Detecting Inconsistencies in Large Biological Networks with Answer Set Programming

Martin Gebser and Torsten Schaub and Sven Thiele
University of Potsdam, Germany

Philippe Veber
Institut Cochin, Paris, France

submitted 21 June 2009; revised 15 October 2009; accepted 8 December 2009

Abstract

We introduce an approach to detecting inconsistencies in large biological networks by using Answer Set Programming (ASP). To this end, we build upon a recently proposed notion of consistency between biochemical/genetic reactions and high-throughput profiles of cell activity. We then present an approach based on ASP to check the consistency of large-scale data sets. Moreover, we extend this methodology to provide explanations for inconsistencies by determining minimal representations of conflicts. In practice, this can be used to identify unreliable data or to indicate missing reactions.

KEYWORDS: answer set programming, bio-informatics, consistency, diagnosis

1 Introduction

Molecular biology has seen a technological revolution with the establishment of high-throughput methods in the last years. These methods allow for gathering multiple orders of magnitude more measured data than was procurable before. Furthermore, there is an increasing number of biological repositories on the web, such as KEGG, Biomodels, Reactome, MetaCyc, and others, incorporating thousands of biochemical reactions and genetic regulations. However, both measurements as well as biological networks are prone to considerable incompleteness, heterogeneity, and mutual inconsistency, which makes it highly non-trivial to draw biologically meaningful conclusions in an automated way. As a consequence, appropriate representation and powerful reasoning tools are needed to model complex biological systems, in the face of incompleteness and inconsistency.

In this paper, we deal with the analysis of high-throughput measurements in molecular biology, like microarray data or metabolic profiles. Up to now, it is still common practice to use expression profiles merely for detecting over- or under-expressed genes under specific conditions, leaving the task of making biological sense out of a multitude of gene identifiers to human experts. However, many efforts have also been made to better utilize high-throughput data, in particular, by integrating them into large-scale models of transcriptional regulations or metabolic processes (Friedman et al. 2000; Klamt and Stelling 2006).

One possible approach consists of investigating the compatibility between experimental measurements and knowledge available in reaction databases. This can be done by using formal frameworks, for instance, the ones developed in (Gutierrez-Rios et al. 2003) and (Siegel et al. 2006). A crucial feature of this methodology is its ability to cope with qualitative knowledge (for instance, reactions lacking kinetic details) and noisy data. In what follows, we rely upon the so-called *Sign Consistency Model* (SCM) due to (Siegel et al. 2006). SCM imposes constraints between experimental measurements and a graph representation of cellular interactions, called an *influence graph* (Soulé 2003). Such a graph provides an over-approximation of the actual biological model, where an "influence" is modeled by a disjunctive causal rule. This is particularly well-suited for dealing with incomplete (missing reactions) or unreliable (noisy data) information.

Building on the SCM framework, we develop declarative techniques based on *Answer Set Programming* (ASP) (Baral 2003; Gelfond 2008) to detect and explain inconsistencies in large data sets. This approach has several advantages. First, it allows us to formulate biological problems in a declarative way, thus easing the communication with biological experts. Second, although we do not detail it here, the rich modeling language facilitates integrating different knowledge representation and reasoning techniques, like abduction, explanation, planning, prediction, etc., in a uniform and transparent way (cf. (Gebser et al. 2010) for such extensions). And finally, modern ASP solvers are based on advanced Boolean constraint solving technology and thus provide us with highly efficient inference engines. Apart from modeling the aforementioned biological problems in ASP, our major concern lies with the scalability of the approach. To this end, we apply our methods to the gene-regulatory network of yeast (Guelzim et al. 2002; Sudarsanam et al. 2000) and, moreover, design an artificial yet biologically meaningful benchmark suite indicating that an ASP-based approach scales well on the considered class of applications. Notably, to the best of our knowledge, the functionalities we provide go beyond the ones of the only comparable approach (Guziolowski et al. 2009).

To begin with, we introduce SCM in Section 2. Section 3 gives the syntax and semantics of ASP used in our application. In Section 4, we develop an ASP formulation of checking the consistency between experimental profiles and influence graphs. We further extend this approach in Section 5 to identifying minimal representations of conflicts if the experimental data is inconsistent with an influence graph. In Section 6, we describe simple yet effective techniques for input reduction along with a connectivity property that is used to refine the encoding presented in Section 5. Section 7 is dedicated to an empirical evaluation of our approach along with an exemplary case study on yeast. For making our methods easily accessible, an available web service is presented in Section 8. Section 9 concludes the paper with a discussion and outlook on future work. Finally, Appendix A and Appendix B contain proofs of soundness and completeness for our problem formulations in ASP.

2 Influence Graphs and Sign Consistency Constraints

Influence graphs (Soulé 2003) are a common representation for a wide range of dynamical systems. In the field of genetic networks, they have been investigated for various classes of systems, ranging from ordinary differential equations (Soulé 2006) to synchronous (Remy et al. 2008) and asynchronous (Richard et al. 2004) Boolean networks. Influence graphs have also been introduced in the field of qualitative reasoning (Kuipers 1994) to describe

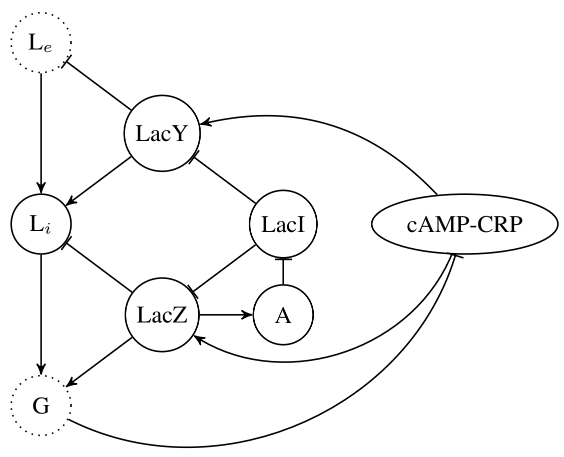


Figure 1. Simplified model of operon lactose in *E. coli*, represented as an influence graph. The vertices represent either genes, metabolites, or proteins, while the edges indicate the regulations among them. Edges with an arrow stand for positive regulations (activations), while edges with a tee head stand for negative regulations (inhibitions). Vertices G and L$_e$ are considered to be inputs of the system, that is, their signs are not constrained via their incoming edges.

physical systems where a detailed quantitative description is unavailable. In fact, this has been the main motivation for using influence graphs for knowledge representation in the context of biological systems.

An *influence graph* is a directed graph whose vertices are the input and state variables of a system and whose edges express the effects of variables on each other.

Definition 2.1 (*Influence Graph*)
An *influence graph* is a directed graph (V, E, σ), where V is a set of vertices, E a set of edges, and $\sigma : E \to \{+, -\}$ a (partial) labeling of the edges.

An edge $j \to i$ means that the variation of j in time influences the level of i. Every edge $j \to i$ of an influence graph can be labeled with a sign, either + or –, denoted by $\sigma(j, i)$, where + (–) indicates that j tends to increase (decrease) i. An example influence graph is given in Figure 1; it represents a simplified model of the operon lactose in *E. coli*.

In SCM, *experimental profiles* are supposed to come from steady state shift experiments where, initially, the system is at steady state, then perturbed using control parameters, and eventually, it settles into another steady state. It is assumed that the data measures the differences between the initial and the final state. Thus, for genes, proteins, or metabolites, we know whether the concentration has increased or decreased, while quantitative values are unavailable, unessential, or unreliable. By $\mu(i)$, we denote the sign, again either + or –, of the variation of a species i between the initial and the final condition. One can easily enhance this setting to also considering null (or more precisely, non-significant) variations, by exploiting the concept of sign algebra (Kuipers 1994).

Given an influence graph (as a representation of cellular interactions) and a labeling of its vertices with signs (as a representation of experimental profiles), we now describe the constraints that relate both. Informally, for every non-input vertex i, its variation $\mu(i)$

Species	L_e	L_i	G	LacY	LacZ	LacI	A	cAMP-CRP
μ_1	–	–	–	–	–	+	–	+
μ_2	+	+	–	+	–	+	–	–
μ_3	+	?	–	?	?	+	?	?
μ_4	?	?	?	–	+	?	?	+

Table 1. *Some vertex labelings (reflecting measurements of two steady states) for the influence graph depicted in Figure 1; unobserved values indicated by question mark '**?**'.*

ought to be explained by the influence of at least one predecessor j of i in the influence graph. Thereby, the *influence* of j on i is given by the sign $\mu(j)\sigma(j,i) \in \{+,-\}$, where the multiplication of signs is derived from that of numbers. Sign consistency constraints can then be formalized as follows.

Definition 2.2 (*Sign Consistency Constraints*)
Let (V,E,σ) be an influence graph and $\mu : V \to \{+,-\}$ a (partial) vertex labeling.

Then, (V,E,σ) and μ are *consistent*, if there are some total extensions $\sigma' : E \to \{+,-\}$ of σ and $\mu' : V \to \{+,-\}$ of μ such that $\mu'(i)$ is consistent for each non-input vertex $i \in V$, where $\mu'(i)$ is consistent, if there is some edge $j \to i$ in E such that $\mu'(i) = \mu'(j)\sigma'(j,i)$.

Note that labelings σ and μ of vertices and edges, respectively, are admitted to be partial. This occurs frequently in practice where the kind of an influence may depend on environmental factors or experimental data may not include all elements of a biological system. In order to decide whether a partially labeled influence graph and a partial experimental profile are mutually consistent, we thus consider the possible totalizations of them. If at least one total edge and one total vertex labeling (extending the given labelings) are such that the signs of all non-input vertices are explained, it is sufficient for mutual consistency.

Table 1 gives four vertex labelings for the influence graph in Figure 1. Total labeling μ_1 is consistent with the influence graph: the variation of each vertex (except for input vertex L_e) can be explained by the effect of one of its regulators. For instance, in μ_1, LacY receives a positive influence from cAMP-CRP as well as a negative influence from LacI, the latter accounting for the decrease of LacY. The second labeling, μ_2, is not consistent: LacY receives only negative influences from cAMP-CRP and LacI, and its increase cannot be explained. Partial vertex labeling μ_3 is consistent with the influence graph in Figure 1, as setting the signs of L_i, LacY, LacZ, A, and cAMP-CRP to +, –, –, –, and +, respectively, extends μ_3 to a consistent total labeling. In contrast, μ_4 cannot be extended consistently.

3 Answer Set Programming

This section provides a brief introduction to ASP, a declarative problem solving paradigm offering a rich modeling language (Lparse Manual; Gebser et al. 2009a) along with highly efficient inference engines based on Boolean constraint solving technology (Giunchiglia et al. 2006; Gebser et al. 2009c; Drescher et al. 2008). The basic idea of ASP is to encode a problem as a logic program such that its answer sets represent solutions.

In view of our application, we take advantage of the elevated expressiveness of disjunctive programs, capturing problems at the second level of the polynomial hierarchy (Eiter and Gottlob 1995). A *disjunctive logic program* P is a finite set of *rules* of the form

$$a_1; \dots ; a_l \leftarrow a_{l+1}, \dots , a_m, \mathit{not}\ a_{m+1}, \dots , \mathit{not}\ a_n\ , \tag{1}$$

where a_i is an *atom* for $1 \leq i \leq n$. A rule r as in (1) is called a *fact* if $l = m = n = 1$, and an *integrity constraint* if $l = 0$. Let $\mathit{head}(r) = \{a_1, \dots , a_l\}$ be the *head* of r, $\mathit{body}(r) = \{a_{l+1}, \dots , a_m, \mathit{not}\ a_{m+1}, \dots , \mathit{not}\ a_n\}$ be the *body* of r, as well let $\mathit{body}(r)^+ = \{a_{l+1}, \dots , a_m\}$ and $\mathit{body}(r)^- = \{a_{m+1}, \dots , a_n\}$.

An interpretation is represented by the set of atoms that are true in it. A *model* of a program P is an interpretation in which all rules of P are true according to the standard definition of truth in propositional logic. Apart from letting ';' and ',' stand for disjunction and conjunction, respectively, this implies treating rules and default negation 'not' as implications and classical negation, respectively. Note that the (empty) head of an integrity constraint is false in every interpretation, while the empty body is true in every interpretation. Answer sets of P are particular models of P satisfying an additional stability criterion. Roughly, a set X of atoms is an answer set, if for every rule of form (1), X contains a minimum of atoms among $a_1, \dots , a_l$ whenever $a_{l+1}, \dots , a_m$ belong to X and no $a_{m+1}, \dots , a_n$ belongs to X. However, the disjunction in heads of rules, in general, is not exclusive. Formally, an *answer set* X of a program P is a $\subseteq$-minimal model of

$$\{\mathit{head}(r) \leftarrow \mathit{body}(r)^+ \mid r \in P, \mathit{body}(r)^- \cap X = \emptyset\}\ .$$

For example, program $\{a; b \leftarrow .\ c; d \leftarrow a, \mathit{not}\ b.\ \leftarrow b.\}$ has answer sets $\{a, c\}$ and $\{a, d\}$.

Although answer sets are usually defined on ground (i.e., variable-free) programs, ASP allows for non-ground problem encodings, where schematic rules stand for their ground instantiations. Grounders, such as *gringo* (Gebser et al. 2009a) and *lparse* (Lparse Manual), are capable of combining a problem encoding and an instance (typically a set of ground facts) into an equivalent ground program, which is then processed by an ASP solver. We follow this methodology and provide encodings for the problems considered below.

4 Checking Consistency

We now come to the first main question addressed in this paper, namely, how to check whether an experimental profile is consistent with a given influence graph. Note that, if the profile provides us with a sign for each vertex of the influence graph, the task can be accomplished simply by checking whether each non-input vertex receives at least one influence matching its variation. However, as soon as the experimental profile has missing values (which is very likely in practice), the problem becomes NP-hard (Veber et al. 2004). In fact, a Boolean satisfiability problem over clauses $C_1, \dots , C_m$ and variables $x_1, \dots , x_n$ can be reduced as follows: introduce unlabeled input vertices $x_1, \dots , x_n$, non-input vertices $C_1, \dots , C_m$ labeled +, and edges $x_j \rightarrow C_i$ labeled + (–) if x_j occurs positively (negatively) in C_i. It is not hard to check that the labeling of $C_1, \dots , C_m$ by + is consistent with the obtained influence graph iff the conjunction of $C_1, \dots , C_m$ is satisfiable.

We next provide a logic program such that each of its answer sets matches a consistent extension of vertex and edge labelings. Our encodings as well as instances are available at (BioASP Tools). The program for consistency checking is composed of three parts, described in the following subsections.

4.1 Problem Instance

An influence graph as well as an experimental profile are given by ground facts. For each species i, we introduce a fact $\mathit{vertex}(i)$, and for each edge $j \rightarrow i$, a fact $\mathit{edge}(j,i)$. If $s \in \{\mathsf{+},\mathsf{-}\}$ is known to be the variation of a species i or the sign of an edge $j \rightarrow i$, it is expressed by a fact $\mathit{observedV}(i,s)$ or $\mathit{observedE}(j,i,s)$, respectively. Finally, a vertex i is declared to be input via a fact $\mathit{input}(i)$.

For example, the negative regulation LacI $\rightarrow$ LacY in the influence graph shown in Figure 1 and observation + for LacI (as with μ_3 in Table 1) give rise to the following facts:

$$\begin{array}{l} \mathit{vertex}(\mathrm{LacI}). \\ \mathit{vertex}(\mathrm{LacY}). \\ \mathit{edge}(\mathrm{LacI},\mathrm{LacY}). \\ \mathit{observedV}(\mathrm{LacI},\mathsf{+}). \\ \mathit{observedE}(\mathrm{LacI},\mathrm{LacY},\mathsf{-}). \end{array} \tag{2}$$

Note that the absence of a fact of form $\mathit{observedV}(\mathrm{LacY},s)$ means that the variation of LacY is unobserved (as with μ_3). In (2), we use LacI and LacY as names for constants associated with the species in Figure 1, but not as first-order variables. Similarly, for uniformity of notations, + and – are written in (2) for constants identifying signs.

4.2 Generating Solution Candidates

As mentioned above, our goal is to check whether an experimental profile is consistent with an influence graph. If so, it is witnessed by total labelings of the vertices and edges, which are generated via the following rules:

$$\begin{array}{r} \mathit{labelV}(V,\mathsf{+});\mathit{labelV}(V,\mathsf{-}) \leftarrow \mathit{vertex}(V). \\ \mathit{labelE}(U,V,\mathsf{+});\mathit{labelE}(U,V,\mathsf{-}) \leftarrow \mathit{edge}(U,V). \end{array} \tag{3}$$

Moreover, the following rules ensure that known labels are respected by total labelings:

$$\begin{array}{r} \mathit{labelV}(V,S) \leftarrow \mathit{observedV}(V,S). \\ \mathit{labelE}(U,V,S) \leftarrow \mathit{observedE}(U,V,S). \end{array} \tag{4}$$

Note that the stability criterion for answer sets demands that a known label derived via a rule in (4) is also derived via (3), thus, excluding the opposite label. In fact, the disjunctive rules used in this section could actually be replaced with non-disjunctive rules via "shifting" (Gelfond et al. 1991),[1] given that our first encoding results in a so-called *head-cycle-free* (HCF) (Ben-Eliyahu and Dechter 1994) ground program. However, similar disjunctive rules are also used in Section 5 where they cannot be compiled away. Also note that HCF programs, for which deciding answer set existence stays in NP, are recognized as such by disjunctive ASP solvers (Leone et al. 2006; Drescher et al. 2008). Hence, the purely syntactic use of disjunction, as done here, is not harmful to efficiency.

The following ground rules are obtained by combining the schematic rules in (3) and (4)

[1] Alternatively, one could also use cardinality constraints (cf. (Lparse Manual)), which would however preclude a comparison with *dlv* in Section 7.

with the facts in (2):

$$\begin{array}{rl}
\mathit{labelV}(\mathrm{LacI},+);\mathit{labelV}(\mathrm{LacI},-) & \leftarrow \mathit{vertex}(\mathrm{LacI}).\\
\mathit{labelV}(\mathrm{LacY},+);\mathit{labelV}(\mathrm{LacY},-) & \leftarrow \mathit{vertex}(\mathrm{LacY}).\\
\mathit{labelE}(\mathrm{LacI},\mathrm{LacY},+);\mathit{labelE}(\mathrm{LacI},\mathrm{LacY},-) & \leftarrow \mathit{edge}(\mathrm{LacI},\mathrm{LacY}).\\
\mathit{labelV}(\mathrm{LacI},+) & \leftarrow \mathit{observedV}(\mathrm{LacI},+).\\
\mathit{labelE}(\mathrm{LacI},\mathrm{LacY},-) & \leftarrow \mathit{observedE}(\mathrm{LacI},\mathrm{LacY},-).
\end{array} \tag{5}$$

One can check that the program consisting of the facts in (2) and the rules in (5) admits two answer sets, the first one including $\mathit{labelV}(\mathrm{LacY},+)$ and the second one including $\mathit{labelV}(\mathrm{LacY},-)$. On the remaining atoms, both answer sets coincide by containing the atoms in (2) along with $\mathit{labelV}(\mathrm{LacI},+)$ and $\mathit{labelE}(\mathrm{LacI},\mathrm{LacY},-)$.

4.3 Testing Solution Candidates

We now check whether generated total labelings satisfy the sign consistency constraints stated in Definition 2.2, requiring an influence of sign s for each non-input vertex i with variation s. We thus define $\mathit{receive}(i,s)$ to indicate that i receives an influence of sign s:

$$\begin{array}{rl}
\mathit{receive}(V,+) & \leftarrow \mathit{labelE}(U,V,S),\mathit{labelV}(U,S).\\
\mathit{receive}(V,-) & \leftarrow \mathit{labelE}(U,V,S),\mathit{labelV}(U,T),S\neq T.
\end{array} \tag{6}$$

Inconsistent labelings, where a non-input vertex does not receive any influence matching its variation, are then ruled out by integrity constraints of the following form:

$$\leftarrow \mathit{labelV}(V,S), \mathit{not}\ \mathit{receive}(V,S), \mathit{not}\ \mathit{input}(V). \tag{7}$$

Note that the schematic rules in (6) and (7) are given in the input language of grounder *gringo* (Gebser et al. 2009a). This allows us to omit an explicit listing of some "domain predicates" in the bodies of rules, which would be necessary when using *lparse* (Lparse Manual). At (BioASP Tools), we provide encodings for *gringo* and also (more verbose ones) for *lparse*.

Starting from the answer sets described in the previous subsection, the included atoms $\mathit{labelE}(\mathrm{LacI},\mathrm{LacY},-)$ and $\mathit{labelV}(\mathrm{LacI},+)$ allow us to derive $\mathit{receive}(\mathrm{LacY},-)$ via a ground instance of the second rule in (6), while $\mathit{receive}(\mathrm{LacY},+)$ is not derivable. After adding $\mathit{receive}(\mathrm{LacY},-)$, the solution candidate containing $\mathit{labelV}(\mathrm{LacY},-)$ satisfies the ground instance of the integrity constraint in (7) obtained by substituting LacY for V and $-$ for S. Assuming LacI to be an input, as it can be declared via fact $\mathit{input}(\mathrm{LacI})$, we thus obtain an answer set containing $\mathit{labelV}(\mathrm{LacY},-)$, expressing a decrease of LacY. In contrast, since $\mathit{receive}(\mathrm{LacY},+)$ is underivable, the solution candidate containing $\mathit{labelV}(\mathrm{LacY},+)$ violates the following ground instance of (7):

$$\leftarrow \mathit{labelV}(\mathrm{LacY},+), \mathit{not}\ \mathit{receive}(\mathrm{LacY},+), \mathit{not}\ \mathit{input}(\mathrm{LacY}).$$

That is, the solution candidate with $\mathit{labelV}(\mathrm{LacY},+)$ does not pass the consistency test.

4.4 Soundness and Completeness

By letting $\tau((V,E,\sigma),\mu)$ denote the set of facts representing the problem instance induced by an influence graph (V,E,σ) and a vertex labeling μ, and P_C the logic program consisting of the rules given in (3), (4), (6), and (7), respectively, we can show the following soundness and completeness results.

Theorem 4.1 (*Soundness*)
Let (V, E, σ) be an influence graph and $\mu : V \to \{+,-\}$ a (partial) vertex labeling.
If there is an answer set of $P_C \cup \tau((V, E, \sigma), \mu)$, then (V, E, σ) and μ are consistent.

Theorem 4.2 (*Completeness*)
Let (V, E, σ) be an influence graph and $\mu : V \to \{+,-\}$ a (partial) vertex labeling.
If (V, E, σ) and μ are consistent, then there is an answer set of $P_C \cup \tau((V, E, \sigma), \mu)$.

The following correspondence result is immediately obtained from Theorem 4.1 and 4.2.

Corollary 4.3 (*Soundness and Completeness*)
Let (V, E, σ) be an influence graph and $\mu : V \to \{+,-\}$ a (partial) vertex labeling.
Then, (V, E, σ) and μ are consistent iff there is an answer set of $P_C \cup \tau((V, E, \sigma), \mu)$.

5 Identifying Minimal Inconsistent Cores

In view of the usually large amount of data, it is crucial to provide concise explanations whenever an experimental profile is inconsistent with an influence graph (i.e., if the logic program given in the previous section has no answer set). To this end, we adopt a strategy that was successfully applied on real biological data (Guziolowski et al. 2007). The basic idea is to isolate minimal subgraphs of an influence graph such that the vertices and edges cannot be labeled consistently. This task is closely related to extracting Minimal Unsatisfiable Cores (MUCs) (Dershowitz et al. 2006) in the context of Boolean satisfiability (SAT). In allusion, we call a minimal subgraph of an influence graph whose vertices and edges cannot be labeled consistently a *Minimal Inconsistent Core* (MIC), whose formal definition is as follows.[2]

Definition 5.1 (*Minimal Inconsistent Core*)
Let (V, E, σ) be an influence graph and $\mu : V \to \{+,-\}$ a (partial) vertex labeling.
Then, a subset W of V is a *Minimal Inconsistent Core* (MIC), if

1. for all total extensions $\sigma' : E \to \{+,-\}$ of σ and $\mu' : V \to \{+,-\}$ of μ, there is some non-input vertex $i \in W$ such that $\mu'(i)$ is inconsistent, and
2. for every $W' \subset W$, there are some total extensions $\sigma' : E \to \{+,-\}$ of σ and $\mu' : V \to \{+,-\}$ of μ such that $\mu'(i)$ is consistent for each non-input vertex $i \in W'$.

To encode MICs, we make use of three important observations made on Definition 5.1. First, the inherent inconsistency of a MIC's vertices stipulated in the first condition must be implied by the MIC and its external regulators, while vertices not connected to the MIC cannot contribute anything. Moreover, the second condition on proper subsets prohibits

[2] We note that verifying a MUC is D^P-complete (Dershowitz et al. 2006; Papadimitriou and Yannakakis 1982), and the same applies to MICs in view of the reduction of SAT described in Section 4. However, solving a decision problem is not sufficient for our application because we also need to provide MIC candidates to verify. As regards checking inconsistency of an (a priori unknown) MIC candidate, we are unaware of ways to accomplish such a co-NP test in non-disjunctive ASP without destroying the candidate at hand.

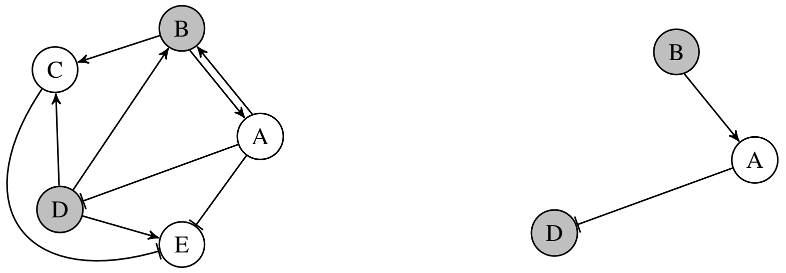


Figure 2. A partially labeled influence graph and a MIC consisting of **A** and **D**.

the inclusion of an input vertex in a MIC, as it could always be removed without affecting inherent (in)consistency of the remaining vertices' variations. Finally, for establishing consistency of all proper subsets of a MIC, it is sufficient to consider subsets excluding a single vertex of the MIC, given that their consistency carries forward to all smaller subsets.

For illustration, consider the influence graph and the MIC in Figure 2. One can check that the observed simultaneous increase of **B** and **D** is not consistent with the influence graph, but the reason for this might not be apparent at first glance. However, once the MIC consisting of **A** and **D** is extracted, we see that the increase of **B** implies an increase of **A**, so that the observed increase of **D** cannot be explained. Note that the elucidation of inherent inconsistency provided by a MIC takes its vertices along with their regulators into account, the latter being incapable of jointly explaining the variations of all vertices in the MIC.

We next provide an encoding for identifying MICs, where a problem instance, that is, an influence graph along with an experimental profile, is represented by facts as specified in Section 4.1. The encoding then consists of three parts: the first generating MIC candidates, the second asserting inconsistency, and the third verifying minimality.

5.1 Generating MIC Candidates

The generating part comprises rules in (4) for deriving known vertex and edge labels. In addition, it includes the following rules:

$$\begin{aligned} \mathit{active}(V); \mathit{inactive}(V) &\leftarrow \mathit{vertex}(V), \mathit{not}\ \mathit{input}(V). \\ \mathit{edgeMIC}(U,V) &\leftarrow \mathit{edge}(U,V), \mathit{active}(V). \\ \mathit{vertexMIC}(U) &\leftarrow \mathit{edgeMIC}(U,V). \\ \mathit{vertexMIC}(V) &\leftarrow \mathit{active}(V). \\ \mathit{labelV}(V,+); \mathit{labelV}(V,-) &\leftarrow \mathit{vertexMIC}(V). \\ \mathit{labelE}(U,V,+); \mathit{labelE}(U,V,-) &\leftarrow \mathit{edgeMIC}(U,V). \end{aligned} \tag{8}$$

The first rule permits guessing non-input vertices forming a MIC candidate. Such vertices are marked as *active*. The subgraph of the influence graph consisting of the active vertices, their regulators, and the connecting edges provides the context of the MIC candidate.[3] The

[3] In Definition 5.1, (in)consistency is checked only for the (non-input) vertices in a MIC, while other vertices' variations do not need to be explained. Hence, guessing unobserved vertex (and edge) labels can be restricted to vertices belonging to or connected to the MIC, which reduces combinatorics.

vertices and edges contributing to this subgraph are identified via *vertexMIC* and *edgeMIC*. The guessing of (unobserved) vertex and edge labels is restricted to them in the last two rules of (8). Finally, note that the rules in (4) propagate known labels also for vertices and edges not correlated to the MIC candidate, viz., to the active vertices. This does not incur additional combinatorics; rather, it reduces derivations depending on MIC candidates.

5.2 Testing for Inconsistency

By adapting the methodology used in (Eiter and Gottlob 1995), the following subprogram makes sure that the active vertices cannot be labeled consistently, taking (implicitly) into account all possible labelings for them, their regulators, and connecting edges:[4]

$$\begin{aligned}
\mathit{opposite}(U,V) &\leftarrow \mathit{labelE}(U,V,\textbf{-}), \mathit{labelV}(U,S), \mathit{labelV}(V,S).\\
\mathit{opposite}(U,V) &\leftarrow \mathit{labelE}(U,V,\textbf{+}), \mathit{labelV}(U,S), \mathit{labelV}(V,T), S \neq T.\\
\mathit{bottom} &\leftarrow \mathit{active}(V), \mathit{opposite}(U,V) : \mathit{edge}(U,V).\\
&\leftarrow \mathit{not\ bottom}.\\
\mathit{labelV}(V,\textbf{+}) &\leftarrow \mathit{bottom}, \mathit{vertex}(V).\\
\mathit{labelV}(V,\textbf{-}) &\leftarrow \mathit{bottom}, \mathit{vertex}(V).\\
\mathit{labelE}(U,V,\textbf{+}) &\leftarrow \mathit{bottom}, \mathit{edge}(U,V).\\
\mathit{labelE}(U,V,\textbf{-}) &\leftarrow \mathit{bottom}, \mathit{edge}(U,V).
\end{aligned} \tag{9}$$

In this (part of the) encoding, $\mathit{opposite}(U,V)$ indicates that the influence of regulator U on V is opposite to the variation of V. If all regulators of an active vertex V have such an opposite influence, the sign consistency constraint for V is violated, in which case atom *bottom* along with all labels for vertices and edges are derived. Note that the stability criterion for an answer set X imposes that *bottom* and all labels belong to X only if the active vertices cannot be labeled consistently. Finally, integrity constraint $\leftarrow \mathit{not\ bottom}$ necessitates the inclusion of *bottom* in any answer set, thus, stipulating an inevitable sign consistency constraint violation for some active vertex.

Reconsidering our example in Figure 2, the ground instances of (8) permit guessing $\mathit{active}(\mathbf{A})$ and $\mathit{active}(\mathbf{D})$. When labeling **A** with **+** (or assuming $\mathit{labelV}(\mathbf{A},\textbf{+})$ to be true), we derive $\mathit{opposite}(\mathbf{A},\mathbf{D})$ and *bottom*, producing in turn all labels for vertices and edges. Furthermore, setting the sign of **A** to **-** (or $\mathit{labelV}(\mathbf{A},\textbf{-})$ to true) makes us derive $\mathit{opposite}(\mathbf{B},\mathbf{A})$, which again gives *bottom* and all labels for vertices and edges. We have thus verified that the sign consistency constraints for **A** and **D** cannot jointly be satisfied, given the observed increases of **B** and **D**. That is, active vertices **A** and **D** are sufficient to explain the inconsistency between the observations and the influence graph.

5.3 Testing for Minimality

It remains to be verified whether the sign consistency constraints for all active vertices are necessary to identify an inherent inconsistency. This test is based on the idea that, excluding

[4] In the language of *gringo* (and *lparse*), the expression $\mathit{opposite}(U,V) : \mathit{edge}(U,V)$ used below refers to the conjunction of all ground atoms $\mathit{opposite}(j,i)$ for which $\mathit{edge}(j,i)$ holds.

any single active vertex, the sign consistency constraints for the other active vertices should be satisfied by appropriate labelings, which can be implemented as follows:

$$\begin{aligned}
\textit{labelV'}(W,V,\textbf{+});\textit{labelV'}(W,V,\textbf{–}) &\leftarrow \textit{active}(W),\textit{vertexMIC}(V).\\
\textit{labelE'}(W,U,V,\textbf{+});\textit{labelE'}(W,U,V,\textbf{–}) &\leftarrow \textit{active}(W),\textit{edgeMIC}(U,V).\\
\textit{labelV'}(W,V,S) &\leftarrow \textit{active}(W),\textit{observedV}(V,S).\\
\textit{labelE'}(W,U,V,S) &\leftarrow \textit{active}(W),\textit{observedE}(U,V,S).\\
\textit{receive'}(W,V,\textbf{+}) &\leftarrow \textit{labelE'}(W,U,V,S),\textit{labelV'}(W,U,S),V\neq W.\\
\textit{receive'}(W,V,\textbf{–}) &\leftarrow \textit{labelE'}(W,U,V,S),\textit{labelV'}(W,U,T),V\neq W,S\neq T.\\
&\leftarrow \textit{labelV'}(W,V,S),\textit{active}(V),V\neq W,\textit{not receive'}(W,V,S).
\end{aligned} \tag{10}$$

This subprogram is similar to the consistency check encoded via the rules in (3), (4), (6), and (7). However, sign consistency constraints are only checked for active vertices, and they must be satisfiable for all but one arbitrary active vertex W. In fact, labelings such that the variations of all active vertices but W are explained witness the fact that W cannot be removed from a MIC candidate without re-establishing consistency. As W ranges over all (non-input) vertices of an influence graph, each active vertex is taken into consideration. Regarding computational complexity, recall from Section 4 that checking consistency is NP-complete. As a consequence, one cannot easily identify conditions to select a particular witness for consistency of a MIC candidate minus some vertex W, and so we do not encode any such conditions. This leads to the potential of multiple answer sets comprising the same MIC but different witnesses, in particular, if many vertices and edges belong to the context of the MIC.

For the influence graph in Figure 2, it is easy to see that the sign consistency constraint for **A** is satisfied by setting the sign of **A** to **+**, expressed by atom $\textit{labelV'}(\mathbf{D},\mathbf{A},\textbf{+})$ in the ground rules obtained from the above encoding part. In turn, the sign consistency constraint for **D** is satisfied by setting the sign of **A** to **–**. This is reflected by atom $\textit{labelV'}(\mathbf{A},\mathbf{A},\textbf{–})$, allowing us to derive $\textit{receive'}(\mathbf{A},\mathbf{D},\textbf{+})$. That is, the ground instance of the above integrity constraint containing $\textit{labelV'}(\mathbf{A},\mathbf{D},\textbf{+})$ is satisfied. The fact that atoms $\textit{labelV'}(\mathbf{D},\mathbf{A},\textbf{+})$ and $\textit{labelV'}(\mathbf{A},\mathbf{A},\textbf{–})$, used for explaining the variation of either **A** or **D**, respectively, disagree on the sign of **A** also shows that jointly considering **A** and **D** yields an inconsistency.

5.4 Soundness and Completeness

Similar to Section 4.4, we can show the soundness and completeness for our MIC extraction encoding P_D, consisting of the rules in (4), (8), (9), and (10), respectively.

Theorem 5.1 (*Soundness*)
Let (V,E,σ) be an influence graph and $\mu : V \rightarrow \{\textbf{+},\textbf{–}\}$ a (partial) vertex labeling.
If X is an answer set of $P_D \cup \tau((V,E,\sigma),\mu)$, then $\{i \mid \textit{active}(i) \in X\}$ is a MIC.

Theorem 5.2 (*Completeness*)
Let (V,E,σ) be an influence graph and $\mu : V \rightarrow \{\textbf{+},\textbf{–}\}$ a (partial) vertex labeling.
If $W \subseteq V$ is a MIC, then there is an answer set X of $P_D \cup \tau((V,E,\sigma),\mu)$ such that $\{i \mid \textit{active}(i) \in X\} = W$.

The following correspondence result is immediately obtained from Theorem 5.1 and 5.2.

Corollary 5.3 (*Soundness and Completeness*)
Let (V, E, σ) be an influence graph and $\mu : V \rightarrow \{+, -\}$ a (partial) vertex labeling.
Then, $W \subseteq V$ is a MIC iff there is an answer set X of $P_D \cup \tau((V, E, \sigma), \mu)$ such that $\{i \mid \mathit{active}(i) \in X\} = W$.

As mentioned above, several answer sets may represent the same MIC because witnesses needed for minimality testing are not necessarily unique.

6 Refinements

In this section, we detail two encoding extensions aiming at the improvement of grounding and solving efficiency. First, input reduction checks for some simple cases to identify and distinguish uncritical vertices. Second, background knowledge about MICs' connectivity can be exploited to more precisely render potential MIC candidates.

6.1 Input Reduction

It is not unlikely in practice that biological networks include simple tractable substructures or that parts of experimental observations are easily explained. Dealing with such particular cases before doing complex computations (like checking consistency or finding MICs) is therefore advisable. Given an influence graph (V, E, σ) and a partial vertex labeling μ capturing experimental data, we below describe conditions to identify vertices that can always be labeled consistently. Such vertices can then be marked as (additional) inputs to exclude their sign consistency constraints from consistency checking and to make explicit that they cannot belong to any MIC. Any of the following conditions is sufficient to identify a vertex i as effectively unconstrained:

1. There is a regulation $i \rightarrow i$ in E such that $\sigma(i, i) = +$, that is, i supports its variation.
2. There is a regulation $j \rightarrow i$ in E such that $\sigma(j, i)$ is undefined. In fact, undetermined regulations are used in practice to model influences that vary, e.g., relative to environmental conditions. Any variation of the target i of such a regulation can be explained by assigning the appropriate label to $j \rightarrow i$ (w.r.t. the label of j).
3. There are regulations $j \rightarrow i, k \rightarrow i$ in E such that $\mu(j)\sigma(j, i) = +$ and $\mu(k)\sigma(k, i) = -$. That is, any variation of i is already explained by the given observations.
4. An observed variation $\mu(i)$ of i is explained if there is some regulation $j \rightarrow i$ in E such that $\mu(j)\sigma(j, i) = \mu(i)$. Any further regulations targeting i can be ignored.
5. If for all regulations $i \rightarrow k$ in E, we have that k is an input, then the variation of i is insignificant for its targets. In this case, if i is unobserved ($\mu(i)$ is undefined) and target of at least one regulation $j \rightarrow i$ in E, we can assign an appropriate label to i (w.r.t. the labels of j and $j \rightarrow i$) without any further conditions.
6. There is a regulation $j \rightarrow i$ in E such that j is unobserved ($\mu(j)$ is undefined), an input, and all targets $k \neq i$ of j ($j \rightarrow k$ belongs to E) are inputs. Without any further conditions, we can assign an appropriate label to j for explaining the variation of i.

The reduction idea is to mark a vertex i as additional input, if it meets one of the above conditions. Since the two last conditions inspect inputs, they may become applicable to

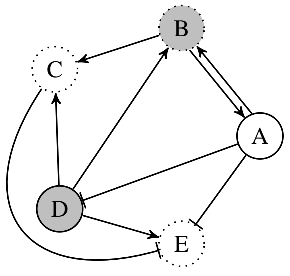


Figure 3. A partially labeled influence graph with uncritical vertices surrounded by dots.

further vertices once inputs are added. Hence, checking the conditions and adding inputs needs to be done exhaustively. As we see below, this can easily be encoded in ASP.

Reconsidering the influence graph and partial observations in Figure 2, we see that vertex **B** receives an influence from **D** matching its observed increase. Thus, the fourth condition applies to already explained vertex **B**. Moreover, vertex **E** is unobserved and does not regulate anything. That is, the fifth condition applies to **E**, and its variation can simply be picked from influences it receives from **A**, **C**, and **D**. After establishing that **E** can be labeled consistently, we find that **C** does not regulate any critically constrained vertex. Applying again the fifth condition, we notice that the variation of **C** is actually insignificant.

Figure 3 shows the situation resulting from the identification of uncritical vertices by iteratively applying the above conditions. The fact that only **A** and **D** are critically constrained tells us that only they can belong to a MIC. As a consequence, the MIC containing **A** and **D**, shown on the right-hand side of Figure 2, is the only one in this example.

The aforementioned idea to mark uncritical vertices as *input* can be encoded as follows:

$$
\begin{aligned}
&\mathit{obs}(V) \leftarrow \mathit{observedV}(V,S).\\
&\mathit{get}(V,\mathbf{+}) \leftarrow \mathit{observedE}(U,V,S), \mathit{observedV}(U,S).\\
&\mathit{get}(V,\mathbf{-}) \leftarrow \mathit{observedE}(U,V,S), \mathit{observedV}(U,T), S \neq T.\\
&\mathit{input}(V) \leftarrow \mathit{observedE}(V,V,\mathbf{+}).\\
&\mathit{input}(V) \leftarrow \mathit{edge}(U,V), \mathit{not}\ \mathit{observedE}(U,V,\mathbf{+}), \mathit{not}\ \mathit{observedE}(U,V,\mathbf{-}).\\
&\mathit{input}(V) \leftarrow \mathit{get}(V,\mathbf{+}), \mathit{get}(V,\mathbf{-}).\\
&\mathit{input}(V) \leftarrow \mathit{observedV}(V,S), \mathit{get}(V,S).\\
&\mathit{input}(V) \leftarrow \mathit{edge}(U,V), \mathit{input}(W) : \mathit{edge}(V,W), \mathit{not}\ \mathit{obs}(V).\\
&\mathit{input}(V) \leftarrow \mathit{edge}(U,V), \mathit{input}(W) : \mathit{edge}(U,W) : W \neq V, \mathit{input}(U), \mathit{not}\ \mathit{obs}(U).
\end{aligned}
$$

Auxiliary predicates *obs* and *get* are used to exhibit whether either variation has been observed for a vertex and whether a particular influence is received for certain, respectively. The last six rules check the described conditions (in the same order) and mark a vertex as *input* if one of them applies. Importantly, the above rules are stratified and thus yield a unique set of derived input vertices. This allows us to perform the reduction efficiently within grounding, without deferring to any procedural implementation external to ASP.

The situation shown in Figure 3 is reflected by the reduction encoding deriving atoms *input*(**B**), *input*(**C**), and *input*(**E**) from an instance (cf. Section 4.1) corresponding to the depicted influence graph and observed variations. Consistency checking and MIC identification (cf. Section 4 and 5) can then focus on the remaining non-input vertices **A** and **D**.

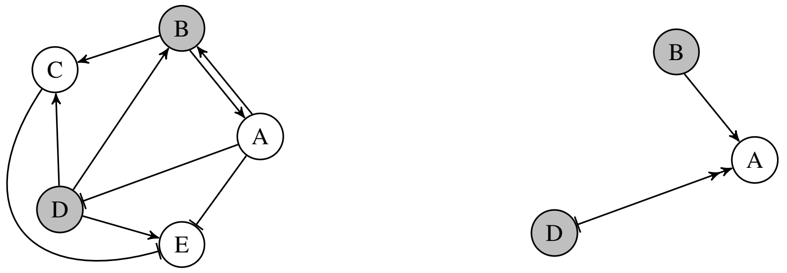


Figure 4. A partially labeled influence graph and the graph $(V[\{\mathbf{A},\mathbf{D}\}], E[\{\mathbf{A},\mathbf{D}\}])$.

6.2 Exploiting Strongly Connected Components for MIC Extraction

In what follows, we introduce a connectivity property of MICs that can be used to further refine the encoding presented in Section 5. Incorporating additional background knowledge into the problem encoding is straightforward (as soon as such knowledge is established). In practice, ancillary (and actually redundant) conditions may significantly narrow and thus speed up both the grounding and the solving process.

MIC Connectivity Property. For analyzing interactions within a MIC, we make use of a graph described in the following. Let (V,E,σ) be an influence graph and $\mu : V \rightarrow \{+,-\}$ be a (partial) vertex labeling, and let $D(\mu)$ denote the set of vertices labeled by μ. For a set $W \subseteq V$ of vertices, we define a graph $(V[W], E[W])$ by:

$$\begin{aligned} V[W] &= W \cup \{j \mid (j \rightarrow i) \in E, i \in W\} \\ E[W] &= \{(j \rightarrow i) \mid (j \rightarrow i) \in E, i \in W\} \cup \{(i \rightarrow j) \mid (j \rightarrow i) \in E, i \in W, j \notin D(\mu)\}\,. \end{aligned}$$

The construction of $(V[W], E[W])$ is based on the idea that a regulator j of some $i \in W$ is connected to i via its sign consistency constraint, and a connection in the opposite direction applies if j is unlabeled by μ. In fact, given some total extensions $\sigma' : E \rightarrow \{+,-\}$ of σ and $\mu' : V \rightarrow \{+,-\}$ of μ, we can check a matching influence of j on i by $\mu'(i) = \mu'(j)\sigma'(j,i)$ or equivalently by $\mu'(j) = \mu'(i)\sigma'(j,i)$. That is, provided that $\mu(j)$ is undefined, $\mu'(i)$ constrains $\mu'(j)$ by contraposition whenever i does not receive a matching influence from any other regulator than j. This observation motivates the inclusion of inverse edges from vertices in W to regulators unlabeled by μ in $E[W]$.

For illustration, the right-hand side of Figure 4 shows graph $(V[\{\mathbf{A},\mathbf{D}\}], E[\{\mathbf{A},\mathbf{D}\}])$ resulting from the partially labeled influence graph on the left-hand side. The single regulator **B** of **A** is labeled, and thus there is no inverse edge from **A** to **B** in $E[\{\mathbf{A},\mathbf{D}\}]$. On the other hand, **A** is an unlabeled regulator of **D**, and so $E[\{\mathbf{A},\mathbf{D}\}]$ includes an inverse edge from **D** to **A**. The addition of this edge turns the subgraph of $(V[\{\mathbf{A},\mathbf{D}\}], E[\{\mathbf{A},\mathbf{D}\}])$ induced by **A** and **D** into a strongly connected component. In view that **A** and **D** belong to a MIC (as discussed in Section 5), we below show that this connectivity is not by chance.

Theorem 6.1 (*MIC Connectivity*)
Let (V,E,σ) be an influence graph and $\mu : V \rightarrow \{+,-\}$ a (partial) vertex labeling.

If $W \subseteq V$ is a MIC, then all vertices in W belong to the same strongly connected component in $(V[W], E[W])$.

The proof is omitted in view of space limitations and can be obtained from the authors.

Optimized MIC Encoding. We now apply Theorem 6.1 to improve the basic MIC extraction encoding (cf. Section 5) in two aspects: adding (redundant) constraints for search space pruning and adding positive body literals for reducing grounding efforts. The following rules pave the way by determining the (non-trivial) strongly connected components in $(V, E[V])$ as an over-approximation of the ones in $(V[W], E[W])$ for any $W \subseteq V$:

$$\begin{aligned}
\mathit{edges}(U,V) &\leftarrow \mathit{edge}(U,V), \mathit{not\ input}(V).\\
\mathit{edges}(V,U) &\leftarrow \mathit{edge}(U,V), \mathit{not\ input}(V), \mathit{not\ observedV}(U,+), \mathit{not\ observedV}(U,-).\\
\mathit{reach}(U,V) &\leftarrow \mathit{edges}(U,V).\\
\mathit{reach}(U,V) &\leftarrow \mathit{edges}(U,W), \mathit{reach}(W,V), \mathit{vertex}(V).\\
\mathit{cycle}(U,V) &\leftarrow \mathit{reach}(U,V), \mathit{reach}(V,U), U \neq V.
\end{aligned} \tag{11}$$

The first rule simply collects edges whose targets are not input, while the second rule adds edges in the inverse direction for unobserved regulators. Reachability w.r.t. the so obtained graph is determined via the third and the fourth rule. Finally, predicate *cycle* indicates whether two (distinct) vertices reach each other in $(V, E[V])$ relative to an influence graph (V, E, σ) and a (partial) vertex labeling μ. In fact, if two vertices belong to a MIC $W \subseteq V$, then mutual reachability in $(V[W], E[W])$ implies the same in $(V, E[V])$, in view that $V[W] \subseteq V$ and $E[W] \subseteq E[V]$. Conversely, if two vertices do not reach each other in $(V, E[V])$, then they cannot jointly belong to any MIC.

The over-approximation of potential MICs provides an easy means to prune the search space by adding the following integrity constraint:

$$\leftarrow \mathit{active}(U), \mathit{active}(V), U < V, \mathit{not\ cycle}(U,V). \tag{12}$$

The constraint makes the fact explicit that distinct vertices of a MIC must reach each other in $(V, E[V])$, and it immediately refutes MIC candidates that do not satisfy this condition.

After making use of Theorem 6.1 to narrow search, we now shift the focus to grounding. As a matter of fact, the quadratic space complexity of the minimality test's ground instantiation, as encoded in (10), is a major bottleneck in scaling. The knowledge about potential pairwisely connected vertices in MICs, represented by integrity constraint (12), also allows us to include positive body literals in order to restrict the scope of minimality tests:

$$\begin{aligned}
\mathit{labelV'}(W,V,+); \mathit{labelV'}(W,V,-) &\leftarrow \mathit{active}(W), \mathit{active}(V), \mathit{cycle}(V,W).\\
\mathit{labelV'}(W,U,+); \mathit{labelV'}(W,U,-) &\leftarrow \mathit{active}(W), \mathit{edgeMIC}(U,V), \mathit{cycle}(V,W).\\
\mathit{labelE'}(W,U,V,+); \mathit{labelE'}(W,U,V,-) &\leftarrow \mathit{active}(W), \mathit{edgeMIC}(U,V), \mathit{cycle}(V,W).\\
\mathit{labelV'}(W,V,S) &\leftarrow \mathit{active}(W), \mathit{observedV}(V,S), \mathit{cycle}(V,W).\\
\mathit{labelV'}(W,U,S) &\leftarrow \mathit{active}(W), \mathit{observedV}(U,S), \mathit{edge}(U,V), \mathit{cycle}(V,W).\\
\mathit{labelE'}(W,U,V,S) &\leftarrow \mathit{active}(W), \mathit{observedE}(U,V,S), \mathit{cycle}(V,W).\\
\mathit{receive'}(W,V,+) &\leftarrow \mathit{labelE'}(W,U,V,S), \mathit{labelV'}(W,U,S).\\
\mathit{receive'}(W,V,-) &\leftarrow \mathit{labelE'}(W,U,V,S), \mathit{labelV'}(W,U,T), S \neq T.\\
&\leftarrow \mathit{labelV'}(W,V,S), \mathit{active}(V), \mathit{cycle}(V,W), \mathit{not\ receive'}(W,V,S).
\end{aligned} \tag{13}$$

In comparison to (10), the extra condition $\mathit{cycle}(V,W)$ in the bodies of the first three rules establishes that labels used for testing minimality are guessed only for pairs W and V of vertices that can potentially jointly belong to a MIC. The same restriction is used in the next three rules forwarding observed vertex and edge labels, but now limited to vertices that can jointly belong to a MIC and to their respective regulators. Finally, the last two rules and the integrity constraint perform the same test as in (10) for a restricted set of pairs W and V. (The fact that *cycle*(V, W) implies $V \neq W$ in *labelE'*(W, U, V, S) also allows us to drop this condition, used in (10), from the bodies of the rules defining *receive'*.)

The complete optimized MIC encoding consists of the original rules in (4), (8), and (9), (11) and (12) as add-ons, and (13) as a replacement for (10). As regards the computational impact, we note that the optimized encoding needs less than two seconds for grounding and finding all MICs on the case study in Section 7.3, which took more than a minute with the unoptimized encoding.

A second version of the optimized encoding is obtained by tightening the consideration of connected vertices in $(V[W], E[W])$ relative to a MIC candidate W. This can be achieved by adding condition *active*(V) to the rules in (11) defining the *edges* predicate. In this way, the static reachability information encoded in (11), which is completely evaluated by grounder *gringo*, is turned into a dynamic relation computed during search. As it turns out, there is no significant performance difference between these two versions of the optimized MIC extraction encoding on the case study in Section 7.3. Hence, more real examples are needed to reliably compare their grounding and solving efficiency.

7 Empirical Evaluation and Application

For assessing the scalability of our approach, we start by conceiving a parameterizable suite of artificial yet biologically meaningful benchmarks. After that, we present a typical application stemming from real biological data, illustrating the exertion in practice. All experiments were performed using input reduction as explained in Section 6.1.

7.1 Checking Consistency

We first evaluate our approach on randomly generated instances, aiming at structures similar to those found in biological applications. Instances are composed of an influence graph, a complete labeling of its edges, and a partial labeling of its vertices. Our random generator takes three parameters: (i) the number α of vertices in the influence graph, (ii) the average degree β of the graph, and (iii) the proportion γ of observed variations for vertices. To generate an instance, we compute a random graph with α vertices (the value of α varying from 500 to 4000) under the model by Erdős-Rényi (1959). Each pair of vertices has equal probability to be connected via an edge, whose label is chosen independently with probability 0.5 for both signs. We fix the average degree β to 2.5, which is considered to be a typical value for biological networks (Jeong et al. 2000). Finally, $\lfloor \gamma\alpha \rfloor$ vertices are chosen with uniform probability and assigned a label with probability 0.5 for both signs. For each number α of vertices, we generated 50 instances using five different values for γ, viz., 0.01, 0.02, 0.033, 0.05, and 0.1. All instances are available at (BioASP Tools).

We used *gringo* (2.0.0) (Gebser et al. 2009a) for combining the generated instances and the encoding given in Section 4 into equivalent ground programs. For checking consistency

α	*claspD Berkmin*	*claspD VMTF*	*claspD VSIDS*	*cmodels*	*dlv*	*gnt*
500	0.14	0.11	0.11	0.16	0.46	0.71
1000	0.41	0.25	0.25	0.35	1.92	3.34
1500	0.79	0.38	0.38	0.53	4.35	7.50
2000	1.33	0.51	0.51	0.71	8.15	13.23
2500	2.10	0.66	0.66	0.89	13.51	21.88
3000	3.03	0.80	0.79	1.07	20.37	31.77
3500	3.22	0.93	0.92	1.15	21.54	34.39
4000	4.35	1.06	1.06	1.36	30.06	46.14

Table 2. *Run-times for consistency checking with claspD, cmodels, dlv, and gnt.*

by computing an answer set (if it exists), we ran disjunctive ASP solvers *claspD* (1.1) (Drescher et al. 2008) with "Berkmin", "VMTF", and "VSIDS" heuristics, *cmodels* (3.75) (Giunchiglia et al. 2006) using *zchaff*, *dlv* (BEN/Oct 11) (Leone et al. 2006), and *gnt* (2.1) (Janhunen et al. 2006). All runs were performed on a Linux machine equipped with an AMD Opteron 2 GHz processor and a memory limit of 2GB RAM.

Table 2 shows average run-times in seconds over 50 instances per number α of vertices, including grounding times of *gringo* and solving times. We checked that grounding times of *gringo* increase linearly with the number α of vertices, and they do not vary significantly over γ. For all solvers, run-times also increase linearly in α.[5] For fixed α values, we found two clusters of instances: consistent ones where total labelings were easy to compute, and inconsistent ones where inconsistency was detected from preassigned labels. This tells us that the influence graphs generated as described above are usually (too) easy to label consistently, and inconsistency only occurs if it is explicitly introduced via fixed labels. However, such constellations are not unlikely in practice (cf. Section 7.3), and isolating MICs from them, as done in the next subsection, turned out to be hard for most solvers. Finally, greater values for γ led to an increased proportion of inconsistent instances, without making them much harder.

7.2 Identifying Minimal Inconsistent Cores

We now investigate the problem of finding a MIC within the same setting as in the previous subsection. Because of the elevated size of ground instantiations and problem difficulty, we varied the number α of vertices from 50 to 300, thus, using considerably smaller influence graphs than before. We again use *gringo* for grounding, now taking the encoding given in Section 5. As regards solving, we restrict our attention to *claspD* because all three of the other solvers showed drastic performance declines.

Table 3 shows average run-times in seconds over 50 instances per number α of vertices. Timeouts, indicated in parentheses, are taken as maximum time of 1800 seconds. We observe a quadratic increase in grounding times of *gringo*, which is in line with the fact that ground instantiations for our MIC encoding grow quadratically with the size of

[5] Longer run-times of *claspD* with "Berkmin" in comparison to the other heuristics are due to a more expensive computation of heuristic values in the absence of conflict information. Furthermore, the time needed for performing "Lookahead" slows down *dlv* as well as *gnt*.

α	*gringo*	*claspD* *Berkmin*	*claspD* *VMTF*	*claspD* *VSIDS*
50	0.24	1.16 (0)	0.65 (0)	0.97 (0)
75	0.55	39.11 (1)	1.65 (0)	3.99 (0)
100	0.87	41.98 (1)	3.40 (0)	4.80 (0)
125	1.37	15.47 (0)	47.56 (1)	10.73 (0)
150	2.02	54.13 (0)	48.05 (0)	15.89 (0)
175	2.77	30.98 (0)	116.37 (2)	23.07 (0)
200	3.82	42.81 (0)	52.28 (1)	24.03 (0)
225	4.94	99.64 (1)	30.71 (0)	41.17 (0)
250	5.98	194.29 (3)	228.42 (5)	110.90 (1)
275	7.62	178.28 (2)	193.03 (4)	51.11 (0)
300	9.45	241.81 (2)	307.15 (7)	124.31 (0)

Table 3. *Run-times for grounding with gringo and solving with claspD.*

influence graphs. In fact, the schematic rules in Section 5.3 give rise to α copies of an influence graph. Considering solving times spent by *claspD* for finding one MIC (if it exists), we observe that they are relatively stable, in the sense that they are tightly correlated to grounding times. This regularity again confirms that, though it is random, the applied generation pattern tends to produce rather uniform influence graphs. Finally, we observed that unsatisfiable instances, i.e., consistent instances without any MIC, were easier to solve than the ones admitting answer sets. We conjecture that this is because consistent total labelings provide a disproof of inconsistency as encoded in Section 5.2.

As our experimental results demonstrate, computing MICs is computationally harder than just checking consistency. This is not surprising because the related (yet simpler) decision problem of verifying a MUC is D^P-complete (Dershowitz et al. 2006; Papadimitriou and Yannakakis 1982) and thus more complex than just deciding satisfiability. With our declarative technique, we spot the quadratic space blow-up incurred by the MIC encoding in Section 5 as a bottleneck. However, there are approaches aiming at a reduction of grounding efforts, and some of them have been presented in Section 6.

7.3 Biological Case Study

In the following, we present the results of applying our approach to real-world data of genetic regulations in yeast. We tested the gene-regulatory network of yeast provided in (Guelzim et al. 2002) against genetic profile data of *snf2* knock-outs (Sudarsanam et al. 2000) from the Saccharomyces Genome Database[6]. The regulatory network of yeast contains 909 genetic or biochemical regulations, all of which have been established experimentally, among 491 genes.

Comparing the yeast regulatory network with the genetic profile of *snf2*, we found the data to be inconsistent with the network, which was easily detected using the approach of Section 4. Applying our diagnosis technique from Section 5, we obtained a total of 19 MICs. While computing the first MIC took less than a second using *gringo* and *claspD* (regardless of the heuristic used), the computation of all MICs was considerably harder. Us-

[6] `http://www.yeastgenome.org`

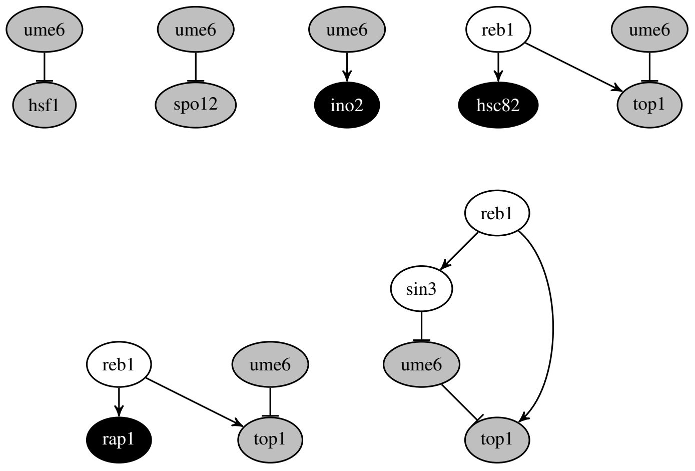


Figure 5. Some MICs obtained by comparing the regulatory network of yeast with a genetic profile.

ing "VMTF" as search heuristic on top of the enumeration algorithm (Gebser et al. 2007) inherited from *clasp* (Gebser et al. 2009c), *claspD* had found all 19 MICs in about 30 seconds, while another 40 seconds were needed to decide that there is no further MIC. With "VSIDS", finding the 19 MICs took about the same time as with "VMTF", but another 80 seconds were used to verify that all MICs had been found. Finally, using "Berkmin" heuristic, 12 MICs had been found before aborting after 30 minutes. The observation that search heuristics matter tells us that investigations into the structure of biological problems and particular methods to solve them efficiently can earn considerable benefits.[7] Furthermore, we note that the potential existence of multiple answer sets encompassing the same MIC did not emerge on the yeast network and *snf2* knock-out data. That is, we obtained 19 answer sets, each one corresponding one-to-one to a MIC.

Six of the computed MICs are exemplarily shown in Figure 5. While the first three of them are pretty obvious, we also identified more complex topologies. However, our example demonstrates that the MICs obtained in practice are still small enough to be understood easily. For finding suitable corrections to the inconsistencies, it is often even more helpful to display the connections between several overlapping MICs. Observe that all six MICs in Figure 5 are related to gene *ume6*. Connecting them yields the subgraph of the yeast regulatory network in Figure 6.

The most obvious problem in Figure 6 is that the observed increase of *ume6* is incompatible with its four targets. This suggests that either the observation on *ume6* is incorrect or that some regulations are missing or wrongly modeled. In the first hypothesis though, one should note that the current model cannot explain a decrease of *ume6*: this would imply an increase of *sin3* and in turn an increase of *reb1*, but then there would be no explanation left for the variation of *hsc82* and *rap1*. So, in either case, our model should be revised. This is

[7] Notably, by exploiting additional background knowledge, the optimized encoding presented in Section 6.2 requires less than two seconds (regardless of heuristics) for grounding and finding all 19 MICs. In fact, its ground instantiation contains only 8481 atoms and 10843 rules, compared to 47260 atoms and 56522 rules with the basic encoding in Section 5. In addition to problem size, also the difficulty drops dramatically: from 23345 conflicts down to 270 conflicts, encountered with "VMTF" heuristic during search for all answer sets.

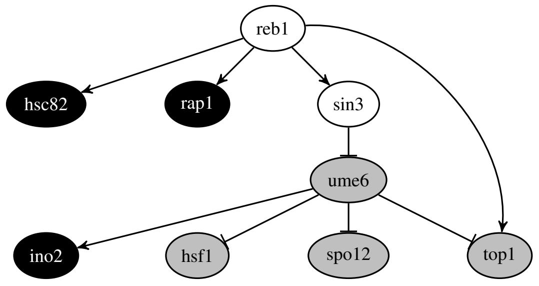


Figure 6. Subgraph obtained by connecting the six MICs given in Figure 5.

not a great surprise: our literature-based network, although very reliable, was presumably far from being complete.

Regarding the biological background, note that *ume6* is a known regulator of sporulation in yeast: in case of nutritional stress, yeast cells stop dividing and produce spores by meiosis. These spores are reproductive structures better adapted to extreme conditions. *ume6* is known as a key inhibitor of early meiotic genes: upon entry in meiosis, this inhibitory effect is released and the target genes are expressed. Notably, a knock-out of *ume6* causes the expression of meiotic genes during vegetative growth (hence its name, *Unscheduled Meiotic Expression*) as well as almost complete failure of sporulation (Washburn and Esposito 2006). *ume6* seems to have activation capabilities as well, though in that case the effect is believed to be indirect (Chen et al. 2007).

In the current view, *ume6* switches from inhibitor to (indirect) activator at the beginning of meiosis: Ume6p (the protein corresponding to the gene *ume6*) has a repressive effect when it forms a complex with Sin3p (note that *sin3* is in our network) and Rdp3p, which is degraded upon entry in meiosis (Mallory et al. 2007). This molecular mechanism can be interpreted in our model and one possible result is given in Figure 7. At least for negative targets, we now have a plausible explanation: the real effector of the inhibition on *hsf1*, *spo12*, *top1*, and *ume6* itself is the complex Ume6p-Sin3p, whose variation is unobserved but depends on the variation of *ume6* and *sin3*. The variation of the targets can be explained if the protein complex decreases, which is in turn possible if *sin3* decreases. Regretfully *sin3* is not observed in our data, but we note that a decrease of this gene is fully compatible with the rest of the network, that is, if we suppose a decrease of *reb1*. Now concerning *ino2*, our network should be updated with more recent evidence: as reviewed in (Chen et al. 2007), *ino2* has several additional regulators, such as *opi1* and *pah1* (see Figure 7). The observed variation of *pah1* is not useful to explain that of *ino2*, but *opi1* is definitely a plausible candidate.

Here we illustrated one main usage of our diagnosis technique: identifying poorly modeled regions of a regulatory network that are incompatible with a given data set. This is definitely a key asset if one wants to build a large-scale regulatory database and check its coherence with newly produced data on a regular basis. Given new data, our diagnosis method produces human-understandable representations of possible incompatibilities with the current model, which serve as the basis for a targeted literature research. With this data-driven approach, a network can then be improved with considerably less effort than with a random traversal of publications, for a much more coherent result.

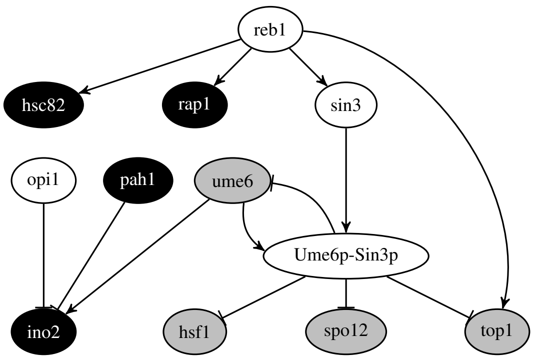


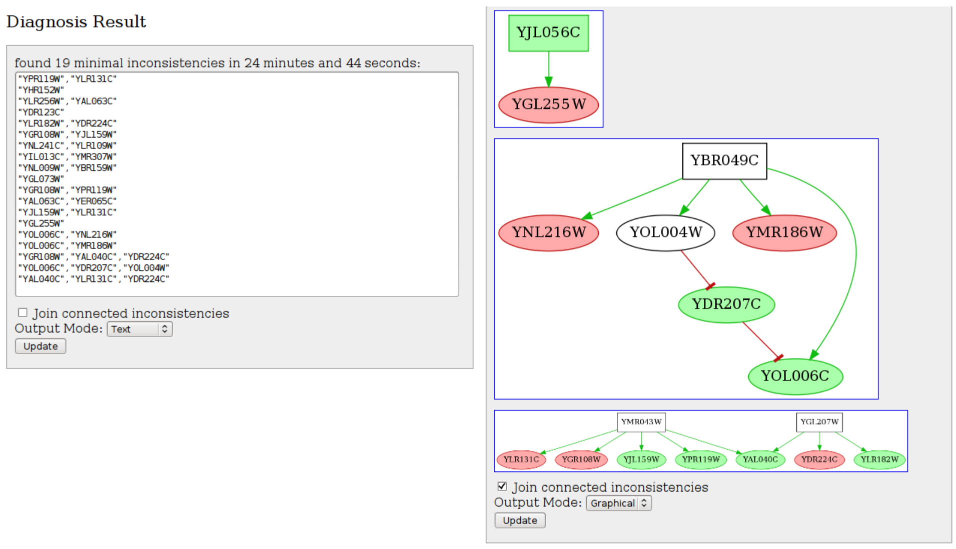
Figure 7. Local correction of the network based on our diagnosis method and literature research.

8 Web Service

To make our methods easily accessible to a biological audience, we built a web service[8] not requiring any locally installed software on the user side except for a web browser. It provides the possibility to upload textual representations of biological networks as well as experimental profiles. Also, a number of predefined examples allows a user to instantly experience the functionalities of the web service. These include consistency checking and diagnosis, i.e., finding MICs, whose implementation has been detailed in Section 4 and 5.

Influence graphs representing biological networks usually contain vertices that are not subject to any regulation. Such entities are understood as controlled by external factors, like environmental or particular experimental conditions. To avoid trivial inconsistencies due to such unregulated and thus unexplainable vertices, the web interface provides an option “Guess input nodes” for automatically declaring all vertices without any predecessor as inputs. While consistency checking simply results in a positive or negative answer, we offer three diagnosis modes: “find one inconsistency”, “find all inconsistencies”, and “approximate all inconsistencies”. The first mode aims at finding a single MIC, and the second at finding all of them. For the latter, we currently use an encapsulating script that repeatedly calls *claspD* while feeding already identified MICs back as integrity constraints, until no further answer set exists. This makes sure that each answer set corresponds to a new MIC and thus avoids potential repetitions. The problem of enumerating answer sets that differ on a set of “relevant” atoms (in our case, on instances of predicate *active*) is addressed in (Gebser et al. 2009b). The integration of this technique into *claspD*, in order to make the wrapper script obsolete, is subject to future work. Once MICs have been computed, they can be represented either textually or graphically, as shown in Figure 8. If the result consists of several MICs, it is possible to view overlapping ones in a combined way, thus highlighting regions of inconsistency. Finally, the third diagnosis mode, “approximate all inconsistencies”, works by marking the vertices of a computed MIC as inputs before proceeding to look for further MICs. This approach has been used in previous work (Guziolowski et al. 2009) and has been integrated into our framework for comparison. However, the results obtained with the third mode depend on the order in which MICs are found

[8] `http://data.haiti.cs.uni-potsdam.de/wsgi/app`


Diagnosis Result

found 19 minimal inconsistencies in 24 minutes and 44 seconds:

```
"YPR119W","YLR131C"
"YHR152W"
"YLR256W","YAL063C"
"YDR123C"
"YLR182W","YDR224C"
"YGR108W","YJL159W"
"YNL241C","YLR109W"
"YIL013C","YMR307W"
"YNL009W","YBR159W"
"YGL073W"
"YGR108W","YPR119W"
"YAL063C","YER065C"
"YJL159W","YLR131C"
"YGL255W"
"YOL006C","YNL216W"
"YOL006C","YMR186W"
"YGR108W","YAL040C","YDR224C"
"YOL006C","YDR207C","YOL004W"
"YAL040C","YLR131C","YDR224C"
```

☐ Join connected inconsistencies
Output Mode: Text

☑ Join connected inconsistencies
Output Mode: Graphical

Figure 8. Representation of identified MICs in textual (left) and graphical (right) mode.

and their vertices declared to be inputs in future computations. Further functionalities, like prediction under consistency (Guziolowski et al. 2007) and inconsistency (Gebser et al. 2010), are also featured by the web service but are outside the scope of this paper.

9 Discussion

We have provided an approach based on ASP to investigate the consistency between experimental profiles and influence graphs. In case of inconsistency, the concept of a MIC can be exploited for identifying concise explanations, pointing to unreliable data and/or missing reactions. The problem of finding MICs is closely related to the extraction of MUCs in the context of SAT. From a knowledge representation point of view, however, we argue for our ASP-based technique, as it provides an easy way to model a problem in terms of a uniform encoding and specific instances.

The BioQuali system (Guziolowski et al. 2009) provides functionalities parallel to our approach. It also works on influence graphs and applies the same consistency notion. In preprocessing, BioQuali reduces an influence graph by iteratively marking unobserved vertices that have no successors as uncritical. This technique is also realized by input reduction, described in Section 6.1. After that, BioQuali transforms the reduced subgraph into a Binary Decision Diagram, used for further computations. While consistency checking with BioQuali yields the same results as our technique, its diagnosis functionality works like the "approximate all inconsistencies" mode, described in the previous section. In contrast to our method, this does in general not admit finding all MICs.

By now, a variety of efficient ASP tools are available, both for grounding and for solving logic programs. Our empirical assessment of them (on random as well as real data) has in principle demonstrated the scalability of the approach. The web service implementation of finding all MICs, which is genuine to our method and not available in any other existing tool, is still based on some workarounds for avoiding redundant answer sets. It is a subject of future work to address this with answer set projection (Gebser et al. 2009b).

As elegance and flexibility in modeling are major advantages of ASP, our current application makes it attractive also for related biological questions, beyond the ones addressed in this paper. For instance, ongoing work deals with repair and prediction under consistency as well as inconsistency (Gebser et al. 2010). In future, it will also be interesting to explore how far the performance of ASP tools can be tuned by varying and optimizing encodings for particular tasks. In turn, challenging applications like the one presented here might contribute to the further improvement of ASP tools, as they might be geared towards efficiency in such domains.

Appendix A Proof of Theorem 4.1 and 4.2

We formalize the representation of instances, as described in Section 4.1, by defining a mapping τ of an influence graph (V, E, σ) and a (partial) vertex labeling $\mu : V \rightarrow \{+,-\}$:

$$\begin{aligned}\tau((V,E,\sigma),\mu) = {} & \{\mathit{vertex}(i). & \mid i \in V\} \\ \cup {} & \{\mathit{edge}(j,i). & \mid (j \rightarrow i) \in E\} \\ \cup {} & \{\mathit{observedE}(j,i,s). & \mid (j \rightarrow i) \in E, \sigma(j,i) = s\} \\ \cup {} & \{\mathit{observedV}(i,s). & \mid i \in V, \mu(i) = s\} \\ \cup {} & \{\mathit{input}(i). & \mid i \in V \text{ is an input}\}\ .\end{aligned} \tag{A1}$$

By P_C, we denote the encoding containing the schematic rules in (3), (4), (6), and (7).

Proof of Theorem 4.1
Assume that X is an answer set of $P_C \cup \tau((V,E,\sigma),\mu)$. Furthermore, let

$$\begin{aligned}P^X = \{ & (\mathit{head}(r) \leftarrow \mathit{body}(r)^+)\theta \mid \\ & r \in P_C \cup \tau((V,E,\sigma),\mu), (\mathit{body}(r)^-\theta) \cap X = \emptyset, \theta : \mathit{var}(r) \rightarrow \mathcal{U}\}\end{aligned}$$

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

Given X, we define σ' and μ' as follows:

$$\begin{aligned}\sigma' &= \{(j \rightarrow i) \mapsto s \mid (j \rightarrow i) \in E, \mathit{labelE}(j,i,s) \in X\} \\ \mu' &= \{i \mapsto s \mid i \in V, \mathit{labelV}(i,s) \in X\}\ .\end{aligned}$$

We show that σ' and μ' are total labelings of edges and vertices, respectively, such that $\mu'(i) = \mu'(j)\sigma'(j,i)$ holds for every non-input vertex $i \in V$ and some edge $j \rightarrow i$ in E.

Regarding the uniqueness of labels assigned by σ' and μ', consider the following rules from (3) and (4) including predicates *labelE* and *labelV* in their heads:

$$\begin{aligned}\mathit{labelV}(V,+); \mathit{labelV}(V,-) &\leftarrow \mathit{vertex}(V). \\ \mathit{labelE}(U,V,+); \mathit{labelE}(U,V,-) &\leftarrow \mathit{edge}(U,V). \\ \mathit{labelV}(V,S) &\leftarrow \mathit{observedV}(V,S). \\ \mathit{labelE}(U,V,S) &\leftarrow \mathit{observedE}(U,V,S).\end{aligned} \tag{A2}$$

Since the given (partial) labelings σ and μ assign unique labels to the elements of their domains, facts defining *observedE* and *observedV* are of the form *observedE*$(j, i, +)$. or *observedE*$(j, i, -)$. and *observedV*$(i, +)$. or *observedV*$(i, -)$., respectively, and at most one of these facts is contained in $\tau((V, E, \sigma), \mu)$ for an edge $(j \rightarrow i) \in E$ or a vertex $i \in V$. Because X is a $\subseteq$-minimal model of P^X, the atoms in the heads of facts are in X, and all atoms in X over predicates *observedE* and *observedV* are derived from facts in $\tau((V, E, \sigma), \mu)$, in view that these predicates do not occur in the head of any rule in P_C. Hence, at most one of the atoms *labelE*$(j, i, +)$ and *labelE*$(j, i, -)$ or *labelV*$(i, +)$ and *labelV*$(i, -)$, respectively, is derivable for an edge $(j \rightarrow i) \in E$ or vertex $i \in V$ from a ground instance of the fourth or third rule in (A2) and then included in X. Furthermore, the second and first rule in (A2) impose that at least one of *labelE*$(j, i, +)$ or *labelE*$(j, i, -)$ and *labelV*$(i, +)$ or *labelV*$(i, -)$ belongs to X for every edge $(j \rightarrow i) \in E$ and vertex $i \in V$, respectively, while the atom containing the opposite label cannot belong to a $\subseteq$-minimal model of P^X. Hence, there is at most one term s such that *labelE*$(j, i, s) \in X$ or *labelV*$(i, s) \in X$ for an edge $(j \rightarrow i) \in E$ or vertex $i \in V$, respectively, and it holds that $s \in \{+, -\}$, which allows us to conclude that σ' and μ' are total labelings.

As regards extending σ and μ, we have that fact *observedE*(j, i, s). or *observedV*(i, s). belongs to $\tau((V, E, \sigma), \mu)$ if $\sigma(j, i) = s$ or $\mu(i) = s$, respectively, is given. This implies that *labelE*$(j, i, s) \in X$ or *labelV*$(i, s) \in X$, respectively, as the fourth or third rule in (A2) would be unsatisfied otherwise. Thus, $\sigma'(j, i) = s$ if $\sigma(j, i) = s$, and $\mu'(i) = s$ if $\mu(i) = s$.

It remains to be shown that $\mu'(i)$ is consistent for each non-input vertex $i \in V$. To this end, we note that the integrity constraint

$$\leftarrow \textit{labelV}(V, S), \textit{not receive}(V, S), \textit{not input}(V).$$

from (7) necessitates *receive*$(i, r) \in X$ if $\mu'(i) = r$ (that is, if *labelV*$(i, r) \in X$) for a non-input vertex $i \in V$. Otherwise, P^X would contain an unsatisfied ground instance in view that *input*$(i) \in X$ exactly if fact *input*(i). is included in $\tau((V, E, \sigma), \mu)$. However, any ground instances of the integrity constraint contributing to P^X do not contain atoms over predicate *receive*. Such atoms can only be derived using the following rules from (6):

$$\begin{array}{l} \textit{receive}(V, +) \leftarrow \textit{labelE}(U, V, S), \textit{labelV}(U, S). \\ \textit{receive}(V, -) \leftarrow \textit{labelE}(U, V, S), \textit{labelV}(U, T), S \neq T. \end{array}$$

Since X is a $\subseteq$-minimal model of P^X, *receive*$(i, +) \in X$ or *receive*$(i, -) \in X$ is possible only if *labelE*$(j, i, s) \in X$ and *labelV*$(j, t) \in X$ such that $s = t$ or $s \neq t$, that is, if $\sigma'(j, i) = s$ and $\mu'(j) = t$ such that $\mu'(j)\sigma'(j, i) = +$ or $\mu'(j)\sigma'(j, i) = -$, respectively. As *labelV*(i, r) is accompanied by *receive*(i, r) in X for each non-input vertex $i \in V$, this allows us to conclude that $\mu'(i) = r$ implies $\mu'(j)\sigma'(j, i) = r$ for some regulator j of i. Hence, we have that $\mu'(i)$ is consistent for each non-input vertex $i \in V$. $\square$

Proof of Theorem 4.2

Assume that (V, E, σ) and μ are consistent. Then, there are total extensions $\sigma' : E \rightarrow \{+, -\}$ of σ and $\mu' : V \rightarrow \{+, -\}$ of μ such that, for each non-input vertex $i \in V$, we have $\mu'(i) = \mu'(j)\sigma'(j, i)$ for some edge $j \rightarrow i$ in E.

We consider the following set X of atoms:

$$\begin{array}{rll} X = & \{\textit{vertex}(i), \textit{labelV}(i,s) & \mid i \in V, \mu'(i) = s\} \\ \cup & \{\textit{edge}(j,i), \textit{labelE}(j,i,s) & \mid (j \rightarrow i) \in E, \sigma'(j,i) = s\} \\ \cup & \{\textit{receive}(i,ts) & \mid (j \rightarrow i) \in E, \sigma'(j,i) = s, \mu'(j) = t\} \\ \cup & \{\textit{observedE}(j,i,s) & \mid (j \rightarrow i) \in E, \sigma(j,i) = s\} \\ \cup & \{\textit{observedV}(i,s) & \mid i \in V, \mu(i) = s\} \\ \cup & \{\textit{input}(i) & \mid i \in V \text{ is an input}\}\ . \end{array}$$

For showing that X is an answer set of $P_C \cup \tau((V,E,\sigma),\mu)$, we need to verify that X is a $\subseteq$-minimal model of

$$\begin{aligned} P^X = \{ & (head(r) \leftarrow body(r)^+)\theta \mid \\ & r \in P_C \cup \tau((V,E,\sigma),\mu), (body(r)^-\theta) \cap X = \emptyset, \theta : var(r) \rightarrow \mathcal{U}\} \end{aligned}$$

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

To start with, we note that X includes an atom *vertex*(i), *edge*(j,i), *observedE*(j,i,s), *observedV*(i,s), and *input*(i), respectively, exactly if there is a fact with the atom in the head in $\tau((V,E,\sigma),\mu)$. Each of these facts belongs also to P^X, is satisfied by X, but not by any set Y of atoms excluding at least one of the head atoms. Furthermore, since σ' and μ' are total mappings, we have that $|\{\textit{labelE}(j,i,+), \textit{labelE}(j,i,-)\} \cap X| = 1$ and $|\{\textit{labelV}(i,+), \textit{labelV}(i,-)\} \cap X| = 1$ for every $(j \rightarrow i) \in E$ and $i \in V$, respectively. Hence, X, but no subset Y of X excluding at least one atom over predicates *labelE* and *labelV*, satisfies all ground instances of the following rules from (3) in P^X:

$$\begin{aligned} \textit{labelV}(V,+); \textit{labelV}(V,-) &\leftarrow \textit{vertex}(V). \\ \textit{labelE}(U,V,+); \textit{labelE}(U,V,-) &\leftarrow \textit{edge}(U,V). \end{aligned}$$

In addition, since σ' and μ' extend σ and μ, respectively, all ground instances of the following rules from (4) in P^X are satisfied by X:

$$\begin{aligned} \textit{labelV}(V,S) &\leftarrow \textit{observedV}(V,S). \\ \textit{labelE}(U,V,S) &\leftarrow \textit{observedE}(U,V,S). \end{aligned}$$

Since *labelE*$(j,i,s) \in X$ and *labelV*$(j,t) \in X$ if $\sigma'(j,i) = s$ and $\mu'(j) = t$, respectively, we have that *receive*$(i,ts) \in X$ exactly if there is a ground instance of the rules

$$\begin{aligned} \textit{receive}(V,+) &\leftarrow \textit{labelE}(U,V,S), \textit{labelV}(U,S). \\ \textit{receive}(V,-) &\leftarrow \textit{labelE}(U,V,S), \textit{labelV}(U,T), S \neq T. \end{aligned}$$

from (6) in P^X such that *labelE*(j,i,s), *labelV*$(j,t) \in X$ occur in the body and *receive*(i,ts) in the head. Hence, no subset Y of X excluding any atom over predicate *receive* is a model of P^X. Finally, since $\mu'(i) = \mu'(j)\sigma'(j,i)$ for each non-input vertex $i \in V$ and some $j \rightarrow i$ in E, *labelV*$(i,r) \in X$ implies that *receive*$(i,r) \in X$. That is, the ground instances of the integrity constraint

$$\leftarrow \textit{labelV}(V,S), \textit{not receive}(V,S), \textit{not input}(V).$$

from (7) that contribute to P^X are satisfied by X.

We have now investigated all rules in $P_C \cup \tau((V,E,\sigma),\mu)$ and shown that their ground instances in P^X are satisfied by X. Furthermore, we have checked for all atoms in X that they cannot be excluded in any model $Y \subset X$ of P^X. That is, X is indeed a $\subseteq$-minimal model of P^X and thus an answer set of $P_C \cup \tau((V,E,\sigma),\mu)$. $\square$

Appendix B Proof of Theorem 5.1 and 5.2

This appendix provides proofs for soundness and completeness of the MIC extraction encoding in Section 5. We use $\tau((V,E,\sigma),\mu)$ as defined in (A1) to refer to the facts representing an influence graph (V,E,σ) and a (partial) vertex labeling $\mu : V \to \{\mathsf{+},\mathsf{-}\}$. By P_D, we denote the encoding consisting of the schematic rules in (4), (8), (9), and (10).

As an auxiliary concept, for any subset $W \subseteq V$, we say that $\sigma' : E \to \{\mathsf{+},\mathsf{-}\}$ and $\mu' : V \to \{\mathsf{+},\mathsf{-}\}$ are *witnessing labelings* for W if the following conditions hold:

1. σ' and μ' are total,
2. if $\sigma(j,i)$ is defined, then $\sigma'(j,i) = \sigma(j,i)$,
3. if $\mu(i)$ is defined, then $\mu'(i) = \mu(i)$, and
4. $\mu'(i)$ is consistent (relative to σ') for each non-input vertex $i \in W$.

The above conditions make sure that σ' and μ' are total extensions of σ and μ, respectively, such that the variations of vertices in W are explained. Comparing Definition 5.1, the first condition requires the absence of witnessing labelings for a MIC W, while the second condition stipulates the existence of witnessing labelings for each $W' \subset W$.

Proof of Theorem 5.1
Assume that X is an answer set of $P_D \cup \tau((V,E,\sigma),\mu)$. Furthermore, let

$$\begin{aligned} P^X = \{ & (\mathit{head}(r) \leftarrow \mathit{body}(r)^+)\theta \mid \\ & r \in P_D \cup \tau((V,E,\sigma),\mu), (\mathit{body}(r)^-\theta) \cap X = \emptyset, \theta : \mathit{var}(r) \to \mathcal{U}\} \end{aligned}$$

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

Let $W = \{i \mid \mathit{active}(i) \in X\}$. We have to show that the following conditions hold:

1. There are witnessing labelings for each $W' \subset W$.
2. There are no witnessing labelings for W.

We below consider these conditions one after the other.

Condition 1. Let $W' = W \setminus \{k\}$ for any $k \in W$. Furthermore, define σ' and μ' as follows:

$$\begin{aligned} \sigma' = {} & \{(j \to i) \mapsto s \mid (j \to i) \in E, \mathit{labelE'}(k,j,i,s) \in X\} \\ & \cup \{(j \to i) \mapsto \mathsf{+} \mid (j \to i) \in E, \mathit{labelE'}(k,j,i,\mathsf{+}) \notin X, \mathit{labelE'}(k,j,i,\mathsf{-}) \notin X\} \\ \mu' = {} & \{i \mapsto s \mid i \in V, \mathit{labelV'}(k,i,s) \in X\} \\ \cup {} & \{i \mapsto \mathsf{+} \mid i \in V, \mathit{labelV'}(k,i,\mathsf{+}) \notin X, \mathit{labelV'}(k,i,\mathsf{-}) \notin X\}\ . \end{aligned}$$

We show that σ' and μ' are witnessing labelings for W'.

Regarding the uniqueness of labels assigned by σ' and μ', consider the following rules from (10) including predicates *labelE'* and *labelV'* in their heads:

$$\begin{aligned} \textit{labelV'}(W,V,\texttt{+});\textit{labelV'}(W,V,\texttt{-}) &\leftarrow \textit{active}(W),\textit{vertexMIC}(V).\\ \textit{labelE'}(W,U,V,\texttt{+});\textit{labelE'}(W,U,V,\texttt{-}) &\leftarrow \textit{active}(W),\textit{edgeMIC}(U,V).\\ \textit{labelV'}(W,V,S) &\leftarrow \textit{active}(W),\textit{observedV}(V,S).\\ \textit{labelE'}(W,U,V,S) &\leftarrow \textit{active}(W),\textit{observedE}(U,V,S). \end{aligned} \tag{B1}$$

Since the given (partial) labelings σ and μ assign unique labels to the elements of their domains, facts defining *observedE* and *observedV* are of the form $\textit{observedE}(j,i,\texttt{+})$. or $\textit{observedE}(j,i,\texttt{-})$. and $\textit{observedV}(i,\texttt{+})$. or $\textit{observedV}(i,\texttt{-})$., respectively, and at most one of these facts is contained in $\tau((V,E,\sigma),\mu)$ for an edge $(j \rightarrow i) \in E$ or vertex $i \in V$. Because X is a $\subseteq$-minimal model of P^X, the atoms in the heads of facts are in X, and all atoms in X over predicates *observedE* and *observedV* are derived from facts in $\tau((V,E,\sigma),\mu)$, in view that these predicates do not occur in the head of any rule in P_D. Hence, at most one of the atoms $\textit{labelE'}(k,j,i,\texttt{+})$ and $\textit{labelE'}(k,j,i,\texttt{-})$ or $\textit{labelV'}(k,i,\texttt{+})$ and $\textit{labelV'}(k,i,\texttt{-})$, respectively, is derivable for an edge $(j \rightarrow i) \in E$ or vertex $i \in V$ from a ground instance of the fourth or third rule in (B1) and then included in X. If either of $\textit{labelE'}(k,j,i,\texttt{+})$ and $\textit{labelE'}(k,j,i,\texttt{-})$ or $\textit{labelV'}(k,i,\texttt{+})$ and $\textit{labelV'}(k,i,\texttt{-})$, respectively, is included in X, then the ground instance of the second or first rule in (B1) for k and an edge $(j \rightarrow i) \in E$ or vertex $i \in V$ is satisfied, so that the atom containing the opposite label cannot belong to a $\subseteq$-minimal model of P^X. Hence, there is at most one term s such that $\sigma'(j,i) = s$ or $\mu'(i) = s$ for an edge $(j \rightarrow i) \in E$ or vertex $i \in V$, respectively, and it holds that $s \in \{\texttt{+},\texttt{-}\}$. Furthermore, looking at the definitions of σ' and μ', it is obvious that both are total, which allows us to conclude that σ' and μ' are total labelings.

As regards extending σ and μ, we have that fact $\textit{observedE}(j,i,s)$. or $\textit{observedV}(i,s)$. belongs to $\tau((V,E,\sigma),\mu)$ if $\sigma(j,i) = s$ or $\mu(i) = s$, respectively, is given. Along with the premise that $\textit{active}(k) \in X$, this implies that $\textit{labelE'}(k,j,i,s) \in X$ or $\textit{labelV'}(k,i,s) \in X$, respectively, as the fourth or third rule in (B1) would be unsatisfied otherwise. Hence, we have $\sigma'(j,i) = s$ if $\sigma(j,i) = s$, and $\mu'(i) = s$ if $\mu(i) = s$.

It remains to be shown that $\mu'(i)$ is consistent for each non-input vertex $i \in W'$. To establish this, we first consider the following rules from (8):

$$\begin{aligned} \textit{edgeMIC}(U,V) &\leftarrow \textit{edge}(U,V),\textit{active}(V).\\ \textit{vertexMIC}(U) &\leftarrow \textit{edgeMIC}(U,V).\\ \textit{vertexMIC}(V) &\leftarrow \textit{active}(V). \end{aligned} \tag{B2}$$

In view that fact $\textit{edge}(j,i)$. belongs to $\tau((V,E,\sigma),\mu)$ for every $(j \rightarrow i) \in E$, we conclude that $\textit{edge}(j,i) \in X$. Along with $\textit{active}(i) \in X$ for every $i \in W$, it follows that $\textit{edgeMIC}(j,i) \in X$ for every $(j \rightarrow i) \in E$ such that $i \in W$, and $\textit{vertexMIC}(i) \in X$ for every $i \in W$. The last observation and the first rule in (B1) imply that $\textit{labelV'}(k,i,\texttt{+}) \in X$ or $\textit{labelV'}(k,i,\texttt{-}) \in X$ for every $i \in W$. For $i \in W'$, i.e., $i \neq k$, the integrity constraint

$$\leftarrow \textit{labelV'}(W,V,S),\textit{active}(V),V \neq W,\textit{not receive'}(W,V,S).$$

from (10) imposes $\textit{receive'}(k,i,\texttt{+}) \in X$ if $\textit{labelV'}(k,i,\texttt{+}) \in X$, and $\textit{receive'}(k,i,\texttt{-}) \in X$ if $\textit{labelV'}(k,i,\texttt{-}) \in X$, while any ground instances of the integrity constraint contributing to P^X do not contain atoms over predicate *receive'*. Such atoms can only be derived using the following rules from (10):

$$\begin{aligned}
&\textit{receive'}(W,V,\texttt{+}) \leftarrow \textit{labelE'}(W,U,V,S), \textit{labelV'}(W,U,S), V \neq W.\\
&\textit{receive'}(W,V,\texttt{-}) \leftarrow \textit{labelE'}(W,U,V,S), \textit{labelV'}(W,U,T), V \neq W, S \neq T.
\end{aligned}$$

Since X is a $\subseteq$-minimal model of P^X, *receive'*$(k,i,\texttt{+}) \in X$ or *receive'*$(k,i,\texttt{-}) \in X$ is possible only if *labelE'*$(k,j,i,s) \in X$ and *labelV'*$(k,j,t) \in X$ such that $s = t$ or $s \neq t$, respectively. Comparing $\tau((V,E,\sigma),\mu)$ and the rules in (B1), (B2), as well as (B3) reveals that $(j \to i) \in E$ is a necessary condition for *labelE'*$(k,j,i,s) \in X$, and the same applies to $j \in V$ and *labelV'*$(k,j,t) \in X$. By the construction of σ' and μ', *labelE'*$(k,j,i,s) \in X$ implies that $\sigma'(j,i) = s$ and *labelV'*$(k,j,t) \in X$ that $\mu'(j) = t$. We conclude that *receive'*$(k,i,\texttt{+}) \in X$ or *receive'*$(k,i,\texttt{-}) \in X$ necessitates $\mu'(j)\sigma'(j,i) = \texttt{+}$ or $\mu'(j)\sigma'(j,i) = \texttt{-}$, respectively, for some regulator j of i. Finally, we have $\mu'(i) = \texttt{+}$ if *labelV'*$(k,i,\texttt{+}) \in X$ (and *receive'*$(k,i,\texttt{+}) \in X$), and $\mu'(i) = \texttt{-}$ if *labelV'*$(k,i,\texttt{-}) \in X$ (and *receive'*$(k,i,\texttt{-}) \in X$). This shows that i receives some influence matching $\mu'(i)$, so that $\mu'(i)$ is consistent. Since $i \in W'$ is arbitrary, σ' and μ' are witnessing labelings for W'.

To conclude the proof of the first condition to verify, we note that witnessing labelings for W' are also witnessing labelings for all subsets of W'. Hence, it is sufficient to check the existence of witnessing labelings for sets $W' = W \setminus \{k\}$ for any $k \in W$. As shown above, an answer set X of $P_D \cup \tau((V,E,\sigma),\mu)$ yields witnessing labelings for them. Hence, the second condition in Definition 5.1 holds for $W = \{i \mid \textit{active}(i) \in X\}$.

Condition 2. We now show by contradiction that there cannot be witnessing labelings for W. To establish this, we first note that vertices in W cannot be input because, if fact *input*(i). belongs to $\tau((V,E,\sigma),\mu)$, then *input*(i) must be included in X, so that the rule

$$\textit{active}(V); \textit{inactive}(V) \leftarrow \textit{vertex}(V), \textit{not input}(V). \tag{B3}$$

from (8) does not contribute a ground instance for i to P^X. Since *active*(i) cannot be derived from any other ground rule in P^X, the fact that X is a $\subseteq$-minimal model of P^X implies that *active*$(i) \notin X$ for any input vertex i. Furthermore, the integrity constraint

$$\leftarrow \textit{not bottom}. \tag{B4}$$

from (9) necessitates *bottom* $\in X$ because X cannot be a model of P^X otherwise. Then, we get *labelV*$(i,\texttt{+})$, *labelV*$(i,\texttt{-}) \in X$ and *labelE*$(j,i,\texttt{+})$, *labelE*$(j,i,\texttt{-}) \in X$ for all vertices $i \in V$ and edges $(j \to i) \in E$, respectively, due to the following rules from (9):

$$\begin{aligned}
\textit{labelV}(V,\texttt{+}) &\leftarrow \textit{bottom}, \textit{vertex}(V).\\
\textit{labelV}(V,\texttt{-}) &\leftarrow \textit{bottom}, \textit{vertex}(V).\\
\textit{labelE}(U,V,\texttt{+}) &\leftarrow \textit{bottom}, \textit{edge}(U,V).\\
\textit{labelE}(U,V,\texttt{-}) &\leftarrow \textit{bottom}, \textit{edge}(U,V).
\end{aligned} \tag{B5}$$

We now show that the existence of witnessing labelings for W yields a contradiction to the fact that X is a $\subseteq$-minimal model of P^X. To this end, assume that σ' and μ' are witnessing labelings for W. Then, let

$$\begin{aligned}
Y = (X \setminus (&\{\mathit{bottom}\} \\
&\cup \{\mathit{labelV}(i,s) \mid \mathit{labelV}(i,s) \in X\} \\
&\cup \{\mathit{labelE}(j,i,s) \mid \mathit{labelE}(j,i,s) \in X\} \\
&\cup \{\mathit{opposite}(j,i) \mid \mathit{opposite}(j,i) \in X\})) \\
\cup\ &\{\mathit{labelV}(i,s) \mid i \in V, \mu'(i) = s\} \\
\cup\ &\{\mathit{labelE}(j,i,s) \mid (j \rightarrow i) \in E, \sigma'(j,i) = s\} \\
\cup\ &\{\mathit{opposite}(j,i) \mid (j \rightarrow i) \in E, \mu'(i) \neq \mu'(j)\sigma'(j,i)\} \ .
\end{aligned}$$

Since *bottom* $\in X \setminus Y$ and X contains a maximum amount of atoms over predicates *labelV*, *labelE*, and *opposite* (the atoms over *opposite* are consequences of the inclusion of atoms over *labelV* and *labelE*), we have that $Y \subset X$, and we show that Y is a model of P^X.

Considering the contributions of the facts in $\tau((V,E,\sigma),\mu)$ and the rules in (10) to P^X, we observe that the atoms over predicates occurring in them are interpreted the same in X and Y. Hence, such facts and rules stay satisfied by Y because they were already satisfied by X. The same applies to the rules from (8) repeated in (B2) and (B3). Furthermore, since σ' and μ' are total and extend σ and μ, respectively, the contributions of the following rules from (4) and (8) to P^X are satisfied by Y:

$$\begin{aligned}
\mathit{labelV}(V,S) &\leftarrow \mathit{observedV}(V,S). \\
\mathit{labelE}(U,V,S) &\leftarrow \mathit{observedE}(U,V,S). \\
\mathit{labelV}(V,\texttt{+}); \mathit{labelV}(V,\texttt{-}) &\leftarrow \mathit{vertexMIC}(V). \\
\mathit{labelE}(U,V,\texttt{+}); \mathit{labelE}(U,V,\texttt{-}) &\leftarrow \mathit{edgeMIC}(U,V).
\end{aligned}$$

Since the integrity constraint in (B4) does not belong to P^X and the rules in (B5) are satisfied by Y in view of *bottom* $\notin Y$, it remains to consider the following rules from (9):

$$\begin{aligned}
\mathit{opposite}(U,V) &\leftarrow \mathit{labelE}(U,V,\texttt{-}), \mathit{labelV}(U,S), \mathit{labelV}(V,S). \\
\mathit{opposite}(U,V) &\leftarrow \mathit{labelE}(U,V,\texttt{+}), \mathit{labelV}(U,S), \mathit{labelV}(V,T), S \neq T. \\
\mathit{bottom} &\leftarrow \mathit{active}(V), \mathit{opposite}(U,V) : \mathit{edge}(U,V).
\end{aligned}$$

The rules defining predicate *opposite* are such that, in order to satisfy their ground instances in P^X, Y must contain $\mathit{opposite}(j,i)$ if $\mathit{labelE}(j,i,r)$, $\mathit{labelV}(j,s)$, and $\mathit{labelV}(i,t)$ belong to Y such that $t \neq sr$. This matches the definition of Y, including $\mathit{labelE}(j,i,r)$ if $\sigma'(j,i) = r$, $\mathit{labelV}(j,s)$ if $\mu'(j) = s$, $\mathit{labelV}(i,t)$ if $\mu'(i) = t$, and $\mathit{opposite}(j,i)$ if $\mu'(i) \neq \mu'(j)\sigma'(j,i)$. Hence, rules defining *opposite* in P^X are satisfied by Y. It remains to be shown that *bottom* is not derivable from any ground instance of the last rule. In this regard, recall that $W = \{i \mid \mathit{active}(i) \in X\} = \{i \mid \mathit{active}(i) \in Y\}$, and we have seen above that $\mathit{active}(i)$ can only belong to X if i is not an input. As σ' and μ' are witnessing labelings for W, for every $i \in W$, there is an edge $(j \rightarrow i) \in E$ such that $\mu'(i) = \mu'(j)\sigma'(j,i)$. By the definition of Y, this implies $\mathit{opposite}(j,i) \notin Y$, while $\mathit{edge}(j,i)$ belongs to X and Y because X and Y are models of $\tau((V,E,\sigma),\mu)$. As a consequence, for every $i \in W$, we have $\{\mathit{opposite}(j,i) \mid \mathit{edge}(j,i) \in Y\} \not\subseteq Y$, so that the ground instance for i in P^X of the rule with *bottom* in the head is satisfied by Y. We have thus established that $Y \subset X$ is indeed a model of P^X, a contradiction to the assumption that X is a $\subseteq$-minimal model of P^X and an answer set of $P_D \cup \tau((V,E,\sigma),\mu)$.

The above contradiction shows that the second condition to verify, which is the first condition in Definition 5.1, holds for $W = \{i \mid \mathit{active}(i) \in X\}$. The fact that the second condition in Definition 5.1 holds for W has been shown before. Hence, W is a MIC. □

Proof of Theorem 5.2
Assume that $W = \{k_1, \dots, k_n\}$ is a MIC. Then, the following conditions hold:

1. There are witnessing labelings $\sigma_1, \mu_1, \dots, \sigma_n, \mu_n$ for $W \setminus \{k_1\}, \dots, W \setminus \{k_n\}$.
2. There are no witnessing labelings for W.

We consider the following set X of atoms:

$$
\begin{array}{rll}
X = & \{\mathit{vertex}(i) & \mid i \in V\} \\
\cup & \{\mathit{edge}(j,i) & \mid (j \to i) \in E\} \\
\cup & \{\mathit{observedE}(j,i,s) & \mid (j \to i) \in E, \sigma(j,i) = s\} \\
\cup & \{\mathit{observedV}(i,s) & \mid i \in V, \mu(i) = s\} \\
\cup & \{\mathit{input}(i) & \mid i \in V \text{ is an input}\} \\
\cup & \{\mathit{active}(i) & \mid i \in W\} \\
\cup & \{\mathit{inactive}(i) & \mid i \in V \setminus W \text{ is not an input}\} \\
\cup & \{\mathit{edgeMIC}(j,i) & \mid (j \to i) \in E, i \in W\} \\
\cup & \{\mathit{vertexMIC}(j) & \mid (j \to i) \in E, i \in W\} \\
\cup & \{\mathit{vertexMIC}(i) & \mid i \in W\} \\
\cup & \{\mathit{labelE'}(k_m,j,i,r) & \mid (j \to i) \in E, i \in W, \sigma_m(j,i) = r, 1 \leq m \leq n\} \\
\cup & \{\mathit{labelE'}(k_m,j,i,r) & \mid (j \to i) \in E, \sigma(j,i) = r, 1 \leq m \leq n\} \\
\cup & \{\mathit{labelV'}(k_m,j,s) & \mid (j \to i) \in E, i \in W, \mu_m(j) = s, 1 \leq m \leq n\} \\
\cup & \{\mathit{labelV'}(k_m,i,s) & \mid i \in W, \mu_m(i) = s, 1 \leq m \leq n\} \\
\cup & \{\mathit{labelV'}(k_m,i,s) & \mid i \in V, \mu(i) = s, 1 \leq m \leq n\} \\
\cup & \{\mathit{receive'}(k_m,i,sr) & \mid (j \to i) \in E, i \in W, \\
 & & \quad \sigma_m(j,i) = r, \mu_m(j) = s, i \neq k_m, 1 \leq m \leq n\} \\
\cup & \{\mathit{receive'}(k_m,i,sr) & \mid (j \to i) \in E, j \in W \text{ or } (j \to k) \in E \text{ for } k \in W, \\
 & & \quad \sigma(j,i) = r, \mu_m(j) = s, i \neq k_m, 1 \leq m \leq n\} \\
\cup & \{\mathit{receive'}(k_m,i,sr) & \mid (j \to i) \in E, \\
 & & \quad \sigma(j,i) = r, \mu(j) = s, i \neq k_m, 1 \leq m \leq n\} \\
\cup & \{\mathit{labelV}(i,+), \mathit{labelV}(i,-) & \mid i \in V\} \\
\cup & \{\mathit{labelE}(j,i,+), \mathit{labelE}(j,i,-) & \mid (j \to i) \in E\} \\
\cup & \{\mathit{opposite}(j,i) & \mid (j \to i) \in E\} \\
\cup & \{\mathit{bottom}\}\ . &
\end{array}
$$

For showing that X is an answer set of $P_D \cup \tau((V,E,\sigma),\mu)$ (such that $\{i \mid \mathit{active}(i) \in X\} = W$), we need to verify that X is a $\subseteq$-minimal model of

$$
\begin{aligned}
P^X = \{ & (\mathit{head}(r) \leftarrow \mathit{body}(r)^+)\theta \mid \\
& \qquad r \in P_D \cup \tau((V,E,\sigma),\mu), (\mathit{body}(r)^-\theta) \cap X = \emptyset, \theta : \mathit{var}(r) \to \mathcal{U}\}
\end{aligned}
$$

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

To start with, we note that X includes an atom $\mathit{vertex}(i)$, $\mathit{edge}(j,i)$, $\mathit{observedE}(j,i,s)$, $\mathit{observedV}(i,s)$, and $\mathit{input}(i)$, respectively, exactly if there is a fact with the atom in the head in $\tau((V,E,\sigma),\mu)$. Each of these facts belongs also to P^X, is satisfied by X, but not by any set Y of atoms excluding at least one of the head atoms.

In view that W cannot contain any input (otherwise, satisfaction of the second condition in Definition 5.1 would immediately imply violation of the first one), we have that either *active*(i) or *inactive*(i) belongs to X for every non-input vertex $i \in V$. Hence, X satisfies all ground instances of the rule

$$\textit{active}(V); \textit{inactive}(V) \leftarrow \textit{vertex}(V), \textit{not input}(V).$$

from (8) belonging to P^X, while no set Y of atoms excluding both *active*(i) and *inactive*(i) for any non-input vertex $i \in V$ satisfies all of these ground instances.

Considering ground instances of the rules

$$\begin{aligned} \textit{edgeMIC}(U,V) &\leftarrow \textit{edge}(U,V), \textit{active}(V). \\ \textit{vertexMIC}(U) &\leftarrow \textit{edgeMIC}(U,V). \\ \textit{vertexMIC}(V) &\leftarrow \textit{active}(V). \end{aligned}$$

from (8), all of them belong to P^X, are satisfied by X, but not by any set Y of atoms such that $\{\textit{edgeMIC}(j,i) \mid \textit{edgeMIC}(j,i) \in X\} \cup \{\textit{vertexMIC}(i) \mid \textit{vertexMIC}(i) \in X\} \not\subseteq Y$ and $\{\textit{active}(i) \mid \textit{active}(i) \in X\} \subseteq \{\textit{active}(i) \mid \textit{active}(i) \in Y\}$, while it has been shown above that $\{\textit{active}(i) \mid \textit{active}(i) \in X\} \not\subseteq \{\textit{active}(i) \mid \textit{active}(i) \in Y\}$ necessitates $\{\textit{inactive}(i) \mid \textit{inactive}(i) \in Y\} \not\subseteq \{\textit{inactive}(i) \mid \textit{inactive}(i) \in X\}$ for Y being a model of P^X. Hence, there cannot be any model $Y \subset X$ of P^X excluding some atom *edgeMIC*(j,i) or *vertexMIC*(i) that belongs to X.

Now turning our attention to atoms of form *labelE'*(k_m, j, i, r) and *labelV'*(k_m, j, s), we note that they are included in X if *edgeMIC*$(j,i) \in X$ and *vertexMIC*$(j) \in X$, respectively, and $\sigma_m(j,i) = r, \mu_m(j) = s$ in witnessing labelings σ_m and μ_m for $W \setminus \{k_m\}$, where $1 \leq m \leq n$, or if $\sigma(j,i) = r, \mu(j) = s$. Then, the fact that *active*$(k_m) \in X$ and labels assigned by σ_m and μ_m are unique and respect those assigned by σ and μ implies that none of the atoms can be removed from X without violating some ground instance of the rules

$$\begin{aligned} \textit{labelV'}(W,V,+); \textit{labelV'}(W,V,-) &\leftarrow \textit{active}(W), \textit{vertexMIC}(V). \\ \textit{labelE'}(W,U,V,+); \textit{labelE'}(W,U,V,-) &\leftarrow \textit{active}(W), \textit{edgeMIC}(U,V). \\ \textit{labelV'}(W,V,S) &\leftarrow \textit{active}(W), \textit{observedV}(V,S). \\ \textit{labelE'}(W,U,V,S) &\leftarrow \textit{active}(W), \textit{observedE}(U,V,S). \end{aligned}$$

from (10) that belongs to P^X. However, X satisfies all of these ground instances by its construction. We further consider the following rules from (10):

$$\begin{aligned} \textit{receive'}(W,V,+) &\leftarrow \textit{labelE'}(W,U,V,S), \textit{labelV'}(W,U,S), V \neq W. \\ \textit{receive'}(W,V,-) &\leftarrow \textit{labelE'}(W,U,V,S), \textit{labelV'}(W,U,T), V \neq W, S \neq T. \end{aligned}$$

As shown above, *labelE'*(k_m, j, i, r) belongs to X if $i \in W$ and $\sigma_m(j,i) = r$, or if $\sigma(j,i) = \sigma_m(j,i) = r$. Furthermore, *labelV'*$(k_m, j, s)$ is included in X if $j \in W$ or $(j \rightarrow k) \in E, k \in W$ and $\mu_m(j) = s$, or if $\mu(j) = \mu_m(j) = s$. Comparing the cross product of these conditions to the definition of X yields that an atom *receive'*(k_m, i, sr) belongs to X exactly if *labelE'*(k_m, j, i, r) and *labelV'*(k_m, j, s) are in X and $i \neq k_m$. Hence, when excluding any of the atoms *receive'*(k_m, i, sr) from X, some ground instance of the above two rules belonging to P^X becomes unsatisfied, and so we have that such atoms cannot be removed from X in order to construct a model $Y \subset X$ of P^X. Moreover, the fact that σ_m and μ_m are witnessing labelings for $W' = W \setminus \{k_m\}$ implies that all ground instances of the integrity constraint

$$\leftarrow \textit{labelV'}(W,V,S), \textit{active}(V), V \neq W, \textit{not receive'}(W,V,S).$$

from (10) that belong to P^X are satisfied by X. In fact, for every $i \in W'$, there is some $(j \rightarrow i) \in E$ such that $\mu_m(i) = \mu_m(j)\sigma_m(j,i)$. Since $\textit{labelE'}(k_m, j, i, \sigma_m(j,i))$ and $\textit{labelV'}(k_m, j, \mu_m(j))$ belong to X, this implies that each atom $\textit{labelV'}(k_m, i, \mu_m(i))$ for $i \in W'$ is accompanied by $\textit{receive'}(k_m, i, \mu_m(i)) = \textit{receive'}(k_m, i, \mu_m(j)\sigma_m(j,i))$ in X, so that the ground instance for k_m, i, and $\mu_m(i)$ of the integrity constraint is not in P^X.

Finally, we consider atoms of the form $\textit{labelV}(i,s)$, $\textit{labelE}(j,i,s)$, and $\textit{opposite}(j,i)$ that belong to X for all $i \in V$ and $(j \rightarrow i) \in E$, respectively, and $s \in \{+,-\}$. Since *bottom* is also in X, it is clear that the ground instances of the following rules from (4), (8), and (9), all of which belong to P^X, are satisfied by X:

$$\begin{aligned}
\textit{labelV}(V,S) &\leftarrow \textit{observedV}(V,S).\\
\textit{labelE}(U,V,S) &\leftarrow \textit{observedE}(U,V,S).\\
\textit{labelV}(V,+); \textit{labelV}(V,-) &\leftarrow \textit{vertexMIC}(V).\\
\textit{labelE}(U,V,+); \textit{labelE}(U,V,-) &\leftarrow \textit{edgeMIC}(U,V).\\
\textit{opposite}(U,V) &\leftarrow \textit{labelE}(U,V,-), \textit{labelV}(U,S), \textit{labelV}(V,S).\\
\textit{opposite}(U,V) &\leftarrow \textit{labelE}(U,V,+), \textit{labelV}(U,S), \textit{labelV}(V,T), S \neq T.\\
\textit{bottom} &\leftarrow \textit{active}(V), \textit{opposite}(U,V) : \textit{edge}(U,V).\\
\textit{labelV}(V,+) &\leftarrow \textit{bottom}, \textit{vertex}(V).\\
\textit{labelV}(V,-) &\leftarrow \textit{bottom}, \textit{vertex}(V).\\
\textit{labelE}(U,V,+) &\leftarrow \textit{bottom}, \textit{edge}(U,V).\\
\textit{labelE}(U,V,-) &\leftarrow \textit{bottom}, \textit{edge}(U,V).
\end{aligned}$$

As shown above, any model $Y \subseteq X$ of P^X must necessarily include $\textit{observedV}(i,s)$ if $\mu(i) = s$, $\textit{observedE}(j,i,s)$ if $\sigma(j,i) = s$, $\textit{vertexMIC}(i)$ if $i \in W$ or $(i \rightarrow k) \in E$ for some $k \in W$, $\textit{edgeMIC}(j,i)$ if $(j \rightarrow i) \in E$ for some $i \in W$, and $\textit{active}(i)$ if $i \in W$. Proceeding by proof by contradiction, assume that there is a model $Y \subset X$ of P^X such that $\textit{labelV}(i,s)$, $\textit{labelE}(j,i,s)$, or $\textit{opposite}(j,i)$ is not in Y for some $i \in V$ or $(j \rightarrow i) \in E$, respectively, and $s \in \{+,-\}$. From the previous considerations and the first two rules repeated above, we know that $\textit{labelV}(i,s)$ and $\textit{labelE}(j,i,s)$ must belong to Y if $\mu(i) = s$ or $\sigma(j,i) = s$, respectively. Furthermore, the third rule necessitates $\{\textit{labelV}(i,+), \textit{labelV}(i,-)\} \cap Y \neq \emptyset$ for every $i \in W$ or $i \in V$ such that $(i \rightarrow k) \in E$ for some $k \in W$, and the fourth rule implies $\{\textit{labelE}(j,i,+), \textit{labelE}(j,i,-)\} \cap Y \neq \emptyset$ for every $(j \rightarrow i) \in E$ such that $i \in W$. In view of the last four rules, we immediately conclude that $\textit{bottom} \notin Y$, which in turn implies that, for every $i \in W$, there is some $(j \rightarrow i) \in E$ such that $\textit{opposite}(j,i)$ does not belong to Y. Comparing the rules defining *opposite*, the exclusion of $\textit{opposite}(j,i)$ is possible only if Y does not include $\textit{labelE}(j,i,r)$, $\textit{labelV}(j,s)$, and $\textit{labelV}(i,t)$ such that $t \neq sr$. As we have shown above that some atoms $\textit{labelE}(j,i,r)$, $\textit{labelV}(j,s)$, and $\textit{labelV}(i,t)$ for $r,s,t \in \{+,-\}$ must belong to Y, we can now conclude that $t = sr$ holds and that the atoms over predicates *labelE* and *labelV* in Y define (partial) labelings σ' and μ' by:

- For every $i \in W$, pick some edge $(j \rightarrow i) \in E$ such that $\textit{opposite}(j,i)$ does not belong to Y, and let $\sigma'(j,i) = r$ if *labelE*$(j,i,r) \in Y$, $\mu'(j) = s$ if *labelV*$(j,s) \in Y$, and $\mu'(i) = t$ if *labelV*$(i,t) \in Y$.

As we have seen above, such an edge $(j \rightarrow i) \in E$ exists for every $i \in W$, and the fact that $t \neq sr$ is not obtained for atoms *labelE*(j,i,r), *labelV*(j,s), and *labelV*(i,t) in Y implies that σ' and μ' assign unique labels to $(j \rightarrow i)$, j, and i, respectively. When we totalize σ' and μ' by setting $\sigma'(j,i) = \sigma(j,i)$ and $\mu'(i) = \mu(i)$ if $\sigma(j,i)$ or $\mu(i)$, respectively, is defined, and $\sigma'(j,i) = +$ as well as $\mu'(i) = +$ for all remaining edges in E and vertices in V, we obtain witnessing labelings for W. But this is a contradiction to the fact that W is a MIC, which allows us to conclude that there cannot be any model $Y \subset X$ of P^X that omits *labelV*(i,s), *labelE*(j,i,s), or *opposite*(j,i) for some $i \in V$ or $(j \rightarrow i) \in E$, respectively, and $s \in \{+,-\}$.

To conclude the proof that X is a $\subseteq$-minimal model of P^X, note that the integrity constraint

$$\leftarrow \mathit{not\ bottom}.$$

from (9) does not contribute any rule to P^X because *bottom* $\in X$. We have now investigated all rules in $P_D \cup \tau((V,E,\sigma),\mu)$ and shown that their ground instances in P^X are satisfied by X. Furthermore, we have checked for all atoms in X that they cannot be excluded in any model $Y \subset X$ of P^X. That is, X is indeed a $\subseteq$-minimal model of P^X and thus an answer set of $P_D \cup \tau((V,E,\sigma),\mu)$. □

Acknowledgments. Philippe Veber was supported by a grant from DAAD. This work was partially funded by the GoFORSYS project[9] (Grant 0313924). The authors are grateful to Carito Guziolowski for profitable discussions and for providing them with the data on yeast, and to Roland Kaminski for helpful comments on encodings.

References

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

BEN-ELIYAHU, R. AND DECHTER, R. 1994. Propositional semantics for disjunctive logic programs. *Annals of Mathematics and Artificial Intelligence 12,* 1-2, 53–87.

BioASP Tools. `http://www.cs.uni-potsdam.de/wv/bioasp`.

CHEN, M., HANCOCK, L., AND LOPES, J. 2007. Transcriptional regulation of yeast phospholipid biosynthetic genes. *Biochimica et Biophysica Acta 1771,* 3, 310–321.

DERSHOWITZ, N., HANNA, Z., AND NADEL, A. 2006. A scalable algorithm for minimal unsatisfiable core extraction. In *Proceedings of the Ninth International Conference on Theory and Applications of Satisfiability Testing (SAT'06)*, A. Biere and C. Gomes, Eds. Springer, 36–41.

DRESCHER, C., GEBSER, M., GROTE, T., KAUFMANN, B., KÖNIG, A., OSTROWSKI, M., AND SCHAUB, T. 2008. Conflict-driven disjunctive answer set solving. In *Proceedings of the Eleventh International Conference on Principles of Knowledge Representation and Reasoning (KR'08)*, G. Brewka and J. Lang, Eds. AAAI Press, 422–432.

EITER, T. AND GOTTLOB, G. 1995. On the computational cost of disjunctive logic programming: Propositional case. *Annals of Mathematics and Artificial Intelligence 15,* 3-4, 289–323.

[9] `http://www.goforsys.org`

ERDŐS, A. AND RÉNYI, P. 1959. On random graphs. *Publicationes Mathematicae 6*, 290–297.

FRIEDMAN, N., LINIAL, M., NACHMAN, I., AND PE'ER, D. 2000. Using Bayesian networks to analyze expression data. *Journal of Computational Biology 7,* 3-4, 601–620.

GEBSER, M., GUZIOLOWSKI, C., IVANCHEV, M., SCHAUB, T., SIEGEL, A., THIELE, S., AND VEBER, P. 2010. Repair and prediction (under inconsistency) in large biological networks with answer set programming. In *Proceedings of the Twelfth International Conference on Principles of Knowledge Representation and Reasoning (KR'10)*, to appear.

GEBSER, M., KAMINSKI, R., OSTROWSKI, M., SCHAUB, T., AND THIELE, S. 2009a. On the input language of ASP grounder *gringo*. In *Proceedings of the Tenth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'09)*, E. Erdem, F. Lin, and T. Schaub, Eds. Springer, 502–508.

GEBSER, M., KAUFMANN, B., NEUMANN, A., AND SCHAUB, T. 2007. Conflict-driven answer set enumeration. In *Proceedings of the Ninth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'07)*, C. Baral, G. Brewka, and J. Schlipf, Eds. Springer, 136–148.

GEBSER, M., KAUFMANN, B., AND SCHAUB, T. 2009b. Solution enumeration for projected Boolean search problems. In *Proceedings of the Sixth International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR'09)*, W. van Hoeve and J. Hooker, Eds. Springer, 71–86.

GEBSER, M., KAUFMANN, B., AND SCHAUB, T. 2009c. The conflict-driven answer set solver *clasp*: Progress report. In *Proceedings of the Tenth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'09)*, E. Erdem, F. Lin, and T. Schaub, Eds. Springer, 509–514.

GELFOND, M. 2008. Answer sets. In *Handbook of Knowledge Representation*, V. Lifschitz, F. van Hermelen, and B. Porter, Eds. Elsevier, 285–316.

GELFOND, M., LIFSCHITZ, V., PRZYMUSINSKA, H., AND TRUSZCZYŃSKI, M. 1991. Disjunctive defaults. In *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning (KR'91)*, J. Allen, R. Fikes, and E. Sandewall, Eds. Morgan Kaufmann Publishers, 230–237.

GIUNCHIGLIA, E., LIERLER, Y., AND MARATEA, M. 2006. Answer set programming based on propositional satisfiability. *Journal of Automated Reasoning 36,* 4, 345–377.

GUELZIM, N., BOTTANI, S., BOURGINE, P., AND KÉPÈS, F. 2002. Topological and causal structure of the yeast transcriptional regulatory network. *Nature Genetics 31*, 60–63.

GUTIERREZ-RIOS, R., ROSENBLUETH, D., LOZA, J., HUERTA, A., GLASNER, J., BLATTNER, F., AND COLLADO-VIDES, J. 2003. Regulatory network of Escherichia coli: Consistency between literature knowledge and microarray profiles. *Genome Research 13,* 11, 2435–2443.

GUZIOLOWSKI, C., BOURDE, A., MOREEWS, F., AND SIEGEL, A. 2009. Bioquali cytoscape plugin: analysing the global consistency of regulatory networks. *BMC Genomics 10.*

GUZIOLOWSKI, C., VEBER, P., LE BORGNE, M., RADULESCU, O., AND SIEGEL, A. 2007. Checking consistency between expression data and large scale regulatory networks: A case study. *Journal of Biological Physics and Chemistry 7,* 2, 37–43.

JANHUNEN, T., NIEMELÄ, I., SEIPEL, D., SIMONS, P., AND YOU, J. 2006. Unfolding partiality and disjunctions in stable model semantics. *ACM Transactions on Computational Logic 7,* 1, 1–37.

JEONG, H., TOMBOR, B., ALBERT, R., OLTVAI, Z., AND BARABÁSI, A. 2000. The large-scale organization of metabolic networks. *Nature 407*, 651–654.

KLAMT, S. AND STELLING, J. 2006. Stoichiometric and constraint-based modelling. In *System Modeling in Cellular Biology: From Concepts to Nuts and Bolts*. MIT Press, 73–96.

KUIPERS, B. 1994. *Qualitative reasoning. Modeling and simulation with incomplete knowledge*. MIT Press.

LEONE, N., PFEIFER, G., FABER, W., EITER, T., GOTTLOB, G., PERRI, S., AND SCARCELLO, F. 2006. The DLV system for knowledge representation and reasoning. *ACM Transactions on Computational Logic 7,* 3, 499–562.

MALLORY, M., COOPER, K., AND STRICH, R. 2007. Meiosis-specific destruction of the Ume6p repressor by the Cdc20-directed APC/C. *Molecular Cell 27,* 6, 951–961.

PAPADIMITRIOU, C. AND YANNAKAKIS, M. 1982. The complexity of facets (and some facets of complexity). In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing (STOC'82).* ACM Press, 255–260.

REMY, É., RUET, P., AND THIEFFRY, D. 2008. Graphic requirements for multistability and attractive cycles in a Boolean dynamical framework. *Advances in Applied Mathematics 41,* 3, 335–350.

RICHARD, A., COMET, J., AND BERNOT, G. 2004. R. Thomas' modeling of biological regulatory networks: Introduction of singular states in the qualitative dynamics. *Fundamenta Informaticae 65,* 4, 373–392.

SIEGEL, A., RADULESCU, O., LE BORGNE, M., VEBER, P., OUY, J., AND LAGARRIGUE, S. 2006. Qualitative analysis of the relation between DNA microarray data and behavioral models of regulation networks. *Biosystems 84,* 2, 153–174.

SOULÉ, C. 2003. Graphic requirements for multistationarity. *Complexus 1,* 3, 123–133.

SOULÉ, C. 2006. Mathematical approaches to differentiation and gene regulation. *Comptes Rendus Biologies 329*, 13–20.

SUDARSANAM, P., IYER, V., BROWN, P., AND WINSTON, F. 2000. Whole-genome expression analysis of snf/swi mutants of Saccharomyces cerevisiae. *Proceedings of the National Academy of Sciences of the United States of America 97,* 7, 3364–3369.

Lparse Manual. SYRJÄNEN, T. *Lparse* 1.0 user's manual. `http://www.tcs.hut.fi/Software/smodels/lparse.ps.gz`.

VEBER, P., LE BORGNE, M., SIEGEL, A., LAGARRIGUE, S., AND RADULESCU, O. 2004. Complex qualitative models in biology: A new approach. *Complexus 2,* 3-4, 140–151.

WASHBURN, B. AND ESPOSITO, R. 2006. Identification of the Sin3-binding site in Ume6 defines a two-step process for conversion of Ume6 from a transcriptional repressor to an activator in yeast. *Molecular and Cellular Biology 21,* 6, 2057–2069.