

Revisiting the Training of Logic Models of Protein Signaling Networks with a Formal Approach based on Answer Set Programming

Santiago Videla, Carito Guziolowski, Federica Eduati, Sven Thiele, Niels Grabe, Julio Saez-Rodriguez, Anne Siegel

▶ **To cite this version:**

Santiago Videla, Carito Guziolowski, Federica Eduati, Sven Thiele, Niels Grabe, et al.. Revisiting the Training of Logic Models of Protein Signaling Networks with a Formal Approach based on Answer Set Programming. CMSB - 10th Computational Methods in Systems Biology 2012, D. Gilbert and M. Heiner, Oct 2012, London, United Kingdom. pp.342-361, 10.1007/978-3-642-33636-2_20 . hal-00737112v1

HAL Id: hal-00737112

https://inria.hal.science/hal-00737112v1

Submitted on 1 Oct 2012 (v1), last revised 21 Dec 2012 (v2)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Revisiting the Training of Logic Models of Protein Signaling Networks with a Formal Approach based on Answer Set Programming

Santiago Videla[1 2 7], Carito Guziolowski[3 7], Federica Eduati[4 5], Sven Thiele[2 1 6], Niels Grabe[3], Julio Saez-Rodriguez[5]§, and Anne Siegel[1 2]§

[1] CNRS, UMR 6074 IRISA, Campus de Beaulieu, 35042 Rennes cedex, France.
[2] INRIA, Centre Rennes-Bretagne-Atlantique, Projet Dyliss, Campus de Beaulieu, 35042 Rennes cedex, France.
[3] University Heidelberg, National Center for Tumor Diseases, Hamamatsu TIGA Center, Im Neuenheimer Feld 267, 69120 Heidelberg, Germany.
[4] Department of Information Engineering, University of Padova, Padova, 31050, Italy.
[5] European Bioinformatics Institute (EMBL-EBI) Wellcome Trust Genome Campus, Cambridge CB10 1SD, UK.
[6] University of Potsdam, Institute for Computer Science, Germany.
[7] These authors contributed equally to this work

Abstract. A fundamental question in systems biology is the construction and training to data of mathematical models. Logic formalisms have become very popular to model signaling networks because their simplicity allows us to model large systems encompassing hundreds of proteins. An approach to train (Boolean) logic models to high-throughput phosphoproteomics data was recently introduced and solved using optimization heuristics based on stochastic methods. Here we demonstrate how this problem can be solved using Answer Set Programming (ASP), a declarative problem solving paradigm, in which a problem is encoded as a logical program such that its answer sets represent solutions to the problem. ASP has significant improvements over heuristic methods in terms of efficiency and scalability, it guarantees global optimality of solutions as well as provides a complete set of solutions. We illustrate the application of ASP with *in silico* cases based on realistic networks and data.

Keywords: Logic modeling, answer set programming, protein signaling networks

1 Introduction

Cells perceive extracellular information via receptors that trigger signaling pathways that transmit this information and process it. Among other effects, these pathways regulate gene expression (transcriptional regulation), thereby defining the response of the cell to the information sensed in its environment. Over

§ Corresponding authors: saezrodriguez@ebi.ac.uk, anne.siegel@irisa.fr

decades of biological research we have gathered large amount of information about these pathways. Nowadays, there exist public repositories such as Pathways Commons [1] and Pathways Interaction Database [2] that contain curated regulatory knowledge, from which signed and oriented graphs can be automatically retrieved [3,4]. These signed-oriented graphs represent molecular interactions inside the cell at the levels of signal transduction and (to a lower extent) of transcriptional regulation. Their edges describe causal events, which in the case of signal transduction are related to the molecular events triggered by cellular receptors. These networks are derived from vast generic knowledge concerning different cell types and they represent a useful starting point to generate predictive models for cellular events.

Phospho-proteomics assays [5] are a recent form of high-throughput or 'omic' data. They measure the level of phosphorylation (correlated with protein-activity) of up to hundreds of proteins at the same moment in a particular biological system [6]. Most cellular key processes, including proliferation, migration, and cell cycle, are ultimatelly controlled by these protein-activity modifications. Thus, measurement of phosphorylation of key proteins under appropriate conditions (experimental designs), such as stimulating or perturbing the system in different ways, can provide useful insights of cellular control.

Computational methods to infer and analyze signaling networks from high-throughput phospho-proteomics data are less mature than for transcriptional data, which has been available for much longer time [6]. In particular, the infererence of gene regulatory networks from transcriptomics data is now an established field (see [7,8] for a review). In comparison to transcriptomics, data is harder to obtain in (phospho) proteomics, but prior knowledge about the networks is much more abundant, and available in public resources as mentioned above.

An approach to integrate the prior knowledge existing in databases with the specific insight provided by phospho-proteomics data was recently introduced and implemented in the tool CellNOpt (CellNetOptimizer; www.cellnopt.org) [9]. CellNOpt uses stochastic optimization algorithms (in particular, a genetic algorithm), to find the Boolean logic model compatible that can best describe the existing data. While CellNOpt has proved able to train networks of realistic size, it suffers from the lack of guarantee of optimum intrinsic of stochastic search methods. Furthermore, it scales poorly since the search space (and thus the computational time) increases exponentially with the network size.

In this paper, we propose a novel method to solve the optimization problem posed in [9] that overcomes its limitations. Our approach trains generic networks based on experimental measures equally as CellNOpt in order to obtain a complete set of global optimal networks specific to the experimental data. This family of optimal networks could be regarded as an explanatory model that is specific to a particular cell type and condition; from these models it should be possible to derive new, more accurate biological insights. To illustrate our approach we used a generic Prior Knowledge Network (PKN) related to signaling events upon stimulation of cellular receptors in hepatocytes, and trained this net-

work with *in silico* simulated phospho-proteomics data. This network was used as a benchmark for network inference in the context of the DREAM (Dialogues for Reverse Engineering Assessment of Methods; www.the-dream-project.org) Predictive Signaling Network Challenge [10].

The proposed solution encodes the optimization problem in Answer Set Programming (ASP) [11]. ASP is a declarative problem solving paradigm from the field of logic programming. Distributed under the GNU General Public Licence, it offers highly efficient inference engines based on Boolean constraint solving technology [12]. ASP allows for solving search problems from the complexity class NP and with the use of disjunctive logic programs from the class Σ_2^P. Moreover, modern ASP tools allow handling complex preferences and multi-criteria optimization, guaranteeing the global optimum by reasoning over the complete solution space.

Our results show significant improvements, concerning computation time and completeness in the search of optimal models, in comparison with CellNOpt. We note that similar features can be obtained by formulation of the problem as an integer linear optimization problem [13]. The perspectives of this work go towards the exploration of the complete space of optimal solutions in order to identify properties such as the robustness of optimal models, and relate them to the quality of the obtained predictions.

2 Formalization

The biological problem that we tackle in this work is essentially a combinatorial optimization problem over the possible logic models representing a given PKN. In this section, first we introduce the graphical representation of logic models by giving a simple example that motivates our formalization. Then, we give a formal definition for the inputs of the problem, we formally define a Protein Signaling Logic Model (PSLM) and we show how predictions are made for a given model. Finally, we define an objective function used for the optimization over the space of possible logic models.

2.1 Motivation

The functional relationships of biological networks, such as PKNs, cannot be captured using only a graph [15,9]. If, for example, two proteins (nodes) A and B have a positive effect on a third one C (encoded in a graph as A $\rightarrow$ C, and B $\rightarrow$ C), is not clear if either A or B can active C, or if both are required (logic OR and AND gate, respectively). To represent such complex (logical) relations between nodes and offer a formal representation of cellular networks, hypergraphs can be used. Since hypergraphs were already described and used to represent logic models of protein signaling networks in [15,16,9,17,14], here we adopt the same formalism and we simply give an example to introduce this representation. For more details, we refer the reader to the cited literature.

Example 1. Given the toy PKN described in Fig. 1(a), an arbitrary compatible logic model is given by the following set of formulas $\{d = (a \wedge b) \vee \neg c; e = c; f = d \wedge e\}$. Moreover, a representation of this logic model is given in Fig. 1(b) as a signed and directed hypergraph. Note that each conjunction clause gives place to a different hyperedge having as its source all the present literals in the clause.

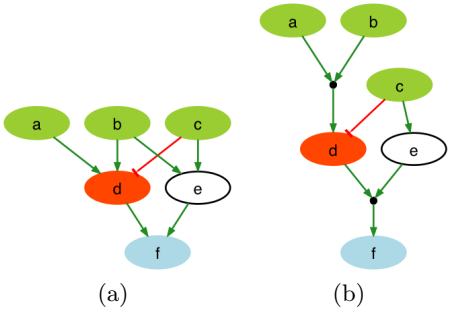


Fig. 1. Hypergraph representation of Logic Models. The green and red edges correspond to activations and inhibitions, respectively. Green nodes represent ligands that can be experimentally stimulated. Red nodes represent those species that can be inhibited by using a drug. Blue nodes represent those species that can be measured by using an antibody. White nodes are neither measured, nor manipulated. **(a)** A toy PKN as a directed and signed graph. **(b)** An arbitrary Logic Model compatible with the PKN shown in (a). Black filled circles represent AND gates, whereas multiple edges arriving to one node model OR gates.

Informally, the training of logic models to phospho-proteomics data consist in finding the hypergraph(s) compatible with a given PKN that best explains the data (using some criteria). Even though a graphical representation is quite intuitive and has been widely used in the literature, it is not the most appropriate way to give a formal and clear formulation of this problem. Thus, in what follows we give a formalization based on propositional logic and in the rest of this work, whenever convenient, we will refer interchangeably to Protein Signaling Logic Models as hypergraphs and to conjunctive clauses as hyperedges.

2.2 Problem Inputs

We identify three inputs to the problem: a Prior Knowledge Network (PKN), a set of experimental conditions or perturbations and for each of them, a set of experimental observations or measurements. For the sake of simplicity, in this work we have considered only PKNs with no feedback loops. They account for the main mechanisms of transmission of information in signaling pathways, but do not include feedback mechanisms that are typically responsible for the switching off of signals once the transmission has taken place [9,6]. In what follows, we give a mathematical definition for each of these inputs.

Definition 1 (Prior Knowledge Network). *A PKN is a signed, acyclic, and directed graph* (V, E, σ) *with* $E \subseteq V \times V$ *the set of directed edges,* $\sigma \subseteq E \times \{1, -1\}$

the signs of the edges , and $V = S \cup K \cup R \cup U$*, the set of vertices where* S *are the stimulus (inputs),* K *are the inhibitors (knock-outs),* R *are the readouts (outputs) and* U *are neither measured, nor manipulated. Moreover, the subsets* S, K, R, U *are all mutually disjoint except for* K *and* R*.*

Note that in the previous definition, σ is defined as a relation and not as a function since it could be the case where both signs are present between two vertices. This is even more likely to happen when a PKN, either extracted from the literature or from one of the mentioned databases, is compressed as described in [9] in order to remove most of the nodes that are neither measured, nor manipulated during the experiments. Also note that the subset of nodes K correspond to those proteins (e.g. kinases) that can be forced to be inactive (inhibited) by various experimental tools such as small-molecule drugs, antibodies or RNAi.

Definition 2 (Experimental condition). *Given a PKN* (V, E, σ) *an experimental condition over* (V, E, σ) *is a function* $\varepsilon : S \cup K \subseteq V \to \{0, 1\}$ *such that if* $v \in S$*, then* $\varepsilon(v) = 1$ *(resp.* 0*) means that the stimuli* v *is present (resp. absent), while if* $v \in K$*, then* $\varepsilon(v) = 1$ *(resp.* 0*) means that the inhibitor for* v *is absent and therefore* v *is not inhibited (resp. the inhibitor for* v *is present and therefore* v *is inhibited).*

Definition 3 (Experimental observation). *Given a PKN* (V, E, σ) *and an experimental condition* ε *over* (V, E, σ)*, an experimental observation under* ε *is a function* $\theta : R(\varepsilon) \subseteq R \to \{0, 1\}$ *such that* $R(\varepsilon)$ *denotes the set of observed readouts under* ε *and if* $v \in R(\varepsilon)$*,* $\theta(v) = 1$ *(resp.* 0*) means that the readout* v *is present (resp. absent) under* ε*.*

Since the phospho-proteomics data used here represents an average across a population of cells, each of which may contain a different number of proteins in active or inactive (1 or 0) state, the values are continuous. Thus, we have to discretize the experimental data somehow in order to fit the previous definition. A simple but yet effective approach is to use a threshold $t = 0.5$ such that values greater than t are set to 1, while values lower that t are set to 0. Other approaches could also be used but, since in this paper we work with discrete *in silico* data, we left this discussion for a future work. Indeed, this paper focuses on comparing the performance of training and formal approaches to the optimization problem, for which *in silico* datasets appear more relevant.

2.3 Protein Signaling Logic Models

Here we state the combinatorial problem as a Constraint Satisfaction Problem (CSP) in order to have a clear and formal definition of a Protein Signaling Logic Model (PSLM) as a solution to this problem. Recall that a CSP is defined by a set of variables X, a domain of values D, and a set of constraints or properties to be satisfied. A solution to the problem is a function $e : X \to D$ that satisfies all constraints [18].

Next, we define two properties that we use later as the constraints of the CSP formulation. The first property defines for a given PKN, the conditions

that must be satisfied by a logical formula in order to define the truth value of any node. For example, if we look the Fig. 1 is quite clear that the hypergraph in (b) is not just some arbitrary hypergraph, but instead is strongly related to the graph in (a). This relation is captured by the following definition.

Definition 4 (PKN evidence property). *Given a PKN (V, E, σ) and $v \in V$, a logical formula φ in Disjunctive Normal Form (DNF) has an evidence in (V, E, σ) with respect to v if and only if for every propositional variable w that occurs positively (resp. negatively) in φ, it exists an edge $(w, v) \in E$ and $((w, v), 1) \in \sigma$ (resp. $((w, v), -1) \in \sigma$).*

The second property identifies those logical formulas in DNF for which exist some equivalent but simpler formula. For example, for two literals X and Y, it is easy to see that $X \vee (X \wedge Y) \equiv X$. In such case we say that $X \vee (X \wedge Y)$ is redundant since, as we will see later, we are interested in minimizing the complexity of the logic models. This concept was previously introduced in [9] as a way to reduce the search space of all possible logical formulas.

Definition 5 (Redundancy property). *Given a logical formula φ in DNF, with $\varphi = \bigvee_{j \geq 1} c_j$ where each c_j is a conjunction clause, φ is a redundant formula if and only if for some $k, l \geq 1$ with $k \neq l$ and some logical conjunction r it holds that $c_k = c_l \wedge r$.*

Now, based on the general form of a CSP given above, and the properties defined in (4) and (5), we define a PSLM as follows.

Definition 6 (Protein Signaling Logic Model). *Given a PKN (V, E, σ) with $V \setminus S = \{v_1, \ldots, v_m\}$ for some $m \geq 1$, let the set of variables $X = \{\psi_{v_1}, \ldots, \psi_{v_m}\}$ and the domain of values D given by all the formulas in DNF having V as the set of propositional variables. Then, a function $e : X \to D$ defines a compatible Protein Signaling Logic Model $\mathcal{B}$ if it holds for $i = 1, \ldots, m$ that $e(\psi_{v_i})$ has an evidence in (V, E, σ) with respect to v_i. Moreover, if it also holds for $i = 1, \ldots, m$ that $e(\psi_{v_i})$ is not redundant, then we say that $\mathcal{B}$ is a non redundant logic model.*

Example 2. Given the PKN in Fig. 1(a) the function that defines the logic model in Fig. 1(b) is given by:

$$e(\psi_v) = \begin{cases} (a \wedge b) \vee \neg c & \text{if } v = d \\ c & \text{if } v = e \\ (d \wedge e) & \text{if } v = f \end{cases}$$

for $\psi_v \in \{\psi_d, \psi_e, \psi_f\}$. Note that in every case, each formula satisfies both properties: PKN evidence (Definition 4) and Non-redundancy (Definition 5).

2.4 Predictive Logic Model

A given Protein Signaling Logic Model (PSLM) describes only the static structure of a Boolean network. Even though Boolean networks are either synchronous or asynchronous, in any case the set of Logical Steady States (LSSs) is the same. Therefore, and since we focus on a Logical Steady State Analysis (LSSA) which

offers a number of applications for studying functional aspects in cellular interactions networks [15], choosing between synchronous and asynchronous is not relevant in this work. Moreover, we do not need to compute all possible LSSs, but only the one that can be reached from a given initial state. Note that the existence of a unique LSS is guaranteed by the assumption of no feedbacks loops in the given PKN. Next, we describe how we compute this LSS in terms of satisfiability of a particular logical formula. This is based on the formalization given by [19] for a related problem named IFFSAT.

Let (V, E, σ) a PKN and $\mathcal{B}$ a compatible PSLM. Recall that $\mathcal{B}$ is defined by a function from every non-stimuli in (V, E, σ) to a DNF formula satisfying the PKN evidence property defined in Definition 4.

First, we define a logical formula $\mathcal{R}$ representing the regulation of every non-stimuli or non-inhibitor node in (V, E, σ).

$$\mathcal{R} = \bigwedge_{v \in V \setminus (S \cup K)} (\mathcal{B}(v) \iff v) \tag{1}$$

Note that we use $(\mathcal{B}(v) \iff v)$ instead of just $(\mathcal{B}(v) \Rightarrow v)$ to enforce that every activation must have a "cause" within the model. Next, we define two logical formulas $\mathcal{S}$ and $\mathcal{K}$ in order to fix the values of *stimulus* and the values or regulations of *inhibitors* in (V, E, σ) under a given experimental condition ε.

$$\mathcal{S} = \bigwedge_{v \in S} \begin{cases} v & \text{if } \varepsilon(v) = 1 \\ \neg v & \text{if } \varepsilon(v) = 0 \end{cases} \qquad \mathcal{K} = \bigwedge_{v \in K} \begin{cases} \mathcal{B}(v) \iff v & \text{if } \varepsilon(v) = 1 \\ \neg v & \text{if } \varepsilon(v) = 0 \end{cases} \tag{2}$$

Thereafter, we look for the truth assignment such that $\mathcal{R} \wedge \mathcal{S} \wedge \mathcal{K}$ evaluates to *true* and which represents the LSS of the network for the given initial conditions. The biological meaning behind this concept is that the input (*stimuli*) signals are propagated through the network by using the faster reactions and after some time, the state of each protein will not change in the future. Thus, we say that the network is stabilized or that it has reached an steady state [15]. Finally, we can define the model prediction under a given experimental condition as follows.

Definition 7 (Model prediction). *Given a PKN (V, E, σ), a PSLM $\mathcal{B}$ compatible with (V, E, σ) and a experimental condition ε over (V, E, σ), the prediction made by $\mathcal{B}$ under the experimental condition ε is given by the truth assignment $\rho : V \to \{0, 1\}$ such that $\mathcal{R} \wedge \mathcal{S} \wedge \mathcal{K}$ evaluates to true.*

Note that without the assumption of no feedbacks loops in the given PKN, the existence of multiple steady states or cycle attractors should be considered. Then, in order to guarantee that ρ is well defined, new constraints should be added to the CSP instance defined in (6), but this is left as a future work.

Example 3. Let $\varepsilon : \{a, b, c, d\} \to \{0, 1\}$ an experimental condition over the PKN given in Fig. 1(a) defined by:

$$\varepsilon(v) = \begin{cases} 1 & \text{if } v \in \{a, b, c\} \\ 0 & \text{if } v = d \end{cases}$$

That is, a, b, c are stimulated while d is inhibited. Then, the prediction made by the PSLM given in Fig. 1(b) under ε, is given by the truth assignment such that the formula

$$((d \wedge e) \iff f) \wedge (c \iff e) \wedge a \wedge b \wedge c \wedge \neg d$$

evaluates to *true*. Thus, e is assigned to 1 and f is assigned to 0.

2.5 Objective function

Given all the PSLMs compatible with a given PKN, our goal is to define an objective function in order to capture under different experimental conditions, the matching between the corresponding experimental observations and model predictions. To this end, we adopt and reformulate the objective function proposed in [9] in terms of our formalization. The objective function represents a balance between fitness of model to experimental data and model size using a free parameter chosen to maximize the predictive power of the model. Of course, other objective functions can be defined in the future, but here we focus on a comparison against one of the state of the art approaches and thus, we choose to use the same objective function.

Before going further, we define the size of a model as follows.

Definition 8 (Size of Protein Signaling Logic Models). *Given a PKN (V, E, σ) and a PSLM $\mathcal{B}$ compatible with (V, E, σ), the size of $\mathcal{B}$ is given by $|\mathcal{B}| = \sum_{v \in V \setminus S} |\mathcal{B}(v)|$ where $|\mathcal{B}(v)|$ denotes the canonical length of logical formulas.*

Example 4. If we consider the PSLM given in Fig. 1(b), its size is given by: $|(a \wedge b) \vee \neg c| + |c| + |(d \wedge e)| = 3 + 1 + 2 = 6$. This can be seen also as the size of the hypergraph, where each hyperedge is weighted by the number of source nodes and the size of the hypergraph is the sum of all weights.

Finally, we define the combinatorial optimization problem of learning PSLMs from experimental observations under several experimental conditions as follows.

Definition 9 (Learning Protein Signaling Logic Models). *Given a PKN (V, E, σ), n experimental conditions $\varepsilon_1, \ldots, \varepsilon_n$ and n experimental observations $\theta_1, \ldots, \theta_n$ with each θ_i defined under ε_i, for a given PSLM $\mathcal{B}$ compatible with (V, E, σ), and n model predictions $\rho_1, \ldots, \rho_n$ over $\mathcal{B}$ with each ρ_i defined under ε_i, we want to minimize*

$$\Theta(\mathcal{B}) = \Theta_f(\mathcal{B}) + \alpha \times \Theta_s(\mathcal{B}) \tag{3}$$

where $\Theta_f(\mathcal{B}) = \frac{1}{n_o} \times \sum_{i=1}^{n} \sum_{v \in R(\varepsilon_i)} (\theta_i(v) - \rho_i(v))^2$ such that n_o, the total number of output measures, is given by $\sum_{i=1}^{n} |R(\varepsilon_i)|$ and $\Theta_s(\mathcal{B}) = \frac{1}{b} \times |\mathcal{B}|$ such that b is the size of the union of all PSLMs compatible with (V, E, σ).

3 Methods

In this section we describe the methods used to perform a comparison between our ASP-based approach and the one presented in [9]. First we provide the ASP implementation that we used to run the experiments and then, we describe the method proposed to systematically generate *in silico* study cases based on realistic networks and data.

3.1 ASP implementation

Our goal is to provide an ASP solution for learning PSLMs from experimental observations under several experimental conditions (Definition 9). Here we provide a logic program representation of the problem described in Section 2 in the input language of the ASP grounder *gringo* [20]. After describing the format of any input instance, we show how we generate non-redundant candidate solutions having an evidence in the given PKN, then we describe how model predictions are made and finally, we show the minimization of the objective function.

Input instance We start by describing the input instance for the PKN given by (V, E, σ), the experimental conditions $\mathcal{E} = \varepsilon_1, \ldots, \varepsilon_n$ and the experimental observations $\mathcal{O} = \theta_1, \ldots, \theta_n$.

$$\begin{aligned}\mathcal{G}((V,E,\sigma),\mathcal{E},\mathcal{O}) =& \{vertex(v) \mid v \in V\} \\ &\cup \{edge(u,v,s) \mid (u,v) \in E, ((u,v),s) \in \sigma\} \\ &\cup \{exp(i,v,s) \mid \varepsilon_i(v) = s, v \in S \cup K\} \\ &\cup \{obs(i,v,s) \mid \theta_i(v) = s, v \in R(\varepsilon_i)\} \\ &\cup \{nexp(n)\} \\ &\cup \{stimuli(v) \mid v \in S\} \\ &\cup \{inhibitor(v) \mid v \in K\} \\ &\cup \{readout(v) \mid v \in R\}\end{aligned} \tag{4}$$

Candidate solutions We follow a common methodology in ASP known as "guess and check" where using non-deterministic constructs, we "guess" candidate solutions and then, using integrity constraints we "check" and eliminate invalid candidates. Since we are interested only on those logical formulas having an evidence in (V, E, σ), first we generate all the possible conjunction clauses having such evidence by computing for every $v \in V$ all the possible subsets between the predecessors of v. This is done by the following rules.

$$\begin{aligned}subset(U, S, null, 1, V) &\leftarrow edge(U, V, S). \\ subset(U, S_U, subset(W, S_W, T), N+1, V) &\leftarrow subset(U, S_U, null, 1, V), \\ &\quad\ subset(W, S_W, T, N, V), \\ &\quad\ vertex(U) < vertex(W).\end{aligned} \tag{5}$$

The idea is to start with the singleton subsets containing only a single predecessor, and to create a bigger subset by recursively extending a singleton subset

with any other subset until a fix point is reached. The first rule defines all the singleton subsets related to V. We represent the subsets here as linked lists where U, the first argument in the predicate $subset/5$, represents the head of the list (5 is the arity of the predicate). The second argument represents the sign of the edge from U to V. The third argument represents the tail of the linked list ($null$ in case of a singleton). The fourth argument represents the subset cardinality, and the last argument keeps track of the target vertex. The head is here used as a identifier such that we can order all subsets. The second rule recursively extends a singular subset identified by head argument U with any subset identified by W as long as $U < W$. We exploit the order between the predicates $vertex/1$ to avoid different permutations of the same subsets.

The following rules define the inclusion relationship between these subsets of predecessors.

$$\begin{aligned} in(U, S, subset(U, S, T)) &\leftarrow subset(U, S, T, N, V). \\ in(W, S_W, subset(U, S_U, T)) &\leftarrow in(W, S_W, T), \\ &\quad subset(U, S_U, T, N, V). \end{aligned} \tag{6}$$

The first rule declares that every subset contains its "head" element. The second rule declares that if W is included in T, and if there is another subset having T as its "tail", then W is also included in it.

Since each subset generated by the rules in (5) represents a possible conjunction clause, we can generate all possible logical formulas in DNF by considering each subset as either present or absent.

$$\begin{aligned} \{clause(subset(U, S, T), N, V)\} &\leftarrow subset(U, S, T, N, V). \\ &\leftarrow clause(C_1, N, V), clause(C_2, M, V), \\ &\quad C_1 \neq C_2, in(U, S, C_2) : in(U, S, C_1). \end{aligned} \tag{7}$$

The first rule is a choice rule that declares the non-deterministic generation of predicates $clause/3$ from a subset. A clause represents the conjunction of all the elements included in the subset. The second rule declares an integrity constraint to avoid the generation of redundant logical formulas by using the predicates generated in (6).

Model predictions Next, we show the representation for the input signals propagation. For each experiment, first the truth values for stimuli and inhibited nodes are fixed and then, truth values are propagated to all nodes by exploiting the fact that in order to assign *true* to any node, it is enough that one conjunction clause over it evaluates to *true*.

$$\begin{aligned} fixed(E, V) &\leftarrow nexp(N), E = 1..N, stimuli(V). \\ fixed(E, V) &\leftarrow inhibitor(V), exp(E, V, 0). \\ active(E, V) &\leftarrow exp(E, V, 1), stimuli(V). \\ inactive(E, V) &\leftarrow exp(E, V, 0). \end{aligned} \tag{8}$$

The first and second rules simply declare which nodes have fixed truth values because they are either an input node, or an inhibited node in a particular

experiment. Thereafter, the third and fourth rules declare the truth assignments that are given by the experimental condition.

The following rules model the signal propagation in every experiment.

$$\begin{aligned} active(E,V) \leftarrow\ & nexp(N), E = 1 \ldots N, \\ & clause(C,M,V), not\ fixed(E,V), \\ & active(E,U) : in(U,1,C), \\ & inactive(E,U) : in(U,-1,C). \\ inactive(E,V) \leftarrow\ & vertex(V), nexp(N), E = 1 \ldots N, \\ & not\ fixed(E,V), not\ active(E,V). \end{aligned} \tag{9}$$

The first rule declares that for each experiment, if there is at least one conjunction clause having all its positive literals assigned to *true* and all its negated literals assigned to *false*, then the complete clause evaluates to *true*. While the second rule declares that every node that is not assigned to *true*, it is assigned by default to *false*.

Optimization Finally, we show the declaration of the objective function. In Section 2 we defined the objective function Θ, but since ASP can only minimize integer functions, we transformed Θ into Θ_{int} trying to lose as less information as possible. To this end, if we assume that the free parameter $\alpha = \frac{N}{D}$ for some $N, D \in \mathbb{N}$, multiplying Θ by $N \times \frac{1}{\alpha} \times n_o \times b$ we define Θ_{int} as follows.

$$\begin{aligned} \Theta_{\text{int}}(\mathcal{B}) &= D \times n_o \times b \times \Theta(\mathcal{B}) \\ &= D \times b \times \sum_{i=1}^{n} \sum_{v \in R(\varepsilon_i)} \left(\theta_i(v) - \rho_i(v)\right)^2 + N \times n_o \times |\mathcal{B}| \end{aligned} \tag{10}$$

This new (integer) objective function is represented as follows in our ASP encoding.

$$\begin{aligned} & \#const\ npenalty = 1. \\ & \#const\ dpenalty = 1000. \\ & penalty_N(npenalty). \\ & penalty_D(dpenalty). \\ & b(B) \leftarrow B = [subset(_,_,N,_) = N]. \\ & n_o(E) \leftarrow E = [obs(_,_,_)]. \\ & mismatch(E,V) \leftarrow obs(E,V,0), active(E,V). \\ & mismatch(E,V) \leftarrow obs(E,V,1), inactive(E,V). \end{aligned} \tag{11}$$

The rules in (11) declare the predicates that we need to give a representation of Θ_{int}. First, we declare two predicates to represent the free parameter α as a fraction of integers. Then, we use a weighted sum to declare the size of the union of all logic models and we count the number of single experimental observations. The two last rules declare in which cases a model prediction does not match the corresponding output measurement.

Last but not least, we require the minimization of the (integer) objective function Θ_{int} simply by using the $\#minimize$ directive.

$$\begin{aligned} \#minimize[\ & mismatch(_,_) : b(B) : penalty_D(PD) = B \times PD, \\ & clause(_,N,_) : n_o(E) : penalty_N(PN) = E \times PN \times N]. \end{aligned} \tag{12}$$

3.2 Benchmark datasets

We wanted to compare the ASP approach with CellNOpt and analyze the scalability of the methods. Also we wanted to determine how the inference of the network is influenced by specific parameters of the problem. For this purpose, we generated meaningful benchmarks that covered a broad range of these influential parameters.

Middle and large-scale benchmark datasets We constructed a middle (see Fig 2) and a large scale (see Fig 3) optimization problem. Both PKNs were derived from literature and in each case we randomly selected compatible PSLMs or hypergraphs (middle: Fig. 2(b), large: Fig. 3(b)), from which we generated *in silico* datasets under several experimental conditions giving place to different numbers of output measures. The main parameters used to compute the objective function for the optimization are shown in the Table 1.

Table 1. Middle and Large optimization problems

Scale	Nodes	Edges	Compatible hyperedges	Size of union hypergraph	Selected hypergraph size	Experimental conditions	Output measures
Middle	17	34	87	162	20	34	210
Large	30	53	130	247	37	56	840

Large set of benchmark datasets We relied on a literature derived PKN for growth and inflammatory signaling [10] to derive compatible PSLMs and generate 240 benchmark datasets with *in silico* observation data. Given the literature derived network (V, E, σ) with $V = S \cup K \cup R \cup U$, we created 4 derivative networks $(V_i, E, \sigma), i = 1 \dots 4$ with $V_i = V$, $V_i = S \cup K_i \cup R_i \cup U_i$, $K_i \subseteq K$, $R_i \subseteq R$, and $U \subseteq U_i$. Each network differing in sets of inhibitors and readouts. For these networks we compressed ('bypassed) nodes that are neither measured, nor manipulated during the experiments, which were not affected by any perturbation, lay on terminal branches or in linear cascades, as described in [9], yielding to 4 compressed networks.

To investigate the influence of the size of the networks in the optimization both in terms of computational times and recovered edges, we randomly selected PSLMs of 3 different sizes (20, 25, or 30) for each compressed network. The size of each model was obtained as defined in Definition 8. Then, 5 different PSLMs were generated for each compressed network and for each size, giving a total of 60 different models (20 of each size). We use these 60 models to run simulated experiments and generate *in silico* experimental observations.

Moreover, we wanted to investigate how the amount of experimental observation data influences the network inference. Therefore, we generated 4 datasets

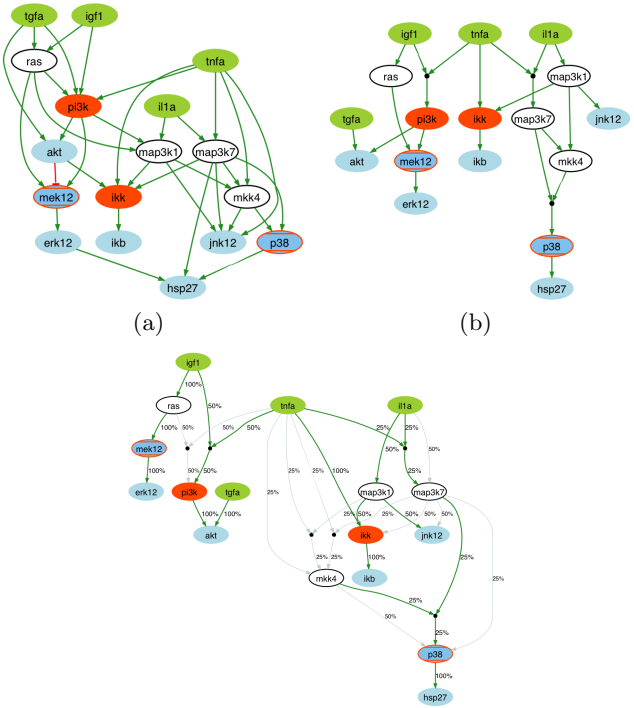


Fig. 2. Input and outputs of a middle-scale optimization problem. (a) A literature-derived Prior Knowledge Network (PKN) of growth and inflammatory signaling. **(b)** An hypergraph which is compatible with the PKN shown in (a). From this model we derive 240 output measures under 34 experimental conditions. **(c)** The ASP optimization enumerated all the minimal PSLMs that predict the *in silico* measures produced in (b) with no mismatches. The union of the 8 optimal models is shown with a specific edge encoding: edges are labeled according to their percentage of occurrence in all the 8 models. The thick green edges correspond to those edges that also appear in the hypergraph used to generate the *in silico* datasets.

$D_1 \dots D_4$ of experimental observations for each model. The first dataset D_1 contained only experimental observations from single-stimulus/single-inhibitor experiments. The other datasets $D_2 \dots D_4$ contained observations from multiple-stimuli/multiple-inhibitors experiments with 30, 50 and 60 experimental conditions respectively. The larger datasets always include the smaller datasets, such that $D_1 \subset D_2 \subset D_3 \subset D_4$. In total we generated 240 different datasets of 4 different sizes, generated from 60 different models of 3 different sizes. The whole method is illustrated in the Fig. 4.

4 Results and discussions

First, we focused in finding minimal PSLMs compatible with the given PKN and predicting the generated dataset for the middle (see Fig. 2) and large-scale (see

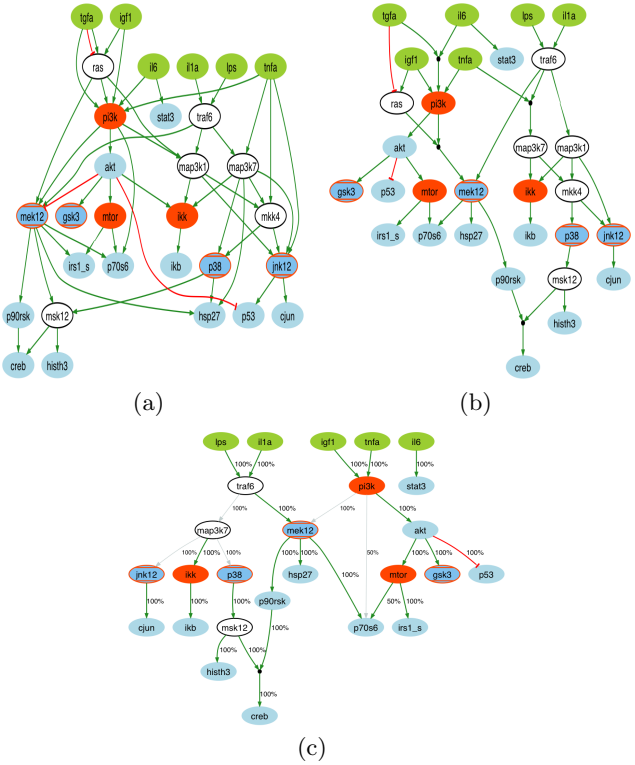


Fig. 3. Large-scale optimization problem. **(a)** A Large literature-derived PKN of growth and inflammatory signaling obtained from [21]. **(b)** An hypergraph which is compatible with the PKN shown in (a). From this model we derive 840 output measures under 56 experimental conditions. **(c)** Union of the two minimal PSLMs predicting the whole dataset with no mismatches.

Fig. 3) benchmarks. Second, general comparisons between our logical approach implemented in ASP and the genetic algorithm implemented in CellNOpt, were performed over the 240 datasets generated as described in Section 3.2.

4.1 Enumeration of solutions to the optimization problems

We used the ASP implementation detailed in Section 3.1 to identify the PSLMs compatible with the middle-scale PKN (respectively the large-scale PKN) having an optimal score with respect to the generated dataset.

Notice that in both cases, by the construction of the datasets, we knew that there exists a compatible PSLM which predicts the whole datasets without mismatches. As a consequence, if $\alpha \in (0, 1)$ (see Eq. 3, Eq. 10), the optimization problem is equivalent to find the logic models with perfect fit and minimal size.

In a first run, the ASP implementation allowed us to compute the minimal score of the optimization problem. Afterwards, we run the ASP solver again to enumerate all the models having a score lower or equal than the minimal score.

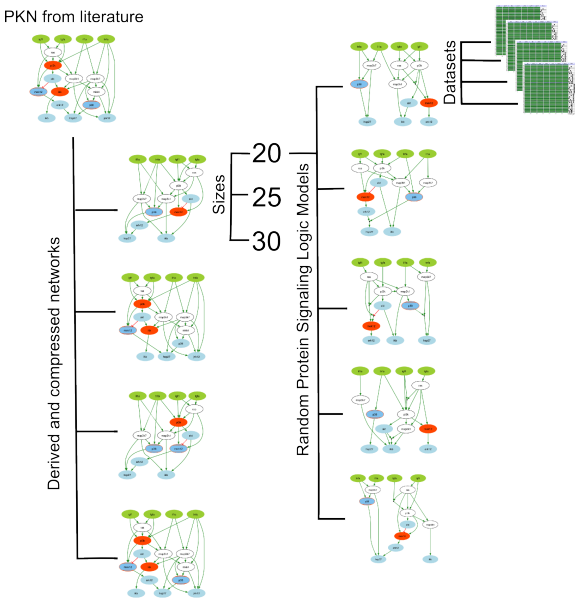


Fig. 4. Pipeline of the generation of the 240 benchmarks

All together, we obtained a complete enumeration of all minimal models. Below, we show the results obtained using the ASP-based approach to solve the middle and large optimization problems.

- **Middle-scale** The minimal score was computed in 0.06 seconds[1]. The enumeration took 0.03 seconds and found 8 global optimal Boolean models with size equal to 16. The union of the 8 optimal models found is shown in Fig. 2(c).
- **Large-scale** The minimal score was computed in 0.4 seconds. The enumeration took 0.07 seconds and found 2 global optimal Boolean models with size equal to 26. In Fig. 3(c) we show the union of the 2 optimal models found.

Both optimization problems were also run with CellNOpt, based on its genetic algorithm (see Materials and Methods section in [9]) performing generations over a population of 500 models[2].

- **Middle-scale** The optimization was run for 9.2 hours and the best score was reached after 7.2 hours (299 generations). During the optimization, 66 Boolean models with perfect fit were found, with sizes going from 16 to 24. Out of the 66 models, only 2 models were minimal (i.e. with size equal to 16).

[1] All ASP computations were run in a MacBook Pro, Intel Core i7, 2.7 GHz and 4 GB of RAM using Gringo 3.0.3 and Clasp 2.0.5 versions.

[2] All CellNOpt computations were run in a cluster of 542 nodes, each with 32 GB of memory, and a total of 9000 cores using CellNOptR 1.0.0.

– **Large-scale** The optimization problem was run for 27.8 hours and the best score was reached after 24.5 hours (319 generations). During the optimization, 206 models with perfect fit were found, with sizes going from 27 to 36. Note that in this case, CellNOpt did not find any of the minimal models (i.e. with size equal to 26).

Our main conclusion here is as follows: in both cases, due to the use of *in silico* data, models with perfect fit were exhibited by both approaches. The main advantage of the formal approach is to be able to explicitly compute the minimal score, allowing us to enumerate all models with this score in a very short time. Meanwhile, genetic algorithms are not able to exhibit this information and therefore cannot develop strategies to compute all minimal models. At the same time, this leads to the question about the biological relevance of optimal models and if it is possible to discriminate between them. A precise study of the biological pathways selected in each optimal model did not allow us to specifically favor one model according to biological evidences. That is why we choose to show the union of them in each case (Fig. 2(c) and Fig. 3(c)).

Nonetheless, the ASP search was strongly supported by the fact that there exists at least one model with perfect fit. This considerably reduces the optimization problem to the search of compatible models with minimal size by canceling the Θ_f term in Eq. (3). Performing optimizations over real data will induce that there will no more exist models with perfect fit, which may have a strong effect over the performance of our formal approach, while for genetic algorithms performances may probably be less affected by real data, but this will have to be studied. An interesting perspective is therefore to test the efficiency of these approaches in a real case experiment.

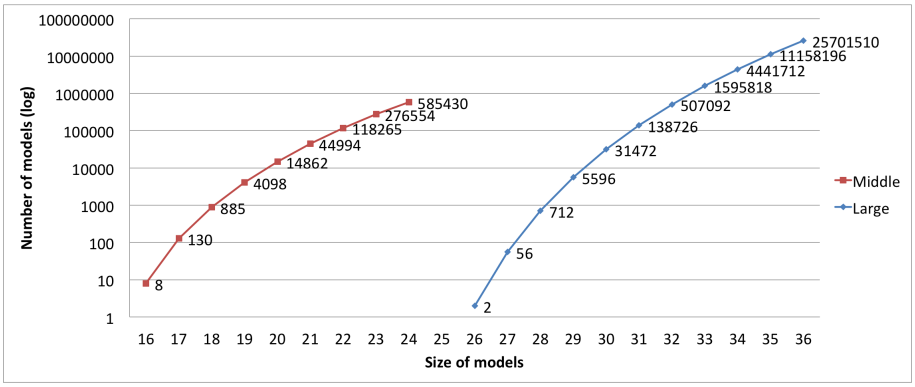
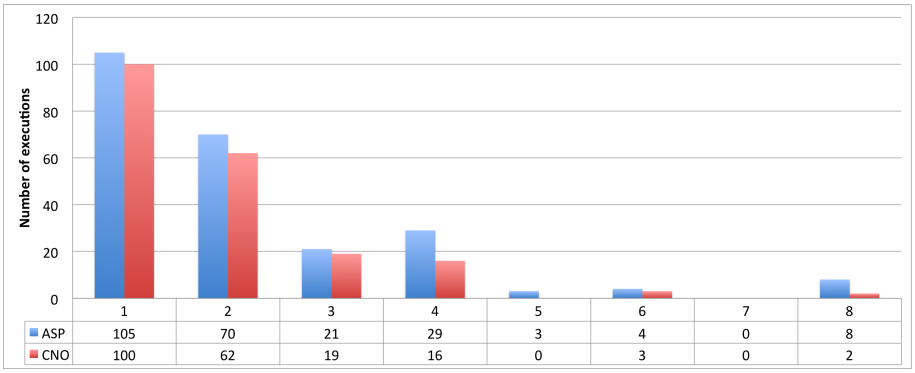


Fig. 5. Number of suboptimal solutions to the middle-scale (red curve) and large-scale (blue curve) optimization problems. Each curve describes the number of models with perfect fit with a given size, where the size ranges from its minimal value to the maximal size of models found by CellNOpt.

4.2 Dependency to the model size

To have a first view of the space of solutions, we investigated the role of the model size over the optimization process. Indeed, the optimization criteria moderates the choice of a model of minimal size -according to a parsimonious principle- by a free parameter related to the fitting between observations and predictions. (see Eq. 3). However, as we mentioned above, in all our experiments we known that there exists at least one model which predicts the whole datasets without mismatches and thus, the optimization problem is focused on finding minimal models. Therefore strongly favoring the size of the model. To evaluate the impact of this for the middle and large optimization problems depicted in Fig. 2 and Fig. 3, we used ASP to enumerate all the models with perfect fit having their size less or equal to the size of the models found by CellNOpt. Results are depicted in Fig. 5, providing a first insight on the structure of the space of compatible PSLMs with perfect fit. It appears that the number of compatible PSLMs increases exponentially with the size of the model. Therefore, optimizing over the size criteria appears quite crucial. A prospective issue is to elucidate whether the topology of the space of suboptimal models informs about the biological relevance of minimal models.



	1	2	3	4	5	6	7	8
ASP	105	70	21	29	3	4	0	8
CNO	100	62	19	16	0	3	0	2

Fig. 6. Executions of ASP and CellNOpt optimizations that found all global optimal models. The total number of runs was of 240 in account of the in-silico data generated. The x-axis represents the number of global optimal models that each problem had. The y-axis, shows the number of executions where ASP and CellNOpt found the total number of global optimums.

4.3 Accuracy of predictions

The study of the middle and large optimization problems evidenced that genetic algorithms may not find all minimal models. In order to elucidate whether this phenomenon is frequent, we used the 240 benchmark datasets generated with the method described in Section 3.2. In Fig. 6 we show the number of executions of the optimization process for ASP and CellNOpt where both approaches found

the complete set of global optimal (minimal) models. Recall that ASP ensures finding the complete set of global optimal models (blue bars in Fig. 6) while this is not the case for CellNOpt. We observed that in 202 executions out of 240 (84%), ASP and CellNOpt both found all the minimal models. This is particularly clear in the 105 executions with a single minimal model, which was found by CellNOpt in 95% of executions. Nonetheless, in the 44 cases with more than 4 optimal models, CellNOpt found all optimal models in only 47% cases. More generally, as the number of minimal solutions to the optimization problem increases, the percentage of minimal solutions identified by CellNOpt decreases.

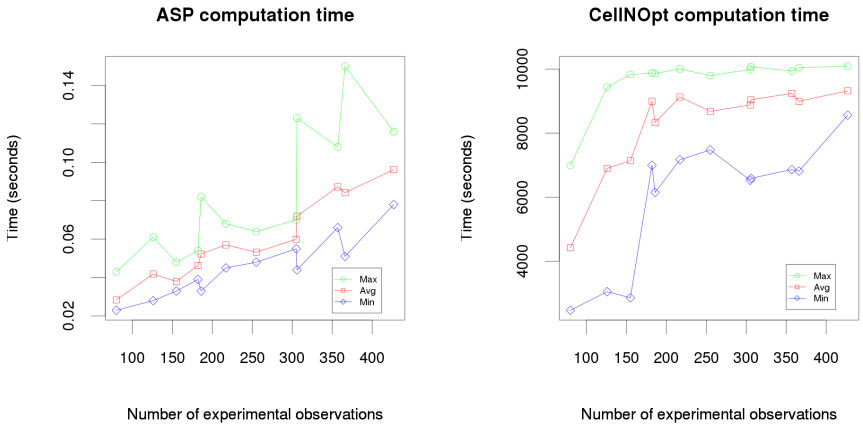


Fig. 7. Computation time of ASP and CellNOpt with respect to the number of experimental observations. The x-axis is the number of experimental observations: sum of number of readouts for each experimental condition. The maximum, average, and minimum computation times are plotted in green, red, and blue respectively.

4.4 Computation times

In Fig. 7 we plot the computation time evolution for ASP and CellNOpt with respect to the number of experimental observations (i.e. output measures) included in the *in silico* datasets used to run the optimizations. Since we generated multiple datasets which contained the same number of experimental observations, for each optimization related to these multiple datasets we obtained minimum, maximum, and average times. We observe that the ASP computation times are in a range that goes from 0.02 to 0.15 seconds, while CellNOpt computation times to find the best score goes from 43 minutes to 2.7 hours, which was set as the limit running time. We see from these results that ASP outperforms CellNOpt in 5 order of magnitude guaranteeing in all cases global optimality. As discussed in a previous subsection, the main prospective issue is to test the relevance of this conclusion when optimizing with real data instead of *in silico* data.

5 Conclusion

We have proposed a formal encoding of a combinatorial optimization problem related to the inference of Boolean rules describing protein signaling networks. We have used ASP, a declarative problem solving paradigm, to solve this optimization problem and compared its performance against the stochastic method implemented by CellNOpt. Our ASP formulation relies on powerful, state-of-the-art and free open source software [20,12]. As main conclusion, we prove that our ASP-based approach ensures to find all optimal models by reasoning over the complete solution space. Moreover, in the experiments presented in this work, ASP outperforms CellNOpt in up to 5 orders of magnitude.

Our analyses provide concrete illustrations of the potential applications, in our opinion under-explored, of ASP in this field. Recently, Integer Linear Programming (ILP) have been used to solve the same problem that we described here [13]. In principle, ILP solvers can also provide the complete set of optimal solutions but a detailed comparison between ASP and ILP for this particular problem remains to be done.

As discussed within the results section, several prospective issues shall now be investigated. We first have to study the robustness of our results when optimizing over real networks and datasets. Second, we shall develop tools to explore the topology of the space of suboptimal models in order to gain in biological relevance in the inference process and try to elucidate whether this topology informs about the biological relevance of minimal models. Finally, by considering the presence of feedbacks loops in the input PKN and by studying the effect of different discretization approaches, we hope to improve the state of the art in protein signaling network inference and offer a useful tool for biologists.

Acknowledgements

The work of the first author was supported by the project ANR-10-BLANC-0218. The work of the second author was financed by the BMBF MEDSYS 0315401B. The work of the third author was partially supported by the 'Borsa Gini' scholarship, awarded by 'Fondazione Aldo Gini', Padova, Italy.

References

1. Cerami, E.G., Gross, B.E., Demir, E., Rodchenkov, I., Babur, O., Anwar, N., Schultz, N., Bader, G.D., Sander, C.: Pathway Commons, a web resource for biological pathway data. Nucleic Acids Research **39**(Database issue) (2011) D685–D690
2. Schaefer, C.F., Anthony, K., Krupa, S., Buchoff, J., Day, M., Hannay, T., Buetow, K.H.: PID: the Pathway Interaction Database. Nucleic Acids Research **37**(Database issue) (2009) D674–D679
3. Zinovyev, A., Viara, E., Calzone, L., Barillot, E.: BiNoM: a Cytoscape plugin for manipulating and analyzing biological networks. Bioinformatics **24**(6) (Mar 2008) 876–877

4. Guziolowski, C., Kittas, A., Dittmann, F., Grabe, N.: Automatic generation of causal networks linking growth factor stimuli to functional cell state changes. FEBS Journal (2012)
5. Palmisano, G., Thingholm, T.E.: Strategies for quantitation of phosphoproteomic data. Expert Review Of Proteomics **7**(3) (2010) 439–456
6. Terfve, C., Saez-Rodriguez, J.: Modeling Signaling Networks Using High-throughput Phospho-proteomics. Advances in experimental medicine and biology **736** (2012) 19–57
7. Bansal, M., Belcastro, V., Ambesi-Impiombato, A., di Bernardo, D.: How to infer gene networks from expression profiles. Mol. Syst. Biol. **3** (2007) 78
8. Hecker, M., Lambeck, S., Toepfer, S., Van Someren, E., Guthke, R.: Gene regulatory network inference: data integration in dynamic models-a review. Bio Systems **96**(1) (2009) 86–103
9. Saez-Rodriguez, J., Alexopoulos, L.G., Epperlein, J., Samaga, R., Lauffenburger, D.A., Klamt, S., Sorger, P.K.: Discrete logic modelling as a means to link protein signalling networks with functional analysis of mammalian signal transduction. Molecular Systems Biology **5**(331) (2009) 331
10. Prill, R.J., Saez-Rodriguez, J., Alexopoulos, L.G., Sorger, P.K., Stolovitzky, G.: Crowdsourcing network inference: the DREAM predictive signaling network challenge. Sci Signal **4**(189) (Sep 2011) mr7
11. Baral, C.: Knowledge Representation, Reasoning and Declarative Problem Solving. Cambridge University Press (2003)
12. Gebser, M., Kaufmann, B., Neumann, A., Schaub, T.: Conflict-driven answer set solving. (2007) 386–392
13. Mitsos, A., Melas, I., Siminelakis, P., Chairakaki, A., Saez-Rodriguez, J., Alexopoulos, L.G.: Identifying Drug Effects via Pathway Alterations using an Integer Linear Programming Optimization Formulation on Phosphoproteomic Data. PLoS Comp. Biol. **5**(12) (September 2009) e1000591
14. Klamt, S., Haus, U.U., Theis, F.J.: Hypergraphs and Cellular Networks. PLoS Comput Biol **5**(5) (May 2009) e1000385
15. Klamt, S., Saez-Rodriguez, J., Lindquist, J., Simeoni, L., Gilles, E.: A methodology for the structural and functional analysis of signaling and regulatory networks. BMC Bioinformatics **7**(1) (2006) 56
16. Saez-Rodriguez, J., Simeoni, L., Lindquist, J., Hemenway, R., Bommhardt, U., Arndt, B., Haus, U.U., Weismantel, R., Gilles, E., Klamt, S., Schraven, B.: A Logical Model Provides Insights into T Cell Receptor Signaling. PLoS Comput Biol **3**(8) (August 2007) e163
17. Christensen, T.S., Oliveira, A.P., Nielsen, J.: Reconstruction and logical modeling of glucose repression signaling pathways in Saccharomyces cerevisiae. BMC systems biology **3** (2009) 7–7
18. Tsang, E.: Foundations of constraint satisfaction. Academic Pr (1993)
19. Haus, U.U., Niermann, K., Truemper, K., Weismantel, R.: Logic integer programming models for signaling networks. J Comput Biol **16**(5) (May 2009) 725–743
20. Gebser, M., Kaminski, R., Ostrowski, M., Schaub, T., Thiele, S.: On the Input Language of ASP Grounder Gringo. Volume 5753 of Lecture Notes in Computer Science. Springer Berlin Heidelberg, Berlin, Heidelberg (2009)
21. Morris, M.K., Saez-Rodriguez, J., Clarke, D.C., Sorger, P.K., Lauffenburger, D.A.: Training signaling pathway maps to biochemical data with constrained fuzzy logic: quantitative analysis of liver cell responses to inflammatory stimuli. PLoS Comput. Biol. **7**(3) (Mar 2011) e1001099

Author contributions

JSR and CG started the project. SV proposed the logic formalization and built the ASP encoding, supported by ST. SV performed ASP experimentations, advised and coordinated by AS. FE conducted the benchmark generation and performed CNO experimentations, advised and coordinated by JSR and CG. Comparisons were coordinated by CG. All were involved in the writing.