JOURNAL OF COMPUTATIONAL BIOLOGY
Volume 27, Number 2, 2020

Pp. 144–155
DOI: 10.1089/cmb.2019.0289

Revision of Boolean Models of Regulatory Networks Using Stable State Observations

FILIPE GOUVEIA, INÊS LYNCE, and PEDRO T. MONTEIRO

ABSTRACT

Models of biological regulatory networks are essential to understand cellular processes. However, the definition of such models is still mostly manually performed, and consequently prone to error. Moreover, as new experimental data are acquired, models need to be revised and updated. Here, we propose a model revision procedure and associated tool, capable of providing the set of minimal repairs to render a model consistent with a set of experimental observations. We consider four possible repair operations, using a lexicographic optimization criterion, giving preference to function repairs over topological ones. Also, we consider observations at stable state discarding the model dynamics. In this article, we extend our previous work to tackle the problem of repairing nodes with multiple reasons of inconsistency. We evaluate our tool on five publicly available logical models. We perform random changes considering several parameter configurations to assess the tool repairing capabilities. Whenever a model is repaired under the time limit, the tool successfully produces the optimal solutions to repair the model. Instances were generated without the previous limitation to validate this extended approach.

Keywords: Boolean functions, logical formalism, model revision, regulatory networks.

1. INTRODUCTION

BIOLOGICAL REGULATORY NETWORKS are composed of genes, proteins, and their interactions to describe complex cellular processes. Modeling such networks is particularly useful to be able to computationally reproduce existing observations, test hypotheses, and identify predictions in silico.

Different formalisms have been proposed to model the dynamical behavior resulting from the network' interacting components with different levels of detail (for a review, see Karlebach and Shamir, 2008). Here, we consider the logical formalism introduced by Thomas (1973). Network components are represented by discrete variables (we consider the Boolean case), edges represent regulatory interactions (either positive or negative), and regulatory effects are represented by Boolean functions.

The definition of such models is typically a manual task performed by a domain expert, in particular for the definition of regulatory effects, where the study of the behaviors generated by the model is compared against existing data (e.g., literature or experimental). However, the study of the generated behaviors is hampered by the combinatorial explosion of the qualitative state space. To tackle this problem, several formal verification techniques have been proposed, such as model checking to automatically verify

Department of Computer Science and Engineering, INESC-ID/Instituto Superior Técnico, Universidade de Lisboa, Lisboa, Portugal.

reachability properties (Monteiro et al., 2008), model reduction to reduce the size of the generated dynamics (Naldi et al., 2011), and the identification of attractors (Naldi et al., 2007; Hopfensitz et al., 2012), among others (Paulevé, 2018).

As the model is extended or new data are acquired, the model may become inconsistent, needing to be revised. Aside from potential topological changes, one crucial step of the model revision process is the redefinition of the component's regulatory function, a manual process that is not formally defined and therefore is prone to error.

Approaches to model revision have been proposed using Answer Set Programming (ASP) (Gebser et al., 2010, 2011; Merhej et al., 2017) and Boolean Satisfiability (SAT) (Guerra and Lynce, 2012). A recent approach was presented by Lemos et al. (2019), where ASP was successfully used to repair inconsistent logical models, although with some limitations, since it does not consider the addition of new regulators as a repair operation or the impact of changing a regulatory function in the model dynamics.

In this article, we propose an ASP-based model revision approach for the Boolean logical formalism, with four possible causes for model inconsistency and the corresponding repair operations. An optimization criterion is also proposed to determine an optimal set of repair operations to render an inconsistent model consistent. We extend a previous work (Gouveia et al., 2019) by tackling the problem of repairing an inconsistent node with multiple reasons of inconsistency, preserving the proposed optimization criteria.

The article is organized as follows. Section 2 describes the logical formalism applied to biological regulatory networks. Section 3 describes the model revision process and the proposed repair operations. The proposed approach is presented in Section 4. The implemented tool is evaluated on five publicly available biological models in Section 5. Section 6 presents the conclusion and future prospects.

2. LOGICAL REGULATORY NETWORKS

Biological regulatory networks are usually represented by a directed graph $\mathcal{G}=(V, E)$, known as regulatory graph, where nodes represent the set of components V, and the set of signed edges $E \subseteq \{(u, v, t)$: $u, v \in V; t \in \{-, +\}\}$ represents regulatory interactions. If the regulatory graph has an edge from v_i to v_j, then v_i is said to be a regulator of v_j. The sign t associated with each edge represents a positive interaction (activation) or a negative interaction (inhibition). Such regulatory graphs define the topology/structure of the network, but lack information on the components' regulatory rules.

2.1. Logical model

A logical model of a regulatory network is defined by a tuple $(V, \mathcal{K})$, where $V=\{v_1, v_2, \ldots, v_n\}$ is the set of n regulatory components of the network, where each v_i is associated with an integer value in $D_i=\{0, \ldots, max_i\}$, representing the component concentration level. A state of the network is thus defined as a vector $s \in S$, where $S=\prod_{v_i \in V} D_i$. Then $\mathcal{K}=\{K_1, K_2, \ldots, K_n\}$ is the set of n regulatory functions where K_i is the regulatory function of v_i and K_i: $S \rightarrow D_i$.

In this work, we consider only Boolean logical models where $\forall_{v_i}\ max_i=1$, that is, each component of the network is represented by a Boolean value, meaning that the component is either present (active) or absent (inactive).

2.2. Boolean functions

Let $\mathcal{B}$ be the set $\{0, 1\}$ and $\mathcal{B}^n$ be the n-dimensional Cartesian product of the set $\mathcal{B}$. Given $(x_1, \ldots, x_n) \in \mathcal{B}^n$, with $1 \leq i \leq n$, a Boolean function $f : \mathcal{B}^n \rightarrow \mathcal{B}$ is positive (resp. negative) in x_i if $f|_{x_i=0} \leq f|_{x_i=1}$ (resp. x_i if $f|_{x_i=0} \geq f|_{x_i=1}$). A given function f is monotone if it is either positive or negative for every x_i (Crama and Hammer, 2011).

A monotone Boolean function f can be represented in disjunctive normal form (DNF) (Crama and Hammer, 2011), as a disjunction of terms where each term is a conjunction of literals (Biere et al., 2009), and each variable x_i appears always as a positive literal (x_i) if f is positive in x_i, or negated ($\neg x_i$) otherwise. In other words, each component regulating another component either has a positive (resp. negative) interaction always appearing as a positive (resp. negated) literal in the DNF terms of the regulatory function.

A nondegenerate Boolean function is a function that depends on all of its variables, that is, all variables have an impact on its result. More formally, a function f is nondegenerate if all of its variables are essential.

A variable x_i is essential in f if $f|_{x_i=0}(X) \neq f|_{x_i=1}(X)$ for some $X \in \mathcal{B}^{n-1}$, and is inessential otherwise (Wegner, 1985).

In this work, we restrict the domain of the regulatory functions to the set of monotone nondegenerate Boolean functions.

2.3. Dynamics

From a given initial state of the network, the value of a component can be updated following its regulatory function, and every component can potentially change its value at any given time. The generation of successors of each state can follow a synchronous update policy, an asynchronous update policy, or other update policies. In a synchronous update policy, every component is called to update its value simultaneously, yielding a single state successor. In an asynchronous update policy, the state has a distinct successor for each component changing its value.

The generated dynamics is represented by a state transition graph (STG), where each node corresponds to a state of the network, and each edge represents a possible transition between states. A key property of interest is the identification of attractors in the STG, which typically denote subsets of states of biological interest (Hopfensitz et al., 2012). There are two types of attractors: complex and point attractors. Complex attractors are sets of mutually reachable states defined as terminal strongly connected components (SCC). If an SCC has a single state, it is denoted a point attractor or stable state, that is, a state without a transition to any other state in the STG.

In this work, we focus on the set of stable states (point attractors) of the Boolean logical model.

3. MODEL REVISION

Models of biological regulatory networks are not always in line with existing experimental observations, that is, the model cannot reproduce some experimental observations. In this case, we say that the model is inconsistent and must be revised and updated.

We consider a model to be consistent, if all its nodes are consistent. A given node is consistent if the value given by its regulatory function is the same as the one given by the experimental observation (if available), and inconsistent otherwise. In the following, we consider all the model stable states as experimental observations.

Given a Boolean logical model, we define four possible causes for inconsistency and the corresponding repair operations as shown in Table 1. Repair operations can be classified as function repairs (class F) thus changing a given regulatory function, or as topological repairs (class T) thus changing the topology of the regulatory graph.

In the following, we describe in detail the repair operations defined as well as the complete model revision procedure, defining a preference order on the proposed repairs of Table 1. In particular, we assume that the domain expert has a higher level of confidence in the correctness of the network topology than in the regulatory functions of the model. We therefore give preference to the use of class F rather than class T repairs.

3.1. Function repair

The definition of logical models still relies on domain experts to choose the best functions for every network component. However, given a component with k regulators, there are 2^{2^k} possible Boolean

Table 1. Causes of Inconsistency and Corresponding Repair Operations

Type	*Cause*	*Repair operation*	*Class*
1	Wrong regulatory function	Function change	F
2	Wrong interaction type	Edge sign flip	T
3	Wrong regulator	Edge removal	T
4	Missing regulator	Edge addition	T

F, function repair; T, topology repair.

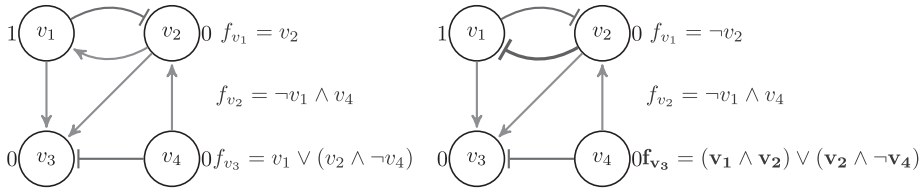

FIG. 1. Example of logical models. On the left is an inconsistent model given the observation besides each node. On the right is a consistent model (repaired).

functions to choose from, rendering this manual process prone to error. In this work, we restrict the function space to the set of monotone nondegenerate Boolean functions, which still yields a large number of functions to choose from, as a function of k.

Let f and f' be two monotone nondegenerate Boolean functions in $\mathcal{B}^n \rightarrow \mathcal{B}$, with the relationship $\preceq$ being defined as follows:

$$f \preceq f' \Leftrightarrow f(X) \Rightarrow f'(X). \tag{1}$$

A partial order set is then defined by the set of all monotone nondegenerate Boolean functions in $\mathcal{B}^n \rightarrow \mathcal{B}$ and the relationship $\preceq$ (Crama and Hammer, 2011; Cury et al., 2019), and is represented by a Hasse diagram (see Fig. 2). Considering the partial order relationship between functions, we rely on the work proposed by Cury et al. (2019) to compute the functions with minimum distance of the original one, that is, their closest neighbors. In Cury et al. (2019), a set of rules is proposed to compute the parents/children of a given function f without the need to compute the whole function space. Given two functions f and f', f' is a parent of f if and only if $f \preceq f'$ and $\nexists f''$ such that $f \preceq f''$ and $f'' \preceq f'$. In this case f is said to be a child of f'. Two functions are said to be comparable if one is an ancestor of the other, that is, if there is a sequence of parents connecting the two.

When a function is inconsistent with the experimental observations, we first determine whether it is necessary to generalize the function (go up in the Hasse diagram), or specify the function (go down in the Hasse diagram). If it is necessary to generalize (resp. specify) the function, we compute the set of parents (resp. children) of the function. We continue to go up (resp. down) the diagram while none of the parents (resp. children) is consistent.

3.2. Topology repair

Changing regulatory functions may not be enough to make a model consistent. In this case, it may be necessary to (also) change the topology of the network. Here, we consider three topology-changing repair operations: flip the sign of an edge; remove an edge; and add an edge.

Flipping the sign of an edge changes the role of a single regulator (the source of the edge). Since we consider the set of monotone nondegenerate Boolean functions, it is not possible to have regulators with dual sign, that is, regulators acting both as activators and as inhibitors. Thus, upon a flip of a sign, a negative regulator (inhibitor) becomes a positive regulator (activator) and vice versa. By adding (resp. removing) an edge, we are adding (resp. removing) regulators from the Boolean function, which effectively changes the dimension of the function space, which may have a greater impact than a function repair.

3.3. Repairing a model

A model is considered to be inconsistent and deemed to be repaired if there is at least one inconsistent node. A node is inconsistent with some experimental observation if the expected value of the node differs from the node value evaluated by the corresponding regulatory function. Here, as previously mentioned, we consider the model's stable states as experimental observations.

Figure 1 (left) shows an example of an inconsistent model. The value of the experimental observation is adjacent to each node in the graph. This model with the corresponding observations has two inconsistent nodes: v_1 and v_3. For example, node v_1 is only positively regulated by v_2 (which has an observed value of 0), and therefore, the function of v_1 evaluates to 0, but the experimental observation of v_1 is 1.

There might be many reasons for a node to be inconsistent. Assuming that we have a higher level of confidence in the correctness of the network topology than in the regulatory functions of the model, to render a node consistent, one tries first to repair the function before considering any topological change in the network. This allows the search for possible repairs in the dimension of the current function before expanding the search space considering different function dimensions. To give preference to function repairs over topological repairs, and to do a guided search for repairs, avoiding considering all the combinations of repair operations at once, the following order for repairs is defined:

(1) Repair the function;
(2) flip the sign of an edge;
(3) add/remove an edge.

In Figure 1 (left), node v_3 is inconsistent and different (function and topological) repair operations can be applied. There are nine monotone nondegenerate Boolean functions with three regulators (Fig. 2). Each of the three incoming edges can flip their sign. Also, we can remove any of the three incoming edges or add the missing edge from node v_3 to itself. If we consider the addition and removal of edges, changing the search space of the Boolean functions, and also that we can apply any combination of repair operations, we obtain a set of 2087 possible combinations of repair operations. Since models of regulatory networks typically have more than four nodes, the number of repair operations rapidly increases due to the combinatorial explosion.

We start by determining the minimum number of inconsistent nodes, to determine the solution with the minimal number of topology repairs. To repair an inconsistent node we proceed as follows:

(1) Try to repair the node by changing the regulatory function.
    - If changing the function is sufficient to make the model consistent, the procedure stops.
(2) Consider applying topological repair operations incrementally.
    (a) Consider flipping sign of edge operations.
        - Consider all the flipping edge operations incrementally until all the combinations are considered or the model is rendered consistent.
    (b) Consider add/remove edge operations incrementally as described above.

Note that whenever a topology repair operation is applied, the regulatory function must be verified for inconsistency again, and a function repair operation is most likely necessary.

Figure 1 (right) shows a repaired version of the left model, where the edge from v_2 to v_1 flipped the sign, making node v_1 consistent, and the regulatory function of v_3 is changed, making node v_3 consistent. This is an optimal model repair with minimal topological changes. Figure 2 shows the Hasse diagram for the set of monotone nondegenerate Boolean functions of node v_3. Marked in blue (dark gray) is the original function of v_3 (Fig. 1, left) and marked in green (light gray) is the regulatory function of v_3 after the repair (Fig. 1, right).

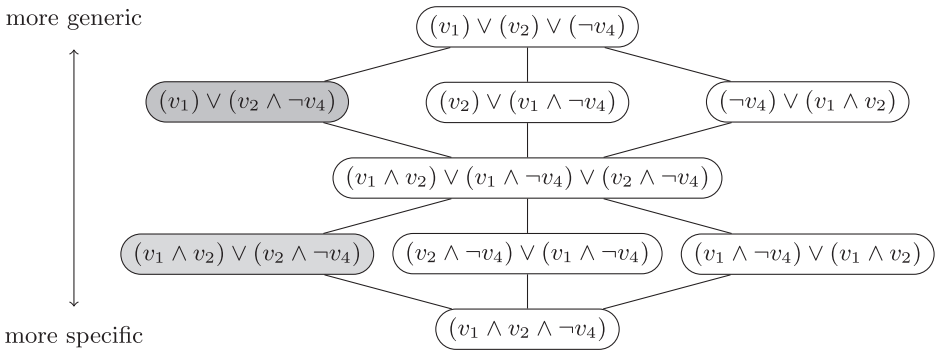

FIG. 2. Hasse diagram for monotone nondegenerate Boolean functions of three regulators (arguments v_1, v_2, and $\neg v_4$).

4. APPROACH

In this section, we describe the approach for the revision of Boolean logical models considering the set of repair operations proposed in Section 3.

As previously mentioned, one must first determine if a model contains inconsistencies to repair it. An ASP (Gebser et al., 2012) was proposed in a previous work to verify the consistency of a model given a set of experimental observations (Gouveia et al., 2018). As previously stated, in this work, we consider the model corresponding stable states, that is, we do not consider the model dynamics between specific states.

Given a model, and a set of experimental observations, the previously published ASP computes the set of minimum inconsistencies (Gouveia et al., 2018). Then, given the model, the set of experimental observations, and the computed set of inconsistencies, we developed a procedure to try to repair the model using the repair operations in Table 1 (Gouveia et al., 2019).

If the model is consistent with the data, no revision is necessary. In case of an inconsistent model, our ASP returns the minimum number of nodes that are inconsistent, that is, the minimum number of nodes to which no value can be assigned. We denote these nodes as inconsistent nodes. Moreover, since we give preference to the repair of regulatory functions, as described in Section 3, we retrieve additional information from the ASP regarding the inconsistent nodes. We define two possible reasons of inconsistency: the regulatory function of an inconsistent node needs to be more specific or the function needs to be more generic. For example, if a regulatory function of a given node produces a 0 (resp. 1) but the value of the node should be 1 (resp. 0) in order for the node to be consistent, it is likely that a more generic (resp. specific) regulatory function is needed. These reasons for inconsistency do not imply the possibility of repairing a model by changing only the regulatory function, but give us a direction to search for possible repairs, thus reducing the search space. Considering these reasons of inconsistency, an inconsistent node may have the following: a single inconsistency, if the function needs to be more specific or more generic, but not both simultaneously; or a double inconsistency, if a function may need to be more specific with respect to one observation and more generic with respect to another observation.

The proposed procedure determines the minimum repair operations necessary to make the model consistent, using the following lexicographic optimization criteria where changes to the regulatory functions are preferred over any topological change:

(1) Minimize the number of add/remove edge operations;
(2) minimize the number of flip signs of an edge operation;
(3) minimize the number of change regulatory function operations.

As an example, let us consider an inconsistent model with one inconsistent node. To determine the optimal solutions (i.e., minimal repair operations to be applied), we start by trying to change the regulatory function of the inconsistent node.

The search for possible function repairs is different in case of single or double inconsistency. The details of each case are described in Sections 4.1 and 4.2, respectively. If no function is found, we proceed to change the regulatory graph topology. In the first stage, we try to change only the sign of the incoming edges of the inconsistent node. For each attempt of flipping the sign of an edge operation, we calculate all the possible function changes again. Then, if flipping the sign of a single edge is not enough to make a model consistent, we consider combinations of two edges. We consider combinations of increasing number of edges until all the edge combinations are considered. If no solution is found, the procedure advances to the next state of topological changes, by repeating this process considering the addition or removal of one edge. If the model is still not consistent, then we consider the same process with two edges. All the combinations are considered increasingly until no more edges can be added or removed.

This process is applied to every inconsistent node. Remember that we consider only the model stable states as experimental observations, and thus repairing the consistency of one node does not impact the consistency of other nodes.

Even though, in most cases, we do not have to compute the complete Hasse diagram of all possible functions for a set of k regulators, the computation of the parents/children of a given function still greatly increases with k. Therefore, we limit all regulatory functions to a maximum of 12 regulators, since most models have regulatory functions with an average of 3 regulators and no more than 8 (Table 2).

Table 2. Boolean Logical Models Considered for Evaluation with the Corresponding: Used Abbreviation, Number of Nodes, Number of Edges, Number of Stable States, Average Number of Regulators Per Node, Maximum Number of Regulators, and Bibliographic Reference

Abbr.	*Model*	*#N*	*#E*	*#SS*	*Avg. Reg.*	*M. Reg.*	*Refs.*
FY	Fission yeast	10	27	12	3	5	Davidich and Bornholdt (2008)
SP	Segment polarity (1 cell)	19	57	7	3	8	Sánchez et al. (2002)
TCR	TCR signalization	40	57	7	1.425	5	Klamt et al. (2006)
MCC	Mammalian cell cycle	10	35	1	3.5	6	Fauré et al. (2006)
Th	Th cell differentiation	23	35	3	1.842	5	Mendoza and Xenarios (2006)

#E, number of edges; #N, number of nodes; #SS, number of stable states; Abbr., used abbreviation; Avg. Reg., average number of regulators per node; M. Reg., maximum number of regulators; Refs., bibliographic references; TCR, T cell receptor; Th, T helper.

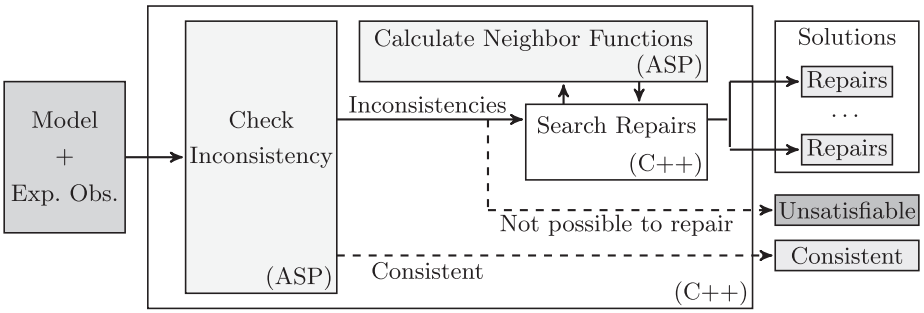

FIG. 3. Diagram of the developed tool. Model and experimental observations are the input of the tool. The tool produces as output all the optimal sets of repair operations. Marked in dashed arrows are alternative flows, where the model is consistent (no need of repair), or it is not possible to repair (no repairs produced). In *light gray* are represented the ASP components of the tool, and in *white* are the C++ components. ASP, Answer Set Programming.

Finally, we developed a tool in C++*, with the behavior illustrated in Figure 3. Given a logical model and a set of experimental observations, the tool decides whether the input is satisfiable. The input is said to be satisfiable either if it is consistent or if a repair can be found. Otherwise it is unsatisfiable.

In addition, we provide a few options to the user. The user can define nodes as fixed nodes, preventing them from being considered inconsistent. We also allow the user to define some edges as fixed, preventing solutions with repair operations that change the sign or remove these edges (see README file in tool code).

4.1. Single inconsistent functions

When a function is inconsistent with a set of observations with a single inconsistency reason, it is considered to replace it either by a more generic or a more specific function, according to the corresponding reason of inconsistency. The inconsistent function is taken as the starting point of the search for a function repair. This is performed using an ASP to compute the immediate neighbors (parents or children) of a given monotone nondegenerate Boolean function, considering the set of rules proposed by Cury et al. (2019).

Using this ASP, our procedure computes all the parents (resp. children) of the regulatory function. If none of the parents (resp. children) is consistent, the corresponding parents (resp. children) are computed, until either a consistent function is found or there are no more functions. This approach guarantees that, if a function repair operation can be applied, then the original function is modified to (one of) the nearest function(s) in the Hasse diagram that is consistent with the experimental observations, that is, having less differences in the truth table with respect to the original function.

Figure 4 (left) illustrates an example of a search for consistent functions where a more generic function is necessary. The search starts at the original function and goes up the Hasse diagram until the nearest consistent function is found.

*Tool available at http://sat.inesc-id.pt/~joaofrg/JCB2019/model_revision.zip

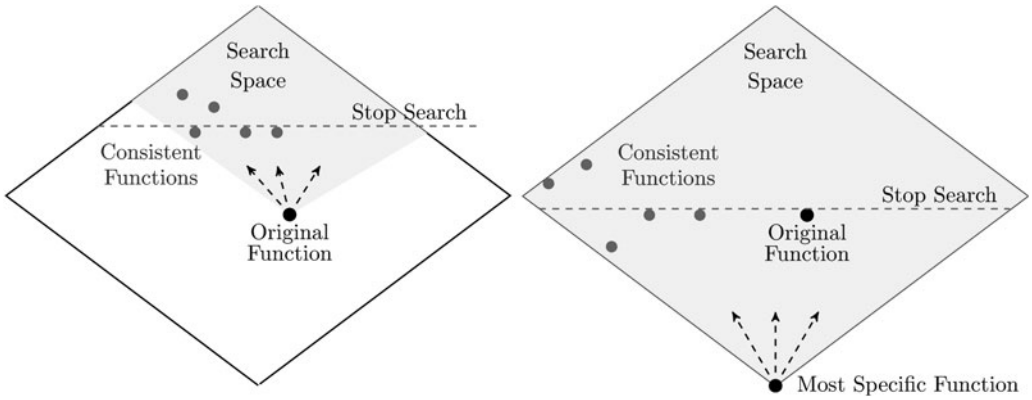

FIG. 4. Search approach of consistent functions in case of single inconsistency (left) and double inconsistency (right). The diamond shape represents the space of monotone nondegenerate Boolean functions. In *light gray* is the search space considered. Direction of the search is indicated by dashed arrows. Search stops at the dashed line.

4.2. Double inconsistent functions

In the case where a function would need to be more specific with respect to one experimental observation and more generic with respect to another observation, a different approach is needed given that, if a consistent function exists, then it must be a noncomparable function, that is, a consistent function that is neither the predecessor nor the ancestor of the original function. Note that in case of double inconsistency, at least one value of the truth table of the regulatory function needs to change from 1 to 0, and at least one value needs to change from 0 to 1. For example, in Figure 2, the function $f = (v_1) \vee (v_2 \wedge \neg v_4)$ is noncomparable to the function $f' = (v_2) \vee (v_1 \wedge \neg v_4)$ since they are neither predecessors nor ancestors of each other. It is easy to verify that these functions differ in two values in the correspondent truth table.

To repair an inconsistent function with double inconsistency, it is necessary to search for a consistent noncomparable function before considering changing the topological repair operations. To achieve this, one can start at the most specific (or most generic) function in the Hasse diagram and go up (resp. down) the diagram until a consistent function is found or the top (resp. bottom) function is reached and no consistent function is found. This approach guarantees that if a consistent function exists, it is found.

The search for a consistent noncomparable function has a big cost in terms of efficiency. In the worst-case scenario, all monotone nondegenerate Boolean functions are analyzed. Note that the search space of Boolean functions grows double exponentially with the number of regulators.

Moreover, in case of a function repair, we need to quickly determine the function (or functions) closest to the original function to minimize the impact on the model dynamics when changing the regulatory function. To do so, we consider the definition of level of a monotone nondegenerate Boolean function proposed by Cury et al. (2019). One can start the search at the most specific function and go up the diagram until a consistent function is found with the same level as the original function, or with the highest level that is lower than the original function or the lowest level that is higher than the original function.

In this work, the model revision procedure, in particular the search for a function repair, was adapted to consider the search for noncomparable functions in case of a double inconsistency. The set of closest consistent functions are produced if they exist, considering the level of the functions. This way, the optimization criteria of producing a function repair with the least impact possible are preserved.

As this search has a great impact on the tool performance since the number of possible functions increases double exponentially with the number of regulators, an optimization was considered. If an inconsistent function is closer to the most generic function than to the most specific function in the Hasse diagram, considering that it is desired to find the closest consistent function if exists, starting the search at the most specific function will probably mean that more functions will be considered than if the search starts at the most generic function. Therefore, to start the search at the closest extreme (bottom or top function), the truth table of the inconsistent function is analyzed. If the truth table of the function has more entries with value 1 than value 0, it means that the function is closer to the most generic function than to the most specific one. In this case, the search for a noncomparable function starts at the top function. Otherwise, the inconsistent function is closer to the most specific function and the search will start at the bottom function.

Figure 4 (right) illustrates an example of a search for consistent functions in case of double inconsistency. As the original function is closer to the bottom function, the search starts at the most specific function and goes up the Hasse diagram considering the whole space of functions. The search stops when consistent functions are found with the same level of the original function, or with the closest level possible.

5. EVALUATION

To evaluate the proposed approach, we considered a set of five known logical models representative of different processes and organisms (Table 2):

- Fission yeast (FY)—the cell cycle regulatory network of fission yeast by Davidich and Bornholdt (2008).
- Segment polarity (SP)—the segment polarity network that plays a role in the fly embryo segmentation by Sánchez et al. (2002).
- T cell receptor (TCR)—the T cell receptor signaling network by Klamt et al. (2006).
- Mammalian cell cycle (MCC)—the core network controlling the mammalian cell cycle by Fauré et al. (2006).
- T helper (Th)—the regulatory network controlling T helper cell differentiation by Mendoza and Xenarios (2006).

We developed an additional tool to make a set of random changes to a logical model according to the four given probabilistic parameters. Our goal was to change a logical model and then assess the repairing

Table 3. Results for Fission Yeast, Segment Polarity, T Cell Receptor Signalization, Mammalian Cell Cycle, and T Helper Cell Differentiation

(%)				*FY*		*SP*		*TCR*		*MCC*		*Th*	
F	*E*	*R*	*A*	*T (s)*	*#TO*	*T (s)*	*#TO*	*T (s)*	*#TO*	*T (s)*	*#TO*	*T (s)*	*#TO*
5	0	0	0	0.028	0	0.030	0	0.037	0	0.016	0	0.031	0
25	0	0	0	0.041	0	0.056	0	0.053	0	0.016	0	0.047	0
50	0	0	0	0.045	0	0.083	0	0.064	0	0.027	0	0.076	0
100	0	0	0	0.058	0	0.136	0	0.099	0	0.035	0	0.129	0
0	5	0	0	0.052	7	0.686	2	0.041	0	0.014	0	0.046	5
0	10	0	0	0.098	19	21.577	18	0.056	0	0.034	0	0.099	9
0	15	0	0	0.157	31	32.201	28	0.090	0	0.064	0	0.187	14
0	20	0	0	0.186	43	34.315	54	0.127	0	0.108	2	0.235	20
0	25	0	0	0.378	49	43.391	65	0.599	0	0.183	2	0.597	27
0	50	0	0	10.956	82	515.652	99	0.713	0	79.366	20	1.606	55
0	75	0	0	0.409	97	—	100	0.874	0	221.829	82	21.866	85
0	0	1	0	0.034	6	0.034	4	0.031	0	0.010	0	0.018	3
0	0	5	0	0.049	14	1.192	15	0.038	0	0.011	0	0.371	7
0	0	10	0	0.091	20	2.526	25	0.061	0	0.011	0	0.414	10
0	0	15	0	0.109	27	3.037	30	5.329	0	0.011	0	0.807	11
0	0	0	1	0.036	0	0.451	23	0.101	0	0.011	0	0.050	13
0	0	0	5	0.383	5	6.329	76	3.268	57	0.036	1	0.590	68
0	0	0	10	3.169	18	416.840	97	4.400	98	0.227	3	16.471	93
0	0	0	15	3.522	34	—	100	—	100	0.281	7	—	100
25	5	0	0	0.057	11	0.289	5	0.063	0	0.025	0	0.066	7
50	25	0	0	0.767	31	86.083	65	0.302	0	0.229	3	0.419	24
100	50	0	0	7.280	69	269.408	93	0.941	0	39.688	30	2.021	56
5	25	5	5	0.806	53	62.721	97	2.712	82	0.610	14	10.743	72
10	10	5	5	0.490	33	31.067	87	3.872	76	0.124	7	1.868	69

F%, E%, R%, and A% are the probabilistic parameters used to change the original model. Time (T), in seconds, is the median of time for the solved instances. #TO is the number of time-outs. One hundred instances were considered per configuration per model.

capabilities of the proposed tool to make the model consistent again. Four probabilistic parameters were considered: changing a function (F%); changing the sign of an edge (E%); removing an existing edge (R%); and adding a missing edge (A%).

Each model was changed with several parameter configurations, and 100 instances were considered per configuration and per model (Table 3). We considered generating instances where only the functions were changed, simulating cases where the topology of the network is correct. We also considered instances where only the sign of the edges was changed, instances where only edges were removed, and instances where only new edges were added. We generated instances with more function changes than topology changes, since we assume greater confidence in the correctness of the topology than of the regulatory functions.

Only instances that have regulatory functions with less than 12 regulators were considered, as well as only instances different from the original models, that is, instances where at least one change was applied. All experiments were run on an Intel(R) Xeon(R) 2.1 GHz 32-core Linux machine, with a time limit of 600 seconds.

Table 3 presents the results of our tool applied to different models (FY, SP, TCR, MCC, and Th) and different changes. The time is presented in seconds and corresponds to the median of times of solved instances. Whenever the model repair was possible within the time limit, the tool successfully repaired the model with a less or equal number of repair operations than the number of changes applied to the original model.

Table 3 shows that most of the repaired models can be repaired under 60 seconds. It is also possible to verify that changing the topology of the network has a bigger impact, increasing the number of time-outs as the number of changes increases. Perturbing the model with the addition of new edges has a bigger impact in the model revision process than the removal of edges. This is due to the dimension increase of the regulatory function, which greatly increases the search space for possible function repairs.

Figure 5 compares the time required to solve instances for each model. We can verify that our tool can solve more instances of the MCC model than the FY model, although these models are the smallest models with 10 nodes each. A difference between these two models is the number of stable states (Table 2). Stable states define the restrictions that are considered for a model to be consistent. Therefore, a greater number of stable states will have an impact on search for possible repair operations to render a model consistent.

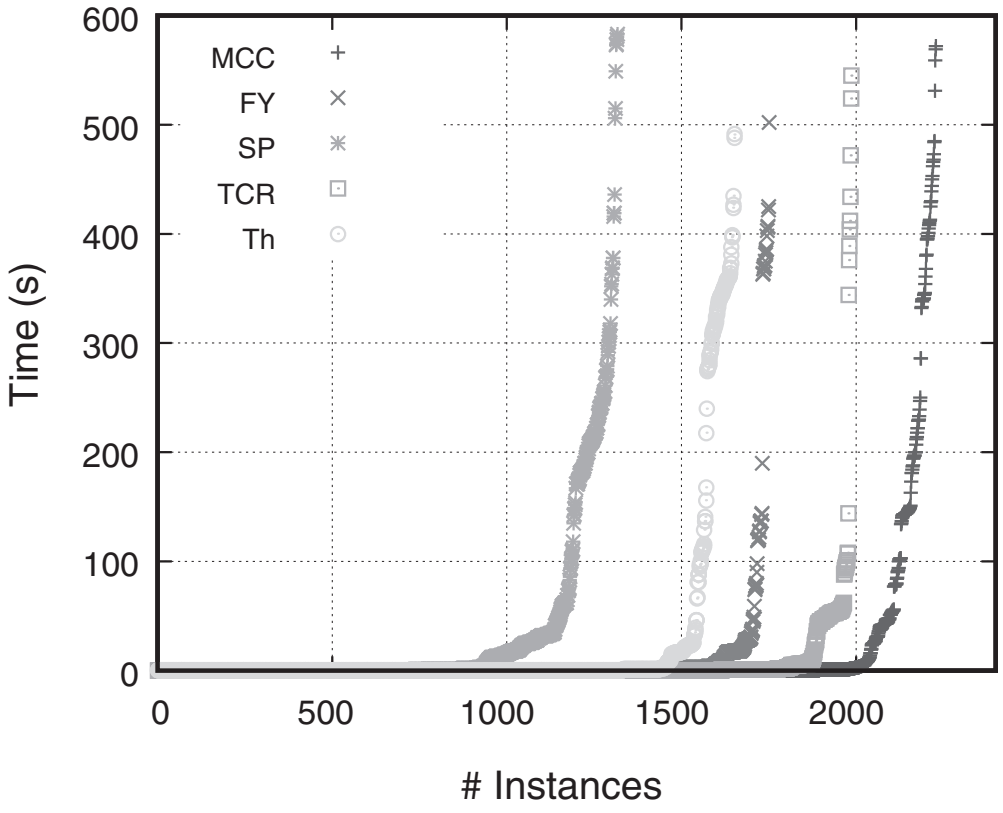

FIG. 5. Time results for each model.

Comparing the results for the SP and TCR models, one can observe that, although the TCR model is composed of more nodes (40) than the SP model (19), repairing the SP model takes more time in general, for the same number of edges (57). This means that the latter has a more interconnected network, with regulatory functions depending on a higher number of regulators. The dimension of the regulatory function greatly impacts the performance of our tool, since the number of monotone nondegenerate Boolean functions to be considered increases with a double exponential with the number of regulators.

Of the 12,000 generated instances, 28.43% have at least one node with double inconsistencies. Our tool was able to find the optimal repair sets under the time limit for 53.84% of these instances. The number of time-outs is, therefore, more notable in instances with double inconsistencies. This result was somehow expected given that, in previous work, we showed that the search for regulatory functions has a great impact on the tool performance. In the case of nodes with double inconsistency, a greater number of functions are considered in function repairs.

6. CONCLUSION AND FUTURE WORK

In this work, we propose a logical model revision tool without considering dynamics. We use ASP in the proposed tool to verify the consistency of a model, retrieve useful information in case of inconsistency, and compute possible regulatory function replacement candidates. We also use C++ to search the set of repair operations. Four repair operations are proposed: regulatory function change; edge sign flipping; edge addition; and edge removal. Our tool receives as an input a logical model and a set of experimental observations, and produces sets of repair operations to render the model consistent, under the defined optimization criteria (Section 4).

We extend our previous work to be able to repair an inconsistent node with multiple reasons of inconsistency, while preserving the optimization criteria.

The tool was successfully tested using several publicly available biological models, being able to repair most of the instances under 60 seconds. We observe that the dimension of the regulatory functions has the biggest impact on the tool performance, since the number of monotone nondegenerate Boolean functions increases.

The results showed that it is possible to repair models with double inconsistencies. Moreover, the impact of the dimension of the regulatory functions on the tool performance is more evident in these cases, since the number of functions considered for a given function repair is greater than in the case of a single inconsistency.

As a future work, time-series data could be used to also consider the model dynamics. Currently, the proposed tool produces all the optimal solutions to repair a model, which can be overwhelming for the nonexpert user. Hence, heuristics could be used to reduce the number of solutions produced [see Merhej et al. (2017) for details].

In addition, we will consider improving the efficiency of function repair search. Profiling our results showed that our tool spends most of the time computing parents and children of functions, and that this impact increases with the number of regulators of the function.

AUTHOR DISCLOSURE STATEMENT

The authors declare they have no competing financial interests.

FUNDING INFORMATION

This work was supported by national funds through Fundação para a Ciência e a Tecnologia (FCT) under contract UID/CEC/50021/2019, project ERGODiC PTDC/EEI-CTP/2914/2014, sabbatical grant SFRH/BSAB/143643/2019 to P.T.M., and PhD grant SFRH/BD/130253/2017 to F.G.

REFERENCES

Biere, A., Heule, M., and van Maaren, H. 2009. *Handbook of Satisfiability*. Volume 185. IOS Press, Amsterdam; Washington, DC.

Crama, Y., and Hammer, P.L. 2011. *Boolean Functions: Theory, Algorithms, and Applications*. Cambridge University Press, Cambridge, England.

Cury, J.E., Monteiro, P.T., and Chaouiya, C. 2019. Partial order on the set of Boolean regulatory functions. arXiv preprint arXiv:1901.07623.

Davidich, M.I., and Bornholdt, S. 2008. Boolean network model predicts cell cycle sequence of fission yeast. *PLoS One*. 3, e1672.

Fauré, A., Naldi, A., Chaouiya, C., et al. 2006. Dynamical analysis of a generic boolean model for the control of the mammalian cell cycle. *Bioinformatics* 22, e124–e131.

Gebser, M., Guziolowski, C., Ivanchev, M., et al. 2010. Repair and prediction (under inconsistency) in large biological networks with Answer Set Programming, 497–507. *In* Twelfth International Conference on the Principles of Knowledge Representation and Reasoning. AAAI Press, North America.

Gebser, M., Kaminski, R., Kaufmann, B., et al. 2012. Answer set solving in practice. *Synth. Lectures Artif. Intell. Machine Learn*. 6, 1–238.

Gebser, M., Schaub, T., Thiele, S., et al. 2011. Detecting inconsistencies in large biological networks with Answer Set Programming. *Theor. Pract. Log. Prog*. 11, 323–360.

Gouveia, F., Lynce, I., and Monteiro, P.T. 2018. Model revision of logical regulatory networks using logic-based tools, 23:1–23:10. *In* Palu', A.D., Tarau, P., Saeedloei, N., and Fodor, P., eds. *Technical Communications of the 34th International Conference on Logic Programming (ICLP 2018)*, volume 64 of *OpenAccess Series in Informatics (OASIcs)*. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl.

Gouveia, F., Lynce, I., and Monteiro, P.T. 2019. Model revision of Boolean regulatory networks at stable state, 100–112. *In* Cai, Z., Skums, P., and Li, M., eds. *International Symposium on Bioinformatics Research and Applications*. Springer International Publishing, Switzerland.

Guerra, J., and Lynce, I. 2012. Reasoning over biological networks using maximum satisfiability, 941–956. *In* Milano, M., ed. *International Conference on Principles and Practice of Constraint Programming*. Springer: Berlin, Heidelberg.

Hopfensitz, M., Müssel, C., Maucher, M., et al. 2012. Attractors in Boolean networks: A tutorial. *Comput. Stat*. 28, 19–36.

Karlebach, G., and Shamir, R. 2008. Modelling and analysis of gene regulatory networks. *Nat. Rev. Mol. Cell Biol.* 9, 770.

Klamt, S., Saez-Rodriguez, J., Lindquist, J.A., et al. 2006. A methodology for the structural and functional analysis of signaling and regulatory networks. *BMC Bioinformatics* 7, 56.

Lemos, A., Lynce, I., and Monteiro, P.T. 2019. Repairing Boolean logical models from time-series data using Answer Set Programming. *Algorithm. Mol. Biol*. 14, 9.

Mendoza, L., and Xenarios, I. 2006. A method for the generation of standardized qualitative dynamical systems of regulatory networks. *Theor. Biol. Med. Model*. 3, 13.

Merhej, E., Schockaert, S., and De Cock, M. 2017. Repairing inconsistent answer set programs using rules of thumb: A gene regulatory networks case study. *Int. J. Approx. Reason*. 83, 243–264.

Monteiro, P.T., Ropers, D., Mateescu, R., et al. 2008. Temporal logic patterns for querying dynamic models of cellular interaction networks. *Bioinformatics* 24, i227–i233.

Naldi, A., Remy, E., Thieffry, D., et al. 2011. Dynamically consistent reduction of logical regulatory graphs. *Theor. Comput. Sci*. 412, 2207–2218.

Naldi, A., Thieffry, D., and Chaouiya, C. 2007. Decision diagrams for the representation and analysis of logical models of genetic networks, 233–247. *In* Calder, M., and Gilmore, S. (eds): *Computational Methods in Systems Biology*. Springer: Berlin, Heidelberg.

Paulevé, L. 2018. Reduction of qualitative models of biological networks for transient dynamics analysis. *IEEE/ACM Transact. Comput. Biol. Bioinformatics* 15, 1167–1179.

Sánchez, L., Chaouiya, C., and Thieffry, D. 2002. Segmenting the fly embryo: Logical analysis of the role of the segment polarity cross-regulatory module. *Int. J. Dev. Biol*. 52, 1059–1075.

Thomas, R. 1973. Boolean formalization of genetic control circuits. *J. Theor. Biol*. 42, 563–585.

Wegner, I. 1985. The critical complexity of all (monotone) Boolean functions and monotone graph properties. *Inform. Control* 67, 212–222.

Address correspondence to:
Filipe Gouveia, MSc
INESC-ID/Instituto Superior Técnico
Universidade de Lisboa
Rua Alves Redol 9
1000-029 Lisboa
Portugal

E-mail: filipe.gouveia@tecnico.ulisboa.pt