Reasoning on the response of logical signaling networks with Answer Set Programming

Torsten Schaub, Anne Siegel, Santiago Videla

Contents

Chapter 1

Reasoning on the response of logical signaling networks with Answer Set Programming

Logical networks provide a simple yet powerful qualitative modeling approach in systems biology. In what follows, we focus on modeling the response of logical signaling networks by means of automated reasoning using Answer Set Programming (ASP). In this context, the problem consisting of learning logical networks is crucial in order to achieve unbiased and robust discoveries. Furthermore, it has been shown that many networks can be compatible with a given set of experimental observations. Thus, first we discuss how ASP can be used to exhaustively enumerate all these logical networks. In practice, this is a key step for the design of new experiments in order to reduce the uncertainty provided by such a family of models. Next, in order to gain control over the system, we look for intervention strategies that force a set of target species into a desired steady state. Altogether, this constitutes a pipeline for reasoning on logical signaling networks providing robust insights to systems biologists. In this chapter, we illustrate the usage of ASP for solving the aforementioned problems and discuss the novelty of our approach with respect to existing methods.

1.1. Introduction

Systems biology and signaling networks

Systems biology is an interdisciplinary field aiming at the investigation and understanding of biology at a system and multi-scale level [IDE 01, KIT 02]. After biological entities have been identified in a specific environment, it remains to elucidate

Chapter written by Torsten SCHAUB, Anne SIEGEL and Santiago VIDELA.

how they interact with each other in order to carry out a particular biological function. Thus, rather than focusing on the components themselves, one is interested in the nature of the links that connect them and the functionalities arising from such interactions. Notably, advances on high-throughput experimental technologies have been one of the main driving forces of systems biology. Such technologies have allowed biologists to study biological systems as a whole rather than as individual components. Nevertheless, the "reductionist" approach of molecular biology has been fundamental for the construction of the large catalogues of biological entities available nowadays. In fact, some authors have considered systems biology not as a new field of research, but instead as an approach to biomedical research combining "reductionist" and "integrationist" techniques [KOH 10].

As it is often the case, the application of novel technologies has led to profound conceptual and philosophical changes in biology. From the early days of molecular biology existed the idea that the DNA sequence dictates most of cell actions, as do the instructions in a computer program. Recently, together with the advent of systems biology, such a mechanistic understanding has been strongly revisited. Instead, an informatic perspective on the role of the genome has been established. From this point of view, the focus is on what the cell does with and to its genome products rather than on what the genome directs the cell to execute [SHA 09]. Then, for any biological system, one can envision at least a three-way interaction between DNA products, the environment and the phenotype [KOH 10]. In this scheme, the group of entities mediating between such interactions are the so-called biological networks. Deciphering the functioning of these complex networks is the central task of systems biology. Importantly, in order to cope with the increasing complexity of large-scale networks, mathematical and computational modeling is required. Hence, the development of such modeling approaches is a major goal in the field.

From the early millennium, many efforts have been made to develop relevant formalisms and modeling frameworks to take into account the specificities of complex biological systems. Among them, one can distinguish between mathematical and computational modeling approaches [FIS 07]. Essentially, mathematical (or quantitative) models are based on denotational semantics, that is, models are specified by mathematical equations describing how certain quantitites change over time. On the other hand, computational (or qualitative) models are based on operational semantics, that is, models are specified in terms of a sequence of steps describing how the states of an abstract machine relate to each other (not necessarily deterministically). Notably, each type of models provides a different level of abstraction enabling to address different kinds of questions. In fact, hybrid modeling precisely aims at exploiting the best of both worlds whenever possible. In any case, it is clear that intuition is not enough to face the complexity of large-scale biological systems. Thus, systematic and elaborated methodological tools are required by (systems) biologists. Moreover, the development of such modeling frameworks is leading to a hypothesis-driven research in biology [IDE 01, KIT 02]. At first, due to the lack of information, multiple

hypotheses are usually generated from prior knowledge by either mathematical, computational or hybrid modeling. Next, decision-making methods can be used to suggest new experiments in order to reduce ambiguous hypotheses [KRE 09]. Finally, new experimental data is produced to test the generated hypotheses, models are refined, and the loop is started over again. Interestingly, to some extent, this iterative process could be automatized allowing an autonomous scientific discovery [SPA 10].

Among the biological networks mediating between genes, the environment, and the phenotype, signal transduction networks are crucial for the understanding of the response to external and internal perturbations. To be more precise, signal transduction occurs when an extracellular signaling molecule binds to a specific cell surface receptor protein. Such a binding causes a conformational change in the receptor that initiates a sequence of reactions leading to a specific cellular response such as growth, survival, apoptosis (cell death), and migration. Post-translational modifications, notably protein phosphorylation, play a key role in signaling. Importantly, signaling networks are involved in biomedical processes and their control has a crucial impact on drug target identification and diagnosis. Nowadays, there exist public repositories such as Pathways Commons [CER 11], Pathways Interaction Database [SCH 09a], and KEGG [KAN 10] that contain curated knowledge about intracellular causal molecular interactions, from which canonical cell signaling networks can be retrieved [GUZ 12]. Such 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 fact, there is emerging experimental evidence that the combinatorics of interactions between cellular components in signaling networks is a primary mechanism for generating highly specialized biological systems [PAP 05]. 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 combinatorial perturbations [ALE 10]. Therefore, moving beyond the aforementioned causal and canonical interactions towards mechanistic and specialized descriptions of signaling networks is a major challenge in systems biology. Importantly, cellular signaling networks operate over a wide range of timescales (from fractions of a seconds to hours). Thus, taking this into account often leads to significant simplifications [PAP 05, MAC 12].

Finally, we conclude this brief introduction to systems biology and signaling networks by noting that biological functionality is multilevel [NOB 10]. That is, signaling networks by no means could function in isolation from metabolic and regulatory

processes. Hence, integrative modeling approaches considering these multiple levels of causation pose indeed a long-term goal in the field.

Logical signaling networks: state of the art and current challenges

Among various discrete dynamical systems, Boolean networks [KAU 69] provide a relatively simple modeling approach yet able to capture interesting and relevant behaviors in the cell. Especially in the case of poorly understood biological systems where quantitative (fine-grained) information is often scarce and hard to obtain. In fact, numerous successful examples together with comprenhesive methodological reviews can be found in [RÉK 08, MOR 10, WAN 12, SAA 13]. In particular, it has been shown that the response in signaling networks can be appropriately modeled with Boolean networks, as illustrated on several signal transduction pathways involved in diverse processes such as proliferation, cell cycle regulation, apoptosis, or differentiation [SAE 07, SAM 09, SAE 11, CAL 10]. Nonetheless, a large majority of the authors working with (Boolean) logic-based models, rely on *ad-hoc* methods to build their models. That is, in most cases, Boolean networks are constructed manually based on information extracted from biological literature and available experimental data. Notably, the manual identification of logic rules underlying the system being studied is often hard, error-prone, and time consuming. In this context, researchers typically aim at modeling a given biological system by means of one Boolean network only. Afterwards, dynamical and structural analysis (often computationally demanding) are conducted over *the* model. Clearly, biological insights and novel hypotheses resulting from these analysis may be incomplete, incorrect or biased if the model was not precise enough. Therefore, automated learning of (Boolean) logic models is required in order to achieve unbiased and robust discoveries.

Reverse engineering in systems biology consists of building mathematical and computational models of biological systems based on the available knowledge and experimental data. Towards the construction of predictive models, one can convert the generic prior knowledge (for instance, canonical cell signaling networks) into a quantitative or qualitative model (for example, a set of differential equations or a set of logic rules) that can be simulated or executed. Next, if enough experimental data is available, the model can be fitted to it in order to obtain the most plausible models for certain environmental conditions or specific cell type. This is normally achieved by defining an objective fitness function to be optimized [BAN 08]. Optimization over quantitative modeling leads to continuous optimization problems. On the other hand, reverse engineering considering qualitative models typically give rise to combinatorial (discrete) optimization problems. Such inferences (or learning) of either quantitative or qualitative models, have been successfully applied for regulatory, signaling, and metabolic networks. Notably, it represents a very active area of research as illustrated by the successive "DREAM" challenges [STO 07].

In the context of logic-based models, the inference of Boolean (genetic) networks from time-series gene expression data has been addressed by several authors under different hypotheses and methods [LIA 98, AKU 00, IDE 00, LÄH 03]. Recently, a brief review and evaluation of these methods has been published in [BER 13]. Importantly, methods for reverse engineering of biological systems are highly dependent on available (amount of) data, prior knowledge and modeling hypotheses. In particular, reverse engineering of Boolean logic models by confronting prior knowledge on causal interactions with phosphorylation activities has been first described in [SAE 09]. A genetic algorithm implementation was proposed to solve the underlying optimization problem, and a software was provided, CellNOpt [TER 12]. 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 some of this shortcomings, mathematical programming approaches were presented in [MIT 09, SHA 12]. Notably, authors in [SAE 09] have shown that the model is very likely to be non-identifiable when we consider the experimental error from measurements. Hence, rather than looking for *the* optimum Boolean model, one is interested in finding (nearly) optimal models within certain tolerance. Interestingly, in the context of quantitative modeling, authors in [CHE 09] have elaborated upon the same argument. Clearly, an exhaustive enumeration of (nearly) optimal solutions would allow for identifying admissible Boolean logic models without any methodological bias. Importantly, previous methods, namely stochastic search and mathematical programming, are not able to cope with this question. Moreover, all subsequent analysis will certainly profit from having such a complete characterization of feasible models. For example, finding "key-players" in signaling networks leading to potential therapeutic interventions is a highly relevant question for cancer research and biomedicine in general. In the context of logic-based models for signaling networks, this question has been recently addressed by [ABD 08, SAM 10, WAN 11, LAY 11]. Despite the specific problem settings and computational approaches in each work, all of them consider a single logic model describing the system. Moreover, due to computational limitations they are restricted to look for a small number (or even single) interventions. Notably, interventions allowing for accomplish certain goals in a given model are very likely to fail in another model which may describe the system similarly well. Therefore, being able to address the same question but considering a family of feasible models may lead to more robust solutions. In fact, this is in line with recent work showing that an ensemble of models often yields more robust predictions than each model in isolation [KUE 07, MAR 12]. In this context, there is an increasing demand of more powerful computational methods in order to achieve robust discoveries in systems biology.

Authors contribution

In this chapter, we present a generic, flexible, and unified framework for modeling logical networks and perform automated reasoning on their response. More precisely,

we characterize the response of logical networks under either two- or three-valued logics based on fixpoint semantics. Furthermore, a representation using Answer Set Programming (ASP; [BAR 03]) is provided. ASP is a declarative problem solving paradigm, in which a problem is encoded as a logic program such that its so-called answer sets represent solutions to the problem. Furthermore, some of the key-aspects of ASP like its ease to express defaults and reachability are particularly relevant for dealing with biological networks. Importantly, available systems nowadays provide a rich yet simple modeling language, high-performance solving capacities, and automated reasoning modes [GEB 12a]. Altogether, ASP provides a powerful computational framework for addressing hard combinatorial problems in systems biology by reasoning over the complete search space. Notably, our ASP representation can be easily elaborated in order to consider specific problem settings as we illustrate in subsequent sections. In particular, in the context of signaling networks, we address the problem consisting of learning Boolean logic models of immediate-early response and the problem consisting of finding minimal intervention strategies in logical networks.

The remainder of the chapter is structured as follows: Section 1.2 introduces ASP and its main features; Section 1.3 describes our framework for modeling logical networks and two specific problems with their corresponding modeling and solving using ASP systems; and Section 1.4 concludes and poses prospective challenges.

1.2. Answer Set Programming at a glance

Answer Set Programming (ASP; [BAR 03, GEB 12a]) provides a declarative framework for modeling combinatorial problems in Knowledge Representation and Reasoning. 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 an 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* [GEB 12b] 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 into a propositional logic program by means of efficient database techniques. 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 but other choices, like computing unsatisfiable cores, are provided as well. The enumeration of all optimal models is done

via the option `--opt-mode=optN`. 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 rather demanding (because of its exhaustive nature). Then, all models possessing the optimum score are enumerated. Notice that this way one can enumerate all (strictly) optimal solutions. Nonetheless, we are often interested in (nearly) optimal answer sets as well. For a concrete example on how we address this in practice, we refer the reader to the encoding provided in Listing 1.8 and its solving in Listing 1.10.

Our encodings are written in the input language of *gringo* 4 series. Such a language implements most of the so-called *ASP-Core-2* standard. [1] In what follows, we introduce its basic syntax and we refer the reader to the available documentation 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. The connectives `:-` and `,` can be read as *if* and *and*, respectively. Furthermore, 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. An atom preceded with default negation, `not`, is satisfied unless the atom is found to be true. In ASP, the semantics of a logic program is given by the *stable models semantics* [GEL 88]. Intuitively, the head of a rule has to be true whenever all its body literals are true. This semantics requires that each true atom must also have some derivation, that is, 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.

We end this quick introduction by three language constructs particularly interesting for our encodings. 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 an answer set, 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

$$l : l_1, \ldots, l_n$$

1. `http://www.mat.unical.it/aspcomp2013/ASPStandardization`

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 (multiple) cost functions in terms of a weighted sum of elements subject to minimization and/or maximization. Such objective functions are expressed in *gringo* 4 in terms of (several) optimization statements of the form

$$\texttt{\#opt}\{w_1@l_1, t_{1_1}, \ldots, t_{m_1} : b_1, \ldots, b_{n_1}; \ldots; w_k@l_k, t_{1_k}, \ldots, t_{m_k} : b_{1_k}, \ldots, b_{n_k}\}.$$

where $\#opt \in \{\text{“}\#minimize\text{”}, \text{“}\#maximize\text{”}\}$, $w_i, l_i, t_{1_i}, \ldots, t_{m_i}$ are terms and $b_{1_i}, \ldots, b_{n_i}$ are literals for $k \geq 0, 1 \leq i \leq k, m_i \geq 0$ and $n_i \geq 0$. Furthermore, w_i and l_i stand for an integer *weight* and priority *level*. Priorities allow for representing lexicographically ordered optimization objectives, greater levels being more significant than smaller ones.

Answer Set Programming for Systems biology

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 [BAR 04]. 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 [DUR 11]. Gebser et al. have addressed the problem consisting of detecting inconsistencies and repairing in large biological networks [GEB 11b, GEB 10]. Fayruzov et al. have used ASP to represent the dynamics in Boolean networks and find their attractors [FAY 11]. Ray et al. have integrated numerical and logical information in order to find the most likely states of a biological system under various constraints [RAY 12]. Furthermore, Ray et al. have used an ASP system to propose revisions to metabolic networks [RAY 10]. Papatheodorou et al. have used ASP to integrate RNA expression with signaling pathway information and infer how mutations affect ageing [PAP 12]. Finally, Schaub and Thiele have first investigated the metabolic network expansion problem with ASP [SCH 09b] and recently, their work has been extendend and applied in a real-case study by Collet et al. [COL 13]. Altogether, this series of contributions illustrates the potential of ASP to address combinatorial search and optimization problems appearing in the field. Nonetheless, its strictly discrete nature poses interesting challenges for future work towards hybrid reasoning system allowing for qualitative and quantitative modeling.

$a \wedge b$	b	
	$\boldsymbol{t}$	$\boldsymbol{f}$
a $\boldsymbol{t}$	$\boldsymbol{t}$	$\boldsymbol{f}$
a $\boldsymbol{f}$	$\boldsymbol{f}$	$\boldsymbol{f}$

$a \vee b$	b	
	$\boldsymbol{t}$	$\boldsymbol{f}$
a $\boldsymbol{t}$	$\boldsymbol{t}$	$\boldsymbol{t}$
a $\boldsymbol{f}$	$\boldsymbol{t}$	$\boldsymbol{f}$

a	$\neg a$
$\boldsymbol{t}$	$\boldsymbol{f}$
$\boldsymbol{f}$	$\boldsymbol{t}$

Table 1.1. *Truth tables for classical (Boolean) logic.*

1.3. Learn and control logical networks with ASP

1.3.1. *Preliminaries*

Propositional logic and mathematical notation

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 (partial) truth assignments over V mapping formulas to truth values $\{\boldsymbol{t}, \boldsymbol{f}, \boldsymbol{u}\}$ according to Kleene's semantics [KLE 50]. Clearly, two-valued assignments are restricted to range $\{\boldsymbol{t}, \boldsymbol{f}\}$ according to classical (Boolean) logic semantics. We recall the truth tables for classical (Boolean) and Kleene's logics in Table 1.1 and Table 1.2, respectively.

Let $f : X \to Y$, be a (partial) function mapping values $x \in X' \subseteq X$ to values $y \in Y$. We denote the set of values x such that $f(x)$ is defined, i.e. X', with $dom(f)$. We sometimes represent mappings extensionally as sets, viz. $\{x \mapsto f(x) \mid x \in dom(f)\}$, for checking containment, difference, etc. To avoid conflicts when composing truth assignments, we define $A \circ B = (A \setminus \overline{B}) \cup B$ where $\overline{B} = \{v \mapsto \overline{s} \mid v \mapsto s \in B\}$ and $\overline{\boldsymbol{t}} = \boldsymbol{f}$, $\overline{\boldsymbol{f}} = \boldsymbol{t}$, $\overline{\boldsymbol{u}} = \boldsymbol{u}$.

1.3.2. *Reasoning on the response of logical networks*

Logical networks

A *logical network* consists of a finite set V of propositional variables and a (partial) function ϕ mapping a variable $v \in V$ to a propositional formula $\phi(v)$ over V.

$a \wedge b$	b		
	$\boldsymbol{t}$	$\boldsymbol{f}$	$\boldsymbol{u}$
a $\boldsymbol{t}$	$\boldsymbol{t}$	$\boldsymbol{f}$	$\boldsymbol{u}$
a $\boldsymbol{f}$	$\boldsymbol{f}$	$\boldsymbol{f}$	$\boldsymbol{f}$
a $\boldsymbol{u}$	$\boldsymbol{u}$	$\boldsymbol{f}$	$\boldsymbol{u}$

$a \vee b$	b		
	$\boldsymbol{t}$	$\boldsymbol{f}$	$\boldsymbol{u}$
a $\boldsymbol{t}$	$\boldsymbol{t}$	$\boldsymbol{t}$	$\boldsymbol{t}$
a $\boldsymbol{f}$	$\boldsymbol{t}$	$\boldsymbol{f}$	$\boldsymbol{u}$
a $\boldsymbol{u}$	$\boldsymbol{t}$	$\boldsymbol{u}$	$\boldsymbol{u}$

a	$\neg a$
$\boldsymbol{t}$	$\boldsymbol{f}$
$\boldsymbol{f}$	$\boldsymbol{t}$
$\boldsymbol{u}$	$\boldsymbol{u}$

Table 1.2. *Truth tables for Kleene's logic.*

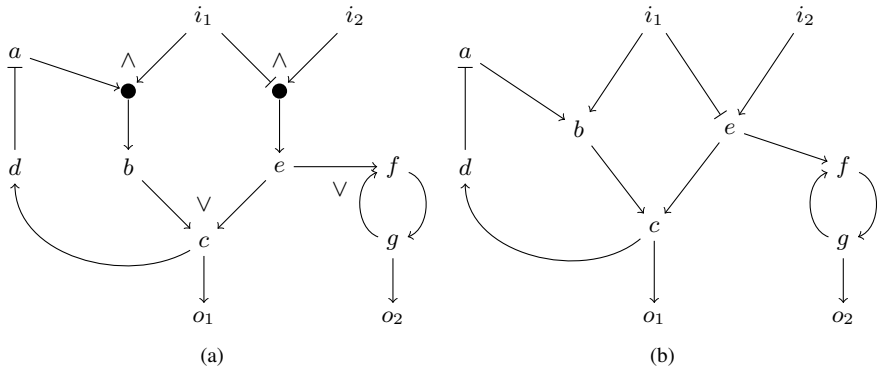

Figure 1.1. *Graphical representaion for logical networks and interaction graphs. Vertices represent biological entities and the interactions among them are represented as follows. Positive interactions are represented by an arrow (→) whereas negative interactions are represented by a T-shape (⊣).* **(a)** *Exemplary logical network represented as a directed hypergraph. Directed hyperedges describe logical interactions.* **(b)** *Interaction graph underlying the logical network in (a). Directed edges describe causal interactions.*

The *logical steady states* of (V, ϕ) are given by truth assignments yielding identical values for v and $\phi(v)$ for all $v \in dom(\phi)$. Generally speaking, such logical networks can be seen as synchronous Boolean networks [KAU 69]. However, since we consider both, two- and three-valued logics, we refrain from using the term "Boolean". Without loss of generality, we assume only formulas in *disjunctive normal form*.[2] For illustration, let us consider the logical network consisting of the set V of species variables $\{i_1, i_2, a, \ldots, g, o_1, o_2\}$ along with the function ϕ defined as:

$$\phi = \left\{ \begin{array}{llllllll} a & \mapsto & \neg d & \quad d & \mapsto & c & \quad g & \mapsto & f \\ b & \mapsto & a \wedge i_1 & \quad e & \mapsto & \neg i_1 \wedge i_2 & \quad o_1 & \mapsto & c \\ c & \mapsto & b \vee e & \quad f & \mapsto & e \vee g & \quad o_2 & \mapsto & g \end{array} \right\}$$

Note that ϕ leaves the specification of the variables i_1 and i_2 undefined. Furthermore, we represent logical networks as (signed) directed hypergraphs as shown in Figure 1.1(a). A *(signed) directed hypergraph* is defined by a pair (V, H) with vertices V and (signed) directed hyperedges H; a (signed) directed hyperedge is a pair (S, t) where S is a finite, non-empty set of pairs (v_i, s_i) with $v_i \in V, s_i \in \{1, -1\}$ and $t \in V$.[3] Then, we say that the (signed) directed hypergraph (V, H) *represents* the logical network (V, ϕ) if and only if for every $v \in dom(\phi)$ and variable $w \in V$ that occurs positively (resp. negatively) in some conjunct ψ of $\phi(v)$, there is a hyperedge

2. Also known as *sum-of-products* [KLA 06b].

3. More generally, a directed hyperedge is a pair (S, T) with $T \subseteq V$. We consider the particular case where T is a singleton. Directed hypergraphs are sometimes referred to as "AND/OR graphs" [GAL 93]

(S_ψ, v) with $(w, 1) \in S_\psi$ (resp. $(w, -1) \in S_\psi$); and vice versa. Following the example shown in Figure 1.1(a), if we consider the mapping $\phi(e) = \neg i_1 \wedge i_2$, we need to verify the existence of the hyperedge $(S_{\neg i_1 \wedge i_2}, e)$ with $S_{\neg i_1 \wedge i_2} = \{(i_1, -1), (i_2, 1)\}$. Similarly, for the mapping $\phi(c) = b \vee e$, we need to verify the existence of the hyperedges (S_b, c) with $S_b = \{(b, 1)\}$ and (S_e, c) with $S_e = \{(e, 1)\}$.

Next, we introduce the notion of the interaction graph underlying a logical network (V, ϕ). An *interaction graph* (V, E, σ) is a signed and directed graph with vertices V, directed edges $E \subseteq V \times V$ and signature $\sigma \subseteq E \times \{1, -1\}$. Moreover, we say that $\Sigma_{(V,\phi)} = (V, E, \sigma)$ is the *underlying interaction graph* of (V, ϕ) if for every edge $(v, w) \in E$ with $((v, w), 1) \in \sigma$ (resp. $((v, w), -1) \in \sigma$), the variable v occurs positively (resp. negatively) in the formula $\phi(w)$. Note that there is a *one-to-many* relation in the sense that the same graph (V, E, σ) corresponds to the underlying interaction graph $\Sigma_{(V,\phi)}$ for possibly many logical networks (V, ϕ). Now, we can rely on standard notions from graph theory to capture several concepts on logical networks. Recall that a path in a graph is a sequence of edges connecting a sequence of vertices. The length of a path is given by the number of edges whereas its sign is the product of the signs of the traversed edges. Herein, we consider only paths with length greater than zero. Thus, an edge (v, v) is required in order to consider the existence of a path from v to v. We say there is a positive (resp. negative) path from v to w in (V, ϕ) if and only if there is a positive (resp. negative) path from v to w in $\Sigma_{(V,\phi)}$. Furthermore, we say there is a positive (resp. negative) *feedback-loop* in (V, ϕ) if and only if for some $v \in V$ there is a positive (resp. negative) path from v to v in $\Sigma_{(V,\phi)}$.

Characterizing the response of the system

Let (V, ϕ) be a logical network describing a biological system of interest. For capturing changes in the environment of such a biological system, for instance, due to an experimental intervention (over-expression or knock-out), we introduce the notion of clamping variables in the network for overriding their original specification. To this end, we define a *clamping* assignment C as a partial two-valued truth assignment over V. Then, we define the mapping $\phi|_C$ as

$$\phi|_C(v) = \begin{cases} \top & \text{if } v \mapsto \boldsymbol{t} \in C \\ \bot & \text{if } v \mapsto \boldsymbol{f} \in C \\ \phi(v) & \text{if } v \in dom(\phi) \setminus dom(C) \end{cases}$$

yielding the modified logical network $(V, \phi|_C)$. Moreover, it is worth noting that $dom(\phi) \subseteq dom(\phi|_C)$. Let us illustrate this with our toy example in Figure 1.1(a). Let C be the clamping assignment defined by $\{i_1 \mapsto \boldsymbol{t}, i_2 \mapsto \boldsymbol{f}, g \mapsto \boldsymbol{f}\}$. Then, $\phi|_C$ is a complete mapping over V defined as:

$$\phi|_C = \left\{ \begin{array}{cccccccccccc} i_1 & \mapsto & \top & a & \mapsto & \neg d & & d & \mapsto & c & & g & \mapsto & \bot \\ i_2 & \mapsto & \bot & b & \mapsto & a \wedge i_1 & & e & \mapsto & \neg i_1 \wedge i_2 & & o_1 & \mapsto & c \\ & & & c & \mapsto & b \vee e & & f & \mapsto & e \vee g & & o_2 & \mapsto & g \end{array} \right\}$$

In practice, clamping assignments are usually restricted to a subset of variables $X \subseteq V$. Moreover, certain variables in X may be further restricted to be clamped either to a single truth value, viz. $\boldsymbol{t}$ or $\boldsymbol{f}$, or not clamped at all. These restriction will be typically related to context-specific application settings, for instance, the kind of biological entity described by each variable and "real-world" experimental limitations over such entity.

Next, for capturing the synchronous updates in a logical network (V, ϕ), we follow [INO 11] and define the *single-step* operator on either two- or three-valued (complete) truth assignments over V:[4]

$$\Omega_{(V,\phi)}(A) = \{v \mapsto A(\phi(v)) \mid v \in dom(\phi)\} \cup \{v \mapsto A(v) \mid v \in V \setminus dom(\phi)\} \ ,$$

where A is extended to formulas in the standard way. Notice that the definition above captures the fact that unmapped variables in ϕ remain unchanged with respect to the value assigned in A. Furthermore, for capturing the trajectory of state A we define the iterative variant of $\Omega_{(V,\phi)}$ as

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

In biological terms, a sequence $(\Omega^j_{(V,\phi)}(A))_{j\in J}$ captures the signal propagation starting in state A. In particular, we are interested in the fixpoint of $\Omega_{(V,\phi)}$ reachable from certain initial assignment A. Importantly, the existence of such a fixpoint is not necessarily guaranteed. In general, it depends on the definition of A and the presence or absence of feedback-loops in (V, ϕ). But in case of existence, such a fixpoint describes a logical steady state which is interpreted as the response of the biological system described by (V, ϕ). To be more precise, the choice of A is related to how we model the absence of information in the context of either two- or three-valued logics. Hence, when we consider three-valued logic, we use the initial assignment $A_{\boldsymbol{u}} = \{v \mapsto \boldsymbol{u} \mid v \in V\}$. Interestingly, in this context a fixpoint is reached regardless of the presence or absence of feedback-loops in the network. Moreover, such a fixpoint poses the property that each of its variables is assigned to $\boldsymbol{u}$ unless there is a cause to assign it to either $\boldsymbol{t}$ or $\boldsymbol{f}$. On the other hand, when we consider two-valued logic, we use the initial assignment $A_{\boldsymbol{f}} = \{v \mapsto \boldsymbol{f} \mid v \in V\}$. Unfortunately, in this context, the presence of feedback-loops typically avoids reaching a fixpoint. Next, let us illustrate the iterated application of $\Omega_{(V,\phi|_C)}$ for our toy example in the context of both, two- and three-valued logics. Recall that we have defined $\phi|_C$ above for clamping assignment $C = \{i_1 \mapsto \boldsymbol{t}, i_2 \mapsto \boldsymbol{f}, g \mapsto \boldsymbol{f}\}$. The resulting assignments from the computation of $\Omega^j_{(V,\phi|_C)}(A)$ with either $A = A_{\boldsymbol{u}}$ or $A = A_{\boldsymbol{f}}$ are shown in Table 1.3. Notably, when we consider three-valued logic, $\Omega^3_{(V,\phi|_C)}(A_{\boldsymbol{u}}) = \Omega^4_{(V,\phi|_C)}(A_{\boldsymbol{u}})$ results

4. The interested reader may notice the resemblance to single-step operators for logic programs introduced in [APT 82] and [FIT 85] for two- and three-valued assignments respectively.

in the fixpoint:

$$\left\{\begin{array}{cccccccccccc} i_1 & \mapsto & \boldsymbol{t} & a & \mapsto & \boldsymbol{u} & d & \mapsto & \boldsymbol{u} & g & \mapsto & \boldsymbol{f} \\ i_2 & \mapsto & \boldsymbol{f} & b & \mapsto & \boldsymbol{u} & e & \mapsto & \boldsymbol{f} & o_1 & \mapsto & \boldsymbol{u} \\ & & & c & \mapsto & \boldsymbol{u} & f & \mapsto & \boldsymbol{f} & o_2 & \mapsto & \boldsymbol{f} \end{array}\right\}$$

Meanwhile, under two-valued logic we obtain $\Omega^1_{(V,\phi|_C)}(A_{\boldsymbol{f}}) = \Omega^9_{(V,\phi|_C)}(A_{\boldsymbol{f}})$ which leads to an oscillatory behavior for variables a, b, c, d and o_1. Notice that these variables correspond to the ones assigned to $\boldsymbol{u}$ in the fixpoint reached for $\Omega^3_{(V,\phi|_C)}(A_{\boldsymbol{u}})$. In this case, the oscillatory behavior is induced by the negative feedback-loop over the path a, b, c, d. Thus, one can verify that, for example, if we remove the mapping $d \mapsto c$ from the definition of ϕ (leaving d undefined in ϕ), then $\Omega^4_{(V,\phi|_C)}(A_{\boldsymbol{f}})$ would result in the fixpoint:

$$\left\{\begin{array}{cccccccccccc} i_1 & \mapsto & \boldsymbol{t} & a & \mapsto & \boldsymbol{t} & d & \mapsto & \boldsymbol{f} & g & \mapsto & \boldsymbol{f} \\ i_2 & \mapsto & \boldsymbol{f} & b & \mapsto & \boldsymbol{t} & e & \mapsto & \boldsymbol{f} & o_1 & \mapsto & \boldsymbol{t} \\ & & & c & \mapsto & \boldsymbol{t} & f & \mapsto & \boldsymbol{f} & o_2 & \mapsto & \boldsymbol{f} \end{array}\right\}$$

In fact, whenever we consider a logical network (V, ϕ) under two-valued logic, we enforce that (V, ϕ) is free of feedback-loops. Notably as detailed below, although not capable of capturing dynamical properties, this simplification guarantees the existence of a fixpoint while it allows us to characterize the so-called *iimmediate-early response* in signaling networks.

Logical networks and their response with Answer Set Programming

Let (V, ϕ) be a logical network. We represent the variables V as facts over the predicate `variable/1`, namely, `variable(`v`)` for all $v \in V$. Recall that we assume $\phi(v)$ to be in disjunctive normal form for all $v \in V$. Hence, $\phi(v)$ is a set of clauses and a clause a set of literals. We represent formulas using predicates `formula/2`, `dnf/2`, and `clause/3`. The facts `formula(`v`,`$s_{\phi(v)}$`)` map variables $v \in V$ to their corresponding formulas $\phi(v)$, facts `dnf(`$s_{\phi(v)}$`,`s_ψ`)` associate $\phi(v)$ with its clauses $\psi \in \phi(v)$, facts `clause(`s_ψ`,`v`,1)` associate clause ψ with its positive literals $v \in \psi \cap V$, and facts `clause(`s_ψ`,`v`,-1)` associate clause ψ with its negative literals $\neg v \in \psi$. Note that each $s_{(.)}$ stands for some arbitrary but unique name in its respective context. Listing 1.1 shows the representation of our toy example logical network in Figure 1.1(a).

Listing 1.1: Logical networks representation as logical facts

```
variable(i1). variable(i2). variable(o2). variable(o1).
variable(a).  variable(b).  variable(c).  variable(d).
variable(e).  variable(f).  variable(g).

formula(a,0). formula(b,2). formula(c,1).  formula(d,4).
formula(e,3). formula(f,6). formula(g,5). formula(o1,4).
formula(o2,7).
```

```
8
9  dnf(0,5). dnf(1,6). dnf(1,0). dnf(2,3). dnf(3,7).
10 dnf(4,1). dnf(5,2). dnf(6,4). dnf(6,6). dnf(7,4).
11
12 clause(0,b,1).  clause(1,c,1).  clause(2,f,1).  clause(3,a,1).
13 clause(3,i1,1). clause(4,g,1).  clause(5,d,-1). clause(6,e,1).
14 clause(7,i2,1). clause(7,i1,-1).
```

The representation of clamping assignments is straightforward. Note that in the following we use 1 and -1 for truth assignments to $\boldsymbol{t}$ and $\boldsymbol{f}$, respectively. Let C be a clamping assignment over V, we represent the assignments in C as facts over the predicate `clamped/2`, namely, `clamped(`v`,`s`)` with $s = 1$ if $C(v) = \boldsymbol{t}$ and $s = -1$ if $C(v) = \boldsymbol{f}$. The example clamping assignment $C = \{i_1 \mapsto \boldsymbol{t}, i_2 \mapsto \boldsymbol{f}, g \mapsto \boldsymbol{f}\}$ is shown in Listing 1.2.

Listing 1.2: Clamping assignment as logical facts

```
14 clamped(i1,1). clamped(i2,-1). clamped(g,-1).
```

Furthermore, we introduce two rules deriving predicates `eval/2` and `free/2`. The predicate `eval/2` captures the fact that clamped variables are effectively fixed to the

	$\phi\vert_C(v)$	$\top$	$\bot$	$\neg d$	$a \wedge i_1$	$b \vee e$	c	$\neg i_1 \wedge i_2$	$e \vee g$	$\bot$	c	g
	$v \in V$	i_1	i_2	a	b	c	d	e	f	g	o_1	o_2
3-valued	$\Omega^0_{(V,\phi\vert_C)}(A_{\boldsymbol{u}})$	$\boldsymbol{u}$	$\boldsymbol{u}$	$\boldsymbol{u}$	$\boldsymbol{u}$	$\boldsymbol{u}$	$\boldsymbol{u}$	$\boldsymbol{u}$	$\boldsymbol{u}$	$\boldsymbol{u}$	$\boldsymbol{u}$	$\boldsymbol{u}$
	$\Omega^1_{(V,\phi\vert_C)}(A_{\boldsymbol{u}})$	$\boldsymbol{t}$	$\boldsymbol{f}$	$\boldsymbol{u}$	$\boldsymbol{u}$	$\boldsymbol{u}$	$\boldsymbol{u}$	$\boldsymbol{u}$	$\boldsymbol{u}$	$\boldsymbol{f}$	$\boldsymbol{u}$	$\boldsymbol{u}$
	$\Omega^2_{(V,\phi\vert_C)}(A_{\boldsymbol{u}})$	$\boldsymbol{t}$	$\boldsymbol{f}$	$\boldsymbol{u}$	$\boldsymbol{u}$	$\boldsymbol{u}$	$\boldsymbol{u}$	$\boldsymbol{f}$	$\boldsymbol{u}$	$\boldsymbol{f}$	$\boldsymbol{u}$	$\boldsymbol{f}$
	$\Omega^3_{(V,\phi\vert_C)}(A_{\boldsymbol{u}})$	$\boldsymbol{t}$	$\boldsymbol{f}$	$\boldsymbol{u}$	$\boldsymbol{u}$	$\boldsymbol{u}$	$\boldsymbol{u}$	$\boldsymbol{f}$	$\boldsymbol{f}$	$\boldsymbol{f}$	$\boldsymbol{u}$	$\boldsymbol{f}$
	$\Omega^4_{(V,\phi\vert_C)}(A_{\boldsymbol{u}})$	$\boldsymbol{t}$	$\boldsymbol{f}$	$\boldsymbol{u}$	$\boldsymbol{u}$	$\boldsymbol{u}$	$\boldsymbol{u}$	$\boldsymbol{f}$	$\boldsymbol{f}$	$\boldsymbol{f}$	$\boldsymbol{u}$	$\boldsymbol{f}$
2-valued	$\Omega^0_{(V,\phi\vert_C)}(A_{\boldsymbol{f}})$	$\boldsymbol{f}$	$\boldsymbol{f}$	$\boldsymbol{f}$	$\boldsymbol{f}$	$\boldsymbol{f}$	$\boldsymbol{f}$	$\boldsymbol{f}$	$\boldsymbol{f}$	$\boldsymbol{f}$	$\boldsymbol{f}$	$\boldsymbol{f}$
	$\Omega^1_{(V,\phi\vert_C)}(A_{\boldsymbol{f}})$	$\boldsymbol{t}$	$\boldsymbol{f}$	$\boldsymbol{t}$	$\boldsymbol{f}$	$\boldsymbol{f}$	$\boldsymbol{f}$	$\boldsymbol{f}$	$\boldsymbol{f}$	$\boldsymbol{f}$	$\boldsymbol{f}$	$\boldsymbol{f}$
	$\Omega^2_{(V,\phi\vert_C)}(A_{\boldsymbol{f}})$	$\boldsymbol{t}$	$\boldsymbol{f}$	$\boldsymbol{t}$	$\boldsymbol{t}$	$\boldsymbol{f}$	$\boldsymbol{f}$	$\boldsymbol{f}$	$\boldsymbol{f}$	$\boldsymbol{f}$	$\boldsymbol{f}$	$\boldsymbol{f}$
	$\Omega^3_{(V,\phi\vert_C)}(A_{\boldsymbol{f}})$	$\boldsymbol{t}$	$\boldsymbol{f}$	$\boldsymbol{t}$	$\boldsymbol{t}$	$\boldsymbol{t}$	$\boldsymbol{f}$	$\boldsymbol{f}$	$\boldsymbol{f}$	$\boldsymbol{f}$	$\boldsymbol{f}$	$\boldsymbol{f}$
	$\Omega^4_{(V,\phi\vert_C)}(A_{\boldsymbol{f}})$	$\boldsymbol{t}$	$\boldsymbol{f}$	$\boldsymbol{t}$	$\boldsymbol{t}$	$\boldsymbol{t}$	$\boldsymbol{t}$	$\boldsymbol{f}$	$\boldsymbol{f}$	$\boldsymbol{f}$	$\boldsymbol{t}$	$\boldsymbol{f}$
	$\Omega^5_{(V,\phi\vert_C)}(A_{\boldsymbol{f}})$	$\boldsymbol{t}$	$\boldsymbol{f}$	$\boldsymbol{f}$	$\boldsymbol{t}$	$\boldsymbol{t}$	$\boldsymbol{t}$	$\boldsymbol{f}$	$\boldsymbol{f}$	$\boldsymbol{f}$	$\boldsymbol{t}$	$\boldsymbol{f}$
	$\Omega^6_{(V,\phi\vert_C)}(A_{\boldsymbol{f}})$	$\boldsymbol{t}$	$\boldsymbol{f}$	$\boldsymbol{f}$	$\boldsymbol{f}$	$\boldsymbol{t}$	$\boldsymbol{t}$	$\boldsymbol{f}$	$\boldsymbol{f}$	$\boldsymbol{f}$	$\boldsymbol{t}$	$\boldsymbol{f}$
	$\Omega^7_{(V,\phi\vert_C)}(A_{\boldsymbol{f}})$	$\boldsymbol{t}$	$\boldsymbol{f}$	$\boldsymbol{f}$	$\boldsymbol{f}$	$\boldsymbol{f}$	$\boldsymbol{t}$	$\boldsymbol{f}$	$\boldsymbol{f}$	$\boldsymbol{f}$	$\boldsymbol{t}$	$\boldsymbol{f}$
	$\Omega^8_{(V,\phi\vert_C)}(A_{\boldsymbol{f}})$	$\boldsymbol{t}$	$\boldsymbol{f}$	$\boldsymbol{f}$	$\boldsymbol{f}$	$\boldsymbol{f}$	$\boldsymbol{f}$	$\boldsymbol{f}$	$\boldsymbol{f}$	$\boldsymbol{f}$	$\boldsymbol{f}$	$\boldsymbol{f}$
	$\Omega^9_{(V,\phi\vert_C)}(A_{\boldsymbol{f}})$	$\boldsymbol{t}$	$\boldsymbol{f}$	$\boldsymbol{t}$	$\boldsymbol{f}$	$\boldsymbol{f}$	$\boldsymbol{f}$	$\boldsymbol{f}$	$\boldsymbol{f}$	$\boldsymbol{f}$	$\boldsymbol{f}$	$\boldsymbol{f}$

Table 1.3. *Exemplary iterated application of $\Omega_{(V,\phi\vert_C)}$ for (V, ϕ) in Figure 1.1(a), clamping assignment $C = \{i_1 \mapsto \boldsymbol{t}, i_2 \mapsto \boldsymbol{f}, c \mapsto \boldsymbol{f}\}$ and initial assignment $A_{\boldsymbol{u}}$ or $A_{\boldsymbol{f}}$.*

corresponding evaluation. Finally, we use the predicate `free/2` to represent the fact that every variable not clamped in C, is subject to the corresponding mapping $\phi(v)$. Both rules are shown in Listing 1.3.

Listing 1.3: Clamped and free variables

```
15 eval(V,S)  :- clamped(V,S).
16 free(V,D)  :- formula(V,D); dnf(D,_); not clamped(V,_).
```

Next, we describe how we model either two- or three-valued logics in ASP. In fact, the rule modeling the propagation of (positive) true values is the same for both logics. Essentially, we exploit the fact that formulas $\phi(v)$ are in disjunctive normal form. Hence, under both logics we derive `eval(`v`,1)` if v is not clamped and there exists a conjunct $\psi \in \phi(v)$ such that all its literals evaluate positively. The rule describing this is shown in Listing 1.4.

Listing 1.4: Positive propagation common to two- and three-valued logics

```
17 eval(V,1) :- free(V,D); eval(W,T) : clause(J,W,T); dnf(D,J).
```

Meanwhile, the propagation of (negative) false values depends on the type of logic under consideration. On the one hand, when we consider two-valued logic, we use the rule shown in Listing 1.5 to derive `eval(`v`,`-1`)` if it cannot be proved that v evaluates positively, that is, `not eval(`v`,1)`.

Listing 1.5: Negative propagation for two-valued logic (with default negation)

```
18 eval(V,-1) :- variable(V); not eval(V,1).
```

On the other hand, when we consider three-valued logic, we use the rules shown in Listing 1.6. Notice that in this case, we derive `eval(`v`,`-1`)` only if it can be proved that all clauses $\psi \in \phi(v)$ evaluate negatively. A clause ϕ evaluates negatively if at least one of its literals evaluates negatively. Clauses evaluating negatively are represented with the predicate `eval_clause/2`.

Listing 1.6: Negative propagation for three-valued logic (with explicit proof)

```
18 eval_clause(J,-1) :- clause(J,V,S); eval(V,-S).
19 eval(V,-1)        :- free(V,D); eval_clause(J,-1) : dnf(D,J).
```

Interestingly, our ASP representation is relatively simple yet flexible enough to be extended and adapted for specific applications as we illustrate in the remainder of this chapter. In what follows, we provide a formal characterization for two very relevant problems over logical networks together with the corresponding modeling and solving based on the presented ASP encodings. First we extend our representation to learn logical networks from a given interaction graph confronting their response with

experimental observations. Moreover, we consider several clamping assignments (describing experimental conditions) simultaneously instead of only one. Afterwards, we adapt our representation again aiming at reasoning over a family of logical networks and finding clamping assignments (describing therapeutic interventions) leading to responses satisfying specific goals.

1.3.3. *Learning models of immediate-early response*

Background

Firstly, we briefly summarize the main biological hypotheses in [SAE 09] providing the foundation for the concept of Boolean logic models of immediate-early response. Concretely, a *Boolean logic model of immediate-early response* is a logical network (V, ϕ) as defined above, without feedback-loops and using classical (Boolean) logics. The main assumption under Boolean logic models of immediate-early response is the following. The response of a biological system to external perturbations occurs at several time scales [PAP 05]. 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) [KLA 06b]. 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 [REM 08, PAU 12] cannot happen until the second phase of signal propagation occurring at a slower time scale. Therefore, eedback-loops are not included in Boolean logic models of immediate-early response assuming that they will become active in a late phase [MAC 12]. Notably, it follows that starting from any initial state, a Boolean logic model of immediate-early response reaches a unique steady state or fixpoint in polynomial time [PAU 12]. Thus, such modeling approach, although not capable of capturing dynamical properties, provides a relatively simple framework for input-output predictive models.

Based on the assumptions and concepts described above, authors in [SAE 09] have proposed a method to learn from an interaction graph and phosphorylation activities at a pseudo-steady state, Boolean logic models of immediate-early response fitting experimental data. In the remaining of this section we provide a precise characterization of this problem using the notions introduced in Section 1.3 and adapting our Answer Set Programming representation accordingly.

Problem

A *prior knowledge network* is an interaction graph (V, E, σ) as defined above. In addition, we distinguish three special subsets of species in V namely, the *stimuli* (V_S),

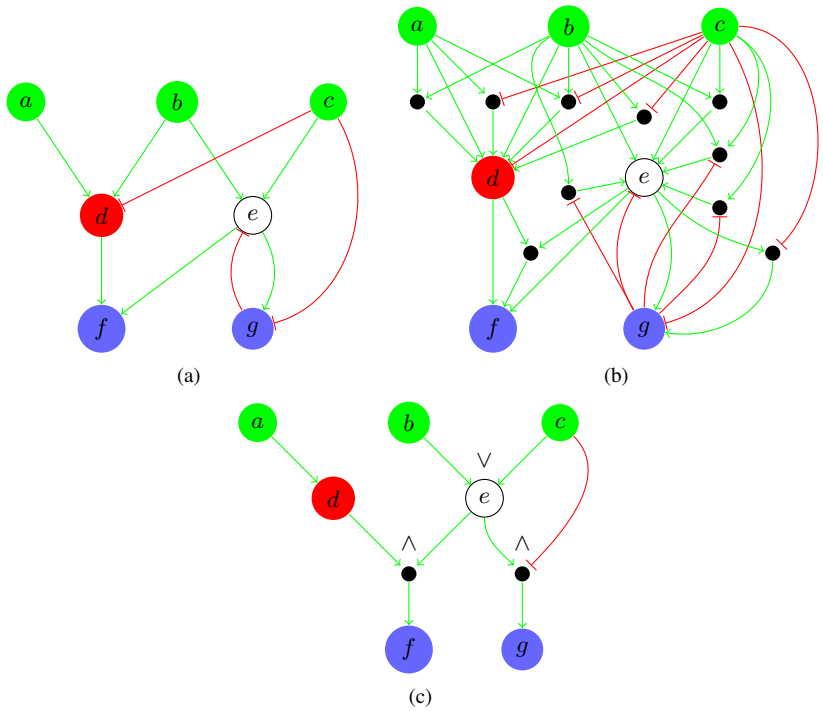

Figure 1.2. *The green and red edges correspond to activations and inhibitions, respectively. Green nodes represent ligands that can be experimentally stimulated. Red nodes represent species that can be inhibited by using a drug. Blue nodes represent species that can be measured by using an antibody. White nodes are neither measured, nor manipulated.* **(a)** *A toy interaction (directed and signed) graph describing causal interactions.* **(b)** *Hypergraph expansion describing all plausible logical interactions based on the prior knowledge network in (a).* **(c)** *A Boolean logic model derived from the prior knowledge network in (a) describing functional relationship defined by the mapping* $\{d \mapsto a; e \mapsto b \vee c; f \mapsto d \wedge e; g \mapsto e \wedge \neg c\}$.

the *knock-outs* (V_K) and the *readouts* (V_R). Nodes in V_S denote extracellular ligands that can be stimulated or knocked-in and thus, we assume they have indegree equal to zero. Nodes in V_K denote intracellular species that can be inhibited or knocked-out 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 setup. 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. An early simplification consists on compressing the PKN in order to collapse most of the nodes in V_U. This often results on a significant reduction of the search space that must be explored during learning. Thus, herein we assume a compressed PKN as an

input and we refer the interested reader to [SAE 09] for a detailed description on this subject.

Given a PKN (V, E, σ), the concept of an *experimental condition* over (V, E, σ) is captured by a clamping assignment over variables $V_S \cup V_K$. Recall that clamping assignments were defined above as partial two-valued assignments. To be more precise, while variables in V_S can be clamped to either $\boldsymbol{t}$ or $\boldsymbol{f}$, variables in V_K can only be clamped to $\boldsymbol{f}$. Next, if C is an experimental condition and $v \in V_S$, then $C(v) = \boldsymbol{t}$ (resp. $\boldsymbol{f}$) indicates that the stimulus v is present (resp. absent), while if $v \in V_K$, then $C(v) = \boldsymbol{f}$ indicates that the species v is inhibited or knocked out. In fact, since extracellular ligands by default are assumed to be absent, for the sake of simplicity we can omit clampings to $\boldsymbol{f}$ over variables in V_S. Therefore, if C is an experimental condition and $v \in dom(C)$ then, either $v \in V_S$ and $C(v) = \boldsymbol{t}$, or $v \in V_K$ and $C(v) = \boldsymbol{f}$. Furthermore, the concept of an *experimental observation* under an experimental condition C is captured by a partial mapping $P_C : V_R \mapsto [0, 1]$. That is, $dom(P_C) \subseteq V_R$ denotes the set of measured readouts under the experimental condition C. If $v \in dom(P_C)$, then $P_C(v)$ represents the phosphorylation activity at a pseudo-steady state of the readout v under C. Notably, it is rather critical to choose a time point that is characteristic for the fast or early events in the biological system under consideration [MAC 12]. 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 (C_i, P_{C_i}) with experimental conditions C_i and experimental observations P_{C_i}. Further, we denote with N_ξ the *size* of ξ given by the number of measured readouts across all experimental conditions $i = 1, \ldots, n$, i.e., $N_\xi = \sum_{i=1}^{n} |dom(P_{C_i})|$.

Let us illustrate the concepts described above with our toy example. Consider the PKN (V, E, σ) defined in Figure 1.2(a). From the graph coloring, we have $V_S = \{a, b, c\}$, $V_K = \{d\}$ and $V_R = \{f, g\}$. Furthermore, let $\xi = ((C_1, P_{C_1}), \ldots, (C_4, P_{C_4}))$ be an example experimental dataset over (V, E, σ) defined by

$$\begin{array}{ll} C_1 = \{a \mapsto \boldsymbol{t}, c \mapsto \boldsymbol{t}\} & P_{C_1} = \{f \mapsto 0.9, g \mapsto 0.0\} \\ C_2 = \{a \mapsto \boldsymbol{t}, c \mapsto \boldsymbol{t}, d \mapsto \boldsymbol{f}\} & P_{C_2} = \{f \mapsto 0.1, g \mapsto 0.9\} \\ C_3 = \{a \mapsto \boldsymbol{t}\} & P_{C_3} = \{f \mapsto 0.0, g \mapsto 0.1\} \\ C_4 = \{a \mapsto \boldsymbol{t}, b \mapsto \boldsymbol{t}\} & P_{C_4} = \{f \mapsto 1.0, g \mapsto 0.8\}. \end{array} \tag{1.1}$$

In words, the experimental conditions $C_1, \ldots, C_4$ can be read as follows. In C_1, stimuli a and c are present, stimulus b is absent and d is not inhibited; in C_2, stimuli a, b, c are like in C_1 but d is inhibited; in C_3, only the stimulus a is present and d is not inhibited; and in C_4, stimuli a and b are present, stimulus c is absent and d is not inhibited. Experimental observations $P_{C_1}, \ldots, P_{C_4}$ give (normalized) phosphorylation activities for readouts f and g under the corresponding experimental condition.

Next, we introduce the notion of Boolean predictions. Let (V, E, σ) be a PKN. Further, let $\xi = (C_i, P_{C_i})$ be an experimental dataset over (V, E, σ) with $i = 1, \ldots, n$.

As detailed above, a Boolean logic model of immediate-early response is defined by a logical network (V, ϕ) without feedback-loops and using classical (Boolean) logics. Hence, now we can define the predictions (output) provided by a Boolean logic model of immediate-early response with respect to a given set of experimental conditions (input). Towards this end, we characterize the response of logical networks using fixpoint semantics as detailed above. More precisely, for $i = 1, \ldots, n$ let F_i be the fixpoint of $\Omega_{(V,\phi|_{C_i})}$ reachable from $A_{\boldsymbol{f}}$. Notice that such a fixpoint always exists given that (V, ϕ) is free of feedback-loops. In words, each F_i describe the logical response (starting from $A_{\boldsymbol{f}}$) of (V, ϕ) with respect to the experimental condition or clamping assignment C_i. Next, we define a straightforward transformation from truth values to binary but numerical values. Such a transformation provides a more convenient notation in order to compare predictions and phosphorylation activities. The *Boolean prediction* of (V, ϕ) with respect to the experimental condition C_i is a function $\pi_i : V \to \{0, 1\}$ defined as

$$\pi_i(v) = \begin{cases} 1 & \text{if } F_i(v) = \boldsymbol{t} \\ 0 & \text{if } F_i(v) = \boldsymbol{f}. \end{cases}$$

As we see below, during learning we aim at explaining the given experimental dataset ξ. Therefore, we are particularly interested on predictions with respect to the experimental conditions included in ξ and over the measured variables in each experimental condition. Nevertheless, predictions with respect to non-performed experimental conditions and/or over non-observed species can be useful to generate testeable hypotheses on the response of the system.

As an example, consider the Boolean logic model (V, ϕ) from Figure 1.2(c) and the experimental condition C_2 from the dataset given in (1.1). Then, the clamped logical network $(V, \phi|_{C_2})$ is defined by the mapping

$$\phi|_{C_2} = \{a \mapsto \top; b \mapsto \bot; c \mapsto \top; d \mapsto \bot; e \mapsto b \vee c; f \mapsto d \wedge e; g \mapsto e \wedge \neg c\}.$$

Next, the fixpoint of $\Omega_{(V,\phi|_{C_2})}$ reachable from $A_{\boldsymbol{f}}$ can be computed yielding the assignment F_2 defined as

$$F_2 = \{a \mapsto \boldsymbol{t}, b \mapsto \boldsymbol{f}, c \mapsto \boldsymbol{t}, d \mapsto \boldsymbol{f}, e \mapsto \boldsymbol{t}, f \mapsto \boldsymbol{f}, g \mapsto \boldsymbol{f}\}.$$

Finally, the Boolean prediction for (V, ϕ) under the experimental condition C_2 is given by

$$\pi_2 = \{a \mapsto 1, b \mapsto 0, c \mapsto 1, d \mapsto 0, e \mapsto 1, f \mapsto 0, g \mapsto 0\}.$$

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. 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. $((w, v), -1) \in \sigma$). Towards this end, we consider a pre-processing step where the given PKN is expanded to generate a (signed) directed hypergraph describing all plausible logical interactions. For each $v \in V$ having non-zero indegree, let $Pred(v)$ be the set of its (signed) predecessors, namely, $Pred(v) = \{(u, s) \mid (u, v) \in E, ((u, v), s) \in \sigma\}$. Furthermore, let $\mathcal{P}(v)$ be the powerset of $Pred(v)$, namely, $2^{Pred(v)}$. Then, (V, H) is the (signed) directed hypergraph *expanded* from (V, E, σ) with nodes V and (signed) directed hyperedges H if for each $v \in V$, $(p, v) \in H$ whenever $p \in \mathcal{P}(v)$. Next, Boolean logic models must essentially result from pruning (V, H). Additionally, we impose two constraints related to the fact that our Boolean logic models essentially aim at providing a framework for input-output predictions.[5] Firstly, for any variable u defined in ϕ all variables $w \in \phi(u)$ must be *reachable* from some stimuli variable. That is, we consider only Boolean logic models (V, ϕ) such that for every $u \in dom(\phi)$ and $w \in \phi(u)$, either $w \in V_S$ or there exist $v \in V_S$ and a path from v to w in the underlying interaction graph $\Sigma_{(V,\phi)}$. Secondly, every variable u defined in ϕ must *reach* some readout variable. That is, we consider only Boolean logic models (V, ϕ) such that for every $u \in dom(\phi)$ there exist $v \in V_R$ and a path from u to v in the underlying interaction graph $\Sigma_{(V,\phi)}$. Finally, let us denote with $\mathbb{M}_{(V,E,\sigma)}$ the *search space* of Boolean logic models satisfying the conditions given above: evidence in (V, E, σ), no feedback-loops, and reachability from/to stimuli/readouts.

In Figure 1.2(a) we show an exemplary PKN and the corresponding expanded (signed) directed hypergraph in Figure 1.2(b). As already described, (signed) directed hypergraphs can be directly linked to logical networks. Thus, by considering each (signed) directed hyperedge in Figure 1.2(b) as either present or absent (and verifying the additional constraints related to feedback-loops and reachability from/to stimuli/readouts), one can generate the search space of Boolean logic models $\mathbb{M}_{(V,E,\sigma)}$ defined above.

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 [SAE 09] put forward the idea of training Boolean logic models by confronting their corresponding Boolean predictions with phosphorylation activities 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 [FRE 04].

5. The interested reader may notice the analogy with the elimination of nodes neither controllable nor (leading to) observable during the compression of the PKN described in [SAE 09].

We now provide the precise formulation for each optimization criteria. Let (V, E, σ) be a PKN. Let $\xi = (C_i, P_{C_i})$ be an experimental dataset over (V, E, σ) with $i = 1, \ldots, n$. Let (V, ϕ) be a Boolean logic model having evidence in (V, E, σ) and let $\pi_1, \ldots, \pi_n$ be its Boolean predictions with each π_i defined under C_i. Firstly, based on the residual sum of squares (RSS) we define the *residual* (Θ_{rss}) of (V, ϕ) with respect to ξ as

$$\Theta_{rss}((V, \phi), \xi) = \sum_{i=1}^{n} \sum_{v \in dom(P_{C_i})} (P_{C_i}(v) - \pi_i(v))^2. \tag{1.2}$$

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

$$\Theta_{size}((V, \phi)) = \sum_{v \in dom(\phi)} |\phi(v)|. \tag{1.3}$$

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 [SAE 09]. Therein, a single-objective function is defined that balances *residual* 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, as detailed in [FRE 04], this 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. 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. $\Theta_{size}((V, \phi)) = 0$. 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. [MAR 04]) could be investigated in the future. To conclude, our lexicographic multi-objective optimization consists of minimizing first Θ_{rss}, and then with lower priority Θ_{size}:

$$(V, \phi_{opt}) = \underset{(V,\phi) \in \mathbb{M}_{(V,E,\sigma)}}{\arg\min} \; (\Theta_{rss}((V, \phi), \xi), \Theta_{size}((V, \phi))). \tag{1.4}$$

Information provided by high-throughput data is intrinsically uncertain due to experimental errors. Therefore, one is not only interested in optimal models but in *nearly* optimal models as well. In this context, authors in [SAE 09] have considered Boolean logic models minimizing Θ_{rss} within certain tolerance, e.g. 10% of the minimum residual. Next, they argue that all models found can explain the data similarly or equally well if one take into account the experimental error. Notice that, in the aforecited work the optimization is addressed using a genetic algorithm. Hence, "minimum residual" refers to the minimum found during the execution of the algorithm which is not necessarily the global minimum. Moreover, due to the incompleteness of stochastic search methods, it is very likely that certain solutions within the allowed tolerance are not found. In practice, one can execute the genetic algorithm several times in order to overcome this issue to some extent. Nonetheless, as we show later in this chapter, a significant number of models may be missing even after several executions. Similarly but in the context of quantitative modeling (based on ordinary differential equations) and using a simulated annealing algorithm, authors in [CHE 09] have elaborated upon the same argument. Interestingly, despite the fact that the model appears to be non-identifiable in both contexts, viz. qualitative and quantitative modeling, biologically relevant insights have been reported in the two aforecited studies. Notably, minimization over size in (1.4) is based on Occam's razor principle. On the one hand, one can consider that larger logic models overfit the available dataset by introducing excessive complexity [SAE 09, PRI 11]. 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 [STE 04]. Therefore, let (V, ϕ_{opt}) be a Boolean logical model as defined in (1.4). Then, considering that tolerance over residual and size may yield biologically relevant models, we are particularly interested in enumerating all (nearly) optimal Boolean logic models (V, ϕ) such that,

$$\Theta_{rss}((V,\phi),\xi) \leq \Theta_{rss}((V,\phi_{opt}),\xi)+t_{rss} \quad \Theta_{size}((V,\phi)) \leq \Theta_{size}((V,\phi_{opt}))+t_{size}$$

with t_{rss} and t_{size} denoting the tolerance over residual and size, respectively.

Next, we introduce the notion of logical input-output behaviors. In practice, the enumeration of (nearly) optimal models often leads to a large number of logical networks, namely, (V, ϕ_j) with $j = 1, \ldots, m$ and $m \gg 1$. Notably, each ϕ_j is a different mapping from variables to propositional formulas. However, it may happen (and it often happens) that for all $v \in V_R$, several logical networks describe exactly the same response to every possible experimental condition (clamping assignments over variables $V_S \cup V_K$). In such a case, we say that those logical networks describe the same input-output behavior. To be more precise, recall that we consider a PKN (V, E, σ). Notice that in each experimental condition over (V, E, σ), every stimulus $v \in V_S$ and inhibitor $v \in V_K$, can be either clamped or not. Thus, let us denote with $\mathcal{C}$ the space of all possible clamping assignments or experimental conditions C over (V, E, σ). Notably, the number of possible clamping assignments is given by $|\mathcal{C}| = 2^{|V_S|+|V_K|}$.

Then, let $(V, \phi_j), (V, \phi_{j'})$ be two (nearly) optimal Boolean logic models. Furthermore, let F_C^j and $F_C^{j'}$ be the fixpoints of $\Omega_{(V,\phi_j|_C)}$ and $\Omega_{(V,\phi_{j'}|_C)}$ reachable from $A_{\boldsymbol{f}}$, respectively. We say that (V, ϕ_j) and $(V, \phi_{j'})$ describe the same *logical input-output behavior* if and only if $F_C^j(v) = F_C^{j'}(v)$ for all $v \in V_R$ and $C \in \mathcal{C}$. Importantly, this abstraction allows us to group logical networks regardless of their "internal wirings" and focus on their input-output predictions. In practice, this also facilitates the analysis and interpretation of results whereas it provides a way to extract robust insights despite the high variability.

Encoding

In order to express and solve the multi-objective optimization described in (1.4) by using ASP, one needs to discretize the function defined in (1.2). A very simple approach converts numerical data into binary data according to a threshold. Furthermore, we propose a finer multi-valued discretization scheme. In fact, the only non-integer variables in (1.2) are the experimental observations $P_{C_i}(v)$. Then, we approximate these values up to $\frac{1}{10^k}$ introducing a parametrized approximation function δ_k (e.g. using the floor or closest integer functions). Next, we define the discrete residual Θ_{rss_k} as

$$\Theta_{rss_k}((V, \phi), \xi) = \sum_{i=1}^{n} \sum_{v \in dom(P_{C_i})} \left[10^k \delta_k(P_{C_i}(v)) - 10^k \pi_i(v)\right]^2 . \tag{1.5}$$

The minimizations of Θ_{rss} and Θ_{rss_k} may yield different Boolean logic models. Nonetheless, one can prove that finding all models minimizing Θ_{rss_k} within a certain tolerance allows us to find all models minimizing Θ_{rss} as well.

Let (V, E, σ) be a PKN and let (V, H) be the directed hypergraph expanded from it. Recall that with $\mathcal{P}(v)$ we denote the powerset of the signed predecessors of $v \in V$, namely, $2^{Pred(v)}$. We represent the directed hypergraph (V, H) using predicates `node/2`, `hyper/3`, and `edge/3`. The facts `node(`v`,`$s_{\mathcal{P}(v)}$`)` map nodes $v \in V$ to their corresponding sets of signed predecessors $\mathcal{P}(v)$, facts `hyper(`$s_{\mathcal{P}(v)}$`,`s_p`,`l`)` associate $\mathcal{P}(v)$ with its sets $p \in \mathcal{P}(v)$ where l denotes their cardinalities, facts `edge(`s_p`,`v`,1)` associate the set p with $(v, 1) \in p$, and facts `edge(`s_p`,`v`,-1)` associate the set p with $(v, -1) \in p$. Note that each $s_{(\cdot)}$ stands for some arbitrary but unique name in its respective context here. Facts over predicates `stimulus`$/1$, `inhibitor`$/1$, and `readout`$/1$ denote nodes in V_S, V_K, and V_R respectively. Next, let $\xi = (C_i, P_{C_i})$ be an experimental dataset over (V, E, σ) with $i = 1, \ldots, n$. Recall that each C_i is a clamping assignment over variables in $V_S \cup V_K$. Then, we extend our representation of clamping assignments given before in order to consider several experimental conditions simultaneously. Towards this end, we represent experimental conditions as facts over predicate `clamped`$/3$, namely `clamped(`i`,`v`,`$C_i(v)$`)` for all $v \in dom(C_i)$ and $i = 1, \ldots, n$. Finally, let k define the discretization scheme. We represent discretized experimental observations as facts over predicate `obs`$/3$, namely,

`obs(`i`,`v`,`$10^k\delta_k(P_{C_i}(v))$`)` for all $v \in dom(P_{C_i})$ and $i = 1, \ldots, n$. We use the predicate `dfactor/1` to denote the discretization factor 10^k.

Using the discretization scheme provided by $k = 1$, Listing 1.7 shows the instance representation for our toy example. That is, the (signed) directed hypergraph in Figure 1.2(b) and the dataset given in (1.1).

Listing 1.7: Toy example input instance (`toy.lp`)

```
node(e,1).  node(d,2). node(g,3).
node(f,4).  node(a,5). node(b,6). node(c,7).

hyper(1,1,1). hyper(2,1,1). hyper(1,8,2).  hyper(2,13,2).
hyper(1,2,1). hyper(2,4,1). hyper(1,9,2).  hyper(2,11,2).
hyper(1,3,1). hyper(2,5,1). hyper(1,10,2). hyper(2,12,2).
hyper(3,5,1). hyper(4,6,1). hyper(3,14,2). hyper(1,16,3).
hyper(3,6,1). hyper(4,7,1). hyper(4,15,2). hyper(2,17,3).

edge(1,b,1).   edge(2,c,1).   edge(3,g,-1).  edge(4,a,1).
edge(5,c,-1).  edge(6,e,1).   edge(7,d,1).   edge(8,b,1).
edge(8,c,1).   edge(9,b,1).   edge(9,g,-1).  edge(10,c,1).
edge(10,g,-1). edge(11,a,1).  edge(11,b,1).  edge(12,a,1).
edge(12,c,-1). edge(13,b,1).  edge(13,c,-1). edge(14,e,1).
edge(14,c,-1). edge(15,d,1).  edge(15,e,1).  edge(16,b,1).
edge(16,c,1).  edge(16,g,-1). edge(17,a,1).  edge(17,b,1).
edge(17,c,-1).

clamped(1,a,1). clamped(1,c,1).
clamped(2,a,1). clamped(2,c,1). clamped(2,d,-1).
clamped(3,a,1).
clamped(4,a,1). clamped(4,b,1).

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

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

dfactor(10).
```

Next we describe our encoding for solving the learning of Boolean logic models as described in the previous section. Our ASP encoding is shown in Listing 1.8.

Listing 1.8: Logic program for learning Boolean logic models (`learning.lp`)

```
variable(V)               :- node(V,_).
formula(V,I)              :- node(V,I); hyper(I,_,_).
{dnf(I,J) : hyper(I,J,N)} :- formula(V,I).
clause(J,V,S)             :- edge(J,V,S); dnf(_,J).

path(U,V) :- formula(V,I); dnf(I,J); edge(J,U,_).
path(U,V) :- path(U,W); path(W,V).
:- path(V,V).
```

```
:- dnf(I,J); edge(J,V,_); not stimulus(V);
   not path(U,V) : stimulus(U).
:- path(_,V); not readout(V); not path(V,U) : readout(U).

exp(E)           :- clamped(E,_,_).
clamped(E,V,-1)  :- exp(E); stimulus(V); not clamped(E,V,1).
clamped(E,V)     :- clamped(E,V,_).
free(E,V,I)      :- formula(V,I); dnf(I,_); exp(E);
                    not clamped(E,V).

eval(E,V, S) :- clamped(E,V,S).
eval(E,V, 1) :- free(E,V,I); eval(E,W,T) : edge(J,W,T); dnf(I,J).
eval(E,V,-1) :- not eval(E,V,1); exp(E); variable(V).

rss(D,V, 1, (F-D)**2) :- obs(E,V,D); dfactor(F).
rss(D,V,-1, D**2)     :- obs(E,V,D).

#minimize{L@1, dnf,I,J : dnf(I,J), hyper(I,J,L)}.
#minimize{W@2, rss,E,V : obs(E,V,D), eval(E,V,S), rss(D,V,S,W)}.

:- formula(V,I); hyper(I,J1,N); hyper(I,J2,M); N < M,
   dnf(I,J1);  dnf(I,J2); edge(J2,U,S) : edge(J1,U,S).

:- formula(V,I); dnf(I,J); edge(J,U,S); edge(J,U,-S).

#const maxsize = -1.
#const maxrss  = -1.

:- maxsize >= 0; maxsize + 1
   #sum {L,dnf,I,J : dnf(I,J), hyper(I,J,L)}.

:- maxrss >= 0; maxrss  + 1
   #sum {W,rss,E,V : obs(E,V,D), eval(E,V,S), rss(D,V,S,W)}.

#show formula/2.
#show dnf/2.
#show clause/3.
```

Lines 1-4 define rules generating the representation of a logical network as described in Section 1.3. Line 1 simply projects node names to the predicate `variable/1`. In Line 2 every node $v \in V$ having non-zero indegree is mapped to a formula $\phi(v)$. Next, in Line 3 each set of signed predecessors $p \in \mathcal{P}(v)$ is interpreted as an abducible conjuctive clause in $\phi(v)$. Then, in Line 4 if $p \in \mathcal{P}(v)$ has been abduced, predicates `clause/3` are derived for every signed predecessor in p.[6] Let us illustrate this on our toy example. In order to describe the mappings $e \mapsto b \vee c$ and $g \mapsto e \wedge \neg c$, one would generate a candidate answer set with atoms `dnf(1,1)`, `dnf(1,2)` and

6. Notice that predicates `clause/3` are only used for the sake of interpretation. One could simply replace Line 45 at the end of the encoding with `#show clause(J,V,S) : edge(J,V,S); dnf(_,J).` and remove Line 4.

`dnf(3,14)` (from Line 2 we derive `formula(e,1)` and `formula(g,3)`). Note that this also force to have atoms `clause(1,b,1)`, `clause(2,c,1)`, `clause(14,e,1)` and `clause(14,c,-1)`. Lines 6-8 eliminate candidate answer sets describing logic models with feedback-loops. Paths from u to v are represented over predicate `path`$/2$ and derived recursively. Thus, the integrity constraint in Line 8 avoids self-reachability in the Boolean logic models. Next, the constraint in Lines 9-10 ensures that for any variable u defined in ϕ all variables $w \in \phi(u)$ are reachable from some stimuli variable. Whereas the constraint in Line 11 guarantees that every variable u defined in ϕ reaches some readout variable. Notice that at this point, we have a representation of the search space of Boolean logic models $\mathbb{M}_{(V,E,\sigma)}$.

Lines 13-21 elaborate on the rules from Listings 1.3, 1.4 and 1.5 given in Section 1.3 in order to consider several clamping assignments simultaneously and compute the fixpoint for each of them accordingly. To be more precise, the response under each experimental condition is represented over predicates `eval/3`, namely `eval(`i`,`v`,`s`)` for experimental condition C_i if variable v is assigned to s. In Lines 23-24 we compute the possible differences (square of residuals) between Boolean predictions and the corresponding experimental observations. We denote such differences over predicate `rss`$/4$, namely `rss(`o`,`v`,`t`,`r`)` for a residual r with respect to the experimental observation o if the Boolean prediction for $v \in V$ is the truth value $t \in \{1, -1\}$. Note that such predicates are independent from every candidate answer set, that is, they can be deduced during grouding. For our example, due to the experimental condition C_2 we have `rss(1,f,1,81)`, `rss(1,f,-1,1)`, `rss(9,g,1,1)` and `rss(9,g,-1,81)`. Therefore, if the fixpoint for f under the experimental condition C_2 is 1, the residual would be 81, whereas if the fixpoint is -1, the residual is only 1. Analogously, but in the opposite way the same holds for g. Next, we describe our lexicographic multi-objective optimization. In Line 26 we declare with lower priority (`@1`) the minimization over the size of logic models (Eq. (1.3)). Meanwhile, in Lines 27 we declare, with higher priority (`@2`), the minimization of the residual sum of squares between the Boolean predictions and experimental observations (Eq. (1.5)).

Lines 29-32 define two relatively simple symmetry-breaking constraints which are particularly relevant during the enumeration of (nearly) optimal solutions. Essentially, these integrity constraints eliminate answer sets describing "trivially" equivalent Boolean logic models with respect to their logical input-output behavior. The constraint in Lines 29-30 eliminates solutions by checking inclusion between conjunctions. For example, for two variables v and w, the formula $v \vee (v \wedge w)$ is logically equivalent to v and hence, the latter is preferred. Next, the constraint in Line 31 simply avoids solutions having mappings in the Boolean logic models of the form $v \wedge \neg v$. 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* [MCC 56]

Lines 34-41 define a rather "standard" mechanism in order to enumerate solutions within given boundaries. Lines 34-35 simply define two constants describing the boundaries for each optimization criterion which are by default set to -1. Lines 37-38 define an integrity constraint in order to eliminate solutions describing Boolean logic models (V, ϕ) if `maxsize` ≥ 0 and `maxsize`$+1 \leq \Theta_{size}((V, \phi))$. Analogously, Lines 40-41 define an integrity constraint in order to eliminate solutions describing Boolean logic models (V, ϕ) if `maxrss` ≥ 0 and `maxrss` $+ 1 \leq \Theta_{rss}((V, \phi), \xi)$.

Solving

In Listing 1.9 we show the optimum answer set found for the toy instance described in Listing 1.7.[7] In this case, the optimum answer set is the thirteenth answer set inspected by the solver (`Answer: 13`). Such answer set describes the Boolean logic model given in Fig. 1.2(c). Furthermore, 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. (1.5)), and 7 for the model size (Eq. (1.3)).

Listing 1.9: Learning an optimum Boolean logic model

```
$ gringo toy.lp learning.lp | clasp --quiet=1
clasp version 3.0.2
Reading from stdin
Solving...
Answer: 13
formula(e,1) formula(d,2) formula(g,3) formula(f,4)\
dnf(1,1) dnf(1,2) dnf(2,4) dnf(3,14) dnf(4,15)\
clause(1,b,1) clause(2,c,1) clause(4,a,1)\
clause(14,e,1) clause(14,c,-1) clause(15,d,1) clause(15,e,1)
Optimization: 88 7
OPTIMUM FOUND

Models       : 13
  Optimum    : yes
Optimization : 88 7
Calls        : 1
Time         : 0.005s (Solving:0.00s 1st Model:0.00s Unsat:0.00s)
CPU Time     : 0.000s
```

Next, the enumeration capabilities of an ASP solver like *clasp* [GEB 07] can be used to find not only one optimal model but all (nearly) optimal models as described earlier. Considering tolerance $t_{rss} = 8$ ($\sim 10\%$ of the optimum residual sum of squares) and size tolerance $t_{size} = 3$, we enumerate all models such that

$$\Theta_{rss_k}((V, \phi), \xi) \leq \Theta_{rss_k}((V, \phi_{opt}), \xi) + t_{rss} = 96$$
$$\Theta_{size}((V, \phi)) \leq \Theta_{size}((V, \phi_{opt})) + t_{size} = 10.$$

7. Using the option `--quiet=1` only the last (optimum) answer set is printed. Notice that the solver prints all the atoms in the answer set in a single line but we have (manually) introduced breaklines to improve readability.

In this example, there are 5 (nearly) optimal Boolean logic models as we show in Listing 1.10.[8] Interestingly, even for this small example, the symmetry-breaking constraints make a significant difference. We note that running the same program but without the symmetry-breaking constraints, yields 17 Boolean logic models instead of only 5. Notably, in real-world problem instances, exploiting these symmetries significantly reduces the number of solutions (without missing any input-output behavior) and hence, it facilitates their post processing and interpretation. Once we have enumerated all (nearly) optimal Boolean logic models with respect to certain tolerances, we can identify the logical input-output behaviors they describe. Towards this end, we have developed a simple algorithm that (using ASP) systematically compares all pairs of models looking for at least one experimental condition, i.e. a clamping assignment, generating a different response over the readouts nodes. We refrain from showing here the algorithm and additional encoding. Nonetheless, it is worth noting that the ASP encoding for deciding whether two Boolean logic models, there exists at least one experimental condition generating a different response over the readout nodes, is a rather straightforward extension from the rules given in Section 1.3. Following with our example, over the 5 (nearly) optimal Boolean logic models enumerated in Listing 1.10, we found 3 logical input-output behaviors.

Listing 1.10: Enumeration of all (nearly) optimal Boolean logic models

```
$ gringo toy.lp learning.lp -c maxrss=96 -c maxsize=10 |\
  clasp --opt-mode=ignore -n0 --quiet
clasp version 3.0.2
Reading from stdin
Solving...
SATISFIABLE

Models      : 5
Calls       : 1
Time        : 0.002s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s)
CPU Time    : 0.000s
```

Importantly, results on a real-case study [GUZ 13] underscore the importance of exploring exhaustively the family of models and take into account experimental noise in order to obtain an adequate picture of the feasible model solutions. 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 logical 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.

8. Option `--opt-mode=ignore` tells the solver to ignore optimize statements; option `-n0` tells the solver to enumerate all solutions; and option `--quiet` avoids printing enumerated solutions.

Hence, in practice this approach may provide a way to extract robust insights despite the high variability. Also, thanks to our exhaustive characterization of these models, we can determine unambiguously which logical interactions are functional in all or none of the models and determine groups of mutually-exclusive mechanisms. Hence, our approach permits the study of the internal combinatorics leading to the variability of the system functioning and provides a tool to suggest new experiments towards the discrimination of plausible input-output behaviors given the available experimental setup. On the computational side, from multiple independent runs (1000 runs with an average of 1000 seconds per run) of the existing genetic algorithm implementation [TER 12], only 20% of them have converged to Boolean logic models within the allowed tolerance. Furthermore, among these runs, the genetic algorithm has retrieved approximately half of the plausible input-output behaviors identified using ASP with an evident bias towards the most common ones. Hence, those behaviors described only by a few logical networks are very unlikely to be found with such stochastic approaches.

1.3.4. *Minimal intervention strategies*

Background

Once a biological system has been properly characterized in agreement with prior knowledge and experimental observations, a major challenge in systems biology is how to systematically control its state, leading to the field of experimental design. Importantly, progress in this area may have a crucial impact on bio-medical research, drug target identification and diagnosis. In fact, the problem of identifying "key-players" in biological systems has been addressed for metabolic, gene, and signaling networks. However, the underlying mathematical formalisms for each of these biological networks allow for different computational approaches.

In the context of logic-based models for signaling networks, this question has been recently addressed by [ABD 08, SAM 10, WAN 11, LAY 11]. Among them, in what follows we focus on the problem defined in [SAM 10]. Based on earlier work [KLA 06a] on metabolic networks, the notion of *minimal intervention sets* was introduced and dedicated algorithms were developed to compute them. Intuitively, an "intervention set" represents a set of *knock-ins* and *knock-outs*. Examples for knock-ins are mutations leading to constitutively activated species or a continuous stimulation with external signals whereas knock-outs may correspond to gene knock-outs or inhibition of a certain species by various experimental tools such as small-molecule drugs, antibodies, or RNAi. In fact, our notion of *clamping assignments* introduced in Section 1.3 was originally motivated as an abstraction (closer to standard logic terminology) of intervention sets. Then, we aim at identifying possible intervention sets leading to a logical steady state describing a specific biological outcome of interest. Furthermore, the aforecited work propose the usage of a three-valued logic which,

though not mentioned, it correspond exactly to Kleene's logic [KLE 50]. It is also observed that under such a three-valued logic, a unique logical steady state follows for any (clamped) logical network and initial state. Hence, in contrast to the previous section where we have restricted ourselves to logical networks under Boolean logic and without feedback-loops, in this chapter, we consider logical networks under Kleene's logic and without any further restrictions. Therefore, the method presented in this section is not only relevant for Boolean logic models of immediate-early response, but for logical networks in general.

Unfortunately, the dedicated algorithms presented in [SAM 10] are computationally demanding due to the highly combinatorial mechanisms in logical signaling networks. Therefore, they are limited to compute small intervention sets and fail to scale over large-scale networks. In general, multiple interventions are necessary to cope with robustness and cellular complexity [STE 04]. Moreover, authors in [SAM 10] have considered looking for interventions in a single logical network. However, as we have shown above, if the inherent experimental noise is considered there are many logical networks compatible with a given dataset of experimental observations. Thus, identified interventions should fulfill the desired goals in every possible logical network. Concretely, the mentioned limitations make it hard to prove that the identified solutions are biologically robust to small perturbations of the system or its environment. Thus, in order to overcome such limitations, more elaborate and more powerful computational methods are needed towards large-scale systems and robust solutions.

Problem

Next, we provide a formal characterization of intervention strategies in logical signaling networks based on the notions introduced in Section 1.3. Given a logical network, the aim of an *intervention strategy* is to identify an intervention set that leads to a steady state satisfying a given goal under some side constraints. In fact, the concepts of an *intervention* (I), *goal* (G), and side *constraints* (C) can be captured as partial two-valued assignments. Moreover, both intervention sets and side constraints are considered clamping assignments as defined in Section 1.3 To be more precise, given a logical network (V, ϕ), an *intervention scenario* is a pair (G, C) of partial two-value assignments over V where C is considered also as a clamping assignment, and an *intervention set* is a clamping assignment I over a set of intervention variables $X \subseteq V$. Recall that for truth assignments A, B we defined the composition of assignments $A \circ B = (A \setminus \overline{B}) \cup B$ where $\overline{B} = \{v \mapsto \overline{s} \mid v \mapsto s \in B\}$ and $\overline{\boldsymbol{t}} = \boldsymbol{f}$, $\overline{\boldsymbol{f}} = \boldsymbol{t}$, $\overline{\boldsymbol{u}} = \boldsymbol{u}$.

Let (V, ϕ) be a logical network, let (G, C) be an intervention scenario, and $X \subseteq V$ be a set of intervention variables. An intervention set I over X is an *intervention strategy* for (G, C) with respect to (V, ϕ), if for some $j \geq 0$, we have that

$$\Omega^{j}_{(V,\phi|_{C \circ I})}(A_u) = \Omega^{j+1}_{(V,\phi|_{C \circ I})}(A_{\boldsymbol{u}}) \qquad G \subseteq \Omega^{j}_{(V,\phi|_{C \circ I})}(A_{\boldsymbol{u}})$$

with $A_{\boldsymbol{u}} = \{v \mapsto \boldsymbol{u} \mid v \in V\}$. In words, $\Omega^j_{(V,\phi|_{C\circ I})}(A_{\boldsymbol{u}})$ is a steady state of the clamped network $(V, \phi|_{C\circ I})$ satisfying the goal conditions in G. Notice the composition of $C \circ I$ indicating that clampings in the intervention set I overwrite clampings in the side constraints C. Finally, the *intervention set problem* consists in deciding whether there is an intervention strategy for an intervention scenario (G, C) wrt a logical network (V, ϕ).

For illustration, let us consider the toy logical network in Figure 1.1(a) along with the intervention scenarios $(G_1, C_1) = (\{o_1 \mapsto \boldsymbol{f}, o_2 \mapsto \boldsymbol{t}\}, \{i_1 \mapsto \boldsymbol{t}\})$ and $(G_2, C_2) = (\{a \mapsto \boldsymbol{t}\}, \emptyset)$. In this example, the first scenario requires the inhibition of o_1 together with the activation of o_2, given that i_1 is stimulated. Furthermore, the second scenario requires the activation of a without any additional side constraints. Next, the intervention set $\{b \mapsto \boldsymbol{f}, e \mapsto \boldsymbol{f}, f \mapsto \boldsymbol{t}\}$ where b and e are inhibited and f is stimulated, satisfies both scenarios yielding the two steady states, respectively:

$$\left\{ \begin{array}{ccc} i_1 \mapsto \boldsymbol{t} \\ i_2 \mapsto \boldsymbol{u} \end{array} \quad \begin{array}{ccc} a \mapsto \boldsymbol{t} & d \mapsto \boldsymbol{f} & g \mapsto \boldsymbol{t} \\ b \mapsto \boldsymbol{f} & e \mapsto \boldsymbol{f} & o_1 \mapsto \boldsymbol{f} \\ c \mapsto \boldsymbol{f} & f \mapsto \boldsymbol{t} & o_2 \mapsto \boldsymbol{t} \end{array} \right\}$$

$$\left\{ \begin{array}{ccc} i_1 \mapsto \boldsymbol{u} \\ i_2 \mapsto \boldsymbol{u} \end{array} \quad \begin{array}{ccc} a \mapsto \boldsymbol{t} & d \mapsto \boldsymbol{f} & g \mapsto \boldsymbol{t} \\ b \mapsto \boldsymbol{f} & e \mapsto \boldsymbol{f} & o_1 \mapsto \boldsymbol{f} \\ c \mapsto \boldsymbol{f} & f \mapsto \boldsymbol{t} & o_2 \mapsto \boldsymbol{t} \end{array} \right\}$$

Authors in [SAM 10], were particularly interested in enumerating all minimal (bounded) intervention strategies with respect to a single logical network. However, as we have illustrated above and other authors have shown by considering real-world networks and data [SAE 09, CHE 09, GUZ 13], it often happens that *the* model is non-identifiable. Therefore, as one can argue that several logical networks can describe a given biological system equally or similarly well, identified intervention strategies should fulfill all intervention scenarios in every possible logical network. Towards this end, herein we extend the problem settings in order to consider a family of logical networks, for instance, resulting from the enumeration of (nearly) optimal Boolean logic models described above.

Now, let us define further intervention strategies relying on a finite family $(V, \phi_i)_{i\in N}$ of logical networks, a finite family $(G_j, C_j)_{j\in J}$ of intervention scenarios and k some positive integer.

– A *multi-scenario intervention strategy* for $(G_j, C_j)_{j\in J}$ wrt $(V, \phi_i)_{i\in N}$ is an intervention strategy for each (G_j, C_j) wrt (V, ϕ_i) for each $j \in J$ and $i \in N$.

– A *bounded intervention strategy* for $(G_j, C_j)_{j\in J}$ wrt $(V, \phi_i)_{i\in N}$ and k is a multi-scenario intervention strategy for $(G_j, C_j)_{j\in J}$ wrt $(V, \phi_i)_{i\in N}$ of cardinality $k' \leq k$.

– A *minimal bounded intervention strategy* for $(G_j, C_j)_{j\in J}$ wrt $(V, \phi_i)_{i\in N}$ and k is a $\subseteq$-minimal multi-scenario intervention strategy for $(G_j, C_j)_{j\in J}$ wrt $(V, \phi_i)_{i\in N}$ of cardinality $k' \leq k$.

In what follows, we focus on the enumeration of all minimal (bounded) intervention strategies for given families of intervention scenarios $(G_j, C_j)_{j\in J}$ and logical networks $(V, \phi_i)_{i\in N}$.

Encoding

Again, the representation of the problem instance is an extension from the one described in Listing 1.1 in order to describe a finite family of logical networks and clamping assignments. To be more precise, instead of having facts over predicates `formula/2`, we consider facts over predicates `formula/3` as follows. Let $(V, \phi_i)_{i\in N}$ be a finite family of logical networks. The facts `formula(`i`,`v`,`$s_{\phi_i(v)}$`)` map variables $v \in V$ to their corresponding formulas $\phi_i(v)$ for each $i \in N$. Meanwhile, facts over predicates `variable/1`, `dnf/2` and `clause/3` remain the same as in Listing 1.1. We use facts over predicate `candidate/1` to denote the intervention variables that can be part of an intervention set. This allows us to control on which variables interventions are permitted, for example one can exclude interventions over constrained or goal variables. Next, we represent the family of intervention scenarios $(G_j, C_j)_{j\in J}$ using predicates `scenario/1`, `goal/3`, and `constrained/3`. The facts `scenario(`j`)` denote the scenarios to consider. The facts `goal(`j`,`v`,`s`)` with $s = 1$ (resp. $s = -1$) if $G_j(v) = \boldsymbol{t}$ (resp. $G_j(v) = \boldsymbol{f}$) and `constrained(`j`,`v`,`s`)` with $s = 1$ (resp. $s = -1$) if $C_j(v) = \boldsymbol{t}$ (resp. $C_j(v) = \boldsymbol{f}$) denote the respective intervention goals and side constraints in each scenario (G_j, C_j).

Listing 1.11 shows the instance representation of our toy example logical network in Figure 1.1(a) together with the two intervention scenarios $(G_1, C_1) = (\{o_1 \mapsto \boldsymbol{f}, o_2 \mapsto \boldsymbol{t}\}, \{i_1 \mapsto \boldsymbol{t}\})$ and $(G_2, C_2) = (\{a \mapsto \boldsymbol{t}\}, \emptyset)$. Notably, for the sake of understanding, we consider a toy example with a single logical network. But in general, this instance representation and the logic program given below, support several logical networks.

Listing 1.11: Toy example problem instance (`toy.lp`)

```
variable(i1). variable(i2). variable(o2). variable(o1).
variable(a).  variable(b).  variable(c).  variable(d).
variable(e).  variable(f).  variable(g).

candidate(i2). candidate(b).  candidate(c).  candidate(d).
candidate(e).  candidate(f).  candidate(g).

formula(1,a,0). formula(1,b,2). formula(1,c,1). formula(1,d,4).
formula(1,e,3). formula(1,f,6). formula(1,g,5). formula(1,o1,4).
formula(1,o2,7).

dnf(0,5). dnf(1,6). dnf(1,0). dnf(2,3). dnf(3,7).
dnf(4,1). dnf(5,2). dnf(6,4). dnf(6,6). dnf(7,4).

clause(0,b,1).  clause(1,c,1).  clause(2,f,1).  clause(3,a,1).
clause(3,i1,1). clause(4,g,1).  clause(5,d,-1). clause(6,e,1).
clause(7,i2,1). clause(7,i1,-1).
```

```
18
19 scenario(1). goal(1,o1,-1). goal(1,o2,1). constrained(1,i1,1).
20 scenario(2). goal(2,a,1).
```

Next we describe our encoding for solving the minimal intervention set problem as described earlier. Our ASP encoding is shown in Listing 1.12.

Listing 1.12: Logic program for finding intervention strategies (`control.lp`)

```
1  goal(T,S)         :- goal(_,T,S).
2  goal(T)           :- goal(T,_).
3  constrained(Z,E)  :- constrained(Z,E,_).
4  constrained(E)    :- constrained(_,E).
5  model(M)          :- formula(M,_,_).
6  formula(W,D)      :- formula(_,W,D).
7
8  satisfy(V,W,S) :- formula(W,D); dnf(D,C); clause(C,V,S).
9  closure(V,T)   :- goal(V,T).
10 closure(V,S*T) :- closure(W,T); satisfy(V,W,S); not goal(V,-S*T).
11
12 { intervention(V,S) : closure(V,S) , candidate(V) }.
13 :- intervention(V,1); intervention(V,-1).
14 intervention(V) :- intervention(V,S).
15
16 eval(M,Z,V,S) :- scenario(Z); intervention(V,S); model(M).
17 eval(M,Z,E,S) :- model(M); constrained(Z,E,S);
18                  not intervention(E).
19 free(M,Z,V,D) :- formula(M,V,D); scenario(Z);
20                  not constrained(Z,V); not intervention(V).
21
22 eval_clause(M,Z,C,-1) :- clause(C,V,S); eval(M,Z,V,-S); model(M).
23
24 eval(M,Z,V, 1) :- free(M,Z,V,D); dnf(D,C);
25                   eval(M,Z,W,T) : clause(C,W,T).
26 eval(M,Z,V,-1) :- free(M,Z,V,D); eval_clause(M,Z,C,-1) : dnf(D,C).
27
28 :- goal(Z,T,S); model(M); not eval(M,Z,T,S).
29
30 #const maxsize=0.
31 :- maxsize>0; maxsize + 1 { intervention(X) }.
32
33 #show intervention/2.
```

In Lines 1-6 we define auxiliary domain predicates used in the remainder of the encoding. Lines 8-10 deserve closer attention since they allow us to reduce significantly the search space of candidate solutions. We incorporate a preprocessing step introduced in [SAM 10] that prunes variable assignments that can never be part of a minimal intervention strategy. The idea is to inductively collect all assignments that could be used to support a goal. First we gather all assignments that make a literal in a clause true and associate it with variable of the associated DNF (Line 8). Starting from the assignments that can satisfy a goal literal directly (Line 9), we inductively consider variable

assignments (Line 10) that can support the assignments collected so far. Let us illustrate this on our toy example. In order to satisfy `goal(1,o2,1)`, one would never consider to intervene variables f or g negatively. Since both reach $o2$ positively, only positive interventions on them could help. The same happens for variable e. However, since e also reaches $o1$ positively and we have `goal(1,o1,-1)`, a negative intervention of e could help for this goal. Next, we use a choice rule in Line 12 to generate candidate solutions. We only choose interventions collected in the preprocessing step above. The integrity constraint in Line 13 eliminates contradictory interventions, e.g. `intervention(e,1)` and `intervention(e,-1)`. Whereas Line 14 simply projects the intervention set to the intervened variables regardless of their signature. For example, one could generate the intervention set consisting of `intervention(e,1)` and `intervention(c,-1)`.

In lines 16-26 elaborate on the rules from Listings 1.3, 1.4 and 1.6 given in Section 1.3 in order to consider several logical networks simultaneously and compute the fixpoint for each of them accordingly. To be more precise, we need to describe which variables are clamped (in all networks) according to the side constraints C_j in each scenario j and the intervention set I, namely, $(V, \phi_i|_{C_j \circ I})$. Towards this end, we use the predicate `eval/4`, namely `eval(`i`,`j`,`v`,`s`)` to represent that in the network (V, ϕ_i) and intervention scenario (G_j, C_j) the variable v is clamped to value s. Following the previous example, this will generate predicates `eval(1,1,i1,1)`, `eval(1,1,e,1)`, `eval(1,2,e,1)`, `eval(1,1,c,-1)` and `eval(1,2,c,-1)`. The remaining rules are adapted accordingly in order to compute for each logical network (V, ϕ_i) and intervention scenario (G_j, C_j), the corresponding fixpoint of $\Omega_{(V,\phi_i|_{C_j \circ I})}$. For our example, we can see how the positive intervention over e is propagated as follows. Since variable f is not intervened, its formula ($f \mapsto e \vee g$) described by predicates `formula(1,f,6)`, `dnf(6,4)`, `dnf(6,6)`, `clause(4,g,1)` and `clause(6,e,1)`, is "free" in all scenarios, namely, `free(1,1,f,6)` and `free(1,2,f,6)`. Furthermore, given that we have `dnf(6,6)` related only to `clause(6,e,1)` and e was intervened positively, the truth value for f is propagated in all scenarios regardless of g, namely, `eval(1,1,f,1)` and `eval(1,2,f,1)`.

Line 28 declares an integrity constraint in order to eliminate answer sets describing intervention sets that do not satisfy all goals in every logical network and intervention scenario. Finally, the statements in Line 30 and 31 allows us to optionally bound the problem by considering only intervention sets up to a given size.

Solving

Normally ASP solvers allow for computing cardinality-minimal solutions whereas we are interested in finding subset-minimal solutions. In [KAM 13] we have shown how one can overcome this limitation by means of meta-programming and disjunctive logic programs [GEB 11a] or by using a specialized solver like *hclasp* [GEB 13]. However, herein we leverage the functionality recently introduced in *clasp* 3 series

which allow for computing subset-minimal solutions *out-of-the-box* by incorporating the features from *hclasp*. Importantly, this requires to use very specific command-line options for *clasp*. We refer the reader to *clasp*'s documentation for more details.

In Listing 1.13 we show the intervention strategies found for the toy instance described in Listing 1.11.

Listing 1.13: Computing all MISs for the toy instance

```
$ gringo control.lp toy.lp |\
  clasp --dom-pref=32 --dom-mod=6 --heu=domain -n0
clasp version 3.0.2
Reading from stdin
Solving...
Answer: 1
intervention(e,-1) intervention(f,1) intervention(b,-1)
Answer: 2
intervention(i2,-1) intervention(f,1) intervention(b,-1)
Answer: 3
intervention(e,-1) intervention(g,1) intervention(b,-1)
Answer: 4
intervention(i2,-1) intervention(g,1) intervention(b,-1)
Answer: 5
intervention(d,-1) intervention(f,1) intervention(b,-1)
Answer: 6
intervention(g,1) intervention(d,-1) intervention(b,-1)
Answer: 7
intervention(c,-1) intervention(f,1)
Answer: 8
intervention(g,1) intervention(c,-1)
Answer: 9
intervention(e,1) intervention(c,-1)
SATISFIABLE

Models       : 9
Calls        : 1
Time         : 0.001s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s)
CPU Time     : 0.000s
```

Interestingly, experiments over real-world biological networks in [KAM 13] have shown that our approach outperforms the previous dedicated algorithms [SAM 10] in up to four orders of magnitude (for small number of interventions (≤ 3) still feasible for the algorithm). This was not very surprising since such algorithms are based on a standard breadth-first search using additional techniques for search space reduction. More importantly, using ASP we are able to search for significantly larger intervention strategies or even solve the unbounded problem (that is, no limit in the number of interventions). While considering a small number of interventions the number of solutions (that is, intervention strategies) is in the order of tens, with a larger number of interventions we have found thousands of feasible solutions. Furthermore, being able to solve the unbounded problem allows us to completely characterize the set of

solutions. Notably, in the light of such a large number of solutions, the way to select among them arises. Herein, we have introduced an extension of the intervention sets problem in order to consider a family of plausible logical networks as those identified in the previous section. This way, we expect to reduce the number of solutions by selecting the more robust of them. That is, intervention sets satisfying each scenario in all logical networks.

Listing 1.14: Software toolbox: *caspo*

```
$ caspo learn pkn.sif dataset.csv 30 --fit 0.1 --size 2

Wrote networks.csv

$ caspo control networks.csv scenarios.csv

Wrote strategies.csv

$ caspo analyze --networks networks.csv --midas dataset.csv 30\
                --strategies strategies.csv

Wrote networks-stats.csv
Wrote networks-mse.csv

Searching input-output behaviors... \
          5 behaviors have been found over 178 logical networks.

Wrote behaviors.csv
Wrote behaviors-mse-len.csv
Wrote variances.csv
Wrote core.csv
Wrote strategies-stats.csv
Wrote summary.txt

caspo analytics summary
=======================
Total Boolean logic networks: 178
Total I/O Boolean logic behaviors: 5
Weighted MSE: 0.0395
Core predictions: 78.12%
Total intervention strategies: 6
```

1.3.5. *Software toolbox: caspo*

In practice, interactions graphs, experimental datasets, logical networks, and similar knowledge in systems biology is often (publicly) available in different kind of "standard" formats. Clearly, converting such a knowledge from their corresponding format to a set of logic facts, e.g. in Listing 1.7 and Listing 1.11, is a tedious and error-prone task if do it by hand. In fact, this can be easily automated by using any

popular scripting language, e.g. python. [9] Analogously, the resulting answer sets from *clasp* can be converted back to "standard" formats for subsequent analysis with available tools or even visualized in order to facilitate their interpretations. Hence, towards this end we have implemented the python package *caspo* which is freely available for download. [10] The aim of *caspo* is to implement a pipeline for automated reasoning on logical signaling networks providing a powerful and easy-to-use software tool for systems biologists. In particular, both problems presented herein together with other related features are available. For the sake of illustration, in Listing 1.14 we show the usage of *caspo* over a real-world instance from the "DREAM" challenge [PRI 11]. For more details on the installation, usage, and available features we refer the reader to the online documentation. More broadly, *caspo* is part of *BioASP*, a collection of python packages leveraging the computational power of ASP for systems biology. [11]

1.4. Conclusion

Systems biology is a research field at the crossover of biology, informatics, and mathematics. Its central task is to decipher the functioning of the so-called biological networks mediating between DNA products, the environment, and the phenotype. Among these networks, signal transduction networks are crucial for the understanding of the response to extra- and intracellular perturbations. Notably, in order to cope with the increasing complexity of large-scale networks, the development of mathematical and computational modeling approaches is a major goal in the field. In this context, despite their high-level of abstraction, logical networks provide a powerful qualitative approach for modeling large-scale biological systems. Importantly, several authors have shown that the response of signaling networks can be appropriately modeled with (Boolean) logical networks. Nevertheless, several challenging problems remain open. For example, the question of identifying the precise logic rules underlying the system being studied or finding appropriate interventions allowing to control it. Existing approaches implementing dedicated algorithms to address these problems have been relatively successful. However, they possess certain shortcomings regarding the scalability and exhaustiveness over large search spaces. In fact, this incompleteness in the search may significantly compromise the robustness of solutions found while it limits the insights provided to biologists.

In contrast to previous approaches, herein we propose to look for robust insights by reasoning over the complete space of feasible solutions. Towards this end, we strongly rely on methods from the area of knowledge representation and reasoning such as Answer Set Programming (ASP). Available systems nowadays provide a rich

9. http://www.python.org/
10. http://bioasp.github.io/caspo/
11. http://bioasp.github.io/

yet simple modeling language, high-performance solving capacities, and automated reasoning modes. Thus, ASP provides a powerful framework for addressing hard combinatorial problems in systems biology by reasoning over the complete search space. In order to illustrate this, we have shown how the aforementioned problems on logical networks can be successfully modeled and solved using ASP. Interestingly, in both cases the computational performance is significantly improved with respect to dedicated algorithms. But more importantly, the exhaustive nature of ASP allows us to find feasible solutions that were missing when using the existing methods. Altogether, our work constitutes a pipeline for automated reasoning on logical signaling networks providing robust insights to systems biologists.

Finally, as mentioned earlier, integrative modeling approaches considering multiple levels and time-scales of causation pose a very challenging goal in systems biology. In order to achieve this goal, we believe that more sophisticated computational methods are needed in order to integrate qualitative and quantitative knowledge. In particular, hybrid reasoning systems leveraging the expressiveness of several technologies and modeling approaches appears as a very promising track for future research and development. Hopefully, advances on this subject will foster the usage of knowledge representation and reasoning methodologies in systems biology towards a better understanding of life.

1.5. Acknowledgements

We would like to thank our collaborators: Axel von Kamp, Carito Guziolowski, Federica Eduati, Jacques Nicolas, Julio Saez-Rodriguez, Martin Gebser, Regina Samaga, Roland Kaminski, Steffen Klamt, Sven Thiele, and Thomas Cokelaer. This work was partially funded by the projects ANR-10-BLANC-0218 and DFG grant SCHA 550/10-1.

1.6. Bibliography

[ABD 08] ABDI A., TAHOORI M. B., EMAMIAN E. S., "Fault diagnosis engineering of digital circuits can identify vulnerable molecules in complex cellular pathways", *Science Signaling*, vol. 1, num. 42, 2008.

[AKU 00] AKUTSU T., MIYANO S., KUHARA S., "Inferring qualitative relations in genetic networks and metabolic pathways", *Bioinformatics*, vol. 16, num. 8, p. 727-734, July 2000.

[ALE 10] ALEXOPOULOS L. G., SAEZ-RODRIGUEZ J., COSGROVE B., LAUFFENBURGER D. A., SORGER P., "Networks inferred from biochemical data reveal profound differences in toll-like receptor and inflammatory signaling between normal and transformed hepatocytes", *Molecular & Cellular Proteomics*, vol. 9, num. 9, p. 1849-1865, 2010.

[APT 82] APT K. R., EMDEN M. H. V., "Contributions to the Theory of Logic Programming", *ACM*, vol. 29, num. 3, p. 841-862, ACM, July 1982.

[BAN 08] BANGA J., "Optimization in computational systems biology", *BMC systems biology*, vol. 2, num. 1, Page 47, 2008.

[BAR 03] BARAL C., *Knowledge Representation, Reasoning and Declarative Problem Solving*, Cambridge University Press, 2003.

[BAR 04] BARAL C., CHANCELLOR K., TRAN N., TRAN N., JOY A., BERENS M., "A knowledge based approach for representing and reasoning about signaling networks", *Proceedings of the Twelfth International Conference on Intelligent Systems for Molecular Biology/Third European Conference on Computational Biology (ISMB'04/ECCB'04)*, p. 15-22, 2004.

[BER 13] BERESTOVSKY N., NAKHLEH L., "An Evaluation of Methods for Inferring Boolean Networks from Time-Series Data", *PLoS ONE*, vol. 8, num. 6, Pagee66031, 2013.

[CAL 10] CALZONE L., TOURNIER L., FOURQUET S., THIEFFRY D., ZHIVOTOVSKY B., BARILLOT E., ZINOVYEV A., "Mathematical modelling of cell-fate decision in response to death receptor engagement", *PLoS Computational Biology*, vol. 6, num. 3, February 2010.

[CER 11] CERAMI E. G., GROSS B. E., DEMIR E., RODCHENKOV I., BABUR Ö., ANWAR N., SCHULTZ N., BADER G. D., SANDER C., "Pathway Commons, a web resource for biological pathway data.", *Nucleic Acids Research*, vol. 39, num. Database issue, p. D685-D690, Oxford University Press, 2011.

[CHE 09] CHEN W. W., SCHOEBERL B., JASPER P. J., NIEPEL M., NIELSEN U. B., LAUFFENBURGER D. A., SORGER P., "Input-output behavior of ErbB signaling pathways as revealed by a mass action model trained against dynamic data", *Molecular Systems Biology*, vol. 5, num. 1, January 2009.

[COL 13] COLLET G., EVEILLARD D., GEBSER M., PRIGENT S., SCHAUB T., SIEGEL A., THIELE S., "Extending the Metabolic Network of Ectocarpus Siliculosus using Answer Set Programming", 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, p. 245-256, 2013.

[DUR 11] DURZINSKY M., MARWAN W., OSTROWSKI M., SCHAUB T., WAGLER A., "Automatic Network Reconstruction using ASP", *Theory and Practice of Logic Programming*, vol. 11, num. 4-5, p. 749-766, 2011.

[FAY 11] FAYRUZOV T., JANSSEN J., VERMEIR D., CORNELIS C., COCK M. D., "Modelling gene and protein regulatory networks with Answer Set Programming.", *International Journal of Data Mining and Bioinformatics*, vol. 5, num. 2, p. 209-229, 2011.

[FIS 07] FISHER J., HENZINGER T. A., "Executable cell biology.", *Nature biotechnology*, vol. 25, num. 11, p. 1239-1249, November 2007.

[FIT 85] FITTING M., "A Kripke-Kleene Semantics for Logic Programs", *Journal of Logic Programming*, vol. 2, num. 4, p. 295-312, 1985.

[FRE 04] FREITAS A., "A critical review of multi-objective optimization in data mining", *ACM SIGKDD Explorations Newsletter*, vol. 6, num. 2, Page 77, December 2004.

[GAL 93] GALLO G., LONGO G., PALLOTTINO S., NGUYEN S., "Directed Hypergraphs and Applications", *Discrete Applied Mathematics*, vol. 42, num. 2-3, p. 177-201, 1993.

[GEB 07] GEBSER M., KAUFMANN B., NEUMANN A., SCHAUB T., "clasp: A Conflict-Driven Answer Set Solver", 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, p. 260-265, 2007.

[GEB 10] GEBSER M., GUZIOLOWSKI C., IVANCHEV M., SCHAUB T., SIEGEL A., THIELE S., VEBER P., "Repair and prediction (under inconsistency) in large biological networks with answer set programming", LIN F., SATTLER U., Eds., *Proceedings of the Twelfth International Conference on Principles of Knowledge Representation and Reasoning (KR'10)*, AAAI Press, p. 497-507, 2010.

[GEB 11a] GEBSER M., KAMINSKI R., SCHAUB T., "Complex Optimization in Answer Set Programming", *Theory and Practice of Logic Programming*, vol. 11, num. 4-5, p. 821-839, 2011.

[GEB 11b] GEBSER M., SCHAUB T., THIELE S., VEBER P., "Detecting Inconsistencies in Large Biological Networks with Answer Set Programming", *Theory and Practice of Logic Programming*, vol. 11, num. 2-3, p. 323-360, 2011.

[GEB 12a] GEBSER M., KAMINSKI R., KAUFMANN B., SCHAUB T., *Answer Set Solving in Practice*, Synthesis Lectures on Artificial Intelligence and Machine Learning, Morgan and Claypool Publishers, 2012.

[GEB 12b] GEBSER M., KAUFMANN B., SCHAUB T., "Multi-threaded ASP Solving with clasp", *Theory and Practice of Logic Programming*, vol. 12, num. 4-5, p. 525-545, 2012.

[GEB 13] GEBSER M., KAUFMANN B., OTERO R., ROMERO J., SCHAUB T., WANKO P., "Domain-specific Heuristics in Answer Set Programming", DESJARDINS M., LITTMAN M., Eds., *Proceedings of the Twenty-Seventh National Conference on Artificial Intelligence (AAAI'13)*, AAAI Press, p. 350-356, 2013.

[GEL 88] GELFOND M., LIFSCHITZ V., "The Stable Model Semantics for Logic Programming", KOWALSKI R., BOWEN K., Eds., *Proceedings of the Fifth International Conference and Symposium of Logic Programming (ICLP'88)*, MIT Press, p. 1070-1080, 1988.

[GUZ 12] GUZIOLOWSKI C., KITTAS A., DITTMANN F., GRABE N., "Automatic generation of causal networks linking growth factor stimuli to functional cell state changes", *FEBS Journal*, vol. 279, num. 18, p. 3462-3474, Blackwell Publishing Ltd, 2012.

[GUZ 13] GUZIOLOWSKI C., VIDELA S., EDUATI F., THIELE S., COKELAER T., SIEGEL A., SAEZ-RODRIGUEZ J., "Exhaustively characterizing feasible logic models of a signaling network using Answer Set Programming", *Bioinformatics*, vol. 29, num. 18, p. 2320-2326, 2013.

[IDE 00] IDEKER T. E., THORSSON V., KARP R. M., "Discovery of regulatory interactions through perturbation: inference and experimental design", ALTMAN R. B., DUNKER A. K., HUNTER L., KLEIN T. E., Eds., *Pacific Symposium on Biocomputing*, vol. 5, p. 305-316, 2000.

[IDE 01] IDEKER T., GALITSKI T., HOOD L., "A new approach to decoding life: systems biology.", *Annual review of genomics and human genetics*, vol. 2, p. 343-372, 2001.

[INO 11] INOUE K., "Logic Programming for Boolean Networks", WALSH T., Ed., *Proceedings of the Twenty-second International Joint Conference on Artificial Intelligence (IJCAI'11)*, IJCAI/AAAI, p. 924-930, 2011.

[KAM 13] KAMINSKI R., SCHAUB T., SIEGEL A., VIDELA S., "Minimal Intervention Strategies in Logical Signaling Networks with Answer Set Programming", *Theory and Practice of Logic Programming*, vol. 13, num. 4-5, p. 675-690, 2013.

[KAN 10] KANEHISA M., GOTO S., FURUMICHI M., TANABE M., M. HIRAKAWA M., "KEGG for representation and analysis of molecular networks involving diseases and drugs.", *Nucleic Acids Research*, vol. 38, num. Database issue, January 2010.

[KAU 69] KAUFFMAN S., "Metabolic stability and epigenesis in randomly constructed genetic nets", *Journal of Theoretical Biology*, vol. 22, num. 3, p. 437-467, February 1969.

[KIT 02] KITANO H., "Systems biology: a brief overview.", *Science*, vol. 295, num. 5560, p. 1662-1664, 2002.

[KLA 06a] KLAMT S., "Generalized concept of minimal cut sets in biochemical networks", *Biosystems*, vol. 83, num. 2-3, p. 233-247, January 2006.

[KLA 06b] 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*, vol. 7, num. 1, Page 56, 2006.

[KLE 50] KLEENE S. C., *Introduction to metamathematics*, Princeton, NJ, 1950.

[KOH 10] KOHL P., CRAMPIN E. J., QUINN T. A., NOBLE D., "Systems Biology: An Approach", *Clinical Pharmacology & Therapeutics*, vol. 88, num. 1, p. 25-33, June 2010.

[KRE 09] KREUTZ C., TIMMER J., "Systems biology: experimental design", *FEBS Journal*, vol. 276, num. 4, p. 923-942, January 2009.

[KUE 07] KUEPFER L., PETER M., SAUER U., STELLING J., "Ensemble modeling for analysis of cell signaling dynamics", *Nature biotechnology*, vol. 25, num. 9, p. 1001-1006, September 2007.

[LÄH 03] LÄHDESMÄKI H., SHMULEVICH I., YLI-HARJA O., "On learning gene regulatory networks under the Boolean network model", *Machine learning*, vol. 52, num. 1-2, p. 147-167, 2003.

[LAY 11] LAYEK R., DATTA A., BITTNER M., DOUGHERTY E. R., "Cancer therapy design based on pathway logic", vol. 27, num. 4, p. 548-555, February 2011.

[LIA 98] LIANG S., FUHRMAN S., SOMOGYI R., "REVEAL, A General Reverse Engineering Algorithm for Inference of Genetic Network Architectures", ALTMAN R. B., DUNKER A. K., HUNTER L., KLEIN T. E., Eds., *Pacific Symposium on Biocomputing*, vol. 3, p. 18-29, 1998.

[MAC 12] MACNAMARA A., TERFVE C., HENRIQUES D., BERNABÉ B. P., SAEZ-RODRIGUEZ J., "State-time spectrum of signal transduction logic models.", *Physical biology*, vol. 9, num. 4, August 2012.

[MAR 04] MARLER R. T., S.ARORA J., "Survey of multi-objective optimization methods for engineering", *Structural and Multidisciplinary Optimization*, vol. 26, num. 6, p. 369–395, April 2004.

[MAR 12] MARBACH D., COSTELLO J., KÜFFNER R., VEGA N., PRILL R., CAMACHO D., ALLISON K., KELLIS M., COLLINS J., STOLOVITZKY G., "Wisdom of crowds for robust gene network inference", *Nature Methods*, vol. 9, num. 8, p. 796-804, July 2012.

[MCC 56] MCCLUSKEY E. J., "Minimization of Boolean functions", *Bell System Technical Journal*, 1956.

[MIT 09] 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 Computational Biology*, vol. 5, num. 12, Pagee1000591, September 2009.

[MOR 10] MORRIS M., SAEZ-RODRIGUEZ J., SORGER P., LAUFFENBURGER D. A., "Logic-based models for the analysis of cell signaling networks", *Biochemistry*, vol. 49, num. 15, p. 3216-3224, 2010.

[NOB 10] NOBLE D., "Biophysics and systems biology.", *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences*, vol. 368, num. 1914, p. 1125-1139, March 2010.

[PAP 05] PAPIN J. A., HUNTER T., PALSSON B. Ø., SUBRAMANIAM S., "Reconstruction of cellular signalling networks and analysis of their properties", *Nature Reviews Molecular Cell Biology*, vol. 6, num. 2, p. 99-111, February 2005.

[PAP 12] PAPATHEODOROU I., ZIEHM M., WIESER D., ALIC N., PARTRIDGE L., THORNTON J. M., "Using Answer Set Programming to Integrate RNA Expression with Signalling Pathway Information to Infer How Mutations Affect Ageing", *PLoS ONE*, vol. 7, num. 12, December 2012.

[PAU 12] PAULEVÉ L., RICHARD A., "Static Analysis of Boolean Networks Based on Interaction Graphs: A Survey", *Electronic Notes in Theoretical Computer Science*, vol. 284, p. 93-104, June 2012.

[PRI 11] 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*, vol. 4, num. 189, Pagemr7, September 2011.

[RAY 10] RAY O., WHELAN K., KING R., "Logic-Based Steady-State Analysis and Revision of Metabolic Networks with Inhibition", BAROLLI L., XHAFA F., VITABILE S., HSU H., Eds., *Proceedings of the Fourth International Conference on Complex, Intelligent and Software Intensive Systems (CISIS'10)*, IEEE Computer Society, p. 661-666, 2010.

[RAY 12] RAY O., SOH T., INOUE K., "Analyzing Pathways Using ASP-Based Approaches", HORIMOTO K., NAKATSUI M., POPOV N., Eds., *Proceedings of the Fourth International Conference on Algebraic and Numeric Biology (ANB'10)*, vol. 6479 of *Lecture Notes in Computer Science*, Springer-Verlag, p. 167-183, 2012.

[RÉK 08] RÉKA A., WANG R., "Discrete dynamic modeling of cellular signaling networks.", *Methods in Enzymology*, vol. 467, p. 281-306, December 2008.

[REM 08] REMY E., RUET P., THIEFFRY D., "Graphic requirements for multistability and attractive cycles in a Boolean dynamical framework", *Advances in Applied Mathematics*, vol. 41, num. 3, p. 335-350, 2008.

[SAA 13] SAADATPOUR A., RÉKA A., "Boolean modeling of biological regulatory networks: A methodology tutorial", *Methods*, vol. 62, num. 1, p. 3-12, 2013.

[SAE 07] SAEZ-RODRIGUEZ J., SIMEONI L., LINDQUIST J., HEMENWAY R., BOMMHARDT U., ARNDT B., HAUS U., WEISMANTEL R., GILLES E., KLAMT S., SCHRAVEN B., "A Logical Model Provides Insights into T Cell Receptor Signaling", *PLOS Computational Biology*, vol. 3, num. 8, August 2007.

[SAE 09] SAEZ-RODRIGUEZ J., ALEXOPOULOS L. G., EPPERLEIN J., SAMAGA R., LAUFFENBURGER D. A., KLAMT S., SORGER P., "Discrete logic modelling as a means to link protein signalling networks with functional analysis of mammalian signal transduction", *Molecular Systems Biology*, vol. 5, num. 331, Nature Publishing Group, 2009.

[SAE 11] SAEZ-RODRIGUEZ J., ALEXOPOULOS L. G., ZHANG M., MORRIS M., LAUFFENBURGER D. A., SORGER P., "Comparing Signaling Networks between Normal and Transformed Hepatocytes Using Discrete Logical Models", *Cancer Research*, vol. 71, num. 16, 2011.

[SAM 09] SAMAGA R., SAEZ-RODRIGUEZ J., ALEXOPOULOS L. G., SORGER P., KLAMT S., "The logic of EGFR/ErbB signaling: theoretical properties and analysis of high-throughput data", *PLoS Computational Biology*, vol. 5, num. 8, August 2009.

[SAM 10] SAMAGA R., KAMP A. V., KLAMT S., "Computing combinatorial intervention strategies and failure modes in signaling networks", *Journal of Computational Biology*, vol. 17, num. 1, p. 39-53, January 2010.

[SCH 09a] SCHAEFER C. F., ANTHONY K., KRUPA S., BUCHOFF J., DAY M., HANNAY T., BUETOW K. H., "PID: the Pathway Interaction Database", *Nucleic Acids Research*, vol. 37, num. Database issue, p. D674–D679, Oxford University Press, 2009.

[SCH 09b] SCHAUB T., THIELE S., "Metabolic Network Expansion with ASP", 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, p. 312-326, 2009.

[SHA 09] SHAPIRO J. A., "Revisiting the central dogma in the 21st century.", *Annals of the New York Academy of Sciences*, vol. 1178, p. 6-28, October 2009.

[SHA 12] SHARAN R., KARP R. M., "Reconstructing Boolean Models of Signaling", *Research in Computational Molecular Biology*, p. 261-271, Springer-Verlag, Berlin Heidelberg, 2012.

[SPA 10] SPARKES A., AUBREY W., BYRNE E., CLARE A., KHAN M. N., LIAKATA M., MARKHAM M., ROWLAND J., SOLDATOVA L. N., WHELAN K. E., YOUNG M., KING R. D., "Towards Robot Scientists for autonomous scientific discovery.", *Automated Experimentation*, vol. 2, p. 1-1, January 2010.

[STE 04] STELLING J., SAUER U., SZALLASI Z., DOYLE F., DOYLE J., "Robustness of Cellular Functions", *Cell*, vol. 118, num. 6, p. 675-685, 2004.

[STO 07] STOLOVITZKY G., MONROE D., CALIFANO A., "Dialogue on reverse-engineering assessment and methods: the DREAM of high-throughput pathway inference", *Annals of the New York Academy of Sciences*, vol. 1115, p. 1-22, December 2007.

[TER 12] TERFVE C. D. A., COKELAER T., HENRIQUES D., MACNAMARA A., GONÇALVES E., MORRIS M., VAN IERSEL M., LAUFFENBURGER D. A., SAEZ-RODRIGUEZ J., "CellNOptR: a flexible toolkit to train protein signaling networks to data using multiple logic formalisms.", *BMC systems biology*, vol. 6, num. 1, October 2012.

[WAN 11] WANG R., ALBERT R., "Elementary signaling modes predict the essentiality of signal transduction network components", *BMC systems biology*, vol. 5, Page 44, 2011.

[WAN 12] WANG R., SAADATPOUR A., ALBERT R., "Boolean modeling in systems biology: an overview of methodology and applications", *Physical biology*, vol. 9, num. 5, September 2012.

Index