LexA, which regulate each other via a mixed feedback loop involving transcriptional regulation and protein-protein interaction. We follow the description

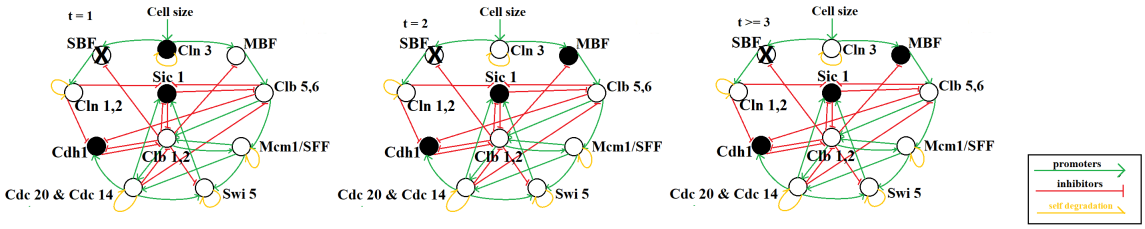

Figure 7. The cycle evolution where gene SBF was silenced

of the boolean network in [32] describing the SOS response in Escherichia coli. The DNA repair is induced in response to the existence of a single stranded DNA (ssDNA). The desired function of this network is as follows: Upon accumulation of ssDNA, RecA is recruited to the single stranded regions of DNA and becomes activated. Activation of RecA releases the inhibition of SOS genes by facilitating self-cleavage of their repressor LexA. The main activator of SOS gene is Sigma70, which belongs to a family of transcription initiation factors responsible for stress response. Its downstream genes can be simplified into two genes, SSB and UmuDC, which are responsible for the repair of DNA damage and inhibition of RecA. When the DNA repair is completed, LexA is activated and the expression of SOS genes is down-regulated [30, 20]. The natural network performing this function is presented in Fig 9. As can be observed, each gene has at most one promoter and always one inhibitor. Each node is self-activating. However, in this case the update rules of the network are slightly different from the ones that we saw for threshold Boolean networks that we saw in Section 3 since in this case the inhibitor role dominates. This means that the activation of the inhibitor always prevent the activation of the inhibited gene. In general, this kind of network are not suitable for our tool because in order to treat them we would need to allow to express some kind of weight on the promotion and inhibition arcs. Of course this will be the focus of a future extension of our translation and tool, however, in this simple case the network can be treated with our tool by simply adding some arcs in the description. The reaction system simulating the boolean network of the SOS response of the Escherichia coli has the following reactions.

```
({ssDNA},{SSB},{RecA})
({Sigma70},{RecA},{LexA})
({One},{LexA},{Sigma70})
({Sigma70},{LexA},{UmuDC})
({Sigma70} {LexA},{SSB})
({ssDNA},{UmuDC},{ssDNA})
({RecA},{SSB},{RecA})
({LexA},{RecA},{LexA})
({Sigma70},{LexA},{Sigma70})
({UmuDC},{LexA},{UmuDC})
({SSB},{LexA},{SSB})
```

Once we have the reaction system simulating the Boolean network this can be executed for simulating the updating of the Boolean network.

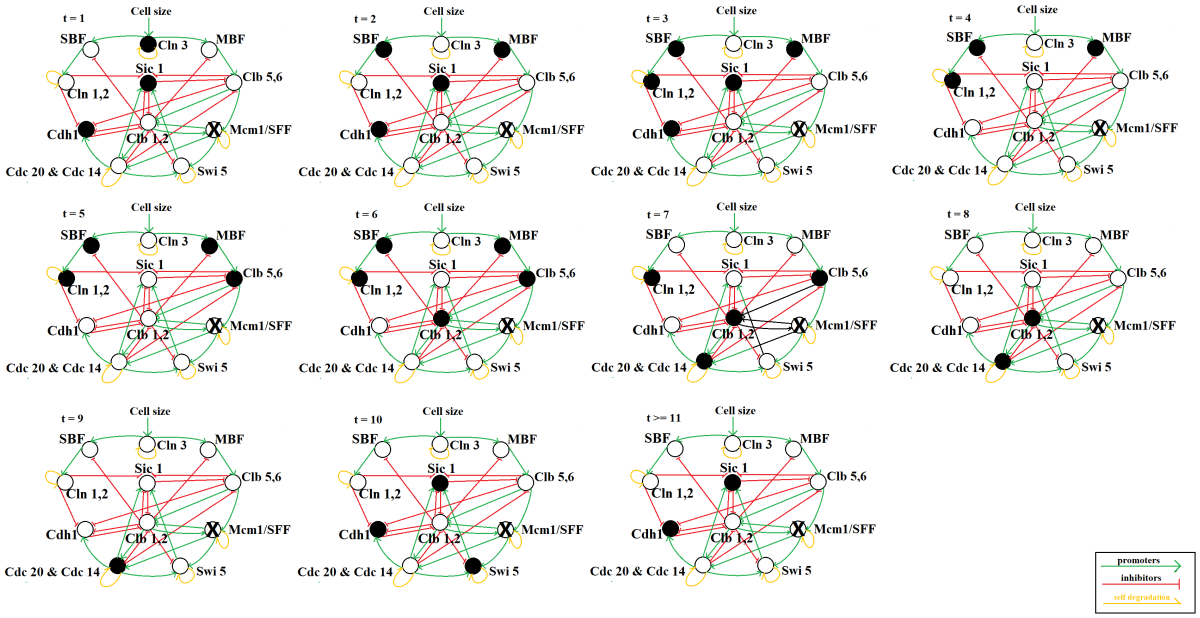

Figure 8. The cycle evolution where gene `Mcm1` was silenced

Figure 10 depicts the evolution of the reaction system that mimics what can be observed in nature. As described above, the DNA repair is induced in response to the existence of a single stranded DNA (`ssDNA`), hence in the initial configuration both `ssDNA` and `LexA` are active. Then, `ssDNA` induces the activation of `RecA` and this starts the reparing cycle which leasts for 8 steps. At the end, `LexA` remain active and waiting for another activation of `ssDNA` starting the reparation cycle again.

8. Conclusions

In this paper we proposed a method and developed a tool for the translation of threshold Boolean networks into Reaction Systems. This allows us to simulate the behaviour of a Boolean network simply by executing the Reaction System we obtain. As a first example, we applied our method to model the extensively studied Cell cycle that allows a yeast cell to split. The translation of the boolean network describing the Cell cycle into a Reaction System allowed us to investigate the role of different genes simply by "playing" with the rules related to the activation of such gene. In this way, we studied the effects of the silencing of some genes such `SBF` and `Mcm1` on the entire cell cycle. As a second example we applied our translation to the boolean network describing the response mechanism of the DNA damage in Escherichia coli (called the SOS response). We used the execution of the obtained Reaction System as a description of the behaviour of this repair mechanism that can be observed in nature.

The main advantage of our translation is that well known methods for reasoning on Reaction Systems can now be applied to study the properties of the boolean networks they simulate. For example, in [2] we applied techniques to detect dynamic causalities relations in Reaction Systems (see [13, 4, 3, 5, 6, 1]) to the translation of the Cell Cycle regulatory network to determine causality relations between genes.

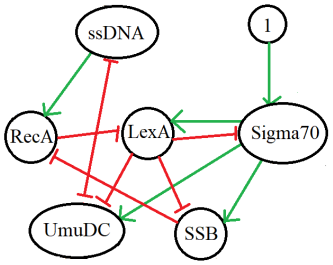

Figure 9. The boolean network of the the SOS response of the Escherichia coli.

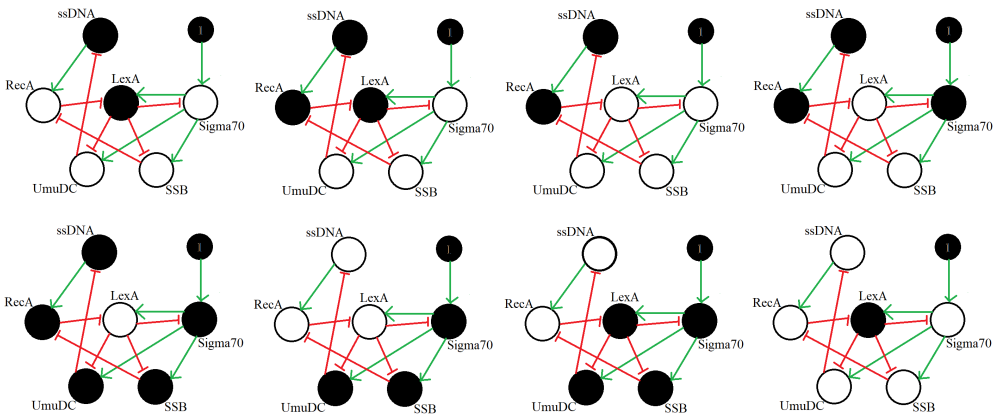

Figure 10. The evolution of the SOS response of the Escherichia coli.

Some notions of causality can be formulated in terms of information flow properties [21, 15, 14], for instance by exploiting the notion of *opacity* of Reaction Systems, introduced in [23, 22]. A development of our approach may consist in the investigation and application of a notion of *initial state opacity* for Reaction Systems obtained as translations of threshold Boolean networks. Let us suppose that we want to learn the initial state of a Boolean network based on the knowledge of a state reached after a given number of steps. The possibility of determining the initial state of the network depends on what information we can observe on the states and on the dynamics of the network itself. Such a possibility can be expressed in terms of opacity of Reaction Systems.

As a future work we plan to extend our method (and tool) to deal with threshold Boolean networks where edges are equipped with an arbitrary weight. That is, allowing a_{ij} in the definition of Boolean networks given in Section 3 to be any natural number instead of just 1 or -1. This would allows us to easily model gene regulatory networks in which inhibitors are dominant as the one described in Section 7. It is worth noting that in the general case of weigthed edges, a difficult part will be to guarantee that our

translation produces a non redundant set of rules with a minimal number of objects.

An additional further development of our approach could consist in considering *asynchronous* Boolean networks, namely models of Gene Regulatory Networks in which the activation state of at most one gene can be updated at each step. The possibilty of dealing with asynchronous networks would allow our approach to become applicable to a larger class of models of Gene Regulatory Networks. However, the synchronous nature of Reaction Systems hampers the translation of asynchronous networks. As a consequence, this development would probably require also the definition of a suitable asynchronous variant of Reaction Systems.

References

[1] Roberto Barbuti, Anna Bernasconi, Roberta Gori, and Paolo Milazzo. Computing preimages and ancestors in reaction systems. In *Theory and Practice of Natural Computing - 7th International Conference, TPNC 2018, Dublin, Ireland, December 12-14, 2018, Proceedings*, pages 23–35, 2018.

[2] Roberto Barbuti, Anna Bernasconi, Roberta Gori, and Paolo Milazzo. Characterization and computation of ancestors in reaction systems. In *Soft Computing*, 2019. Submitted for pubblication.

[3] Roberto Barbuti, Roberta Gori, Francesca Levi, and Paolo Milazzo. Specialized predictor for reaction systems with context properties. In *Proc. of the 24th Int. Workshop on Concurrency, Specification and Programming, CS&P 2015*, pages 31–43, 2015.

[4] Roberto Barbuti, Roberta Gori, Francesca Levi, and Paolo Milazzo. Investigating dynamic causalities in reaction systems. *Theoretical Computer Science*, 623:114–145, 2016.

[5] Roberto Barbuti, Roberta Gori, Francesca Levi, and Paolo Milazzo. Specialized predictor for reaction systems with context properties. *Fundamenta Informaticae*, 147(2-3):173–191, 2016.

[6] Roberto Barbuti, Roberta Gori, Francesca Levi, and Paolo Milazzo. Generalized contexts for reaction systems: definition and study of dynamic causalities. *Acta Inf.*, 55(3):227–267, 2018.

[7] Roberto Barbuti, Roberta Gori, and Paolo Milazzo. Multiset patterns and their application to dynamic causalities in membrane systems. In *Membrane Computing - 18th International Conference, CMC 2017, Bradford, UK, July 25-28, 2017, Revised Selected Papers*, pages 54–73, 2017.

[8] Roberto Barbuti, Roberta Gori, and Paolo Milazzo. Predictors for flat membrane systems. *Theor. Comput. Sci.*, 736:79–102, 2018.

[9] Chiara Bodei, Andrea Bracciali, and Davide Chiarugi. On deducing causality in metabolic networks. *BMC Bioinformatics*, 9(S-4), 2008.

[10] Chiara Bodei, Roberta Gori, and Francesca Levi. An analysis for causal properties of membrane interactions. *Electr. Notes Theor. Comput. Sci.*, 299:15–31, 2013.

[11] Chiara Bodei, Roberta Gori, and Francesca Levi. Causal static analysis for brane calculi. *Theor. Comput. Sci.*, 587:73–103, 2015.

[12] Robert Brijder, Andrzej Ehrenfeucht, Michael G. Main, and Grzegorz Rozenberg. A tour of reaction systems. *Int. J. Found. Comput. Sci.*, 22(7):1499–1517, 2011.

[13] Robert Brijder, Andrzej Ehrenfeucht, and Grzegorz Rozenberg. A note on causalities in reaction systems. *ECEASST*, 30, 2010.

[14] J Bryans, M Koutny, and P Ryan. Modelling non-deducibility using petri nets. In *Proc. of the 2nd International Workshop on Security Issues with Petri Nets and other Computational Models*, 2004.

[15] Jeremy W Bryans, Maciej Koutny, Laurent Mazaré, and Peter YA Ryan. Opacity generalised to transition systems. In *International Workshop on Formal Aspects in Security and Trust*, pages 81–95. Springer, 2005.

[16] Nadia Busi. Causality in membrane systems. In *Membrane Computing, 8th International Workshop, WMC 2007, Thessaloniki, Greece, June 25-28, 2007 Revised Selected and Invited Papers*, pages 160–171, 2007.

[17] Alberto Dennunzio, Enrico Formenti, Luca Manzoni, and Antonio E Porreca. Ancestors, descendants, and gardens of eden in reaction systems. *Theoretical Computer Science*, 608:16–26, 2015.

[18] Bernard Derrida. Dynamical phase transition in nonsymmetric spin glasses. *Journal of Physics A: Mathematical and General*, 20(11):L721–L725, 1987.

[19] Andrzej Ehrenfeucht and Grzegorz Rozenberg. Reaction systems. *Fundamenta informaticae*, 75(1-4):263–280, 2007.

[20] Timothy S. Gardner, Diego di Bernardo, David Lorenz, and James J. Collins. Inferring genetic networks and identifying compound mode of action via expression profiling. *Science*, 301(5629):102–105, 2003.

[21] Joseph A Goguen and José Meseguer. Security policies and security models. In *1982 IEEE Symposium on Security and Privacy*, pages 11–11. IEEE, 1982.

[22] Roberta Gori, Damas Gruska, and Paolo Milazzo. Hidden states in reaction systems. In *27th International Workshop on Concurrency, Specification and Programming, CS and P 2018*, volume 2240, pages 1–14. CEUR-WS, 2018.

[23] Roberta Gori, Damas Gruska, and Paolo Milazzo. Studying opacity of reaction systems through formula based predictors. *Fundamenta Informaticae*, 165(3-4):303–319, 2019.

[24] Roberta Gori and Francesca Levi. Abstract interpretation based verification of temporal properties for bioambients. *Inf. Comput.*, 208(8):869–921, 2010.

[25] K.E. Kürten. Critical phenomena in model neural networks. *Physics Letters A*, 129(3):157 – 160, 1988.

[26] Fangting Li, Tao Long, Ying Lu, Qi Ouyang, and Chao Tang. The yeast cell-cycle network is robustly designed. *Proceedings of the National Academy of Sciences*, 101(14):4781–4786, 2004.

[27] Andrew Murray and Tim Hunt. *The Cell Cycle*. Oxford Univ.Press, New York, 1993.

[28] Thimo Rohlf and Stefan Bornholdt. Criticality in random threshold networks: annealed approximation and beyond. *Physica A: Statistical Mechanics and its Applications*, 310(1):245 – 259, 2002.

[29] R. Rudell and A. Sangiovanni-Vincentelli. Multiple Valued Minimization for PLA Optimization. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 6(5):727–750, 1987.

[30] Mandana Sassanfar and Jeffrey W. Roberts. Nature of the sos-inducing signal in escherichia coli: The involvement of dna replication. *Journal of Molecular Biology*, 212(1):79 – 96, 1990.

[31] Thomas Schlitt and Alvis Brazma. Current approaches to gene regulatory network modelling. *BMC bioinformatics*, 8(6):S9, 2007.

[32] Bin Shao, Xiang Liu, Dongliang Zhang, Jiayi Wu, and Qi Ouyang. From boolean network model to continuous model helps in design of functional circuits. *PLOS ONE*, 10:1–12, 06 2015.

[33] P.T. Spellman, G. Sherlock, M.Q. Zhang, V.R. Iyer, K. Anders, M.B. Eisen, P.O Brown., D. Botstein, and B. Futcher. Comprehensive identification of cell cycle-regulated genes of the yeast saccharomyces cerevisiae by microarray hybridization. *Mol. Biol. Cell*, 9(12):3273–3297, 1998.