Learning Boolean logic models of signaling networks with ASP

Santiago Videla, Carito Guziolowski, Federica Eduati, Sven Thiele, Martin Gebser, Jacques Nicolas, Julio Saez-Rodriguez, Torsten Schaub, Anne Siegel

Learning Boolean logic models of signaling networks with ASP

Santiago Videla[a,b,c], Carito Guziolowski[d], Federica Eduati[e], Sven Thiele[b,a], Martin Gebser[c], Jacques Nicolas[b,a], Julio Saez-Rodriguez[e], Torsten Schaub[c], Anne Siegel[a,b]

[a] *UMR CNRS 6074 IRISA, Campus de Beaulieu, 35042 Rennes, France*
[b] *INRIA, Dyliss project, Campus de Beaulieu, 35042 Rennes, France*
[c] *Universität Potsdam, Institut für Informatik, August-Bebel-Str. 89, D-14482, Deutschland*
[d] *École Centrale de Nantes, IRCCyN UMR CNRS 6597, 1 rue de la Noë, 44321, Nantes, France*
[e] *European Molecular Biology Laboratory, European Bioinformatics Institute, Hinxton CB10 1SD, UK*

Abstract

Boolean networks provide a simple yet powerful qualitative modeling approach in systems biology. However, manual identification of logic rules underlying the system being studied is in most cases out of reach. Therefore, automated inference of Boolean logical networks from experimental data is a fundamental question in this field. This paper addresses the problem consisting of learning from a prior knowledge network describing causal interactions and phosphorylation activities at a pseudo-steady state, Boolean logic models of immediate-early response in signaling transduction networks. The underlying optimization problem has been so far addressed through mathematical programming approaches and the use of dedicated genetic algorithms. In a recent work we have shown severe limitations of stochastic approaches in this domain and proposed to use Answer Set Programming (ASP), considering a simpler problem setting. Herein, we extend our previous work in order to consider more realistic biological conditions including numerical datasets, the presence of feedback-loops *in the prior knowledge network* and the necessity of multi-objective optimization. In order to cope with such extensions, we propose several discretization schemes and elaborate upon our previous ASP encoding. Towards real-world biological data, we evaluate the performance of our approach over *in silico* numerical datasets based on a real and large-scale prior knowledge network. The correctness of our encoding and discretization schemes are dealt with in a separate appendix.

Keywords: answer set programming, signaling transduction networks, boolean logic models, combinatorial multi-objective optimization, systems biology

1. Introduction

Systems biology is an emerging field aiming at the investigation and understanding of biology at a system and multi-scale level. After biological entities have been identified in a specific environment, it remains to elucidate how they interact with each other in order to carry out a particular biological function. Therefore the construction of mathematical and predictive models is a fundamental goal of this field.

Cells respond to their environment by activating signaling networks that trigger processes such as growth, survival, apoptosis (cell death), and migration. Post-translational modifications, notably protein phosphorylation, play a key role in signaling. Nowadays, there exist public repositories such as Pathways Commons [10], Pathways Interaction Database [53] and KEGG [35] that contain curated knowledge about intracellular causal molecular interactions, from which canonical cell signaling networks can be retrieved [29]. Such

Email addresses: santiago.videla@irisa.fr (Santiago Videla), carito.guziolowski@irccyn.ec-nantes.fr (Carito Guziolowski), federica_eduati@hotmail.com (Federica Eduati), sven.thiele@irisa.fr (Sven Thiele), gebser@cs-uni.potsdam.de (Martin Gebser), jacques.nicolas@irisa.fr (Jacques Nicolas), saezrodriguez@ebi.ac.uk (Julio Saez-Rodriguez), torsten@cs-uni.potsdam.de (Torsten Schaub), anne.siegel@irisa.fr (Anne Siegel)

biological networks are derived from vast generic knowledge compiled from different cell types. Nevertheless, little is known about the exact chaining and composition of signaling events within these networks in specific cells and specific conditions. For example, in cancer cells, signaling networks frequently become compromised, leading to abnormal behaviors and responses to external stimuli. Many current and emerging cancer treatments are designed to block nodes in signaling networks, thereby altering signaling cascades. Thus, advancing our understanding of how these networks are deregulated across specific environments will ultimately lead to more effective treatment strategies for patients. In this context, phosphorylation assays are a recent form of high-throughput data providing information about protein-activity modifications in a specific cell type upon various perturbations. Towards the construction of predictive models, one can convert the generic prior knowledge (canonical cell signaling networks) into a mathematical model (e.g. a set of differential equations or a set of logic rules) that can be simulated. Next, if enough experimental data is available, the model can be fitted to the data (for example, by determining kinetic constants in a biochemical model) to obtain the most plausible model for a specific cell type. This is normally achieved by defining an objective fitness function to be optimized [4].

Boolean logical networks [36, 61] provide a simple yet powerful qualitative modeling approach which has become very popular during the last decade [51, 43, 63]. In contrast to quantitative methods (which permit fine-grained (kinetic) analysis), qualitative approaches allow for addressing large-scale biological networks. In this context, the manual identification of logic rules underlying the system being studied is often hard, error-prone and time consuming. Further, it has been shown that, if the inherent experimental noise is considered, many different logical networks can be compatible with a set of experimental observations [52]. Thus, automated inference of Boolean logical networks from experimental data would allow for identifying admissible large-scale logic models saving a lot of efforts and without any a priori bias.

Notably, the inference of Boolean networks have been addressed by several authors under different hypotheses and methods as we show in Section 6. Specifically, in this paper we focus on the problem initially described in [52]. Therein, a genetic algorithm implementation was proposed to solve the underlying combinatorial multi-objective optimization problem, and a software was provided, CellNOpt [59]. Nonetheless, stochastic search methods cannot characterize the models precisely: they are intrinsically unable not just to provide a complete set of solutions, but also to guarantee that an optimal solution is found. To overcome this limitation, approaches based on Integer Linear Programming (ILP) [42, 55] and Answer Set Programming (ASP) [62] have been applied, providing a proof of concept that a global optimum can be identified. ASP [5, 21] is a declarative problem solving paradigm, in which a problem is encoded as a logic program such that its answer sets (i.e. stable models) represent solutions to the problem. Moreover, modern ASP tools allow handling complex preferences and multi-objective optimization, guaranteeing the global optimum by reasoning over the complete solution space. In fact, combinatorial optimization in computational biology has been reviewed in [28] from the perspective of mathematical programming pointing out the importance of exact methods in this subject. Further, multi-objective optimization in the context of bioinformatics and computational biology has been recently reviewed in [31] showing its increasing relevance in this field. In this context, ASP offers a unique pairing of declarativeness and performance to address combinatorial multi-objective optimization problems.

Herein we extend our work in [62] as follows. First, we consider numerical datasets instead of only binary. Essentially, this allows us to simulate realistic experimental datasets and test our approach towards real-world data. Also, in order to cope with such numerical datasets, we introduce and compare several discretization schemes. Next, in contrast to [52] and our previous work in [62], we formalize the learning problem as a lexicographic multi-objective optimization. In fact, this is similar to ILP approaches [42, 55]. By doing this, we avoid the usage of additional artifacts in order to transform the problem into a single-objective optimization. Finally, as detailed in the next section, we aim at learning Boolean logic models without feedback-loops. Nevertheless, in this work we allow for prior knowledge networks with feedback-loops, which are often present. Thus, previous methods (including ours) which consider acyclic prior knowledge networks as an input, would require an expert in order to decide where to "cut" the loops in advance. In fact, this is easily integrated as a constraint in ASP reducing manual pre-processing and the risk of missing admissible models. Clearly, we have elaborated upon our previous ASP encoding in order to cope with the mentioned extensions. That is, (1) numerical datasets, (2) lexicographic multi-objective optimization

and (3) feedback-loops in the prior knowledge. We have validated our approach using a real-world prior knowledge network related to signaling events upon stimulation of cellular receptors in hepatocytes and *in silico* generated datasets.

The paper is structured as follows: Section 2 recalls main biological hypotheses underlying the learning of Boolean logic models; Section 3 provides a formal characterization of our problem; Section 4 introduces ASP and shows how to learn Boolean logic models using it; Section 5 presents benchmarks evaluating the performance of our approach; Section 6 reviews related work; and Section 7 concludes.

2. Background

2.1. Logical preliminaries

Given a finite set V of propositional variables, we form propositional formulas from V with the connectives $\bot$, $\top$, $\neg$, $\vee$, and $\wedge$ in the standard way. Further, we consider truth assignments mapping formulas to truth values $\{0, 1\}$ according to classical logic semantics and interprete, 0 as *false* and 1 as *true*.

2.2. Boolean logic models of immediate-early response

In what follows we briefly summarize the main biological hypotheses in [52] providing the foundation for the concept of *Boolean logic models of immediate-early response*. Generally speaking, a Boolean logic model (V, ϕ) can be seen as a Boolean network [36, 61]. That is, it consists of a finite set V of propositional variables describing biological species or compounds and a function ϕ mapping variables $v \in V$ to a propositional formula $\phi(v)$ over V. We say that ϕ is *complete* if and only if $v \in dom(\phi)$ for every $v \in V$. Furthermore, we say that ϕ is *acyclic* if and only if there are no feedback-loops in the Boolean logic model (V, ϕ). Importantly, Boolean logic models of immediate-early response are simpler than other more elaborate settings in Boolean networks, for instance, asynchronous (multivalued) [60] or probabilistic [56]. Such approaches are often used to describe and study complex dynamical properties which is not the goal of our work.

2.2.1. Biological hypotheses

The main assumption under Boolean logic models as treated in [52] is the following. The response of a biological system to external perturbations occurs at several time scales. Thus, one can discriminate between fast and slow events. Under this assumption, at a given time after perturbation, the system reaches a state on which fast events are relevant, but slow events (such as protein degradation) have a relatively insignificant effect. In this context, we say that the system has reached a *pseudo-steady state* describing the early events or immediate-early response. Qualitatively, these states can be computed as logical steady states in the Boolean network (V, ϕ) [38]. That is, truth assignments over V yielding identical values for v and $\phi(v)$ for all $v \in dom(\phi)$. Let us illustrate this with our toy example shown in Fig. 1(b). In this case, the Boolean logic model (V, ϕ) is defined over variables $V = \{a, \ldots, g\}$ and the mapping ϕ

$$\{d \mapsto a; e \mapsto b \vee c; f \mapsto d \wedge e; g \mapsto e \wedge \neg c\}.$$

Furthermore, let A be the following truth assignment over V

$$\{a \mapsto 1, b \mapsto 0, c \mapsto 1, d \mapsto 1, e \mapsto 1, f \mapsto 1, g \mapsto 0\}.$$

One can verify that $A(v) = A(\phi(v))$ for all $v \in dom(\phi)$ where A is extended to formulas in the standard way. Hence, the truth assignment A describes a logical steady state in the Boolean logic model (V, ϕ).

In fact, the discrimination between fast and slow events has an important consequence. Since we focus on fast or early events, it is assumed that oscillation or multi-stability caused by feedback-loops [49, 45] cannot happen until the second phase of signal propagation occurring at a slower time scale. Therefore, feedback-loops are not included in Boolean logic models of immediate-early response assuming that they will become active in a late phase [39]. Notably, it follows that starting from any initial state, a Boolean logic model of immediate-early response reaches a unique steady state in polynomial time [45]. Thus, such modeling approach, although not capable of capturing dynamical properties, provides a relatively simple framework for input-output predictive models.

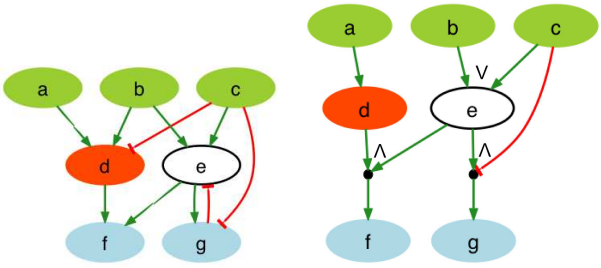


Figure 1: **Directed hypergraph representation of logic models**. The green and red edges correspond to activations and inhibitions, respectively. Green nodes represent ligands that can be experimentally stimulated. Red nodes represent species that can be inhibited by using a drug. Blue nodes represent species that can be measured by using an antibody. White nodes are neither measured, nor manipulated. **(a)** A toy interaction (directed and signed) graph describing causal interactions among proteins. **(b)** An arbitrary Boolean logic model derived from the interaction graph shown in (a) describing functional relationship defined by the mapping $\{d \mapsto a; e \mapsto b \vee c; f \mapsto d \wedge e; g \mapsto e \wedge \neg c\}$.

2.2.2. Boolean logic models as directed hypergraphs

Graph theory is a standard tool used to model biological networks. Nodes in the graph typically describe biological species (genes, proteins, or metabolites) whereas the edges represent causal relations among them. However, functional relationships in biological networks cannot be captured using only graph theory [37]. If two proteins modeled by nodes a and b have a positive effect on a third one d, this would be described in a graph by edges $a \to d$, $b \to d$ like in Fig. 1(a). Nevertheless, it is unclear whether a or b can independently activate d, or if both are required. In order to describe such logical functional relations between species, and to offer a formal representation of cellular networks, directed hypergraphs like the one in Fig. 1(b) can be used.[1] A *directed hypergraph* $H = (V, E)$ is a generalization of a directed graph $G = (V, A)$, where V is the set of nodes and E the set of *directed hyperedges*. While edges in G connect pairs of nodes $a, b \in V$, *directed hyperedges* in H connect pairs of *sets of nodes* $S, T \subseteq V$. Without loss of generality, assuming only formulas in disjunctive normal form provides a straightforward link between Boolean logic models and directed hypergraphs. Since hypergraphs were already described and used to represent Boolean logic models in [38, 52], we adopt the same formalism and we simply give the example in Fig. 1 to introduce this representation. For more details, we refer the reader to the aforecited literature.

2.2.3. Learning Boolean logic models

Based on the assumptions and concepts described above, authors in [52] have proposed a method to learn from a prior knowledge network describing causal interactions (Fig. 1(a)) and phosphorylation activities at a pseudo-steady state, Boolean logic models (Fig. 1(b)) fitting experimental data. In particular, given a network encoding our knowledge of signal transduction and a dataset measuring the activation of proteins (outputs) in this network upon various perturbations (inputs), one can derive from the network Boolean logic models fitting the data. By default, it is assumed a "resting" network (no stimuli present). Then, the value of a variable that has no mapping in ϕ is given by its default truth value, i.e. 0. On the other hand, the value of all other variables is overwritten by signals propagated from the inputs (stimuli and knock-outs) according to their mappings in ϕ. Finally, the identification of the Boolean logic models whose input-output predictions best fit the data is posed as an optimization problem.

3. Learning Boolean logic models

3.1. Problem inputs

Boolean logic models must be learned from three inputs: a prior knowledge network (PKN), a set of experimental conditions, and for each of them, the corresponding experimental observation. A PKN

[1] Directed hypergraphs are sometimes referred to as "AND/OR graphs" or "labelled graphs" [19].

is a signed and directed graph (V, E, σ) with nodes V, directed edges $E \subseteq V \times V$ and signature $\sigma \subseteq E \times \{1, -1\}$. Nodes in V describe biological species whereas the signed and directed edges in E represent causal relationships among them, i.e., activatory or inhibitory effects. Further, we distinguish three special subsets in V namely, the *stimuli* (V_S), the *knock-outs* (V_K) and the *readouts* (V_R). Nodes in V_S denote extracellular signals and thus, we assume they have indegree equal to zero. Nodes in V_K denote species that can be inhibited by various experimental tools such as small-molecule drugs, antibodies, or RNAi. Finally, nodes in V_R denote species that can be measured by using an antibody. Notably, species in none of these sets, are neither measured, nor manipulated for the given experimental settings. Let us denote with V_U the set of such nodes. Then, except for V_R and V_K that may intersect, the sets V_S, V_K, V_R and V_U are pairwise mutually disjoint. Note that the signature σ is defined as a relation and not as a function since it could be the case that both signs are present. This is more likely to happen when the PKN is compressed as described in [52] in order to remove most of the nodes in V_U.

Given a PKN (V, E, σ), the concept of an *experimental condition* over (V, E, σ) is captured by a truth assignment over variables $V_S \cup V_K$. If ε is an experimental condition and $v \in V_S$, then $\varepsilon(v) = 0$ (resp. 1) indicates that the stimulus v is absent (resp. present), while if $v \in V_K$, then $\varepsilon(v) = 0$ (resp. 1) indicates that the species v is inhibited (resp. not inhibited). Furthermore, the concept of an *experimental observation* under ε is captured by a partial mapping $\omega : V_R \mapsto [0, 1]$. That is, $dom(\omega) \subseteq V_R$ denotes the set of observed readouts under the experimental condition ε. If $v \in dom(\omega)$, then $\omega(v)$ represents the phosphorylation activity of the readout v under ε. Since phosphorylation assays represents an average across a population of cells, the phosphorylation activity for each readout is usually normalized to $[0, 1]$. Finally, an *experimental dataset* ξ is a finite set of pairs $(\varepsilon_i, \omega_i)$ with each ω_i defined under ε_i. Further, we denote with N_ξ the *size* of ξ given by the number of observed readouts across all experiments, i.e., $N_\xi = \sum_{i=1}^{n} |dom(\omega_i)|$.

Let us illustrate the problem inputs with our toy example. Consider the PKN (V, E, σ) defined in Fig. 1(a). From the graph coloring, we have $V_S = \{a, b, c\}$, $V_K = \{d\}$ and $V_R = \{f, g\}$. Finally, let $\xi = ((\varepsilon_1, \omega_1), \ldots, (\varepsilon_4, \omega_4))$ be an example experimental dataset over (V, E, σ) defined by

$$\begin{array}{ll} \varepsilon_1 = \{a \mapsto 1, b \mapsto 0, c \mapsto 1, d \mapsto 1\} & \omega_1 = \{f \mapsto 0.9, g \mapsto 0.0\} \\ \varepsilon_2 = \{a \mapsto 1, b \mapsto 0, c \mapsto 1, d \mapsto 0\} & \omega_2 = \{f \mapsto 0.1, g \mapsto 0.9\} \\ \varepsilon_3 = \{a \mapsto 1, b \mapsto 0, c \mapsto 0, d \mapsto 1\} & \omega_3 = \{f \mapsto 0.0, g \mapsto 0.1\} \\ \varepsilon_4 = \{a \mapsto 1, b \mapsto 1, c \mapsto 0, d \mapsto 1\} & \omega_4 = \{f \mapsto 1.0, g \mapsto 0.8\}. \end{array} \tag{1}$$

3.2. Predictive Boolean logic models

We aim at learning Boolean logic models from a PKN and an experimental dataset. In fact, any learned model has to be supported by some *evidence* in the prior knowledge. For example, looking at Fig. 1, it is clear that the logic model in (b) is not just some arbitrary model, but it is strongly related to the PKN in (a). To be more precise, given a PKN (V, E, σ) we consider only Boolean logic models (V, ϕ) without feedback-loops and such that, for each variable $v \in V$, if w occurs positively (resp. negatively) in $\phi(v)$ then, there exists an edge $(w, v) \in E$ and $((w, v), 1) \in \sigma$ (resp. $((w, v), -1) \in \sigma$). Notice that feedback-loops may occur in the PKN, i.e. a cycle in the graph (V, E, σ), but not in the Boolean logic models.

In order to characterize the trajectories and steady states of Boolean logic models, we follow the investigations of [33] and reformulate the *immediate consequence operator* T_P introduced in [3] for a logic program P. Let (V, ϕ) be a Boolean logic model with a complete mapping ϕ over V. Furthermore, let A be a truth assignment over V. First we define $T_{(V,\phi)}$ as

$$T_{(V,\phi)}(A) = \{v \mapsto A(\phi(v)) \mid v \in V\}.$$

where A is extended to formulas in the standard way. Thus, $T_{(V,\phi)}(A)$ is a mapping from variables in V to truth values $\{0, 1\}$, i.e., a truth assignment over V. Next, we define the iterative variant of $T_{(V,\phi)}$ as

$$T^0_{(V,\phi)}(A) = A \quad \text{and} \quad T^{j+1}_{(V,\phi)}(A) = T_{(V,\phi)}\big(T^j_{(V,\phi)}(A)\big).$$

In biological terms, a sequence $(T^j_{(V,\phi)}(A))_{j \geq 0}$ represents the signal propagation starting in state A.

Let (V, E, σ) be a PKN and let ε be an experimental condition over (V, E, σ). In order to capture the perturbations generated by ε, to the resting system described by a Boolean logic model (V, ϕ), we define $\phi|_\varepsilon$ for each $v \in V$ as

$$\phi|_\varepsilon(v) = \begin{cases} \top & \text{if } v \in V_S \text{ and } \varepsilon(v) = 1 \\ \bot & \text{if } (v \in V_S \cup V_K \text{ and } \varepsilon(v) = 0) \text{ or } v \notin dom(\phi) \cup V_S \\ \phi(v) & \text{otherwise} \end{cases}$$

yielding the modified Boolean logic model $(V, \phi|_\varepsilon)$. Importantly, $\phi|_\varepsilon$ is a complete and acyclic mapping over V. Thus, $T_{(V,\phi|_\varepsilon)}$ has a unique fixpoint which can be computed in polynomial time [45] via iterated applications of $T_{(V,\phi|_\varepsilon)}$ starting at any state. Essentially, by propagating the constant truth values for $\top$ and $\bot$.

Finally, given a Boolean logic model (V, ϕ), the concept of model *prediction* ρ under an experimental condition ε is captured by the truth assignment $T^j_{(V,\phi|_\varepsilon)}(A_0)$ such that, A_0 is any truth assignment over V and for some $j \geq 0$, we have that $T^j_{(V,\phi|_\varepsilon)}(A_0) = T^{j+1}_{(V,\phi|_\varepsilon)}(A_0)$. That is, ρ is the unique fixpoint of $T_{(V,\phi|_\varepsilon)}$.

As an example, in the experimental condition ε_2 from (1) we have:

$$\{a \mapsto 1, b \mapsto 0, c \mapsto 1, d \mapsto 0\}.$$

That is, a and c are stimulated while b and d are inhibited. Next, given the Boolean logic model (V, ϕ) in Fig. 1(b), $(V, \phi|_{\varepsilon_2})$ is defined by the mapping

$$\{a \mapsto \top; b \mapsto \bot; c \mapsto \top; d \mapsto \bot; e \mapsto b \vee c; f \mapsto d \wedge e; g \mapsto e \wedge \neg c\}.$$

Furthermore, let $A_0 = \{v \mapsto 0 \mid v \in V\}$ be a truth assigment over V. Then, the prediction ρ_2 for (V, ϕ) under ε_2 can be computed via iterated applications of $T_{(V,\phi|_{\varepsilon_2})}$ until reaching a fixpoint:

$$\begin{aligned}
&T^0_{(V,\phi|_{\varepsilon_2})}(A_0) = A_0 \\
&T^1_{(V,\phi|_{\varepsilon_2})}(A_0) = T_{(V,\phi|_{\varepsilon_2})}(A_0) = \{a \mapsto 1, b \mapsto 0, c \mapsto 1, d \mapsto 0, e \mapsto 0, f \mapsto 0, g \mapsto 0\} = A_1 \\
&T^2_{(V,\phi|_{\varepsilon_2})}(A_0) = T_{(V,\phi|_{\varepsilon_2})}(A_1) = \{a \mapsto 1, b \mapsto 0, c \mapsto 1, d \mapsto 0, e \mapsto 1, f \mapsto 0, g \mapsto 0\} = A_2 \\
&T^3_{(V,\phi|_{\varepsilon_2})}(A_0) = T_{(V,\phi|_{\varepsilon_2})}(A_2) = \{a \mapsto 1, b \mapsto 0, c \mapsto 1, d \mapsto 0, e \mapsto 1, f \mapsto 0, g \mapsto 0\} = A_2.
\end{aligned}$$

When $T^1_{(V,\phi|_{\varepsilon_2})}(A_0)$ is computed giving the truth assigment A_1, variables a, c and b, d are assigned to the constant truth values for $\top$ and $\bot$ respectively, whereas variables e, f and g remain assigned to 0. Next, when $T^2_{(V,\phi|_{\varepsilon_2})}(A_0)$ is computed resulting in A_2, the variable e is assigned to 1 since $A_1(\phi|_{\varepsilon_2}(e)) = A_1(b \vee c) = 1$. To conclude, $T^3_{(V,\phi|_{\varepsilon_2})}(A_0)$ is computed showing that a fixpoint, namely A_2, has been reached. Hence, such a fixpoint defines the prediction ρ_2 for (V, ϕ) under ε_2.

3.3. Learning as optimization

For a given PKN (V, E, σ), there are exponentially many candidate Boolean logic models (V, ϕ) having an evidence on it. Therefore, authors in [52] put forward the idea of training Boolean logic models by comparing their corresponding predictions to experimental observations at a pseudo-steady state. In this context, two natural optimization criteria arise in order to conduct the learning: (1) model accuracy (biologically meaningful), and (2) model complexity (Occam's razor principle). In fact, this is a typical scenario on automatized learning of predictive models [18].

We now provide the precise formulation for each optimization criteria as defined in [52]. Let (V, E, σ) be a PKN. Let $\xi = ((\varepsilon_1, \omega_1), \ldots, (\varepsilon_n, \omega_n))$ be an experimental dataset over (V, E, σ). Let (V, ϕ) be a Boolean logic model having evidence in (V, E, σ) and let $\rho_1, \ldots, \rho_n$ be its Boolean predictions with each ρ_i defined under ε_i. Firstly, based on the residual sum of squares (RSS) we define the *fitness* (Θ_f) of (V, ϕ) with respect to ξ as

$$\Theta_f((V, \phi), \xi) = \sum_{i=1}^{n} \sum_{v \in dom(\omega_i)} (\omega_i(v) - \rho_i(v))^2. \tag{2}$$

Secondly, for a given logical formula $\phi(v)$, let us denote its length by $|\phi(v)|$. Then, we define the *size* (Θ_s) of (V, ϕ) as

$$\Theta_s((V,\phi)) = \sum_{v \in dom(\phi)} |\phi(v)|. \tag{3}$$

A popular and relatively simple approach to cope with multi-objective optimization is to transform it into a single-objective optimization. Towards this end, one usually combines all criteria by defining a function using free parameters in order to assign different weights to each criteria. In fact, this is exactly the approach adopted in [52]. Therein, a single-objective function is defined that balances *fitness* and *size* using a parameter α chosen to maximize the predictive power of the model. Moreover, it has been shown that "predictive power" is best for $\alpha < 0.1$. However, such approach suffers from known drawbacks. First, it depends on "magic values" for each weight often based on intuition or empirically determined. Second, it combines different scales of measurements that need to be normalized. Third, it combines non-commensurable criteria producing meaningless quantities [18]. On the other hand, the lexicographic approach allows us to assign different priorities to different objectives in a qualitative fashion. Notably, in our context logic models providing high predictive power are significantly more relevant than the sizes of such models. Thus, the lexicographic approach is very convenient to cope with the multi-objective nature of our optimization problem. Yet another popular approach is to look for Pareto optimal models. However, this method will lead to a large number of models providing either none or very low predictive power. For example, consider the Boolean logic model (V, ϕ) with $\phi = \emptyset$, i.e. the *empty* model. Such a model is trivially consistent with any input PKN (V, E, σ) while it minimizes the objective function *size*, i.e. $\Theta_s((V, \phi)) = 0$. Therefore, (V, ϕ) is Pareto optimal although it does not provide any valuable information. Similarly, one can show that many other (*non-empty*) models will be Pareto optimal as well although they provide very low predictive power. Hence, Pareto optimality is not well suited for our problem. Notwithstanding, other multi-objective optimization methods (cf. [40]) could be investigated in the future.

Finally, let $\mathbb{M}_{(V,E,\sigma)}$ be the space of Boolean logic models without feedback-loops having evidence in (V, E, σ). Then, our lexicographic multi-objective optimization consists of minimizing first Θ_f, and then with lower priority Θ_s:

$$(V, \phi_{opt}) = \underset{(V,\phi) \in \mathbb{M}_{(V,E,\sigma)}}{\arg\min} \; (\Theta_f((V,\phi),\xi), \Theta_s((V,\phi))). \tag{4}$$

4. Learning Boolean logic models with Answer Set Programming

4.1. Introduction to Answer Set Programming

Answer Set Programming (ASP; [5, 21]) provides a declarative framework for modeling various Knowledge Representation and Reasoning problems. The unique pairing of declarativeness and performance in state-of-the-art ASP solvers allows for concentrating on an actual problem, rather than a smart way of implementing it. The basic idea of ASP is to express a problem in a logical format so that the models of its representation provide the solutions to the original problem. Problems are expressed as logic programs and the resulting models are referred to as answer sets. Although determining whether a program has a answer set is the fundamental decision problem in ASP, more reasoning modes are needed for covering the variety of reasoning problems encountered in applications. Hence, a modern ASP solver, like *clasp* [25] supports several reasoning modes for assessing the multitude of answer sets, among them, regular and projective enumeration, intersection and union, and multi-criteria optimization. As well, these reasoning modes can be combined, for instance, for computing the intersection of all optimal models. This is accomplished in several steps. At first, a logic program with first-order variables is turned by efficient database techniques into a propositional logic program. This is in turn passed to a solver computing the answer sets of the resulting program by using advanced Boolean constraint technology. For optimization, a solver like *clasp* uses usually branch-and-bound algorithms (other choices, like computing unsatisfiable cores, exist). The enumeration of all optimal models, as in our paper, is done in two steps. At first an optimal model is determined along with its optimum value. This computation has itself two distinct phases. First, an optimal model candidate must be found and second, it must be shown that there is no better candidate; the latter amounts to a proof

of unsatisfiability and is often complex. Then, all models possessing the same value are enumerated in a second step.

Our encodings are written in the input language of gringo 3 [22, 25]. In what follows we introduce its basic syntax and we refer the reader to the aforecited literature for more details. An *atom* is a predicate symbol followed by a sequence of terms (e.g. `p(a,b)`,`q(X,f(a,b))`). A *term* is a constant (e.g. `c`, `42`) or a function symbol followed by a sequence of terms (e.g. `f(a,b)`, `g(X,10)`) where uppercase letters denote first-order variables. Then, a rule is of the form

$$H \texttt{:-}\ B_1, \ldots, B_n.$$

where H (head) is an atom and any B_j (body) is a literal of the form A or `not` A for an atom A where the connective `not` corresponds to default negation. Further, a rule without body is a *fact*, whereas a rule without head is an *integrity constraint*. A logic program consists of a set of rules, each of which is terminated by a period. The connectives `:-` and `,` can be read as *if* and *and*, respectively. A statement starting with `not` is satisfied unless its enclosed proposition is found to be true. The semantics of a logic program is given by the *stable models semantics* [27]. Intuitively, the head of a rule has to be true whenever all its body literals are true. In ASP every atom needs some derivation, i.e., an atom cannot be true if there is no rule deriving it. This implies that only atoms appearing in some head can appear in answer sets, i.e. stable models.

We end this quick introduction by three language constructs particularly interesting for our encoding. First, the so called *choice rule* of the form,

$$\{H_1, \ldots, H_m\} \texttt{:-}\ B_1, \ldots, B_n.$$

allows us to express choices over subsets of atoms. Any subset of its head atoms can be included in a stable model, provided the body literals are satisfied. Note that using a choice rule one can easily *generate* an exponential search space of candidate solutions. Second, a conditional literal is of the form

$$L : L_1 : \cdots : L_n$$

The purpose of this language construct is to govern the instantiation of the literal L through the literals $L_1, \ldots, L_n$. In this respect, the conditional literal above can be regarded as the list of elements in the set $\{L \mid L_1, \ldots, L_n\}$. Finally, for solving (multi-criteria) optimization problems, ASP allows for expressing cost functions in terms of a weighted sum of elements subject to minimization and/or maximization. Such objective functions are expressed in ASP in terms of optimization statements of the form

$$\texttt{\#minimize}\{L_1 = W_1@P_1, \ldots, L_N = W_N@P_N\}.$$

where every L_j is a literal and every W_j an integer weight. Further, P_i provides an integer priority level. Priorities allow for representing lexicographically ordered minimization objectives, greater levels being more significant than smaller ones. More complex preferences, for instance, inclusion-based minimization or Pareto efficiency, can be addressed by means of meta-programming as described in [23].

4.2. Data discretization schemes

In order to express and solve the multi-objective optimization described in (4) by using ASP, one needs to discretize the function defined in (2). A very simple approach converts numerical data into binary data according to a threshold. Further, we propose a finer multi-valued discretization scheme. In fact, the only non-integer variables in (2) are the experimental observations $\omega_i(v)$. Then, we introduce a first parametrized discretization scheme of these variables as follows. Such scheme corresponds to the simplest method which converts real numbers into Boolean values, where $\lfloor x \rfloor$ stands for the integer part of a real number x.

$$\forall x \in [0,1]\,,\ \delta_0(x) = \begin{cases} \lfloor 2x \rfloor & 0 \leq x < 1 \\ 1 & x = 1 \end{cases}$$

Alternatively, one may require to have better approximations (up to $\frac{1}{10}, \frac{1}{100}, \ldots, \frac{1}{10^k}$) by using truncations:

$$\delta_1(x) = \frac{\lfloor 10x \rfloor}{10}, \quad \delta_2(x) = \frac{\lfloor 100x \rfloor}{100}, \quad \ldots \quad \delta_k(x) = \frac{\lfloor 10^k x \rfloor}{10^k}.$$

We note that these finer approximations are not direct extensions of the binary approximation δ_0 since they are not continuous for $x = 1$. Although they are not exactly similar, they appear to be the most commonly used in practice. Hence, we choose to study all of them in a common framework. Next, we define the discretized fitness Θ_{f_k} as

$$\Theta_{f_k}((V,\phi),\xi) = \sum_{i=1}^{n} \sum_{v \in dom(\omega_i)} \left[10^k \delta_k(\omega_i(v)) - 10^k \rho_i(v)\right]^2. \tag{5}$$

Since $10^k \delta_k(\omega_i(v))$ are all integer values, we have that $\Theta_{f_k}((V,\phi),\xi)$ contains only integer variables.

The minimizations of Θ_f and Θ_{f_k} may yield different Boolean logic models. Nonetheless, the following proposition guarantees that finding all models minimizing Θ_{f_k} within a certain tolerance allows us to find all models minimizing Θ_f. We refer the reader to the appendix for a detailed proof.

Proposition 4.1. *Let (V,E,σ) be a PKN. Let ξ be an experimental dataset over (V,E,σ) with size N_ξ. Let $k \in \mathbb{N}$ define the discretization scheme. Let us denote with μ and μ_k, the corresponding minima for Θ_f and Θ_{f_k} over the space of models $\mathbb{M}_{(V,E,\sigma)}$ and with respect to ξ:*

$$\mu = \min_{(V,\phi)\in\mathbb{M}_{(V,E,\sigma)}} \Theta_f((V,\phi),\xi) \qquad \mu_k = \min_{(V,\phi)\in\mathbb{M}_{(V,E,\sigma)}} \Theta_{f_k}((V,\phi),\xi).$$

Then $10^{-2k}\mu_k$ converges to μ when k increases, with an exponential speed:

$$\mu_k = 10^{2k}\mu + O(10^k).$$

Moreover, any Boolean logic model minimizing Θ_f, also minimizes Θ_{f_k} within the following tolerance t_k:

$$t_k = 2\sqrt{\frac{N_\xi}{\mu_k}} + \frac{N_\xi}{\mu_k}.$$

Notice that μ_k increases exponentially with k. Furthermore, as we shall see it in Section 5, in practice, μ_k is significantly greater than N_ξ provided that $k \geq 1$. Hence, in practice, the tolerance t_k is relatively small. Importantly, this justifies that the minimization of Θ_f can be successfully addressed by enumerating suboptimal models of Θ_{f_k} using a multi-valued discretization scheme together with the tolerance t_k. We illustrate this with our experiments and we refer the reader to Section 7 for further discussion.

4.3. Input instance

Let (V,E,σ) be a PKN. We represent the nodes in V as facts over the predicate `vertex`/1, namely `vertex(`v`)` for all $v \in V$.[2] Further, facts over the predicate `edge`/3 represent edges in E with their signature, that is, `edge(`v`,`w`,`s`)` for all $(v,w) \in E$ and $((v,w),s) \in \sigma$. Facts over predicates `stimulus`/1, `inhibitor`/1, and `readout`/1 denote nodes in V_S, V_K, and V_R respectively. Let $\xi = ((\varepsilon_1,\omega_1),\ldots,(\varepsilon_n,\omega_n))$ be an experimental dataset over (V,E,σ). Recall that each ε_i is a truth assignment over variables in $V_S \cup V_K$. Then, we represent experimental conditions as facts over the predicate `exp`/3, namely `exp(`i`,`v`,`$\varepsilon_i(v)$`)` for all $v \in V_S \cup V_K$ and $1 \leq i \leq n$. Finally, let k define the discretization scheme as in Section 4.2. We represent discretized experimental observations as facts over the predicate `obs`/3, namely, `obs(`i`,`v`,`$10^k\delta_k(\omega_i(v))$`)` for all $v \in dom(\omega_i)$ and $1 \leq i \leq n$. We use the predicate `dfactor/1` to denote the discretization factor 10^k.

Using the discretization scheme provided by $k = 1$, Listing 1 shows the instance representation for our toy example. That is, the PKN in Fig. 1(a) and the dataset given in (1).

[2] We use `p`/n to indicate that predicate `p` has arity n.

Listing 1: Toy example input instance

```
vertex(a). vertex(b). vertex(c). vertex(d). vertex(e). vertex(f). vertex(g).

edge(a,d,1). edge(b,d,1). edge(b,e,1). edge(c,d,-1). edge(c,e,1).
edge(d,f,1). edge(e,f,1). edge(e,g,1). edge(g,e,-1). edge(c,g,-1).

stimulus(a). stimulus(b). stimulus(c). inhibitor(d). readout(f). readout(g).

exp(1,a,1). exp(1,b,0). exp(1,c,1). exp(1,d,1).
exp(2,a,1). exp(2,b,0). exp(2,c,1). exp(2,d,0).
exp(3,a,1). exp(3,b,0). exp(3,c,0). exp(3,d,1).
exp(4,a,1). exp(4,b,1). exp(4,c,0). exp(4,d,1).

obs(1,f,9). obs(2,f,1). obs(3,f,0). obs(4,f,10).
obs(1,g,0). obs(2,g,9). obs(3,g,1). obs(4,g,8).

dfactor(10).
```

4.4. Logic program

We now describe our encoding for learning Boolean logic models as described in Section 3. Our ASP encoding is shown in Listing 2.

Listing 2: Logic program

```
sub(set(U,S,nil),1,V)             :- edge(U,V,S).
sub(set(U,SU,set(W,SW,T)),N+1,V) :- edge(U,V,SU), sub(set(W,SW,T),N,V), U<W.

in(U,S,set(U,S,T))   :- sub(set(U,S,T),N,V).
in(W,SW,set(U,SU,T)) :- in(W,SW,T), sub(set(U,SU,T),N,V).

{conjunction(C,N,V)} :- sub(C,N,V).

path(U,V) :- conjunction(C,_,V), in(U,_,C).
path(U,V) :- conjunction(C,_,V), in(W,_,C), path(U,W).
:- path(V,V).

:- conjunction(C1,N,V), conjunction(C2,M,V), N<M, in(U,S,C2) : in(U,S,C1).

exp(E) :- exp(E,_,_).
mapped(V) :- conjunction(_,_,V).
fixed(E,V) :- exp(E,V,0).
fixed(E,V) :- exp(E), vertex(V), not mapped(V).

active(E,V) :- exp(E,V,1), stimulus(V).
active(E,V) :- exp(E), conjunction(S,M,V), not fixed(E,V),
               active(E,U) : in(U,1,S), not active(E,U) : in(U,-1,S).

residual(D,V,1,#pow(F-D,2)) :- obs(E,V,D), dfactor(F), D<F.
residual(D,V,0,#pow(D,2))   :- obs(E,V,D), D>0.

#minimize[conjunction(_,N,_)=N@1].
#minimize[active(E,V) : obs(E,V,D) : residual(D,V,1,W)=W@2,
          not active(E,V) : obs(E,V,D) : residual(D,V,0,W)=W@2].

#hide.
#show conjunction/3.
```

Since we are only interested in logical formulas having an evidence in (V, E, σ), we construct all possible conjunctions having such evidence by computing for each $v \in V$ all possible subsets of predecessors of v. We denote such subsets over the predicate `sub`/3 and the function `set`/3. The idea is to start with singleton subsets containing only one predecessor, and to create larger sets by recursively extending singletons until all non empty subsets are constructed. This is done in lines 1 and 2. We exploit the order between the

vertices U and W to avoid different permutations of the same subsets. Then, we define the membership relation between vertices and subsets in lines 4 and 5.

In line 7 we use a choice rule to consider each subset as either present or absent. Since each subset describes a conjunction of literals, we generate over predicate `conjunction`/3, all possible logical formulas in disjunctive normal form. That is, we represent a Boolean logic model (V, ϕ) as a set of facts over the predicate `conjunction`/3, namely `conjunction(`C`,`n`,`v`)` for each conjunct in $\phi(v)$ with length n and such that, C is a term of the form `set`/3 describing the set of literals in the conjunct. Lines 9-12 eliminate candidate answer sets describing logic models with feedback-loops. Paths from U to V are represented over the predicate `path`/2 and derived recursively. Thus, the integrity constraint in line 11 avoids self-reachability in the Boolean logic models. Next, in line 13 we use an integrity constraint to avoid redundant models by checking inclusion between conjunctions. For example, for two literals a and b, we say that $a \vee (a \wedge b)$ is *redundant* since it is logically equivalent to a. This concept was previously introduced in [52] as a way to reduce the search space during learning inspired on Sperner systems [8]. Notably, other logical redundancies could be considered as well. However, a complete treatment of redundancies would lead to the NP-complete problem known as *minimization of Boolean functions* [41].

Lines 15-18 define auxiliary domain predicates describing fixed nodes in each experimental condition, i.e., stimuli, knock-outs, and unmapped nodes. Then, for each experimental condition, we represent truth assignments over nodes in V by the presence or absence of the predicate `active`/2. Further, we exploit the default negation in order to use *false* as the default truth value. Then, lines 20-22 allows us to compute under each experimental condition ε_i the fixpoint of $T_{(V,\phi|_{\varepsilon_i})}$ by the inductive propagation of the truth values for the fixed nodes.

Finally, in lines 24 and 25 we compute the possible differences (square of residuals) between Boolean predictions and the corresponding experimental observations. We denote such differences over the predicate `residual`/4. Next, we describe our lexicographic multi-objective optimization. In line 27 we declare with lower priority (`@1`) the minimization over the size of logic models (Eq. (3)). Meanwhile, in lines 28 and 29 we declare, with higher priority (`@2`), the minimization of the residual sum of squares between the Boolean predictions and experimental observations (Eq. (5)).

The next result shows that our ASP encoding is sound and complete with respect to the multi-objective optimization problem described in Section 3 and according to some k defining the discretization scheme. We refer the reader to the appendix for a detailed proof.

Proposition 4.2. *Let (V, E, σ) be a PKN and let ξ be an experimental dataset over it. Let k define the discretization scheme. Let L be the logic program given in Listing 2 and let $\tau((V, E, \sigma), \xi, k)$ be the instance encoding as described above (e.g. Listing 1). Then, X is an answer set of $L \cup \tau((V, E, \sigma), \xi, k)$ iff X describes a Boolean logic model (V, ϕ_{opt}) such that,*

$$(V, \phi_{opt}) = \underset{(V,\phi)\in\mathbb{M}_{(V,E,\sigma)}}{\arg\min} \; (\Theta_{f_k}((V, \phi), \xi), \Theta_s((V, \phi)))$$

minimizing first Θ_{f_k}, and then with lower priority Θ_s.

4.5. Solving

We use the ASP solver *clasp* [24] which implements advanced Boolean constraint technology together with branch-and-bound algorithms for dealing with multi-objective optimization. In Listing 3 we show the optimal answer set found for the toy instance described in Listing 1.[3] In this case, the optimum answer set is the tenth answer set inspected by the solver (`Answer: 10`). Such answer set describes the Boolean logic model given in Fig. 1(b). Further, the values for the optimization criteria are given ordered by their priorities (`Optimization: 88 7`). That is, 88 for the discretized residual sum of squares (Eq. (5)), and 7 for the model size (Eq. (3)). Next, one could run the solver with option `--opt-all=88,7` in order to enumerate all optimal answer sets.

[3] Using the option `--quiet=1` only the last (optimum) answer set is printed.

Listing 3: Learning logic models for the toy instance

```
$ gringo encoding.lp toy.lp | clasp --quiet=1
clasp version 2.1.3
Reading from stdin
Solving...
Answer: 10
conjunction(set(a,1,nil),1,d) conjunction(set(b,1,nil),1,e) conjunction(set(c,1,nil),1,e)
conjunction(set(c,-1,set(e,1,nil)),2,g) conjunction(set(d,1,set(e,1,nil)),2,f)
Optimization: 88 7
OPTIMUM FOUND

Models       : 1
  Enumerated: 10
  Optimum    : yes
Optimization: 88 7
Time         : 0.001s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s)
CPU Time     : 0.000s
```

5. Benchmarks

5.1. Benchmark generation

In order to study the performance of our ASP-based approach over realistic biological data, we generated *in silico* numerical datasets based on a real and large-scale prior knowledge network (PKN). We use a generic PKN related to signaling events upon stimulation of cellular receptors in hepatocytes. After compressing the network as described in [52], the corresponding graph has 30 nodes and 56 edges. Meanwhile, there are 180 possible hyperedges and thus, 2^{180} hypergraphs describing Boolean logic models can be derived from this PKN.

The method used to generate our benchmarks is illustrated in Fig. 2. We start deriving a (random) Boolean logic model (V, ϕ_{gold}) from the PKN. Then, using this model as our gold standard, a Boolean dataset ξ_0 is generated by computing the model predictions (outputs) for several experimental conditions (inputs). Then, several numerical datasets ξ_i, i.e. values in $[0, 1]$, are generated by adding a Gaussian noise with zero mean and standard deviation from to 0.1 to 0.6. This way, we generated several numerical datasets based on the predictions made by (V, ϕ_{gold}). Finally, experiments were run considering each *in silico* numerical dataset together with the PKN, and using either binary, or multi-valued discretization.[4]

5.2. Experiments

Overall performance. We show the overall performance of the experiments run in Table 1. Datasets ξ_0 to ξ_3 lead to the same optimal fitness to data $(\Theta_f N_{\xi_i}^{-1})$, optimal model size (Θ_s) and number of optimal models $(\#_{opt})$, using similar CPU times. However, for datasets ξ_3 to ξ_6, CPU times grow clearly faster when using binary discretization than multi-valued. Moreover, learning on datasets ξ_4 and ξ_5 leads to more fitted Boolean logic models when using multi-valued discretization. Therefore, we conclude that multi-valued discretization provides a better overall performance than binary discretization for both, finding an optimal model and enumerating all of them.

Completeness. Interestingly, for dataset ξ_6 despite of the difference in CPU times, the fitness to data $(\Theta_f N_{\xi_i}^{-1})$ and model size (Θ_s) are the same regardless of the discretization scheme. Nonetheless, using binary discretization we found 4 optimal models whereas only 2 of them are found using multi-valued discretization. This opens the way to two discussions. First, we observe that some optimal solutions are lost between the binary and the multi-valued discretization. This issue is related to the fact that we defined the multi-valued discretization scheme in terms of the floor function. Hence, while in the binary scheme

[4]Experiments were performed on a MacBook Pro, Intel Core i7, 2.7 GHz and 4 GB of RAM using the ASP grounder *gringo 3.0.3* and ASP solver *clasp 2.1.3*

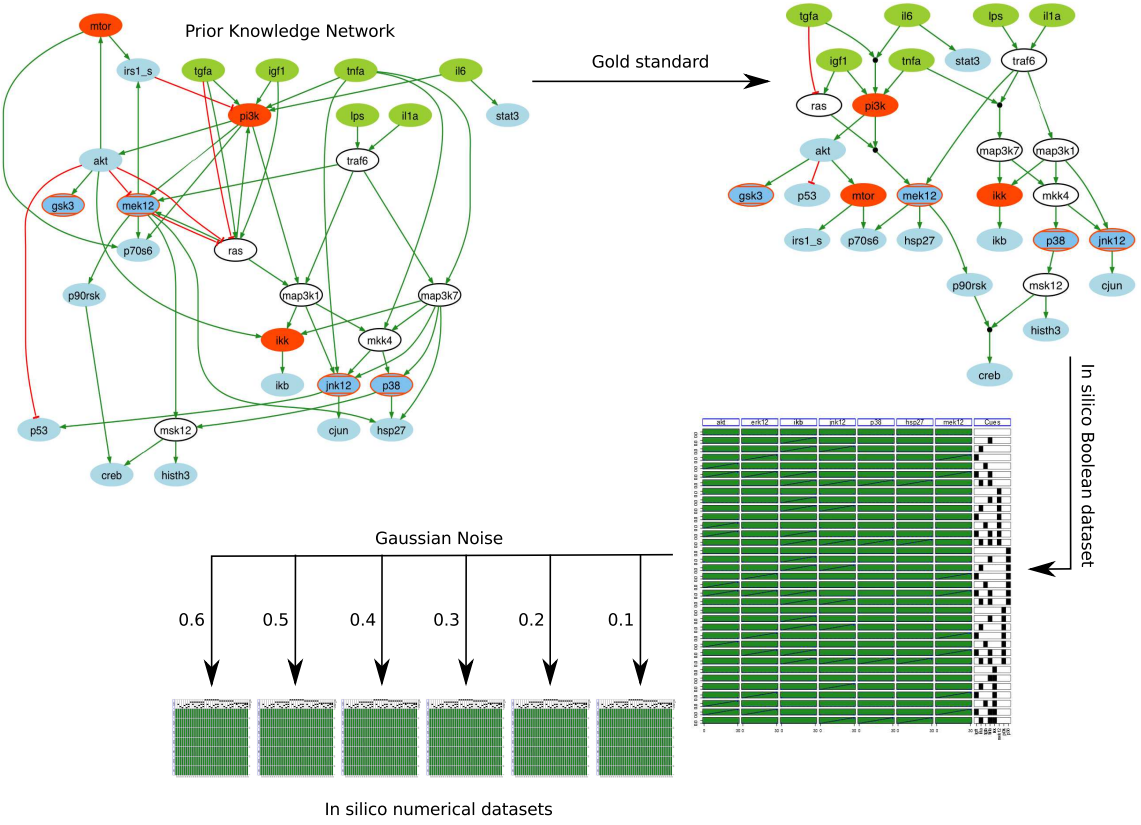


Figure 2: **Generation of in silico numerical datasets:** First, a Boolean logic model (V, ϕ_{gold}) is derived from the prior knowledge network (PKN). Then, using this model, a Boolean dataset ξ_0 is generated by computing the outputs (Boolean predictions) for several combinations of inputs. Next, several numerical datasets ξ_i, i.e. values in $[0, 1]$, are generated by adding a Gaussian noise with zero mean and standard deviation from to 0.1 to 0.6.

values at both sides of the threshold (i.e. 0.5) are equally discretized to either 0 or 1, in the multi-valued scheme lower values are preferred and may result in a change of the set of optimal models. This shows that the discretization scheme has a strong impact over the solutions of the optimization problem. Second, this example confirms that the discretized optimization problem and the real one may not have the same set of solutions. Thus, in order to overcome this issue, we rely on our theoretical result in Proposition 4.1. Therein, we prove that the minimization of Θ_f can be successfully addressed by enumerating suboptimal models of Θ_{f_k}. Towards this end, we allow the enumeration of suboptimal logic models by considering a tolerance over model fitness or model size. To be more precise, let $T_f, T_s \in \mathbb{N}$ be the tolerances under consideration over model fitness and size, respectively. Then, for each ξ_i and some optimal model (V, ϕ_{opt_i}) with respect to ξ_i, we enumerate suboptimal Boolean logic models (V, ϕ) such that,

$$\Theta_{f_k}((V, \phi), \xi_i) \leq \Theta_{f_k}((V, \phi_{opt_i}), \xi_i) + T_f \qquad \Theta_s((V, \phi)) \leq \Theta_s((V, \phi_{opt_i})) + T_s.$$

For example, for dataset ξ_6, according to Proposition 4.1 and using $k = 3$ (i.e. 1000-valued) one should consider up to 0.5% of tolerance over the optimum. By doing this, in a few seconds we are able to enumerate 22181 Boolean models which can be easily inspected towards the identification of the optimal ones with respect to Θ_f. This study confirmed that the 4 optimal models for binary discretization (without tolerance) are the only one minimizing Θ_f. Notably, for $k \leq 2$, the tolerance to consider is at least 5% of the optimum. Unfortunately, over ξ_6, such tolerance yields millions of suboptimal models which is useless in practice. For further discussion, we refer the reader to Section 7. In what follows we report experiments corresponding to enumerate suboptimal models according to the discretization scheme given by $k = 2$.

Table 1: **Computing all optimal Boolean logic models.** Datasets ξ_0 to ξ_6 correspond to the *in silico* datasets generated as illustrated in Fig 2. For each discretization scheme we report the CPU time (seconds) to find 1 optimal model (t_{opt}), the CPU time to find all optimal models (t_{all}), the number of optimal models ($\#_{opt}$), the normalized optimum RSS, i.e., mean squared error, from the real ($\Theta_f N_{\xi_i}^{-1}$) and discrete fitness ($\Theta_{f_k} 10^{-2k} N_{\xi_i}^{-1}$), and the optimum size (Θ_s).

ξ_i	Discretization																	
	Binary						10-valued						100-valued					
	t_{opt}	t_{all}	$\#_{opt}$	$\frac{\Theta_f}{N_{\xi_i}}$	$\frac{\Theta_{f_k}}{10^{2k} N_{\xi_i}}$	Θ_s	t_{opt}	t_{all}	$\#_{opt}$	$\frac{\Theta_f}{N_{\xi_i}}$	$\frac{\Theta_{f_k}}{10^{2k} N_{\xi_i}}$	Θ_s	t_{opt}	t_{all}	$\#_{opt}$	$\frac{\Theta_f}{N_{\xi_i}}$	$\frac{\Theta_{f_k}}{10^{2k} N_{\xi_i}}$	Θ_s
ξ_0	0.67	0.08	2	0	0	26	0.16	0.07	2	0	0	26	0.13	0.08	2	0	0	26
ξ_1	0.67	0.07	2	0.0048	0	26	0.23	0.07	2	0.0048	0.0046	26	0.14	0.07	2	0.0048	0.0047	26
ξ_2	0.38	0.14	2	0.0221	0.0094	26	0.11	0.07	2	0.0221	0.0200	26	0.19	0.07	2	0.0221	0.0217	26
ξ_3	1.85	0.29	2	0.0458	0.0458	26	0.38	0.18	2	0.0458	0.0422	26	0.7	0.15	2	0.0458	0.0452	26
ξ_4	32.83	6.0	4	0.0838	0.1187	32	7.47	0.67	4	0.0833	0.0771	29	4.71	0.7	4	0.0833	0.0825	29
ξ_5	256.98	38.64	4	0.1174	0.1625	30	17.42	3.47	4	0.1164	0.1108	32	30.14	3.32	4	0.1164	0.1158	32
ξ_6	179.64	73.07	4	0.1514	0.2000	29	97.26	9.32	2	0.1514	0.1473	29	56.98	9.2	2	0.1514	0.1509	29

Tolerance over model fitness. First, we evaluate the impact of several tolerances only over model fitness. Towards this end, we have fixed $T_s = 0$ and considered tolerances T_f corresponding to $2\%, 4\%, 6\%$ and 8% of the optimum fitness with respect to ξ_3. For the sake of illustration, we take tolerances always relative to ξ_3 since it provides an intermediate fitness to data. Results are shown in Fig. 3. Firstly, for datasets ξ_0 and ξ_1 we found the same behavior with small impact of the tolerances under consideration. Secondly, datasets ξ_2 and ξ_3 have shown not the same, but very similar behavior with respect to the number of models found for each tolerance. Thirdly, datasets ξ_4 to ξ_6 have also shown similar results for each tolerance. Notably, for datasets ξ_1, ξ_2 and ξ_3 the number of suboptimal models grows clearly slower than for datasets ξ_4, ξ_5 and ξ_6. It is worth noting that for the latter cases, even if the considered tolerance correspond to a relatively small ratio of the optimum (i.e. 1% to 4%), the number of suboptimal models reaches quantities which are useless in practice. Therefore, this promotes the use of a more refined discretization scheme for such datasets.

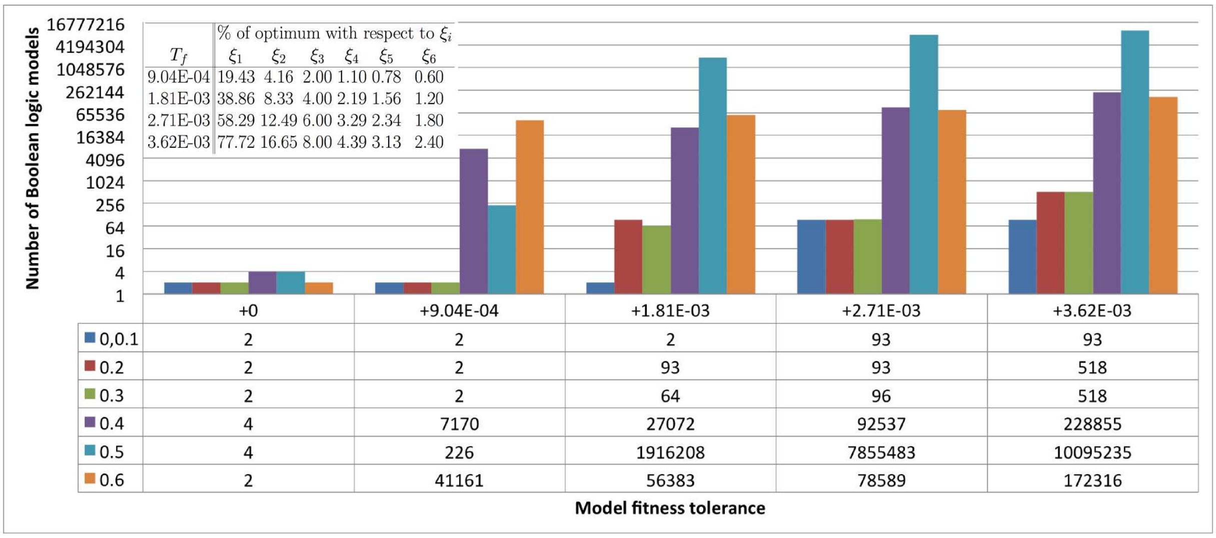
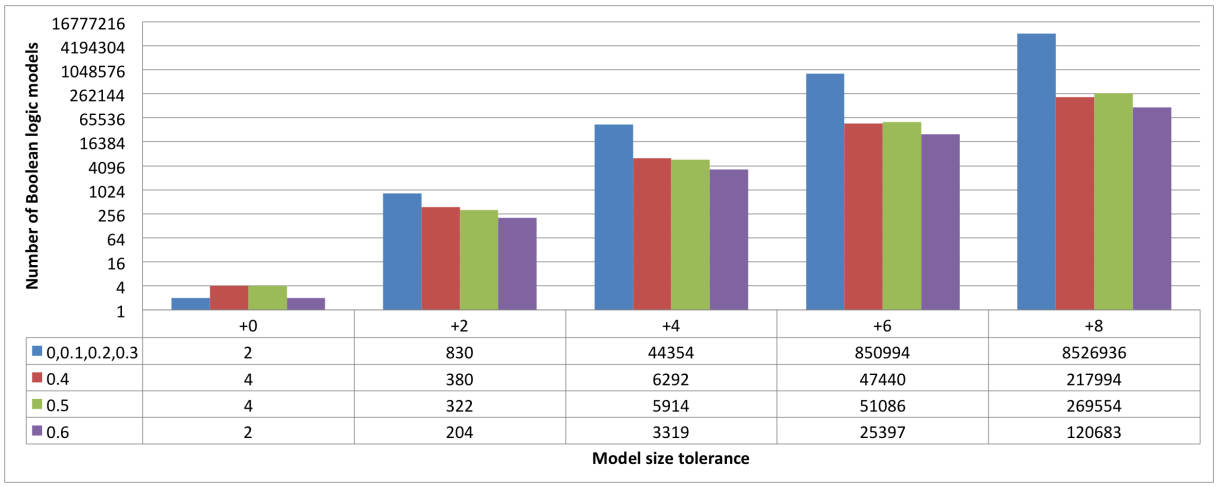


Figure 3: **Number of Boolean logic models with respect to model fitness tolerances.** Each bar describes, for each dataset, the number of models having Θ_{f_2} lower or equal than the minimum plus a tolerance, and model size Θ_s lower or equal than the optimum size (reported in Table 1). Considered tolerances T_f correspond to 0.02, 0.04, 0.06 and 0.08 times the optimum with respect to ξ_3. The corresponding percentage with respect to each dataset ξ_i is reported in the table at the top left corner of the figure. Datasets ξ_0 and ξ_1 have shown the same behavior (blue bar). For datasets with low levels of noise the number of suboptimal models grows clearly slower than for datasets with high level of noise.



	+0	+2	+4	+6	+8
0,0.1,0.2,0.3	2	830	44354	850994	8526936
0.4	4	380	6292	47440	217994
0.5	4	322	5914	51086	269554
0.6	2	204	3319	25397	120683

Figure 4: **Number of Boolean logic models with respect to model size tolerances.** Each bar describes, for each dataset, the number of models having optimal Θ_{f_2} and model size Θ_s lower or equal than the minimum plus t with $t \in \{2, 4, 6, 8\}$. Datasets ξ_0, ξ_1, ξ_2 and ξ_3 have shown the same behavior (blue bar). For all datasets, the number of Boolean logic models grows exponentially.

Tolerance over model size. Next, we evaluate the impact of several tolerances only over model size. We have fixed $T_f = 0$ and considered tolerances $T_s = 2, 4, 6, 8$. Results are shown in Fig. 4. Datasets ξ_0 to ξ_3 have shown exactly the same behavior for each tolerance value. Further, for all datasets the number of Boolean logic models grows exponentially as we increase the tolerance over model size. Notably, minimization over model size is based on Occam's razor principle (parsimony). Hence, there is no experimental evidence in order to select among this large number of models. On one hand, one can consider that larger logic models overfit the available dataset by introducing excessive complexity [52, 46]. On the other hand, one can argue that it is actually necessary to consider such "spurious" links in order to capture cellular robustness and complexity [58].

6. Related work

We follow the investigations of Saez-Rodriguez et al. in [52], which we first revisited in [62] and now extended herein. Some variants of our problem were addressed by Mitsos et al. in [42], and by Sharan and Karp in [55], both using Integer Linear Programming (ILP).

More generally, the inference of Boolean networks has been addressed by several authors under different hypotheses and methods. Liang et al. have used information theoretic principles of mutual information to infer Boolean networks from gene expression data [50]. Similarly, Ideker et al. have studied the problem of identifiability and experimental design under the Boolean networks framework [32]. Akutsu et al. have studied several problems consisting of completing a given Boolean network so that the input-output behavior is consistent with given examples [2]. In one of such problems, namely the *Consistency Problem*, they look for a Boolean network consistent with all Boolean examples. An interesting generalization is the so-called *Best-Fit Extension Problem* described by Shmulevich et al. [57]. Therein, Boolean networks with weighted inconsistencies are allowed and an error function is defined subject to minimization. Recently, an evaluation of some of these methods has been published in [7]. Closer to experimental design, Akutsu et al. also have studied the problem of identifying a genetic Boolean network from experimental data in regard to the number and the complexity of experiments needed [1]. Overall, our work presents some significant differences with the aforecited literature. To start with, all of them are focused on gene regulatory networks and gene expression time-series data, whereas we work on signaling transduction networks and phosphorylation activities at a pseudo-steady state. Further, they work only with Boolean experimental observations which would correspond to adopt the binary discretization scheme in our framework. Moreover, except for the

Best-Fit Extension Problem, they look for Boolean networks fully consistent with the time-series Boolean data. Meanwhile, herein we consider an objective function which describes the goodness of the model based on the numerical data that is subsequently optimized. Finally, all these contributions focus on a "local" inference in the following sense. They aim at learning the Boolean function for each node based on (local) input-output behaviors for such node. On the other hand, we aim at a "global" learning given the prior knowledge network and (global) behaviors over the input-output layers in the network.

Our work contributes to a growing list of ASP applications in systems biology. Almost a decade ago, Baral et al. have proposed applying knowledge representation and reasoning methodologies to the problem of representing and reasoning about signaling networks [6]. More recently, several authors have addressed the question of pruning or identification of biological networks using ASP. Durzinsky et al. have studied the problem consisting of reconstructing all possible networks consistent with experimental time series data [14]. Gebser et al. have addressed the problem consisting of detecting inconsistencies and repairing in large biological networks [26, 20]. Fayruzov et al. have used ASP to represent the dynamics in Boolean networks and find their attractors [15, 16]. Ray et al. have integrated numerical and logical information in order to find the most likely states of a biological system under various constraints [47]. Further, Ray et al. have used an ASP system to propose revisions to metabolic networks [48]. Papatheodorou et al. have used ASP to integrate RNA expression with signaling pathway information and infer how mutations affect ageing [44]. Kaminski et al. have addressed the problem consisting of finding minimal intervention strategies in logical signaling networks [34]. Finally, Schaub and Thiele have first investigated the metabolic network expansion problem with ASP [54] and recently, their work has been extendend and applied in a real-case study by Collet et al. [11]. Altogether, this series of contributions illustrates the potential of ASP to address combinatorial and multi-objective optimization problems appearing in the field. In particular, our work emphasizes the power of ASP for performing an exhaustive enumeration of feasible solutions within certain tolerance with respect to an optimum. More broadly, among other formal approaches applied to the inference of biological models, we find related to ours, Calzone et al. using temporal logic [9], Folschette et al. relying on the process hitting framework [17], and Corblin et al. based on constraint programming [12, 13]. Moreover, some of them have also adopted ASP among their methodologies [17, 13]. Nonetheless, they are mainly focused on the characterization of dynamical properties emerging from available models for a given biological system. Hence, combining the enumeration capabilities of ASP to find feasible models with the characterization of dynamical properties common to all models, poses an interesting challenge for future work.

7. Conclusion

Boolean networks provide a simple yet powerful qualitative modeling approach in systems biology. In this context, we have formalized the problem consisting of learning from a prior knowledge network (PKN) describing causal interactions and phosphorylation activities at a pseudo-steady state, Boolean logic models of immediate-early response in signaling transduction networks. Previous work addressing this problem consists of dedicated genetic algorithms [52] and mathematical programming approaches [42, 55]. Further, in a recent work [62] we have proposed to use Answer Set Programming (ASP) considering a simpler problem setting. Nonetheless, we have shown the shortcomings of genetic algorithms: they are intrinsically unable not only to provide a complete set of solutions, but also to guarantee that an optimal solution is found. Meanwhile, modern ASP tools allow handling complex preferences and multi-objective optimization, guaranteeing the global optimum by reasoning over the complete solution space. In this context, ASP offers a unique pairing of declarativeness and performance to address combinatorial multi-objective optimization problems like the one at hand.

Herein, we have extended our previous work in order to consider: (1) numerical datasets, (2) a multi-objective optimization formulation and (3) feedback-loops in the PKN. Notably, in order to cope with such extensions, we have proposed several discretization schemes and elaborated upon our previous ASP encoding. In order to study the performance of our ASP-based approach towards real-world biological data, we have generated *in silico* numerical datasets based on a real and large-scale PKN. Experiments performed have shown that the discretization scheme has a stronger impact on the computation times than on the fitness

of optimal models to data. In general, multi-valued discretization has shown to be one order of magnitude faster than binary discretization. However, some optimal models may be lost due to the discretization scheme. Hence, in order to overcome this issue, we have proven that the continuous optimization can be successfully addressed by enumerating suboptimal models of the discretized optimization within a certain (small) tolerance. Further, we have shown that the number of optimal models is rather small (tens), but if one considers tolerances over fitness or size, this number can increase significantly (thousands or millions). Notably, being able to perform such exhaustive enumeration is a key feature of our approach. Nonetheless, either real or more elaborated *in silico* (e.g. based on ordinary differential equations) datasets would be required for a further and biologically significant analysis. Moreover, when using real-world datasets, mechanistic descriptions available in popular databases [53, 35, 10] might be used (if they exists) to validate the inferred logical interactions. Although, the method shown herein is highly dependent on the specificity of experiments and thus, a proper biological validation would imply reproducing the experiments on the same or very similar settings.

Altogether, our experiments have shown that current ASP tools are mature enough to cope with real-world problem instances. Nevertheless, given the ability to enumerate such a large number of Boolean models, the way to select among them arises in order to provide new insights to biologists. In fact, rather than select among models one can consider all of them regardless of their different topologies. That is, to take into account only their input-output behaviors. Although not shown in this work, we have found that the variability of input-output behaviors is significantly lower than models topologies. Recent advances in this direction and considering real-world experimental data, can be found in [30]. Briefly, in the aforecited work it is shown that if the experimental error is considered, several thousands of Boolean logic models fit the available data similarly well. Nonetheless, such a large number of models can be grouped into less than a hundred input-output behaviors. Next, these behaviors have been characterized in terms of the number of Boolean logic models they gather and their fitness to data. Moreover, it was found that for 30% of the space of possible inputs, all behaviors agree on the given outputs. Hence, in practice this approach may provide a way to extract robust insights despite the high variability.

Several interesting issues could be investigated in the future. Firstly, it would be interesting to consider other discretization schemes based on the nearest integer or ceiling functions. Also, rather than truncation using discrete levels as a power of 10, one could consider discrete levels as a power of 2 or any other base. Notably, each discretization scheme may have an impact on both, performance and fitness to data. Moreover, considering real-world datasets and their inherent noise, opens very interesting questions. In particular, one could try to identify what is the most significant information in a given dataset, based on what it is actually used by each scheme and the models they recovered. Secondly, the computation of input-output behaviors can be done in a post-processing step but also could be considered as a new search problem by itself. Such a problem would consist of finding all input-output behaviors within a given tolerance over fitness or size. Clearly, without performing the exhaustive enumeration of Boolean logic models. Thirdly, questions related to experimental design under the Boolean networks framework [1, 32] can be investigated with ASP. Assuming that input-output behaviors within certain tolerance describe the data equally (or similarly) well, the model is said to be non-identifiable. Thus, one needs to perform further experiments towards lower variability. Since experiments are usually expensive and time-consuming, one would like to know which experiments are more likely to bring new insights to the optimization process. Thus, a proper experimental design enables a maximum informative analysis of the experimental data. Finally, it would be interesting to perform a detailed comparison between ASP and ILP approaches in order to elucidate their strengths and complementary features on the learning of Boolean logic models as we have described herein.

Acknowledgements

The work of the first author was supported by the project ANR-10-BLANC-0218. We thank the financial aid from the EU through project "BioPreDyn" (ECFP7-KBBE-2011-5 Grant number 289434). This work was partially funded by DFG grant SCHA 550/10-1.

References

[1] Akutsu, T., Kuhara, S., Maruyama, O., Miyano, S., Mar. 2003. Identification of genetic networks by strategic gene disruptions and gene overexpressions under a boolean model. Theoretical Computer Science 298 (1), 235–251.

[2] Akutsu, T., Tamura, T., Horimoto, K., 2009. Completing Networks Using Observed Data. In: Algorithmic Learning Theory. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 126–140.

[3] Apt, K. R., van Emden, M. H., Jul. 1982. Contributions to the theory of logic programming. J. ACM 29 (3), 841–862.

[4] Banga, J., 2008. Optimization in computational systems biology. BMC systems biology 2 (1), 47.

[5] Baral, C., 2003. Knowledge Representation, Reasoning and Declarative Problem Solving. Cambridge University Press.

[6] Baral, C., Chancellor, K., Tran, N., Tran, N., Joy, A., Berens, M., 2004. A knowledge based approach for representing and reasoning about signaling networks. In: Proceedings of the Twelfth International Conference on Intelligent Systems for Molecular Biology/Third European Conference on Computational Biology (ISMB'04/ECCB'04). pp. 15–22.

[7] Berestovsky, N., Nakhleh, L., 2013. An Evaluation of Methods for Inferring Boolean Networks from Time-Series Data. PLoS ONE 8 (6), e66031.

[8] Bollobás, B., 1986. Combinatorics: Set Systems, Hypergraphs, Families of Vectors, and Combinatorial Probability. Cambridge University Press.

[9] Calzone, L., Chabrier-Rivier, N., Fages, F., Soliman, S., 2006. Machine learning biochemical networks from temporal logic properties. In: Priami, C., Plotkin, G. (Eds.), Transactions on Computational Systems Biology VI. Vol. 4220 of Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 68–94.

[10] Cerami, E. G., Gross, B. E., Demir, E., Rodchenkov, I., Babur, O., Anwar, N., Schultz, N., Bader, G. D., Sander, C., 2011. Pathway Commons, a web resource for biological pathway data. Nucleic Acids Research 39 (Database issue), D685–D690.

[11] Collet, G., Eveillard, D., Gebser, M., Prigent, S., Schaub, T., Siegel, A., Thiele, S., 2013. Extending the metabolic network of Ectocarpus siliculosus using answer set programming. In: Cabalar, P., Son, T. (Eds.), Proceedings of the Twelfth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'13). Vol. 8148 of Lecture Notes in Artificial Intelligence. Springer-Verlag, pp. 245–256.

[12] Corblin, F., Fanchon, E., Trilling, L., 2010. Applications of a formal approach to decipher discrete genetic networks. BMC Bioinformatics 11 (1), 385.

[13] Corblin, F., Fanchon, E., Trilling, L., Chaouiya, C., Thieffry, D., 2012. Automatic inference of regulatory and dynamical properties from incomplete gene interaction and expression data. In: Lones, M., Smith, S., Teichmann, S., Naef, F., Walker, J., Trefzer, M. (Eds.), Information Processign in Cells and Tissues. Vol. 7223 of Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 25–30.

[14] Durzinsky, M., Marwan, W., Ostrowski, M., Schaub, T., Wagler, A., 2011. Automatic network reconstruction using ASP. Theory and Practice of Logic Programming 11, 749–766.

[15] Fayruzov, T., De Cock, M., Cornelis, C., Vermeir, D., Nov. 2009. Modeling Protein Interaction Networks with Answer Set Programming. In: Conference on Bioinformatics and Biomedicine, 2009. BIBM '09. IEEE International. pp. 99–104.

[16] Fayruzov, T., Janssen, J., Vermeir, D., Cornelis, C., Cock, M. D., 2011. Modelling gene and protein regulatory networks with answer set programming. International Journal of Data Mining and Bioinformatics 5 (2), 209–229.

[17] Folschette, M., Paulevé, L., Inoue, K., Magnin, M., Roux, O., 2012. Concretizing the process hitting into biological regulatory networks. In: Gilbert, D., Heiner, M. (Eds.), Computational Methods in Systems Biology. LNCS. Springer Berlin Heidelberg, pp. 166–186.

[18] Freitas, A. A., Dec. 2004. A critical review of multi-objective optimization in data mining. ACM SIGKDD Explorations Newsletter 6 (2), 77.

[19] Gallo, G., Longo, G., Pallottino, S., Nguyen, S., 1993. Directed Hypergraphs and Applications. Discrete Applied Mathematics 42 (2-3), 177–201.

[20] Gebser, M., Guziolowski, C., Ivanchev, M., Schaub, T., Siegel, A., Thiele, S., Veber, P., 2010. Repair and prediction (under inconsistency) in large biological networks with answer set programming. In: Lin, F., Sattler, U. (Eds.), Proceedings of the Twelfth International Conference on Principles of Knowledge Representation and Reasoning (KR'10). AAAI Press, pp. 497–507.

[21] Gebser, M., Kaminski, R., Kaufmann, B., Schaub, T., 2012. Answer Set Solving in Practice. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan and Claypool Publishers.

[22] Gebser, M., Kaminski, R., Ostrowski, M., Schaub, T., Thiele, S., 2009. On the input language of asp grounder gringo. In: Erdem, E., Lin, F., Schaub, T. (Eds.), Proceedings of the Tenth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'09). Vol. 5753 of Lecture Notes in Artificial Intelligence. Springer-Verlag, pp. 502–508.

[23] Gebser, M., Kaminski, R., Schaub, T., 2011. Complex optimization in answer set programming. Theory and Practice of Logic Programming 11 (4-5), 821–839.

[24] Gebser, M., Kaufmann, B., Neumann, A., Schaub, T., 2007. clasp: A conflict-driven answer set solver. In: Baral, C., Brewka, G., Schlipf, J. (Eds.), Proceedings of the Ninth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'07). Vol. 4483 of Lecture Notes in Artificial Intelligence. Springer-Verlag, pp. 260–265.

[25] Gebser, M., Kaufmann, B., Schaub, T., 2012. Multi-threaded ASP solving with clasp. Theory and Practice of Logic Programming 12 (4-5), 525–545.

[26] Gebser, M., Schaub, T., Thiele, S., Veber, P., 2011. Detecting inconsistencies in large biological networks with answer set programming. Theory and Practice of Logic Programming 11 (2-3), 323–360.

[27] Gelfond, M., Lifschitz, V., 1988. The stable model semantics for logic programming. In: Kowalski, R., Bowen, K. (Eds.), Proceedings of the Fifth International Conference and Symposium of Logic Programming (ICLP'88). MIT Press, pp. 1070–1080.

[28] Greenberg, H. J., Hart, W. E., Lancia, G., Jun. 2004. Opportunities for Combinatorial Optimization in Computational Biology. INFORMS Journal on Computing 16 (3), 211–231.
[29] Guziolowski, C., Kittas, A., Dittmann, F., Grabe, N., 2012. Automatic generation of causal networks linking growth factor stimuli to functional cell state changes. FEBS Journal 279 (18), 3462–3474.
[30] Guziolowski, C., Videla, S., Eduati, F., Thiele, S., Cokelaer, T., Siegel, A., Saez-Rodriguez, J., 2013. Exhaustively characterizing feasible logic models of a signaling network using answer set programming. Bioinformatics, ahead of print, doi 10.1093/bioinformatics/btt393.
[31] Handl, J., Kell, D. B., Knowles, J., Apr. 2007. Multiobjective optimization in bioinformatics and computational biology. IEEE/ACM Trans. Comput. Biol. Bioinformatics 4 (2), 279–292.
[32] Ideker, T. E., Thorsson, V., Karp, R. M., 2000. Discovery of regulatory interactions through perturbation: inference and experimental design. Pacific Symposium on Biocomputing 5, 305–316.
[33] Inoue, K., 2011. Logic programming for boolean networks. In: Walsh, T. (Ed.), Proceedings of the Twenty-second International Joint Conference on Artificial Intelligence (IJCAI'11). IJCAI/AAAI, pp. 924–930.
[34] Kaminski, R., Schaub, T., Siegel, A., Videla, S., 2013. Minimal intervention strategies in logical signaling networks with answer set programming. Theory and Practice of Logic Programming 13 (4-5), 675–690.
[35] Kanehisa, M., Goto, S., Furumichi, M., Tanabe, M., Hirakawa, M., Jan. 2010. KEGG for representation and analysis of molecular networks involving diseases and drugs. Nucleic acids research 38 (Database issue), D355–60.
[36] Kauffman, S., Feb. 1969. Metabolic stability and epigenesis in randomly constructed genetic nets. Journal of Theoretical Biology 22 (3), 437–467.
[37] Klamt, S., Haus, U.-U., Theis, F. J., May 2009. Hypergraphs and Cellular Networks. PLoS Computational Biology 5 (5), e1000385.
[38] Klamt, S., Saez-Rodriguez, J., Lindquist, J., Simeoni, L., Gilles, E., 2006. A methodology for the structural and functional analysis of signaling and regulatory networks. BMC Bioinformatics 7 (1), 56.
[39] Macnamara, A., Terfve, C., Henriques, D., Bernabé, B. P., Saez-Rodriguez, J., Aug. 2012. State-time spectrum of signal transduction logic models. Physical biology 9 (4), 045003.
[40] Marler, R. T., S.Arora, J., Apr. 2004. Survey of multi-objective optimization methods for engineering. Structural and Multidisciplinary Optimization 26 (6), 369–395.
[41] McCluskey, Jr, E. J., 1956. Minimization of Boolean functions. Bell System Technical Journal.
[42] Mitsos, A., Melas, I., Siminelakis, P., Chairakaki, A., Saez-Rodriguez, J., Alexopoulos, L. G., Sep. 2009. Identifying Drug Effects via Pathway Alterations using an Integer Linear Programming Optimization Formulation on Phosphoproteomic Data. PLoS Computational Biology 5 (12), e1000591.
[43] Morris, M., Saez-Rodriguez, J., Sorger, P., Lauffenburger, D. A., 2010. Logic-based models for the analysis of cell signaling networks. Biochemistry 49 (15), 3216–3224.
[44] Papatheodorou, I., Ziehm, M., Wieser, D., Alic, N., Partridge, L., Thornton, J. M., Dec. 2012. Using Answer Set Programming to Integrate RNA Expression with Signalling Pathway Information to Infer How Mutations Affect Ageing. PLoS ONE 7 (12), e50881.
[45] Paulevé, L., Richard, A., Jun. 2012. Static Analysis of Boolean Networks Based on Interaction Graphs: A Survey. Electronic Notes in Theoretical Computer Science 284, 93–104.
[46] Prill, R. J., Saez-Rodriguez, J., Alexopoulos, L. G., Sorger, P. K., Stolovitzky, G., Sep 2011. Crowdsourcing network inference: the DREAM predictive signaling network challenge. Sci Signal 4 (189), mr7.
[47] Ray, O., Soh, T., 2012. Analyzing Pathways Using ASP-Based Approaches. Algebraic and Numeric Biology.
[48] Ray, O., Whelan, K., King, R., 2010. Logic-Based Steady-State Analysis and Revision of Metabolic Networks with Inhibition. Proceedings of the 2010 International Conference on Complex, Intelligent and Software Intensive Systems (CISIS '10) 0, 661–666.
[49] Remy, E., Ruet, P., Thieffry, D., 2008. Graphic requirements for multistability and attractive cycles in a Boolean dynamical framework. Advances in Applied Mathematics 41 (3), 335–350.
[50] S. Liang, S. F., Somogyi, R., 1998. REVEAL, A General Reverse Engineering Algorithm for Inference of GeneticNetwork Architectures. Pacific Symposium on Biocomputing 3, 19–29.
[51] Saadatpour, A., Albert, R., Nov. 2012. Boolean modeling of biological regulatory networks: A methodology tutorial. Methods.
[52] Saez-Rodriguez, J., Alexopoulos, L. G., Epperlein, J., Samaga, R., Lauffenburger, D. A., Klamt, S., Sorger, P. K., 2009. Discrete logic modelling as a means to link protein signalling networks with functional analysis of mammalian signal transduction. Molecular Systems Biology 5 (331).
[53] Schaefer, C. F., Anthony, K., Krupa, S., Buchoff, J., Day, M., Hannay, T., Buetow, K. H., 2009. PID: the Pathway Interaction Database. Nucleic Acids Research 37 (Database issue), D674–D679.
[54] Schaub, T., Thiele, S., 2009. Metabolic network expansion with ASP. In: Hill, P., Warren, D. (Eds.), Proceedings of the Twenty-fifth International Conference on Logic Programming (ICLP'09). Vol. 5649 of Lecture Notes in Computer Science. Springer-Verlag, pp. 312–326.
[55] Sharan, R., Karp, R. M., 2012. Reconstructing Boolean Models of Signaling. In: Research in Computational Molecular Biology. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 261–271.
[56] Shmulevich, I., Dougherty, E. R., Kim, S., Zhang, W., Jan. 2002. Probabilistic boolean networks: a rule-based uncertainty model for gene regulatory networks. Bioinformatics 18 (2), 261–274.
[57] Shmulevich, I., Saarinen, A., Yli-Harja, O., Astola, J., 2003. Inference of genetic regulatory networks via best-fit extensions. In: Zhang, W., Shmulevich, I. (Eds.), Computational and Statistical Approaches to Genomics. Springer US, pp. 197–210.
[58] Stelling, J., Sauer, U., Szallasi, Z., Doyle, F., Doyle, J., 2004. Robustness of Cellular Functions. Cell 118 (6), 675–685.

[59] Terfve, C. D. A., Cokelaer, T., Henriques, D., Macnamara, A., Gonçalves, E., Morris, M., van Iersel, M., Lauffenburger, D. A., Saez-Rodriguez, J., Oct. 2012. CellNOptR: a flexible toolkit to train protein signaling networks to data using multiple logic formalisms. BMC systems biology 6 (1), 133–133.

[60] Thomas, R., 1991. Regulatory networks seen as asynchronous automata: A logical description. Journal of Theoretical Biology 153 (1), 1–23.

[61] Thomas, R. R., Nov. 1973. Boolean formalization of genetic control circuits. Journal of Theoretical Biology 42 (3), 563–585.

[62] Videla, S., Guziolowski, C., Eduati, F., Thiele, S., Grabe, N., Saez-Rodriguez, J., Siegel, A., 2012. Revisiting the Training of Logic Models of Protein Signaling Networks with ASP. In: Gilbert, D., Heiner, M. (Eds.), Computational Methods in Systems Biology. LNCS. Springer Berlin Heidelberg, pp. 342–361.

[63] Wang, R.-S. R., Saadatpour, A. A., Albert, R. R., Sep. 2012. Boolean modeling in systems biology: an overview of methodology and applications. Physical biology 9 (5).

Appendices

A. Data discretization schemes

Let (V, E, σ) be a PKN. Let ξ be an experimental dataset over (V, E, σ) with size N_ξ. Let $k \in \mathbb{N}$ define the discretization scheme. Let us denote with μ and μ_k, the corresponding minima for Θ_f and Θ_{f_k} over the space of models $\mathbb{M}_{(V,E,\sigma)}$ and with respect to ξ:

$$\mu = \min_{(V,\phi)\in\mathbb{M}_{(V,E,\sigma)}} \Theta_f((V,\phi),\xi) \qquad \mu_k = \min_{(V,\phi)\in\mathbb{M}_{(V,E,\sigma)}} \Theta_{f_k}((V,\phi),\xi).$$

Then $10^{-2k}\mu_k$ converges to μ when k increases, with an exponential speed:

$$\mu_k = 10^{2k}\mu + O(10^k).$$

Moreover, any Boolean logic model minimizing Θ_f, also minimizes Θ_{f_k} within the following tolerance t_k:

$$t_k = 2\sqrt{\frac{N_\xi}{\mu_k}} + \frac{N_\xi}{\mu_k}.$$

Proof. Let (V, ϕ) be any Boolean logic model having evidence in (V, E, σ). Let $\rho_1, \ldots, \rho_n$ be the n Boolean predictions of (V, ϕ) with each ρ_i defined under ε_i. The difference Θ_f and $10^{-2k}\Theta_{f_k}$ over (V, ϕ) with respect to ξ is given by:

$$\left|[\Theta_f - 10^{-2k}\Theta_{f_k}]((V,\phi),\xi)\right| = \left|\sum_{i=1}^{n}\sum_{v\in dom(\omega_i)} [\omega_i(v)-\rho_i(v)]^2 - [\delta_k(\omega_i(v))-\rho_i(v)]^2\right|$$

For each triplet (ω_i, ρ_i, v) let $\gamma_i(v) = \delta_k(\omega_i(v)) - \rho_i(v)$ and $\delta_i(v) = \omega_i - \delta_k(\omega_i(v))$. Therefore:

$$\begin{aligned}\left|[\Theta_f - 10^{-2k}\Theta_{f_k}]((V,\phi),\xi)\right| &= \left|\sum_{i=1}^{n}\sum_{v\in dom(\omega_i)} [\gamma_i(v)+\delta_i(v)]^2 - \gamma_i(v)^2\right| \\ &= \left|\sum_{i=1}^{n}\sum_{v\in dom(\omega_i)} \delta_i(v)\ [2\gamma_i(v)+\delta_i(v)]\,.\right|\end{aligned}$$

Recall that $N_\xi = \sum_{i=1}^{n} |dom(\omega_i)|$. From the Cauchy-Schwarz inequality we have that

$$\sum_{i=1}^{n}\sum_{v\in dom(\omega_i)} \Big|\gamma_i(v)\Big| \leq \sqrt{N_\xi}\sqrt{\sum_{i=1}^{n}\sum_{v\in dom(\omega_i)} \gamma_i(v)^2} = \sqrt{N_\xi}\sqrt{10^{-2k}\Theta_{f_k}((V,\phi),\xi)}.$$

Notice that from the discretization scheme we have that $\delta_i(v) < 10^{-k}$ for every (i, v). It follows that:

$$\begin{aligned}\left|[\Theta_f - 10^{-2k}\Theta_{f_k}]((V,\phi),\xi)\right| &\leq 10^{-k}\left(\sum_{i=1}^{n}\sum_{v\in dom(\omega_i)} 2\Big|\gamma_i(v)\Big| + \sum_{i=1}^{n}\sum_{v\in dom(\omega_i)} \Big|\delta_i(v)\Big|\right) \\ &\leq 10^{-k}\left(2\sqrt{N_\xi}\sqrt{10^{-2k}\Theta_{f_k}((V,\phi),\xi)} + 10^{-k}N_\xi\right).\end{aligned}$$

We deduce that

$$\mu \leq 10^{-2k}\mu_k + 10^{-k}\left(2\sqrt{N_\xi}\sqrt{10^{-2k}\mu_k} + 10^{-k}N_\xi\right) \tag{A.1}$$

$$\leq 10^{-2k}\mu_k\left(1 + 2\times 10^{-k}\sqrt{\frac{N_\xi}{10^{-2k}\mu_k}} + 10^{-2k}\frac{N_\xi}{10^{-2k}\mu_k}\right) = 10^{-2k}\mu_k\left(1 + \underbrace{2\sqrt{\frac{N_\xi}{\mu_k}} + \frac{N_\xi}{\mu_k}}_{=t_k}\right)$$

With a similar reasoning introducing $\gamma'_i(v) = w_i(v) - \rho_i(v)$ instead of $\gamma(v)$, we have the following relation.

$$\begin{aligned}\left|[\Theta_f - 10^{-2k}\Theta_{f_k}]((V,\phi),\xi)\right| &= \left|\sum_{i=1}^{n}\sum_{v\in dom(\omega_i)} \delta_i(v)\ [2\gamma'_i(v) - \delta_i(v)]\right| \\ &\leq 10^{-k}\left(2\sqrt{N_\xi}\sqrt{\Theta_f((V,\phi),\xi)} + 10^{-k}N_\xi\right).\end{aligned}$$

Therefore,

$$\begin{aligned}\mu_k &\leq 10^{2k}\mu + 10^k\left(2\sqrt{N_\xi}\sqrt{\mu} + 10^{-k}N_\xi\right) \\ &\leq 10^{2k}\mu + 10^k\underbrace{\left(2\sqrt{N_\xi}\sqrt{\mu} + N_\xi\right)}_{=B}\end{aligned}$$

Introducing this inequality in (A.1), we deduce that:

$$\begin{aligned}10^{2k}\mu &\leq \mu_k + 10^k\left(2\sqrt{N_\xi}\sqrt{10^{-2k}\mu_k} + 10^{-k}N_\xi\right) \\ &\leq \mu_k + 10^k\left(2\sqrt{N_\xi}\sqrt{\mu + 10^{-k}B} + N_\xi\right) \\ &\leq \mu_k + 10^k\left(\underbrace{2\sqrt{N_\xi}\sqrt{\mu + B} + N_\xi}_{=C}\right)\end{aligned}$$

Altogether, we have that there exists $D = \max\{B, C\}$, which is independent from k, such that

$$\left|10^{2k}\mu - \mu_k\right| \leq D10^k.$$

□

B. ASP encoding correctness

Let (V, E, σ) be a PKN and let $\xi = ((\varepsilon_1, \omega_1), \ldots, (\varepsilon_m, \omega_m))$ be an experimental dataset over it. Let k define the discretization scheme. Let L be the logic program given in Listing 2. Let $\tau((V, E, \sigma), \xi, k)$ be the instance encoding as described in Section 4 (e.g. Listing 1).

Next, let us define the following sets of terms needed for our proofs. For every $v \in V$ let S_v^1 be the set of terms describing the singleton predecessors of v in (V, E, σ):

$$S_v^1 = \{\texttt{set}(w, s, \texttt{nil}) \mid (w, v) \in E, ((w, v), s) \in \sigma\}.$$

Furthermore, for $1 < i \leq |S_v^1|$ let S_v^i be defined recursively describing the sets of i predecessors of v,

$$\begin{aligned}S_v^i = \{\texttt{set}(u, s_u, \texttt{set}(w, s_w, t)) \mid \\ \texttt{set}(u, s_u, \texttt{nil}) \in S_v^1, \texttt{set}(w, s_w, t) \in S_v^{i-1}, u < w\}.\end{aligned}$$

where $<$ denotes a total ordering over V and either $t = \texttt{nil}$, or $t \in S_v^{i-2}$.

In what follows we show that our ASP encoding $L \cup \tau((V, E, \sigma), \xi, k)$ is sound and complete with respect to the multi-objective optimization problem described in Section 3 and according to the discretization scheme given by k. That is, we prove that X is an answer set of $L \cup \tau((V, E, \sigma), \xi, k)$ iff X describes a Boolean logic model (V, ϕ_{opt}) such that,

$$(V, \phi) = \underset{(V,\phi')\in\mathbb{M}_{(V,E,\sigma)}}{\arg\min} (\Theta_{f_k}((V, \phi'), \xi), \Theta_s((V, \phi')))$$

minimizing first Θ_{f_k}, and then with lower priority Θ_s.

First, let us recall some standard notation for logic rules. For a rule r of the form

$$A_0 \texttt{:-}\ A_1, ..., A_m, \texttt{not}\ A_{m+1}, \ldots, \texttt{not}\ A_n.$$

with atoms A_i, we define $head(r) = A_0$, $body^+(r) = \{A_1, \ldots, A_m\}$, and $body^-(r) = \{A_{m+1}, \ldots, A_n\}$. Further, we often use the connective $\leftarrow$ instead of `:-` .

Soundness. Let X be an answer set of $L \cup \tau((V, E, \sigma), \xi, k)$. Furthermore, let

$$P^X = \{(head(r) \leftarrow body(r)^+)\theta \mid r \in L \cup \tau((V, E, \sigma), \xi, k), (body(r)^-\theta) \cap X = \emptyset, \theta : var(r) \rightarrow \mathcal{A}\}$$

where $var(r)$ is the set of all variables that occur in a rule r, $\mathcal{A}$ is the set of all constants appearing in $L \cup \tau((V, E, \sigma), \xi)$, and θ is a ground substitution for the variables in r. Then, by definition of an answer set, we know that X is a $\subseteq$-minimal model of P^X.

Next, let us define

$$\mathcal{C}_v = \{c \mid \texttt{conjunction}(c, n, v) \in X, c \in S_v^n\}.$$

Further, for every $c \in \mathcal{C}_v$, let $\mathcal{L}_c = \mathcal{L}_c^+ \cup \mathcal{L}_c^-$ with

$$\mathcal{L}_c^+ = \{w \mid \texttt{in}(w, 1, c) \in X\} \qquad \mathcal{L}_c^- = \{\neg w \mid \texttt{in}(w, -1, c) \in X\}.$$

be the set of literals occurring in c. Next, we define the mapping ϕ for every v such that $\mathcal{C}_v \neq \emptyset$ as

$$\phi(v) = \bigvee_{c \in \mathcal{C}_v} \bigwedge_{l \in \mathcal{L}_c} l \quad .$$

Moreover, for every $(\varepsilon_i, \omega_i) \in \xi$ and $v \in V$ let us define the Boolean predictions of (V, ϕ) under each ε_i as,

$$\rho_i(v) = \begin{cases} 1 & \text{if } \texttt{active}(i, v) \in X \\ 0 & \text{otherwise.} \end{cases}$$

We show that (V, ϕ) is a Boolean logic model such that,

$$(V, \phi) = \underset{(V,\phi') \in \mathbb{M}_{(V,E,\sigma)}}{\arg\min} (\Theta_{f_k}((V, \phi'), \xi), \Theta_s((V, \phi')))$$

minimizing first Θ_{f_k}, and then with lower priority Θ_s. Towards this end, first we show that (V, ϕ) is a Boolean logic model having evidence in (V, E, σ) and without feedback-loops, i.e., $(V, \phi) \in \mathbb{M}_{(V,E,\sigma)}$. Thereafter, we show that ρ_i is the fixpoint of $T_{(V,\phi|_{\varepsilon_i})}$ for each $i = 1, \ldots, m$. Finally, we show that (V, ϕ) is an optimum model with respect to the lexicographic multi-objective optimization, minimizing first Θ_{f_k} and then Θ_s.

Consider the rules in Listing 4:

Listing 4: Lines 1-5 from Listing 2

```
1 sub(set(U,S,nil),1,V)             :- edge(U,V,S).
2 sub(set(U,SU,set(W,SW,T)),N+1,V) :- edge(U,V,SU), sub(set(W,SW,T),N,V), U<W.
3
4 in(U,S,set(U,S,T))    :- sub(set(U,S,T),N,V).
5 in(W,SW,set(U,SU,T)) :- in(W,SW,T), sub(set(U,SU,T),N,V).
```

Together with the instance encoding $\tau((V, E, \sigma), \xi, k)$, the rules in Listing 4 enforce that for every $v \in V$ and $s \in S_v^n$ we have $\texttt{sub}(s, n, v) \in X$ with $1 \leq n \leq |S_v^1|$. Further, if $\texttt{set}(u, s, t) \in S_v^n$ then $\texttt{in}(u, s, \texttt{set}(u, s, t)) \in X$. Moreover, if $\texttt{in}(w, s_w, t) \in X$ and $\texttt{set}(u, s_u, t) \in S_v^n$, then $\texttt{in}(w, s_w, \texttt{set}(u, s_u, t)) \in X$. Therefore, the choice rule in Listing 5:

Listing 5: Line 7 from Listing 2

```
7 {conjunction(C,N,V)} :- sub(C,N,V).
```

guarantees that $\phi(v)$ has an evidence in (V, E, σ). Hence, for every $\subseteq$-minimal model X it holds that if $\texttt{conjunction}(c, n, v) \in X$ then $\texttt{sub}(c, n, v) \in X$ for $c \in S_v^n$. Next, consider the rules in Listing 6:

Listing 6: Lines 9-11 from Listing 2

```
9  path(U,V) :- conjunction(C,_,V), in(U,_,C).
10 path(U,V) :- conjunction(C,_,V), in(W,_,C), path(U,W).
11 :- path(V,V).
```

The rules above eliminate candidate answer sets describing logic models with feedback-loops. Paths from U to V are derived recursively in lines 9 and 10. Then, the integrity constraint in line 11 avoids self-reachability. Notably, it follows that (V, ϕ) defines a Boolean logic model without feedback-loops. Otherwise, there must exists $v_0, \ldots, v_p$ and $c \in \mathcal{C}_{v_i}$ such that $\texttt{in}(v_{(i+1)\%p}, s, c) \in X$ with $s \in \{-1, 1\}$ for every $i = 0, \ldots, p$.[5] In such a case, the rule in line 9 enforce that $\texttt{path}(v_{(i+1)\%p}, v_i) \in X$ for every $\subseteq$-minimal model X. Further, by the rule in line 10, if $\texttt{path}(v_{(i+1)\%p}, v_i) \in X$ and $\texttt{path}(v_{(i+2)\%p}, v_{(i+1)\%p}) \in X$, then $\texttt{path}(v_{(i+2)\%p}, v_i) \in X$. Hence, it also holds that $\texttt{path}(v_i, v_i) \in X$ for every $i = 0, \ldots, p$. Thus, due the integrity constraint in line 11, X cannot be a model of P^X. Therefore, (V, ϕ) defines a Boolean logic model without feedback-loops.

Next, we show that ρ_i as defined above is the fixpoint of $T_{(V,\phi|_{\varepsilon_i})}$, i.e., $T_{(V,\phi|_{\varepsilon_i})}(\rho_i) = \rho_i$. Consider the rules in Listing 7:

Listing 7: Lines 15-22 from Listing 2

```
15 exp(E) :- exp(E,_,_).
16 mapped(V) :- conjunction(_,_,V).
17 fixed(E,V) :- exp(E,V,0).
18 fixed(E,V) :- exp(E), vertex(V), not mapped(V).
19
20 active(E,V) :- exp(E,V,1), stimulus(V).
21 active(E,V) :- exp(E), conjunction(S,M,V), not fixed(E,V),
22                active(E,U) : in(U,1,S), not active(E,U) : in(U,-1,S).
```

The rules in lines 15-18 in Listing 7 together with the instance encoding $\tau((V, E, \sigma), \xi, k)$ enforce that for every $\subseteq$-minimal model X, $\texttt{fixed}(i, v) \in X$ iff $\varepsilon_i(v) = 0$ (line 17) or $\mathcal{C}_v = \emptyset$, i.e., $v \notin dom(\phi)$ (line 18). Further, the rules in lines 20-22 enforce that $\texttt{active}(i, v) \in X$ iff $v \in V_S$ and $\varepsilon_i(v) = 1$, or $\texttt{fixed}(i, v) \notin X$ and there exists $c \in \mathcal{C}_v$ such that for every $w \in \mathcal{L}_c^+$, $\texttt{active}(i, w) \in X$ and for every $\neg w \in \mathcal{L}_c^-$, $\texttt{active}(i, w) \notin X$.

Let us compute $T_{(V,\phi|_{\varepsilon_i})}(\rho_i)$ for every $v \in V$. If $v \in V_S$ and $\varepsilon_i(v) = 1$ then $\phi|_{\varepsilon_i}(v) = \top$ and $\texttt{active}(i, v) \in X$, thus

$$T_{(V,\phi|_{\varepsilon_i})}(\rho_i)(v) = \rho_i(\top) = 1 = \rho_i(v).$$

If $v \in V_S \cup V_K$ and $\varepsilon_i(v) = 0$, or $v \notin dom(\phi) \cup V_S$ then $\phi|_{\varepsilon_i}(v) = \bot$, $\texttt{fixed}(i, v) \in X$ and $\texttt{active}(i, v) \notin X$, thus

$$T_{(V,\phi|_{\varepsilon_i})}(\rho_i)(v) = \rho_i(\bot) = 0 = \rho_i(v).$$

Otherwise, $\phi|_{\varepsilon_i}(v) = \phi(v)$ and we have

$$T_{(V,\phi|_{\varepsilon_i})}(\rho_i)(v) = \rho_i(\phi(v))$$

thus, we need to show that $\rho_i(\phi(v)) = \rho_i(v)$. By definition, $\rho_i(v) = 1$ iff $\texttt{active}(i, v) \in X$. Further, as shown above $\texttt{active}(i, v) \in X$ iff $v \in V_S$ and $\varepsilon_i(v) = 1$, or $\texttt{fixed}(i, v) \notin X$ and there exists $c \in \mathcal{C}_v$ such that for every $w \in \mathcal{L}_c^+$, $\texttt{active}(i, w) \in X$ ($\rho_i(w) = 1$) and for every $\neg w \in \mathcal{L}_c^-$, $\texttt{active}(i, w) \notin X$ ($\rho_i(w) = 0$). Therefore, it holds $\rho_i(\bigwedge_{l \in \mathcal{L}_c} l) = 1$. Further, since $\phi(v)$ is in disjunctive normal form, it also

[5] With % we denote the *modulo* operator.

holds $\rho_i(\phi(v)) = 1$. Analogously, it can be shown $\rho_i(v) = 0$ iff $\rho_i(\phi(v)) = 0$. Thus, we have that ρ_i is the fixpoint of $T_{(V,\phi|_{\varepsilon_i})}$.

Finally, it remains to show that the following holds,

$$(V,\phi) = \operatorname*{arg\,min}_{(V,\phi')\in\mathbb{M}_{(V,E,\sigma)}} (\Theta_{f_k}((V,\phi'),\xi), \Theta_s((V,\phi')))$$

minimizing first Θ_{f_k}, and then with lower priority Θ_s. Consider the rules in Listing 8:

Listing 8: Lines 24-29 from Listing 2

```
24 residual(D,V,1,#pow(F-D,2)) :- obs(E,V,D), dfactor(F), D<F.
25 residual(D,V,0,#pow(D,2))   :- obs(E,V,D), D>0.
26
27 #minimize[conjunction(_,N,_)=N@1].
28 #minimize[active(E,V) : obs(E,V,D) : residual(D,V,1,W)=W@2,
29           not active(E,V) : obs(E,V,D) : residual(D,V,0,W)=W@2].
```

Rules in lines 24 and 25 together with the instance encoding $\tau((V,E,\sigma),\xi,k)$ enforce that $\texttt{residual}(d,v,x,w) \in X$ if there exists $(\varepsilon_i,\omega_i) \in \xi$ and $v \in dom(\omega_i)$ such that $d = A_k\delta_k(\omega_i(v))$ with $w = (A_k\delta_k(\omega_i(v)) - A_kx)^2$ and $x \in \{0,1\}$. Moreover, for every atom $\texttt{residual}(d,v,x,w) \in X$, we have $w > 0$. Therefore, the minimization statement in lines 31 and 32 enforces that (V,ϕ) is minimal with respect to Θ_{f_k}. To be more precise,

$$\begin{aligned}
\sum_{\substack{\texttt{residual}(d,v,1,w)\in X\\ \texttt{obs}(i,v,d)\in X\\ \texttt{active}(i,v)\in X}} w + \sum_{\substack{\texttt{residual}(d,v,0,w)\in X\\ \texttt{obs}(i,v,d)\in X\\ \texttt{active}(i,v)\notin X}} w &= \\
&= \sum_{\substack{\texttt{obs}(i,v,d)\in X\\ \texttt{active}(i,v)\in X}} (A_k\delta_k(\omega_i(v)) - A_k)^2 + \sum_{\substack{\texttt{obs}(i,v,d)\in X\\ \texttt{active}(i,v)\notin X}} (A_k\delta_k(\omega_i(v)))^2 \\
&= \sum_{\texttt{obs}(i,v,d)\in X} (A_k\delta_k(\omega_i(v)) - A_k\rho_i(v))^2 \\
&= \sum_{i=1}^{m} \sum_{v\in dom(\omega_i)} (A_k\delta_k(\omega_i(v)) - A_k\rho_i(v))^2 = \Theta_{f_k}((V,\phi),\xi).
\end{aligned}$$

Further, with lower priority, the minimization statement in line 30 guarantees that among the models minimizing Θ_{f_k}, (V,ϕ) is also minimal with respect to Θ_s. To be more precise,

$$\sum_{\texttt{conjunction}(c,n,v)\in X} n = \sum_{v\in V}\sum_{c\in\mathcal{C}_v} |\mathcal{L}_c| = \sum_{v\in dom(\phi)} |\phi(v)| = \Theta_s((V,\phi)).$$

□

Completeness. Let (V,ϕ) be a Boolean logic model and let $\rho_1,\ldots,\rho_m$ be its Boolean predictions with each ρ_i defined under ε_i such that,

$$(V,\phi) = \operatorname*{arg\,min}_{(V,\phi')\in\mathbb{M}_{(V,E,\sigma)}} (\Theta_{f_k}((V,\phi'),\xi), \Theta_s((V,\phi')))$$

minimizing first Θ_{f_k}, and then with lower priority Θ_s.

Next, for every $v \in V$, let $\mathcal{C}_v$ be the set of conjuncts occurring in $\phi(v)$. Further, for every $c \in \mathcal{C}_v$, let $\mathcal{L}_c$ be the set of literals occurring in c. Furthermore, we denote with $\mathcal{L}_c^+$ and $\mathcal{L}_c^-$, the set of positive and negative literals occurring in c, respectively. Then, a path in ϕ from u_n to v is such that there exists literals $l_1,\ldots,l_n$ of the form u_i or $\neg u_i$, such that $l_1 \in \mathcal{L}_c$ for some $c \in C_v$, and $l_{i+1} \in \mathcal{L}_{c_i}$ for some $c_i \in \mathcal{C}_{u_i}$ with $i = 1,\ldots,n-1$.

We consider the following set X of atoms. First, all atoms in $\tau((V,E,\sigma),\xi,k)$ are included in X. Next, for every $v \in V$ and $s \in S_v^n$ we consider $\texttt{sub}(s,n,v) \in X$ with $1 \leq n \leq |S_v^1|$. Further, if $\texttt{set}(u,s,t) \in S_v^n$ then $\texttt{in}(u,s,\texttt{set}(u,s,t)) \in X$. Moreover, if $\texttt{in}(w,s_w,t) \in X$ and $\texttt{set}(u,s_u,t) \in S_v^n$, then $\texttt{in}(w,s_w,\texttt{set}(u,s_u,t)) \in X$. Next, for every $v \in V$ and $c \in \mathcal{C}_v$ with length n, let $s \in S_v^n$ such that for every $w \in \mathcal{L}_c^+$, $\texttt{in}(w,1,s) \in X$ and for every $\neg w \in \mathcal{L}_c^-$, $\texttt{in}(w,-1,s) \in X$. Then, we consider atoms $\texttt{conjunction}(s,n,v) \in X$. Next, for every path from u to v in ϕ, we consider atoms $\texttt{path}(u,v) \in X$. Recall that $\xi = ((\varepsilon_1,\omega_1),\ldots,(\varepsilon_m,\omega_m))$, then we consider $\texttt{exp}(i) \in X$ for every $i = 1,\ldots,m$. Further, for every $v \in V_S \cup V_K$ if $\varepsilon_i(v) = 0$ we consider $\texttt{fixed}(i,v) \in X$. Moreover, for every $v \in V$ if $\mathcal{C}_v = \emptyset$ then, we consider $\texttt{fixed}(i,v) \in X$ for every $i = 1,\ldots,m$. Furthermore, for every $v \in V$ such that $\mathcal{C}_v \neq \emptyset$, let $\texttt{mapped}(v) \in X$. Next, we consider $\texttt{active}(i,v) \in X$ for every $i = 1,\ldots,m$ such that $\rho_i(v) = 1$. Finally, we consider $\texttt{residual}(d,v,x,w) \in X$ if there exists $(\varepsilon_i,\omega_i) \in \xi$ and $v \in dom(\omega_i)$ such that $d = A_k\delta_k(\omega_i(v))$ with $w = (A_k\delta_k(\omega_i(v)) - A_kx)^2$ and $x \in \{0,1\}$. Moreover, we consider only atoms $\texttt{residual}(d,v,x,w) \in X$ with $0 < w$.

We need to show that X is an answer set of $L \cup \tau((V,E,\sigma),\xi,k)$ verifying that X is a $\subseteq$-minimal model of

$$P^X = \{(head(r) \leftarrow body(r)^+)\theta \mid r \in L \cup \tau((V,E,\sigma),\xi,k), (body(r)^-\theta) \cap X = \emptyset, \theta : var(r) \rightarrow \mathcal{A}\}$$

where $var(r)$ is the set of all variables that occur in a rule r, $\mathcal{A}$ is the set of all constants appearing in $L \cup \tau((V,E,\sigma),\xi,k)$, and θ is a ground substitution for the variables in r.

To start with, we note that X includes all the facts in $\tau((V,E,\sigma),\xi,k)$. Each of these facts belongs also to P^X. Thus, any set Y of atoms excluding at least one of them, cannot be a model of P^X. Next, consider the rules in Listing 9:

Listing 9: Lines 1-5 from Listing 2

```
1 sub(set(U,S,nil),1,V)              :- edge(U,V,S).
2 sub(set(U,SU,set(W,SW,T)),N+1,V) :- edge(U,V,SU), sub(set(W,SW,T),N,V), U<W.
3
4 in(U,S,set(U,S,T))    :- sub(set(U,S,T),N,V).
5 in(W,SW,set(U,SU,T)) :- in(W,SW,T), sub(set(U,SU,T),N,V).
```

The ground instances of the rules above belongs to P^X. Furthermore, all of them are satisfied by X, but not by any set $Y \subset X$. Rules in lines 1 and 2 enforce that for every $s \in S_v^n$, an atom $\texttt{sub}(s,n,v)$ is included in every $\subseteq$-minimal model. In addition, rules in lines 4 and 5 enforce that any set excluding from X at least one atom over predicate $\texttt{in}/3$ cannot be a model of P^X.

Next, we consider the choice rule in Listing 10:

Listing 10: Line 7 from Listing 2

```
7 {conjunction(C,N,V)} :- sub(C,N,V).
```

In fact, a choice rule of the form:

$$\{\texttt{H}\} \leftarrow \texttt{B}.$$

can be translated into 3 rules:

$$\texttt{A} \leftarrow \texttt{B}. \qquad \texttt{H} \leftarrow \texttt{A}, \texttt{not } \overline{\texttt{H}}. \qquad \overline{\texttt{H}} \leftarrow \texttt{not H}.$$

by introducing new atoms $\texttt{A}$ and $\overline{\texttt{H}}$. Therefore, for every atom $\texttt{conjunction}(c,n,v) \in X$ with $c \in S_v^n$, P^X includes rules of the form:

$$\texttt{A} \leftarrow \texttt{sub}(c,n,v). \qquad \texttt{conjunction}(c,n,v) \leftarrow \texttt{A}.$$

Recall that for every $c \in S_v^n$, an atom $\texttt{sub}(c,n,v)$ is included in every $\subseteq$-minimal model. Thus, we have that X satisfies such rules and any set excluding from X at least one atom over predicate $\texttt{conjunction}/3$ cannot be a model of P^X.

Now, consider the rules in Listing 11:

Listing 11: Lines 9-11 from Listing 2

```
9  path(U,V) :- conjunction(C,_,V), in(U,_,C).
10 path(U,V) :- conjunction(C,_,V), in(W,_,C), path(U,W).
11 :- path(V,V).
```

All the ground instance of the rules above are in P^X. Further, by construction of X, we have that $\texttt{conjunction}(s,n,v) \in X$ if there is some $c \in \mathcal{C}_v$ such that for every $w \in \mathcal{L}_c^+$, $\texttt{in}(w,1,s) \in X$ and for every $\neg w \in \mathcal{L}_c^-$, $\texttt{in}(w,-1,s) \in X$. Thus, the rules in lines 9 and 10 enforce to have an atom $\texttt{path}(u,v)$ included in every $\subseteq$-minimal model of P^X for each path from u to v in ϕ. Therefore, any set excluding from X at least one atom over predicate $\texttt{path}/3$ cannot be a model of P^X. Further, since (V,ϕ) is feedback-loops free, the integrity constraint in line 11 is also satisfied by X.

Listing 12: Line 13 from Listing 2

```
13 :- conjunction(C1,N,V), conjunction(C2,M,V), N<M, in(U,S,C2) : in(U,S,C1).
```

Moreover, since among the models minimizing Θ_{f_k}, (V,ϕ) is also minimal with respect to Θ_s, X satisfies the integrity constraint in Listing 12 as well. Otherwise, for some $v \in V$ there must exists $c_1, c_2 \in \mathcal{C}_v$ such that $\mathcal{L}_{c_1} \subset \mathcal{L}_{c_2}$. Notably, in such a case, $c_1 \vee c_2$ is logically equivalent to c_1. Therefore, there is a Boolean logic model (V,ϕ') such that $\Theta_{f_k}((V,\phi'),\xi) = \Theta_{f_k}((V,\phi),\xi)$ and $\Theta_s((V,\phi')) < \Theta_s((V,\phi))$.

Consider the rules in Listing 13:

Listing 13: Lines 15-22 from Listing 2

```
15 exp(E) :- exp(E,_,_).
16 mapped(V) :- conjunction(_,_,V).
17 fixed(E,V) :- exp(E,V,0).
18 fixed(E,V) :- exp(E), vertex(V), not mapped(V).
19
20 active(E,V) :- exp(E,V,1), stimulus(V).
21 active(E,V) :- exp(E), conjunction(S,M,V), not fixed(E,V),
22               active(E,U) : in(U,1,S), not active(E,U) : in(U,-1,S).
```

Since for every $(\varepsilon_i,\omega_i) \in \xi$ and $v \in V_S \cup V_K$ there is an atom $\texttt{exp}(i,v,\varepsilon_i(v))$ included in every $\subseteq$-minimal model of P^X, the rule in line 15 enforces that there is also an atom $\texttt{exp}(i)$ included in every $\subseteq$-minimal model of P^X for every $i = 1,\ldots,m$. Furthermore, the rule in line 16 guarantees that for every $v \in V$ such that $\mathcal{C}_v \neq \emptyset$, we have an atom $\texttt{mapped}(v)$ included in every $\subseteq$-minimal model of P^X. Next, the rules in lines 17 and 18 enforce that an atom $\texttt{fixed}(i,v)$ is included in every $\subseteq$-minimal model of P^X if either $\varepsilon_i(v) = 0$ with $v \in V_S \cup V_K$, or if $\mathcal{C}_v = \emptyset$ with $v \in V$ and $i = 1,\ldots,m$. Hence, any set excluding from X at least one atom over predicates $\texttt{exp}/1$, $\texttt{mapped}/1$ or $\texttt{fixed}/2$ cannot be a model of P^X. Now, let us show that for every $v \in V$ and $i = 1,\ldots,m$, an atom $\texttt{active}(i,v)$ is included in every $\subseteq$-minimal model of P^X if $\rho_i(v) = 1$. To start with, recall that ρ_i is the fixpoint of $T_{(V,\phi|_{\varepsilon_i})}$. Thus, for every $v \in V$ it holds that, $\rho_i(v) = \rho_i(\phi|_{\varepsilon_i}(v))$. Then, $\rho_i(v) = 1$ iff $\phi|_{\varepsilon_i}(v) = \top$ or $\phi|_{\varepsilon_i}(v) = \phi(v)$ and for some $c \in \mathcal{C}_v$ it holds that, $\rho_i(w) = 1$ for every $w \in \mathcal{L}_c^+$ and $\rho_i(w) = 0$ for every $w \in \mathcal{L}_c^-$. The rule in line 20 enforces that an atom $\texttt{active}(i,v)$ is included in every $\subseteq$-minimal model of P^X if $\varepsilon_i(v) = 1$ with $v \in V_S$ and $i = 1,\ldots,m$. In such case, we have $\phi|_{\varepsilon_i}(v) = \top$. Meanwhile, the rule in lines 21 and 22 enforces that an atom $\texttt{active}(i,v)$ is included in every $\subseteq$-minimal model of P^X if $\phi|_{\varepsilon_i}(v) = \phi(v)$ and for some $c \in \mathcal{C}_v$ it holds that, $\rho_i(w) = 1$ for every $w \in \mathcal{L}_c^+$ and $\rho_i(w) = 0$ for every $w \in \mathcal{L}_c^-$. Therefore, any set excluding from X at least one atom over predicate $\texttt{active}/2$ cannot be a model of P^X.

Finally, consider the rules in Listing 14:

Listing 14: Lines 24-29 from Listing 2

```
24 residual(D,V,1,#pow(F-D,2)) :- obs(E,V,D), dfactor(F), D<F.
25 residual(D,V,0,#pow(D,2))   :- obs(E,V,D), D>0.
26
27 #minimize[conjunction(_,N,_)=N@1].
28 #minimize[active(E,V) : obs(E,V,D) : residual(D,V,1,W)=W@2,
29           not active(E,V) : obs(E,V,D) : residual(D,V,0,W)=W@2].
```

All the ground instances of the rules in lines 24 and 25 belong to P^X. Furthermore, such rules enforce that every $\subseteq$-minimal model of P^X includes an atom $\texttt{residual}(d, v, x, w)$ if there exists $(\varepsilon_i, \omega_i) \in \xi$ and $v \in dom(\omega_i)$ such that $d = A_k\delta_k(\omega_i(v))$ with $w = (A_k\delta_k(\omega_i(v)) - A_kx)^2$ and $x \in \{0, 1\}$. In fact, only atoms $\texttt{residual}(d, v, x, w)$ with $0 < w$ are included. Thus, any set excluding from X at least one atom over predicate $\texttt{residual}/4$ cannot be a model of P^X. To conclude the proof, given that (V, ϕ) minimize Θ_{f_k}, X satisfies the minimization statement in lines 31 and 32. Moreover, given that among the Boolean logic models minimizing Θ_{f_k}, (V, ϕ) also minimize Θ_s, X satisfies the minimization statement in line 30 as well. For more details on the equivalence between the objective functions and the minimization statements, we refer the reader to the proof of soundness.

We have investigated all rules in $L \cup \tau((V, E, \sigma), \xi, k)$ and shown that their ground instances in P^X are satisfied by X. Moreover, we have checked that any set excluding from X at least one atom is not a model of P^X. Hence, X is a $\subseteq$-minimal model of P^X and thus an answer set of $L \cup \tau((V, E, \sigma), \xi, k)$.

□