Model Revision of Boolean Regulatory Networks at Stable State

Filipe Gouveia(✉), Inês Lynce, and Pedro T. Monteiro

INESC-ID/Instituto Superior Técnico, Universidade de Lisboa, Lisbon, Portugal
{filipe.gouveia,ines.lynce,pedro.tiago.monteiro}@tecnico.ulisboa.pt

Abstract. Models of biological regulatory networks are essential to understand the cellular processes. However, the definition of such models is still mostly manually performed, and consequently prone to error. Moreover, as new experimental data is acquired, models need to be revised and updated. Here, we propose a model revision tool, capable of proposing the set of minimum repairs to render a model consistent with a set of experimental observations. We consider four possible repair operations, giving preference to function repairs over topological ones. Also, we consider observations at stable state, *i.e.*, we do not consider the model dynamics. We evaluate our tool on five known 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. Also, the number of repair operations required is less than or equal to the number of random changes applied to the original model.

1 Introduction

Biological regulatory networks are composed of genes, proteins and their interactions to describe complex cellular processes. Modelling 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 (see [12] for a review). Here, we consider the logical formalism introduced by Thomas [20]. Network components are represented by discrete variables (here 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 are compared against existing

Fundação para a Ciência e a Tecnologia PhD grant SFRH/BD/130253/2017, national funds UID/CEC/50021/2019, grant SFRH/BSAB/143643/2019 and project grant PTDC/EEI-CTP/2914/2014.

Z. Cai et al. (Eds.): ISBRA 2019, LNBI 11490, pp. 100–112, 2019.
https://doi.org/10.1007/978-3-030-20242-2_9

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 reachability properties [16], model reduction to reduce the size of the generated dynamics [17], and the identification of attractors [11], among others [18].

As the model is extended or new data is acquired, the model may become inconsistent, and in that case needs to be revised. One crucial step of the model revision process is the redefinition of the component's Boolean functions, 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) [6,8,15] and Boolean Satisfiability (SAT) [10]. Here, 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.

The paper is organised 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 Sect. 4. The implemented tool is evaluated on five well-known biological models in Sect. 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. In a regulatory graph, nodes represent the set of components V and the set of edges $E \subseteq \{(u, v, t) : u, v \in V; t \in \{-, +\}\}$ represent 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. We can associate a sign t to each edge representing a positive interaction (activation) or a negative interaction (inhibition). Such regulatory graphs define the topology/structure of the network, lacking information on the components regulatory rules.

2.1 Logical Model

A logical model of a regulatory network is defined by a tuple (V, 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 $s \in S = \prod_{v_i \in V} D_i$. Then $K = \{K_1, K_2, \ldots, K_n\}$ 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 with $\forall_i \; max_i = 1$, *i.e.*, each components 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 $(x_1, \ldots, x_n) \in \mathcal{B}^n$, 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 $f|_{x_i=0} \geq f|_{x_i=1}$). Function f is *monotone* if it is either *positive* or *negative* for every x_i [2].

A monotone Boolean function f can be represented in Disjunctive Normal Form (DNF) [2], where a DNF formula is a disjunction of terms where each term is a conjunction of literals [1], and each variable x_i appears always as a positive literal (x_i) if f is positive in x_i, or it always appears 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 of the regulatory function.

A nondegenerate Boolean function is a function that depends on all of its variables, *i.e.*, 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* for 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 [21].

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, where every component is called to update their value simultaneously, yielding a single state successor; an asynchronous update policy, where the state has a distinct successor for each component changing its value; among other update policies (see [5] for details).

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 [11]. 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, *i.e.* a state without a transition to any other state in the STG.

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

3 Model Revision

Models of biological regulatory networks are not always in line with existing experimental observations, *i.e.*, the model cannot explain some experimental

Table 1. Causes of inconsistency and corresponding repair operations. Class F stands for function repair and class T stands for topology repair.

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

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). Otherwise, it is inconsistent. 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 the regulatory functions or as topology 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 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 relation $\preceq$ being defined as:

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

A partial order set (POset) is then defined by the set of all monotone nondegenerate Boolean functions in $\mathcal{B}^n \rightarrow \mathcal{B}$ and the relation $\preceq$ [2,3], and is represented by an Hasse diagram (see Fig. 3). Considering the partial order relation

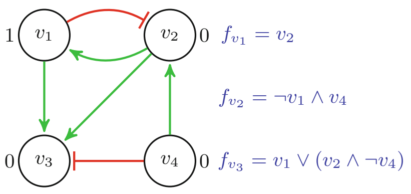
1
v_1
v_2
0
$f_{v_1} = v_2$
$f_{v_2} = \neg v_1 \wedge v_4$
0
v_3
v_4
0
$f_{v_3} = v_1 \vee (v_2 \wedge \neg v_4)$

Fig. 1. Inconsistent logical model (Color figure online).

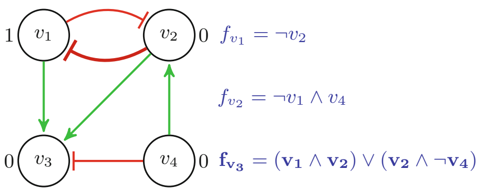
1
v_1
v_2
0
$f_{v_1} = \neg v_2$
$f_{v_2} = \neg v_1 \wedge v_4$
0
v_3
v_4
0
$\mathbf{f_{v_3} = (v_1 \wedge v_2) \vee (v_2 \wedge \neg v_4)}$

Fig. 2. Repaired logical model (Color figure online).

between functions, we rely on the work proposed by Cury *et al.* [3] to compute the functions with minimal impact to the original one, *i.e.*, their immediate neighbours. In Cury *et al.* [3] a set of rules is proposed to compute the *father/children* of a given function f without the need to compute the whole function space. Given two functions f and f', f' is a *father* 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'.

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 *fathers* (resp. *children*) of the function. We continue to go up (resp. down) the diagram if none of the *fathers* (resp. *children*) is consistent.

3.2 Topology Repair

Changing regulatory functions may not suffice 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. Since we consider the set of monotone nondegenerate Boolean functions, there are no dual regulators, *i.e.* regulators acting both as activators and as inhibitors. Thus, 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 its dimension which will likely have a greater impact than a function repair.

3.3 Repairing a Model

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

Figure 1 shows an example of an inconsistent model. The experimental observation is next 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. To render a node consistent, one tries first to repair the function before considering any topological change in the network. This allows to search for possible repairs in the dimension of the current function before expanding the search space considering different function dimensions. The ordering is therefore as follows:

1. Repair the function;
2. Flip the sign of an edge;
3. Add/remove an edge.

In Fig. 1, node v_3 is inconsistent and different (function and topological) repair operations can be applied. There are 9 monotone nondegenerate Boolean functions with 3 regulators (see Fig. 3). Each of the 3 incoming edges can flip their sign. Also, we can remove any of the 3 incoming edges or add the missing edge from node v_3 to itself. If we consider the addition and removal of edges, which changes the search space of the Boolean functions, and 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 4 nodes, the number of repair operations clearly explodes.

We start by first determining the minimum number of inconsistent nodes, to determine the solution with the minimum number of topology repairs. Therefore, first we try to repair a node by changing the regulatory function. If changing the function is not sufficient to make the model consistent we then proceed with topological repairs incrementally. We start by considering applying one topological repair operation, flipping the sign of an edge, for each possible edge. Then we consider applying two topology repair operations, and so on, until no more operations are possible. 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 2 shows the repaired model of Fig. 1, where the edge from v_2 to v_1 flipped the sign, making node v_1 consistent, and the regulatory function of v_3 changed, making node v_3 consistent. This is an optimal model repair with minimum topology changes. Figure 3 shows the Hasse diagram for the set of monotone nondegenerate Boolean functions of node v_3. Marked in blue (dark grey) is the original function of v_3 (Fig. 1) and marked in green (light grey) is the regulatory function of v_3 after the repair (Fig. 2).

4 Approach

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

As previously mentioned, one must first determine if a model contains inconsistencies in order to repair it. In previous work, we proposed an Answer Set

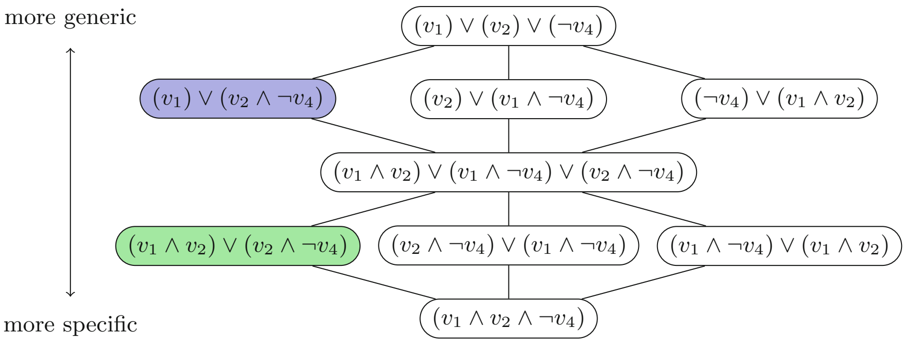
more generic
more specific
$(v_1) \lor (v_2) \lor (\neg v_4)$
$(v_1) \lor (v_2 \land \neg v_4)$
$(v_2) \lor (v_1 \land \neg v_4)$
$(\neg v_4) \lor (v_1 \land v_2)$
$(v_1 \land v_2) \lor (v_1 \land \neg v_4) \lor (v_2 \land \neg v_4)$
$(v_1 \land v_2) \lor (v_2 \land \neg v_4)$
$(v_2 \land \neg v_4) \lor (v_1 \land \neg v_4)$
$(v_1 \land \neg v_4) \lor (v_1 \land v_2)$
$(v_1 \land v_2 \land \neg v_4)$

Fig. 3. Hasse diagram for monotone nondegenerate Boolean functions of three regulators (arguments v_1, v_2, and $\neg v_4$).

Programming (ASP) [7] program to verify the consistency of a model given a set of experimental observations [9]. As previously stated, in this work, we consider the model corresponding stable states, *i.e.*, we do not consider the model dynamics between specific states.

Given a model, a set of experimental observations, and a set of inconsistencies, we developed a procedure to try to repair the model using the repair operations in Table 1.

We start by verifying the consistency of the model, using the previously developed ASP program [9]. If the model is consistent with the data, no revision is necessary. In case of an inconsistent model, our ASP program returns the minimum number of nodes that are inconsistent, *i.e.*, the minimum number of nodes to which no value can be assigned. We call these nodes *inconsistent nodes*. Moreover, since we give preference to the repair of regulatory functions, as described in Sect. 3, we retrieve additional information from the ASP program regarding the inconsistent nodes. We define two possible reasons of inconsistency: the regulatory function of an inconsistent node needs to be either more specific or more generic. For example, if a regulatory function for 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 inconsistent do not imply that we can repair a model by only changing the regulatory function, but give us a direction to search for possible repairs.

The proposed procedure determines the minimum repair operations necessary to make the model consistent, using a lexicographic optimization criterion with the following order:

1. Minimize the number of add/remove edge operations;
2. Minimize the number of flip sign of an edge operations;
3. Minimize the number of change regulatory function operations.

This order of optimization gives preference to applying changes to the regulatory functions over any topological change.

As an example, let us consider an inconsistent model with a single inconsistent node. To determine the optimum solutions (*i.e.*, minimal repair operations to be applied), we start by trying to change the regulatory function of the inconsistent node, by replacing it either by a more generic or a more specific one, according to the corresponding reason of inconsistency. This is performed using an ASP program to compute the immediate neighbours (*fathers* or *children*) of a given monotone nondegenerate Boolean function, considering the set of rules proposed by Cury *et al.* [3].

Using this ASP program, our procedure computes all the fathers (resp. children) of the regulatory function. If none of the fathers (resp. children) is consistent, it is computed the corresponding fathers (resp. children), until it finds either a consistent function 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 data, *i.e.* having less differences in the truth table. If no function is found, it proceeds to change the regulatory graph topology. In the first stage, it tries to change only the sign of the incoming edges of the inconsistent node. For each attempt of flipping the sign of an edge operation, it calculates all the possible function changes again. Then, if flipping the sign of a single edge is not enough to make a model consistent, it considers combinations of two edges, and so on until all the edges are considered. If no solution is found, the procedure advance to the next state of topological changes, by repeating this process considering adding or removing one edge. If the model is still not consistent, then considers the same process with two edges, and so on 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 data, and thus repairing the consistency of one node does not impact the consistency of other nodes.

Even though we do not have to compute the complete Hasse diagram of all possible functions for a given n, the computation of the fathers/children of a given function still greatly increases with n. 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 (see Table 2).

Also, we can only repair a model if an inconsistent node has a single reason for its inconsistency: it either needs to be more specific or more generic, but not both simultaneously. If a function would need to be more specific in one experimental observation and more generic in another observation, in order to repair the model we would have to consider the set of all monotone nondegenerate functions. Here, we only consider the set of monotone nondegenerate functions that are *comparable* with the original function. Having this limitation allows for a reduction of the search space when repairing a function.

Finally, we developed a tool in C++[1], with the behaviour illustrated in Fig. 4. 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.

[1] Tool available at http://sat.inesc-id.pt/~joaofrg/ISBRA2019/model_revision.zip.

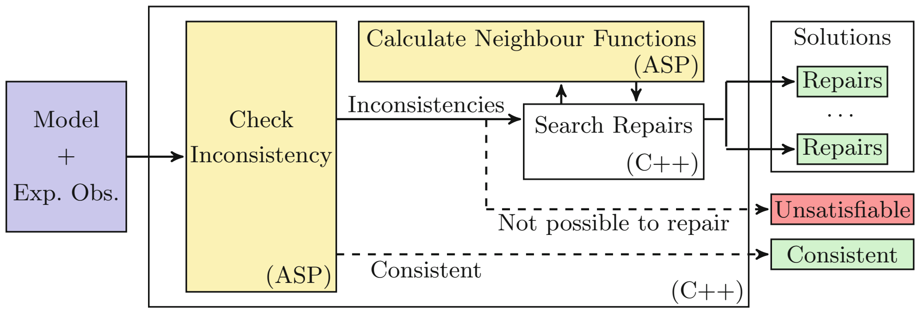

Fig. 4. Diagram of the developed tool. Model and experimental observations are the input of the tool. The tool produces as output all the optimal sets of repair operations. Marked in dashed arrows are alternative flows, where the model is consistent (no need of repair), or it is not possible to repair (no repairs produced). In yellow are represented the ASP components of the tool, and in white are the C++ components (Color figure online).

Table 2. Boolean logical models considered for evaluation with corresponding: used abbreviation (Abbr.), number of nodes (#Nodes), number of edges (#Edges), number of stable states (#SS), average number of regulators per node (Avg.Reg.), maximum number of regulators (M.Reg.), and bibliographic reference (Ref.).

Abbr.	Model	#Nodes	#Edges	#SS	Avg.Reg.	M.Reg.	Ref.
FY	Fission yeast	10	27	12	3	5	[4]
SP	Segment polarity (1 cell)	19	57	7	3	8	[19]
TCR	TCR signalisation	40	57	7	1, 425	5	[13]
MCC	Mammalian cell cycle	10	35	1	3, 5	6	[5]
Th	Th cell differentiation	23	35	3	1, 842	5	[14]

Additionally, we provide a few options to the user. First, we allow the user to prevent some repair operations. Second, 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 fixed edges.

5 Evaluation

In order to evaluate the proposed approach, we considered a set of five known logical models representative of different processes and organisms (see Table 2): the cell-cycle regulatory network of fission yeast by Davidich and Bornholdt [4]; the segment polarity network which plays a role in the fly embryo segmentation by Sanchèz *et al.* [19]; the T-Cell Receptor (TCR) signaling network by Klamt *et al.* [13]; the core network controlling the mammalian cell cycle by Fauré *et al.* [5]; and the regulatory network controlling T-helper cell differentiation by Mendoza and Xenarios [14].

Table 3. Results for FY, SP, TCR, MCC and Th. 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 timeouts. 10 instances were considered per configuration per model.

(%)				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	0,034	0	0,036	0	0,047	0	0,021	0	0,028	0
25	0	0	0	0,059	0	4,734	0	0,063	0	0,021	0	0,061	0
50	0	0	0	0,060	0	14,003	2	0,097	0	0,033	0	0,677	0
100	0	0	0	0,072	0	18,937	2	0,129	0	0,046	0	0,751	0
0	5	0	0	0,070	0	0,105	0	0,050	0	0,033	0	0,061	1
0	10	0	0	0,070	0	1,566	1	0,050	0	0,101	0	0,044	0
0	15	0	0	0,035	1	0,168	3	0,050	0	0,039	0	0,051	1
0	20	0	0	0,071	2	0,284	4	0,050	0	0,136	0	0,062	1
0	0	1	0	0,034	0	0,635	0	0,045	0	0,020	0	0,025	1
0	0	5	0	0,069	0	5,021	1	0,046	0	0,020	0	0,026	1
0	0	10	0	0,095	2	24,481	4	0,060	0	0,019	0	0,589	2
0	0	15	0	0,083	2	32,896	3	7,106	0	0,029	0	1,613	2
0	0	0	1	0,874	0	0,130	2	0,152	0	0,020	0	0,028	3
0	0	0	5	0,096	0	42,684	7	2,518	3	0,219	0	0,497	8
0	0	0	10	0,842	1	-	10	-	10	0,234	1	-	10
0	0	0	15	6,003	4	-	10	-	10	0,622	0	258,022	9
25	5	0	0	0,062	0	5,358	0	0,063	0	0,032	0	0,108	0
50	25	0	0	0,127	2	13,989	4	0,187	0	0,570	0	0,724	1
5	25	5	5	0,453	4	-	10	3,979	8	0,549	1	0,781	9
10	10	5	5	0,601	2	24,637	8	50,662	6	0,142	1	0,745	7

We developed a tool to make a set of random changes to a logical model according to four given probabilistic parameters. Our goal was to change a logical model, and then assess the repairing capabilities of the proposed tool to make the model consistent again. The set of 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%).

We changed each model with several parameters configurations, and considered 10 instances per configuration per model (see 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 we only removed edges, and instances where only were added new edges. We generated instances with more functions changes than topology changes, since we assume greater confidence in the correctness of the topology than of the regulatory functions.

Due to the limitations of our tool, we only considered instances that have regulatory functions with less than 12 regulators, and instances with a single reason for inconsistency per node (as described in Sect. 4). We also only considered instances different from the original models, *i.e.*, instances where at least one change was applied. All experiments were run on an AMD Opteron(TM) 1.4 GHz 32-core Linux machine, with a time limit of 600 s.

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 under 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 s. It is also possible to verify that changing the topology of the network has a bigger impact, increasing the number of timeouts 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.

Comparing the results for the SP and TCR models we can observe that, although the TCR model is composed of more nodes (40) than 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.

6 Conclusion and Future Work

In this work we propose a logical model revision tool without considering dynamics. We use ASP to verify the consistency of a model and retrieve useful information in case of inconsistency, and compute possible regulatory functions replacement candidates. And 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 an optimization criteria (Sect. 4).

The tool was successfully tested using several well-known biological models, being able to repair most of the instances under 60 s. We were able to conclude that the dimension of the regulatory functions has the biggest impact on the tool performance, since the number of monotone nondegenerate Boolean functions increases.

As a future work, time-series data could be used to also consider the model dynamics. Also, the possibility to repair models with inconsistent nodes with multiple reasons for inconsistency (see Sect. 4 for more details). The proposed tool produces all the optimal solutions to repair a model. Heuristics could be used to reduced the number of solutions produced (see [15] for details).

References

1. Biere, A., Heule, M., van Maaren, H.: Handbook of Satisfiability, vol. 185. IOS Press, Amsterdam (2009)
2. Crama, Y., Hammer, P.L.: Boolean Functions: Theory, Algorithms, and Applications. Cambridge University Press, Cambridge (2011)
3. Cury, J.E., Monteiro, P.T., Chaouiya, C.: Partial Order on the set of Boolean Regulatory Functions. arXiv preprint arXiv:1901.07623 (2019)
4. Davidich, M.I., Bornholdt, S.: Boolean network model predicts cell cycle sequence of fission yeast. PLoS ONE **3**(2), e1672 (2008)
5. Fauré, A., Naldi, A., Chaouiya, C., Thieffry, D.: Dynamical analysis of a generic Boolean model for the control of the mammalian cell cycle. Bioinformatics **22**(14), e124–e131 (2006)
6. Gebser, M., et al.: Repair and prediction (under inconsistency) in large biological networks with answer set programming. In: KR (2010)
7. Gebser, M., Kaminski, R., Kaufmann, B., Schaub, T.: Answer set solving in practice. Synth. Lect. Artif. Intell. Mach. Learn. **6**(3), 1–238 (2012)
8. Gebser, M., Schaub, T., Thiele, S., Veber, P.: Detecting inconsistencies in large biological networks with answer set programming. TPLP **11**(2–3), 323–360 (2011)
9. Gouveia, F., Lynce, I., Monteiro, P.T.: Model revision of logical regulatory networks using logic-based tools. In: ICLP 2018 (Technical Communications). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik (2018)
10. Guerra, J., Lynce, I.: Reasoning over biological networks using maximum satisfiability. In: Milano, M. (ed.) CP 2012. LNCS, pp. 941–956. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33558-7_67
11. Hopfensitz, M., Müssel, C., Maucher, M., Kestler, H.A.: Attractors in Boolean networks: a tutorial. Comput. Stat. **28**(1), 19–36 (2012)
12. Karlebach, G., Shamir, R.: Modelling and analysis of gene regulatory networks. Nat. Rev. Mol. Cell Biol. **9**(10), 770 (2008)
13. Klamt, S., Saez-Rodriguez, J., Lindquist, J.A., Simeoni, L., Gilles, E.D.: A methodology for the structural and functional analysis of signaling and regulatory networks. BMC Bioinform. **7**(1), 56 (2006)
14. Mendoza, L., Xenarios, I.: A method for the generation of standardized qualitative dynamical systems of regulatory networks. Theor. Biol. Med. Model. **3**(1), 13 (2006)
15. Merhej, E., Schockaert, S., De Cock, M.: Repairing inconsistent answer set programs using rules of thumb: a gene regulatory networks case study. Int. J. Approx. Reason. **83**, 243–264 (2017)
16. Monteiro, P.T., Ropers, D., Mateescu, R., Freitas, A.T., De Jong, H.: Temporal logic patterns for querying dynamic models of cellular interaction networks. Bioinformatics **24**(16), i227–i233 (2008)
17. Naldi, A., Remy, E., Thieffry, D., Chaouiya, C.: Dynamically consistent reduction of logical regulatory graphs. Theor. Comput. Sci. **412**(21), 2207–2218 (2011)
18. Paulevé, L.: Reduction of qualitative models of biological networks for transient dynamics analysis. IEEE/ACM Trans. Comput. Biol. Bioinform. **15**, 1167–1179 (2017)
19. Sánchez, L., Chaouiya, C., Thieffry, D.: Segmenting the fly embryo: logical analysis of the role of the segment polarity cross-regulatory module. Int. J. Dev. Biol. **52**(8), 1059–1075 (2002)

20. Thomas, R.: Boolean formalization of genetic control circuits. J. Theor. Biol. **42**(3), 563–585 (1973)
21. Wegner, I.: The critical complexity of all (monotone) Boolean functions and monotone graph properties. Inf. Control. **67**(1–3), 212–222 (1985)