# ModRev - Model Revision Tool for Boolean Logical Models of Biological Regulatory Networks

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

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

**Abstract.** Biological regulatory networks can be represented by computational models, which allow the study and the analysis of biological behaviours, therefore providing a better understanding of a given biological process. However, as new information is acquired, biological models may need to be revised, in order to also account for this new information. Here, we present a model revision tool, capable of repairing inconsistent Boolean biological models. Moreover, the tool is able to confront the models, both with steady state observations, as well as time-series data, considering both synchronous and asynchronous update schemes. The tool was tested with a well-known biological model that was corrupted with different random changes. The presented tool was able to successfully repair the majority of the corrupted models.

## 1 Introduction

Computational models of biological regulatory networks are of great interest in Systems Biology [7]. These models, representing complex biological processes, allow to study and analyse such processes and the corresponding biological behaviours. Such computational models accommodate the test of hypotheses, the identification of predictions *in silico*, and the identification of network properties of biological regulatory networks.

As new experimental data become available, computational models may become inconsistent, i.e., models may not be able to reproduce the new information acquired. In this case, models need to be revised [8]. However, this model revision process is mainly a manual task, performed by a modeler, and therefore prone to error. Moreover, repairing an inconsistent model is not an easy task, due to the inherent combinatorial problem associated to all the possible changes that can be made to render a model consistent. Furthermore, the construction of biological models is also typically a manual task, thus accentuating the importance of the model revision process.

This work was supported by national funds through Fundação para a Ciência e a Tecnologia (FCT) with reference SFRH/BD/130253/2017 (PhD grant) and UIDB/50021/2020 (INESC-ID multi-annual funding).


A. Abate et al. (Eds.): CMSB 2020, LNBI 12314, pp. 339–348, 2020.
https://doi.org/10.1007/978-3-030-60327-4_18

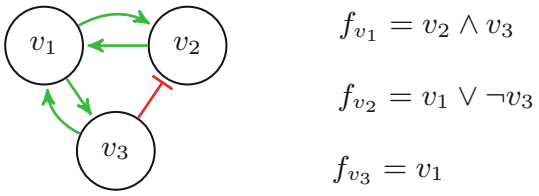


**Fig. 1.** Example of a Boolean logical model.

This paper presents a new tool, ModRev, that combines and implements the methods for model revision from previously published works [5,6]. ModRev is capable of assessing whether a Boolean logical model of a biological regulatory network is consistent with new experimental observations. In case of inconsistency, the tool repairs the model to render it consistent.

ModRev is able to consider steady state observations or time-series observations as experimental data. Moreover, both synchronous and asynchronous update schemes are supported when considering time-series observations.

The paper is organised as follows. Section 2 presents the background and related work. The ModRev tool is presented in Sect. 3. Experimental results are shown in Sect. 4. We discuss the tool features and prospects in Sect. 5.

## 2 Preliminaries

Biological regulatory networks are composed of biological compounds, and the corresponding interactions, representing complex biological processes. Different formalisms can be use to build computational models of regulatory networks, such as Ordinary Differential Equations (ODE) [7], Piecewise Linear Differential Equations (PLDE) [2], Logical Formalism [14], Sign Consistency Model (SCM) [13], among others [2]. Here, we consider the Boolean logical formalism [14], which has proven useful to study and analyse biological behaviours.

A Boolean logical model is usually represented by a *regulatory graph* and a set of *regulatory functions*. A *regulatory graph* is defined as a tuple $(V, E)$ where $V$ is a set of nodes representing the biological compounds, and $E$ is the set of directed edges representing interactions between biological compounds. Each node is associated with a Boolean variable, representing whether the corresponding compound is present or absent. Edges are associated with a sign, representing positive interactions (activations) or negative interactions (inhibitions). If there is an edge from node $v_1$ to node $v_2$ we say that $v_1$ is a *regulator* of $v_2$. Each node is also associated with a *regulatory function*, which is a Boolean function that given the value of that node regulators determines its next value.

Figure 1 shows an example of a regulatory graph and corresponding regulatory functions, where green pointed arrows represent positive interactions, and red blunt arrows represent negative interactions.

### 2.1 Related Work

Few approaches of model revision processes have been proposed. A first approach of model revision over Thomas' logical formalism [14] considers that a model is inconsistent if it is over-constrained [10]. The revision process removes constraints until the model becomes consistent, probably leading to under-constrained models, not representing correctly the real biological process. Some approaches to model revision consider the Sign Consistency Model formalism [3,13]. This formalism, although similar to the logical formalism, relies on the sign algebra for the sign of the regulatory functions. Therefore, this type of models lacks in expressiveness in the definition of regulatory functions when compared to the logical models. In [9], properties found in literature, called rule of thumbs, are considered to repair inconsistent models. However, this approach has limitations regarding the repair operations that can be performed, the definition of regulatory functions, and the generation of the networks' dynamics. Recently, a model revision approach was proposed for Boolean logical models, with more expressiveness regarding the definition of regulatory functions [8]. However, it does not take into account the impact of the regulatory function on the networks' dynamics. Also, it does not consider adding a missing regulator in the model as a possible repair operation.

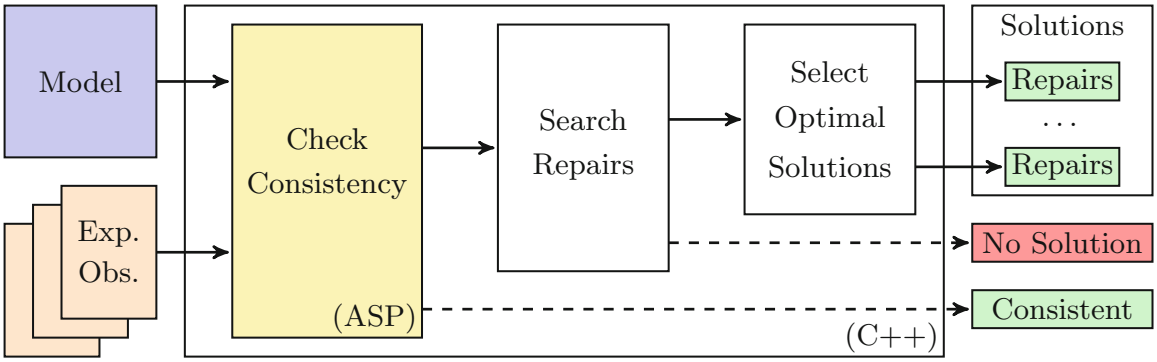


**Fig. 2.** Tool architecture.

## 3 ModRev Tool

ModRev is a freely available model revision tool for Boolean logical models of biological regulatory networks[1]. This paper presents a tool that implements the model revision methods presented in [5] to repair inconsistent models under steady state, and implements the method presented in [6] for time-series observations (see [5,6] for a detailed description of the methods).

Considering a Boolean logical model and a set of experimental observations, ModRev determines whether the model is consistent with the observations. In case of inconsistency, it determines the minimum set of nodes that must be

[1] https://filipegouveia.github.io/ModelRevisionASP/.

repaired. Four possible repair operations are considered: change a regulatory function; change the type of interaction (from activation to inhibition and vice-versa); remove a regulator; and add a regulator.

In order to repair an inconsistent model, the following lexicographic optimisation criteria is defined to minimise the number of operations of: 1) add/remove regulator; 2) change interaction type; 3) function change. These criteria allows to give preference to function changes over changes in the structure of the network.

Figure 2 illustrates the tool architecture. Dashed arrows represent alternative flows, where it is not possible to repair a model, or the model is already consistent and no repair is needed.

### 3.1 Input and Output

Regulatory functions supported by ModRev are monotone non-degenerate Boolean functions. Biologically, a monotone function means that each regulator only has one role, either an activator, or an inhibitor, but not both. A non-degenerate function means that each regulator influences the output of the regulatory function. Otherwise, it should not be a regulator. This model revision process requires the regulatory functions to be represented in Blake Canonical Form[1], which is a disjunction of all the prime implicants of the function [5,6].

The ModRev tool is based on Answer Set Programming (ASP) [4], and the input is defined using ASP predicates. To represent a Boolean logical model we use the predicate `vertex(V)`, to indicate that `V` is a node of the regulatory graph, and the predicate `edge(V1,V2,S)` to represent an edge from `V1` to `V2` with a sign `S` $\in \{0, 1\}$, where 0 (1) represents a negative (positive) interaction. The predicate `vertex` may be omitted if the node can be inferred from `edge` predicates. To represent regulatory functions, we use the predicate `functionOr(V,1..N)` that indicates that the regulatory function of `V` is a disjunction of `N` terms. The predicate `functionAnd(V,T,R)` is then used to represent that node `R` is a regulator of `V` and is present in the term `T` of the regulatory function.

ModRev is able to confront a model with a set of experimental observations, either in steady state, or a time-series data. To represent the set of experimental observations, the predicate `exp(E)` is used to identify an experimental observation `E`. To represent the observed values of an experiment `E`, we use the predicate `obs_vlabel(E,V,S)`, which means that node `V` in experiment `E` has an observed value `S` $\in \{0, 1\}$ considering steady state observations. If time-series observations are to be considered instead, a similar predicate (`obs_vlabel(E,T,V,S)`) is used, where `T` represents the time-step of the observed value.

If ModRev identifies that a given model is not consistent with a set of observations, it produces all the optimum solutions (repairs) that render the model consistent. Considering the optimisation criteria defined above, the optimum set of repair operations are produced.

## 4 Experimental Evaluation

We tested our tool using a Boolean logical model of the segment polarity (SP) network which plays a role in the fly embryo segmentation [12]. We corrupted the model using four probability parameters (in percentage): F, the probability of changing a regulatory function; E, the probability of changing the sign of an edge; R, the probability of removing a regulator; and A, the probability of adding a regulator. Table 1 shows 24 combinations of these parameters that have been considered. For each parameter configuration, 100 corrupted instances were generated. Also, five time-series observations with twenty time-steps were considered.

Given a corrupted model and a set of experimental observations, our tool is able to repair most of the models under a time limit of one hour. Figure 3 shows the median solving times for each configuration. Considering the synchronous update scheme, it is possible to observe that, for the configurations with added or removed regulators, a greater repair time is needed. This is due to the change in the dimension of the regulatory function, which has a big impact on the tool performance. Considering the asynchronous update scheme, we can verify that the tool repairs the corrupted models in less than 2 s. This difference between the

**Table 1.** Percentage values of F, E, R, and A parameters, of the 24 configurations.

| Config. | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| F | 5 | 25 | 50 | 100 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 25 | 50 | 100 | 5 | 10 |
| E | 0 | 0 | 0 | 0 | 5 | 10 | 15 | 20 | 25 | 50 | 75 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 5 | 25 | 50 | 25 | 10 |
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 5 | 10 | 15 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 5 | 5 |
| A | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 5 | 10 | 15 | 0 | 0 | 0 | 5 | 5 |

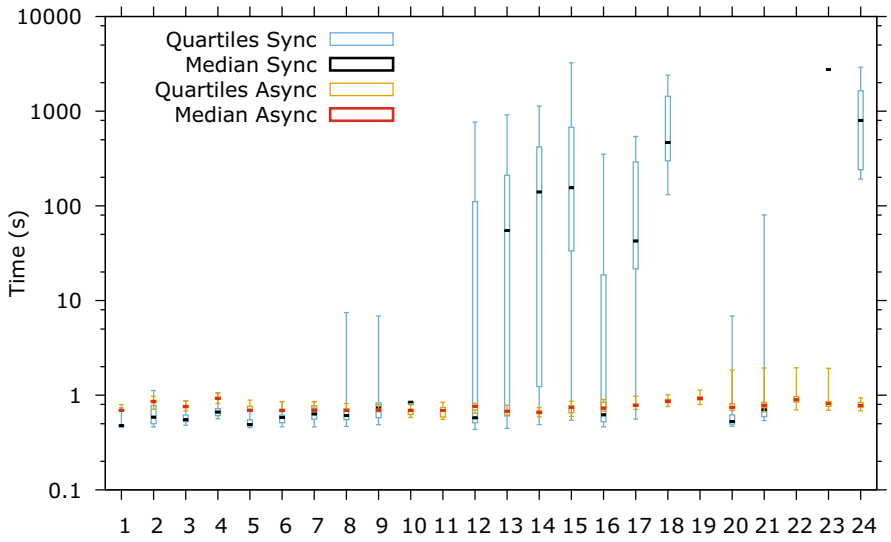


**Fig. 3.** Median time in seconds of solved instances for each corruption configuration, under synchronous and asynchronous update schemes.

two update schemes relies on the fact that, in the asynchronous update, only one regulatory function is updated at each time step. Therefore, fewer constraints must be verified when looking for possible repair operations.

## 5 Discussion

Currently, the interaction (input/output) with the ModRev tool is based on ASP predicates. To be able to facilitate the future interoperability with the qualitative modelling community, we plan to implement an import/export facility to be integrated into the BioLQM library [11]. Additionally, we are currently improving the comparison with time-series data through the implementation of the fully asynchronous update scheme. This will allow to be more permissive on the generated dynamics.

## A Tutorial

The model shown in Fig. 1 is represented by the following listing:

```
vertex(v1). vertex(v2). vertex(v3).
edge(v1,v2,1). edge(v1,v3,1). edge(v2,v1,1).
edge(v3,v1,1). edge(v3,v2,0).
functionOr(v1,1..1).
functionAnd(v1,1,v2). functionAnd(v1,1,v3).
functionOr(v2,1..2).
functionAnd(v2,1,v1). functionAnd(v2,2,v3).
functionOr(v3,1..1).
functionAnd(v3,1,v1).
```

Now let us consider that we want to define a steady state observation in which $v_1$ has value 0, $v_2$ has value 0, and $v_3$ has value 1, as following:

```
exp(p1).
obs_vlabel(p1,v1,0). obs_vlabel(p1,v2,0). obs_vlabel(p1,v3,1).
```

Using ModRev tool, giving the model defined above (as a file `model.lp`) and the steady state (as a file `obsSS.lp`), execute the following command:

```
$ ./modrev -m model.lp -obs obsSS.lp -ss
```

```
### Found solution with 1 repair operation.
    Inconsistent node v3.
        Repair #1:
            Flip sign of edge (v1,v3).
```

This output means that the model in Fig. 1 can be repaired by changing the interaction type between $v_1$ and $v_3$. If we repair the model and execute the above command again, the result will be:

```
This network is consistent!
```

Now let us assume that the user knows that the interaction between $v_1$ and $v_3$ is correct, and wants to prevent repairs over it. The predicate `fixed(v1,v3).` can be used to define that the edge between these nodes can not be changed or removed. Adding this predicate to the model and running the command above, we obtain the following result:

```
### Found solution with 2 repair operations.
    Inconsistent node v3.
        Repair #1:
            Change function of v3 to (v1) || (v3)
            Add edge (v3,v3) with sign 1.
```

A different set of repair operations is obtained that does not change the fixed edge. Now assume that the user wants to prevent any repair over the node $v_3$. The predicate `fixed(v3).` can be used to prevent that node to be inconsistent. However, in this example, if we prevent any change to node $v_3$, considering its regulatory function, and that $v_1$ has value 0 and $v_3$ has value 1, and we are in the presence of a steady state, it becomes impossible to repair the network. In this case, when the model is over-constrained, using the same command as before, the tool produces the following message:

```
It is not possible to repair this network.
```

Consider now that we have, for the same model in Fig. 1, a time-series data as shown in Table 2. Consider that this experimental observation with three time-steps (0, 1 and 2) is considering a synchronous update scheme.

**Table 2.** Synchronous time-series data

| | | Time | | |
|---|---|---|---|---|
| | | 0 | 1 | 2 |
| Node | $v_1$ | 0 | 1 | 0 |
| | $v_2$ | 0 | 0 | 0 |
| | $v_3$ | 1 | 0 | 0 |

We can represent the time-series data using the following listing:

```
#const t = 2.
exp(p2).
obs_vlabel(p2,0,v1,0). obs_vlabel(p2,0,v2,0).
obs_vlabel(p2,0,v3,1).
obs_vlabel(p2,1,v1,1). obs_vlabel(p2,1,v2,0).
obs_vlabel(p2,1,v3,0).
obs_vlabel(p2,2,v1,0). obs_vlabel(p2,2,v2,0).
obs_vlabel(p2,2,v3,0).
```

Note that we start the file indicating the maximum value of time step with `#const t = 2`.

Using ModRev to verify whether the model is consistent, while considering the above time-series data (as a file `obsTS01.lp`) under a synchronous update scheme, execute the following command:

```
$ ./modrev -m model.lp -obs obsTS01.lp -up s
```

This will produce the following result:

```
### Found solution with 5 repair operations.
    Inconsistent node v1.
        Repair #1:
            Change function of v1 to (v2) || (v3)
   Inconsistent node v2.
        Repair #1:
            Change function of v2 to (v1 && v3)
            Flip sign of edge (v1,v2).
        Repair #2:
            Change function of v2 to (v1 && v3)
            Flip sign of edge (v3,v2).
   Inconsistent node v3.
        Repair #1:
            Change function of v3 to (v1 && v2)
            Add edge (v2,v3) with sign 1.
        Repair #2:
            Change function of v3 to (v1 && v3)
            Add edge (v3,v3) with sign 1.
```

Note that now we have multiple choices to render the model consistent. To repair node $v_2$, for example, one can apply the operations in `Repair #1` or in `Repair #2`. The same applies to repair node $v_3$.

If instead of a time-series data under a synchronous update scheme, we are under an asynchronous update scheme, the previous command would change from `-up s` to `-up a`. The option `-up` indicates the update scheme to be considered, with argument `s` for synchronous and `a` for asynchronous.

ModRev also supports incomplete time-series data. Assume that we have the experimental observation shown in Table 3, where node $v_3$ was not observed, and a value of $v_1$ was also not observed.

Consider the following representation of an incomplete time-series data:

```
#const t = 2.
exp(p3).
obs_vlabel(p3,0,v1,0). obs_vlabel(p3,0,v2,1).
obs_vlabel(p3,1,v2,0).
obs_vlabel(p3,2,v1,1). obs_vlabel(p3,2,v2,0).
```

**Table 3.** Incomplete synchronous time-series data

| Node | Time 0 | 1 | 2 |
|---|---|---|---|
| $v_1$ | 0 | | 1 |
| $v_2$ | 1 | 0 | 0 |
| $v_3$ | | | |

Executing the following command, while considering the above experimental observation (as a file `obsTS02.lp`) under synchronous update scheme, produces the result below.

```
$ ./modrev -m model.lp -obs obsTS02.lp -up s
```

```
### Found solution with 3 repair operations.
    Inconsistent node v1.
        Repair #1:
            Change function of v1 to (v2) || (v3)
            Flip sign of edge (v2,v1).
   Inconsistent node v2.
        Repair #1:
            Change function of v2 to (v1 && v3)
```

ModRev tool also supports confronting a model with multiple experimental observations at the same time. For example, we could confront the model of Fig. 1 with the two time-series data above, using the command:

```
$ ./modrev -m model.lp -obs obsTS01.lp obsTS02.lp -up s
```

Note that the directive `#const t = 2` must only be defined once.

## References


1. Crama, Y., Hammer, P.L.: Boolean Functions: Theory, Algorithms, and Applications. Cambridge University Press, Cambridge (2011)
2. De Jong, H.: Modeling and simulation of genetic regulatory systems: a literature review. J. Comput. Biol. **9**(1), 67–103 (2002)
3. Gebser, M., et al.: Repair and prediction (under inconsistency) in large biological networks with answer set programming. In: Lin, F., Sattler, U., Truszczynski, M. (eds.) Principles of Knowledge Representation and Reasoning: Proceedings of the Twelfth International Conference, KR 2010. AAAI Press (2010)
4. Gebser, M., Kaminski, R., Kaufmann, B., Schaub, T.: Answer set solving in practice. In: Synthesis Lectures on Artificial Intelligence and Machine Learning, vol. 6, no. 3, pp. 1–238 (2012)
5. Gouveia, F., Lynce, I., Monteiro, P.T.: Revision of Boolean models of regulatory networks using stable state observations. J. Comput. Biol. **27**(2), 144–155 (2020)

6. Gouveia, F., Lynce, I., Monteiro, P.T.: Semi-automatic model revision of Boolean regulatory networks: confronting time-series observations with (a)synchronous dynamics. bioRxiv preprint (2020) https://doi.org/10.1101/2020.05.10.086900
7. Karlebach, G., Shamir, R.: Modelling and analysis of gene regulatory networks. Nat. Rev. Mol. Cell Biol. **9**(10), 770 (2008)
8. Lemos, A., Lynce, I., Monteiro, P.T.: Repairing Boolean logical models from time-series data using Answer Set Programming. Algorithms Molecular Biol. **14**(1), 9 (2019)
9. Merhej, E., Schockaert, S., Cock, M.D.: Repairing inconsistent answer set programs using rules of thumb: a gene regulatory networks case study. Int. J. Approximate Reason. **83**, 243–264 (2017)
10. Mobilia, N., Rocca, A., Chorlton, S., Fanchon, E., Trilling, L.: Logical modeling and analysis of regulatory genetic networks in a non monotonic framework. In: Ortuño, F., Rojas, I. (eds.) IWBBIO 2015. LNCS, vol. 9043, pp. 599–612. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-16483-0_58
11. Naldi, A.: BioLQM: A java toolkit for the manipulation and conversion of logical qualitative models of biological networks. Front. Physiol. **9** (2018)
12. 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)
13. Siegel, A., Radulescu, O., Le Borgne, M., Veber, P., Ouy, J., Lagarrigue, S.: Qualitative analysis of the relation between DNA microarray data and behavioral models of regulation networks. Biosystems **84**(2), 153–174 (2006)
14. Thomas, R.: Boolean formalization of genetic control circuits. J. Theor. Biol. **42**(3), 563–585 (1973)