Markus Durzinsky | Wolfgang Marwan | Max Ostrowski
Torsten Schaub | Annegret Wagler

Automatic network reconstruction using ASP

Suggested citation referring to the original publication:
Theory and Practice of Logic Programming 11 (2011) 4–5, pp. 749–766
DOI https://doi.org/10.1017/S1471068411000287
ISSN (print) 1471-0684
ISSN (online) 1475-3081

TLP **11** (4–5): 749–766, 2011. © Cambridge University Press 2011
doi:10.1017/S1471068411000287

Automatic network reconstruction using ASP

MARKUS DURZINSKY and WOLFGANG MARWAN
Magdeburg Centre for Systems Biology, Universität Magdeburg, Germany

MAX OSTROWSKI and TORSTEN SCHAUB*
Universität Potsdam, Germany

ANNEGRET WAGLER
Université Blaise Pascal, Clermont-Ferrand, France

Abstract

Building biological models by inferring functional dependencies from experimental data is an important issue in Molecular Biology. To relieve the biologist from this traditionally manual process, various approaches have been proposed to increase the degree of automation. However, available approaches often yield a single model only, rely on specific assumptions, and/or use dedicated, heuristic algorithms that are intolerant to changing circumstances or requirements in the view of the rapid progress made in Biotechnology. Our aim is to provide a declarative solution to the problem by appeal to Answer Set Programming (ASP) overcoming these difficulties. We build upon an existing approach to Automatic Network Reconstruction proposed by part of the authors. This approach has firm mathematical foundations and is well suited for ASP due to its combinatorial flavor providing a characterization of all models explaining a set of experiments. The usage of ASP has several benefits over the existing heuristic algorithms. First, it is declarative and thus transparent for biological experts. Second, it is elaboration tolerant and thus allows for an easy exploration and incorporation of biological constraints. Third, it allows for exploring the entire space of possible models. Finally, our approach offers an excellent performance, matching existing, special-purpose systems.

1 Introduction

The creation of biological models by inferring functional dependencies from experimental data is a key issue in molecular biology. A common approach is to construct descriptive models from series of experiments. This (manual) process usually starts from a model defined using existing biological knowledge which is then gradually refined by appeal to data gathered in subsequent experiments. A model obtained this way is however merely consistent with the gathered experimental data, and, besides simulation, no true indication can be given how well the resulting model captures the biological system. For instance, it is unclear whether the obtained model is one among many or few alternative models. Moreover, it is of great interest to

* Affiliated with Simon Fraser University, Canada, and Griffith University, Australia.

know the difference among alternative models to design new experiments for further discriminating the best-fitting model.

This problem is addressed in the area of *Automatic Network Reconstruction* (ANR) (Liang *et al.* 1998; Gifford and Jaakkola 2001; Yeung *et al.* 2002; Repsilber *et al.* 2002). However, the available approaches often yield a single model only, rely on specific assumptions, and/or use dedicated, heuristic algorithms for constructing a model from experimental data. Moreover, all these approaches are intolerant to changing circumstances or requirements in the view of the rapid progress made in Biotechnology. Unlike this, we provide a declarative solution to the problem by appeal to Answer Set Programming (ASP; Baral 2003). To this end, we build upon the approach to ANR proposed in Durzinsky *et al.* (2010) and Marwan *et al.* (2008). This approach has firm mathematical foundations and is well suited for ASP due to its combinatorial flavor providing a characterization of all models explaining a set of experiments. The usage of ASP has several benefits over the existing heuristic algorithms. First, it is declarative and thus transparent for biological experts. Second, it is elaboration tolerant and thus allows for an easy exploration and incorporation of biological constraints. Third, it allows for exploring the entire space of possible models. Finally, our approach offers an excellent performance, matching existing, special-purpose systems.

The next section gives a formal introduction to ANR, as provided in Durzinsky *et al.* (2010) and Marwan *et al.* (2008), followed by a brief introduction to ASP in Section 3. Section 4 is dedicated to our solution to ANR in ASP. We empirically evaluate our approach in Section 5 and conclude with a discussion and a summary in Sections 6 and 7.

2 Automatic network reconstruction

Automatic Network Reconstruction aims at constructing all models explaining a set of (perturbation) experiments reflecting a certain biological process. Our approach starts from experimental time-series data and generates all interaction networks that account for the observed mass or signal flow. We briefly describe the steps of this approach proposed in (Durzinsky *et al.* 2008; Marwan *et al.* 2008; Durzinsky *et al.* 2010).

We represent a collection S of n observable species as a vector $(s_1, \ldots, s_n)$ being considered to be crucial for describing the studied biological phenomenon, along with a corresponding vector $(D_1, \ldots, D_n)$ of associated capacities over N_0. Accordingly, species s_i is assigned a value from capacity D_i for $1 \leqslant i \leqslant n$. A *state* x of species $(s_1, \ldots, s_n)$ is a vector $(x_1, \ldots, x_n)$ such that $x_i \in D_i$ for $1 \leqslant i \leqslant n$. Thus, x_i provides the value of species s_i in state x for $1 \leqslant i \leqslant n$. Note that our concept of a state is only partial because it is confined to the observable species in S. In what follows, we leave the set S of species implicit whenever clear from the context.

A (perturbation) *experiment* $\mathscr{E}(x^0) = (x^0; x^1, \ldots, x^k)$ over S is a sequence of states reflecting the time-dependent response $(x^1, \ldots, x^k)$ of a biological system to a (specific) perturbation of the system in state x^0. We associate with each response state $x^i \in \mathscr{E}(x^0)$ its *terminal state* $x^k \in \mathscr{E}(x^0)$ and define $t(x^i) = x^k$ for all $1 \leqslant i < k$.

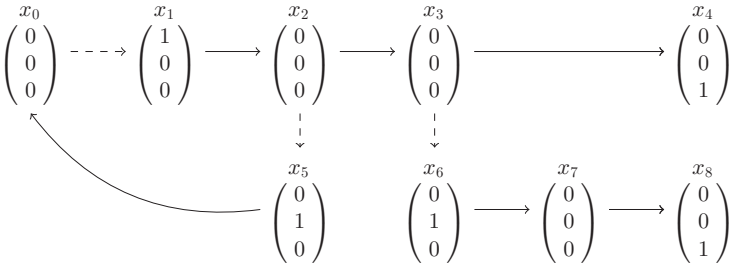

Fig. 1. An Experiment Graph $G(\mathscr{E})$.

Typically, several experiments $\mathscr{E}(x^0)$ starting from different initial states x^0 are necessary to describe a biological phenomenon. We encode a set $\mathscr{E}$ of different experiments in terms of an *experiment graph* $G(\mathscr{E}) = (X, E_P \cup E_R)$ over S, which is a directed graph such that X is the multi-set of states in $\mathscr{E}$, and E_P and E_R are disjoint sets of perturbation and response edges, respectively. That is, for each $\mathscr{E}(x^0) = (x^0; x^1, \ldots, x^k) \in \mathscr{E}$, we have $(x^0, x^1) \in E_P$ and $(x^i, x^{i+1}) \in E_R$ for $1 \leqslant i < k$. For illustration, consider Figure 1 showing an experiment graph over species $\{fr, r, spo\}$, encoding three experiments $\mathscr{E}(x^0) = (x^0; x^1, \ldots, x^4)$, $\mathscr{E}(x^2) = (x^2; x^5, x^0)$, and $\mathscr{E}(x^3) = (x^3; x^6, \ldots, x^8)$. The entries in each state vector give the respective values of each species; continuous arrows represent response edges, dashed ones give perturbation edges.

An experiment graph $G(\mathscr{E}) = (X, E_P \cup E_R)$ is *valid* if

I. every state $x \in X$ has at most one outgoing arc in E_R,
II. $x = x'$ implies $t(x) = t(x')$ for all $x, x' \in X$, and
III. $(x' - x) \notin \mathbb{N}^n$ holds for all $(x, x') \in E_R$,

Condition I stipulates that an experiment graph is deterministic, while II requires that no equal[1] states lead to different terminal states. III demands that there must be at least one species that decreases between two consecutive response states. In fact, the experiment graph in Figure 1 violates two validity conditions: II is violated through states x^5 and x^6, as these states are equal but lead to differing terminal states x^0 and x^8, respectively. III is violated by response edge (x^2, x^3).

For the reconstruction, we use the paradigm that system states can be changed by applying reactions. A *reaction* over n species is described by a vector $r \in \mathbb{Z}^n$, where $r_i < 0$ for some $1 \leqslant i \leqslant n$. So, a reaction must have at least one negative entry to consume at least one species. A reaction r is *enabled* in a state x over n species with capacities $(D_1, \ldots, D_n)$, if we have $x_i + r_i \in D_i$ for all $1 \leqslant i \leqslant n$, i.e. if neither nonnegativity nor capacity constraints are violated. For instance, reaction $r = (0, -1, 0)$ is enabled in x^6 because $x^6 + r = (0, 0, 0)$ belongs to the species' capacity.

Given an experiment graph $(X, E_P \cup E_R)$ and a response edge $(x, x') \in E_R$, we say that this response is *realized* by a sequence $\sigma((x, x')) = (r^1, \ldots, r^l)$ of reactions, if

[1] Recall that X is a multiset; two states are equal if their vector of species is equal.

IV. $y^i + r^i = y^{i+1}$ for all $1 \leqslant i \leqslant l$, and
V. $(y^1, y^2, \ldots, y^{l+1})$ is a sequence of states such that $x = y^1$ and $x' = y^{l+1}$,
VI. $r^i_k \cdot r^j_k \geqslant 0$ for all $1 \leqslant i, j \leqslant l$ and all $0 \leqslant k \leqslant n$.

All reactions subsequently applied to state x fulfill the response edge and ultimately lead to the consecutively observed state x' in E_R. For example, the reaction $r = (0, -1, 0)$ constitutes a singleton sequence $\sigma((x^6, x^7))$ as $x^6 + r = x^7$ realizes (x^6, x^7). Note that VI stipulates that all reactions in such a sequence must be *monotone*;[2] at microscopic level, a species cannot be produced and consumed (or vice versa) by two reactions, see Durzinsky *et al.* (2008) for details.

To also account for the experimentally observed mass or signal flow, Marwan *et al.* (2008) propose to use a partial order on the set of reactions to reflect their relative rates. A sequence $(r^1, \ldots, r^l)$ of reactions is said to respect such a partial order $\prec$, if r^i is the unique $\prec$-minimal reaction enabled in an (intermediate) state y^i for each $1 \leqslant i < l$. Note that the reaction order $\prec$ must be sufficiently strong to guarantee a unique fastest reaction at each step. This implies for each state to have a unique successor state, ensuring the system's determinism.

Following Marwan *et al.* (2008), a *regulatory structure* $(\mathscr{R}, \prec)$ over species S consists of a set of reactions $\mathscr{R}$ and a partial order $\prec$ among them.[3] A regulatory structure $(\mathscr{R}, \prec)$ is *conformal* with a valid experiment graph $(X, E_P \cup E_R)$, if

VII. for all $r \in \mathscr{R}$, r is not enabled in any terminal state of X,
VIII. for all $e \in E_R$, there is a $\prec$-respecting realizing sequence[4] $\sigma(e) \subseteq \mathscr{R}$, and
IX. there exists no $r \in \mathscr{R}$ where r is not an element of some $\sigma(e)$.

As defined in Durzinsky *et al.* (2010), the *Network Reconstruction Problem* for a valid experiment graph consists in finding all regulatory structures conformal with the graph.

An invalid experiment graph can be recovered by adding new, artificial species to S.[5] Given an (invalid) experiment graph $(X, E_P \cup E_R)$, an extension $(X', E_P \cup E_R)$ with a species is obtained by replacing each state $(x_1, \ldots, x_n)$ in X with $(x_1, \ldots, x_n, x_{n+1}, \ldots, x_{n+a})$ such that $x_{n+i} \in \{0, 1\}$ for $1 \leqslant i \leqslant a$; all other capacities and edges are left intact. Note that an experiment graph has 2^a extensions.

An extension $(X', E_P \cup E_R)$ of an experiment graph with a species is *valid*, if

X. $(X', E_P \cup E_R)$ is a valid experiment graph and
XI. $x_{n+i} = x'_{n+i}$ for each $(x, x') \in E_P$ and $1 \leqslant i \leqslant a$.

The latter condition stipulates that additional species are not direct targets of experimental perturbations, but they certainly respond in successive states. Similarly,

[2] This is a significant constraint on the quality of time series data. The response of the system must be measured with sufficient time resolution, such that oscillation between measurements can be excluded.
[3] $\mathscr{R}$ is also referred to as a *network* because such reaction sets are easily converted to *Petri nets*, as done in Marwan *et al.* (2008). This is however beyond the scope of this paper.
[4] We slightly abuse notation, and take $\sigma(e) \subseteq \mathscr{R}$ to mean that each element of $\sigma(e)$ is also in $\mathscr{R}$.
[5] This allows for differentiating seemingly equal yet different states, enabling new reactions by decreasing additional species, or avoiding reactions in terminal states.

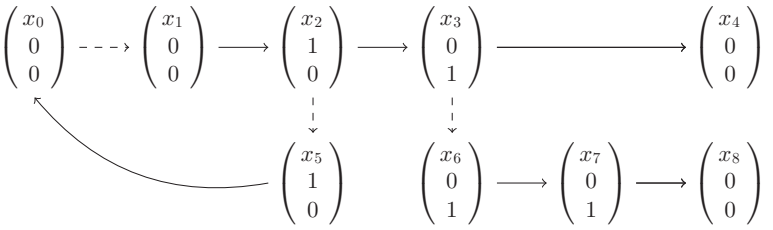

Fig. 2. First Extension $G(\mathscr{E}_1)$ of the Experiment Graph $G(\mathscr{E})$ in Figure 1.

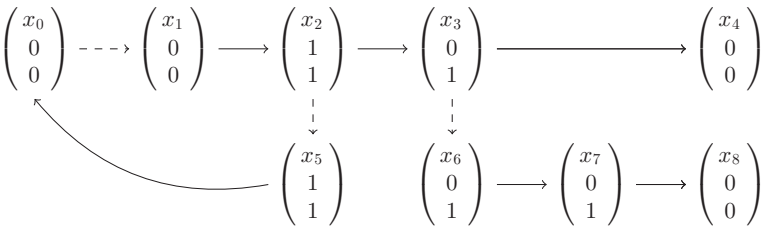

Fig. 3. Second Extension $G(\mathscr{E}_2)$ of the Experiment Graph $G(\mathscr{E})$ in Figure 1.

we want to reduce the changes of additional species in response edges: a response edge $(x, x') \in E_R$ is subject to an *additional change*, if $x_{n+j} \neq x'_{n+j}$ for some $1 \leqslant j \leqslant a$.

At last, given an invalid experiment graph, the *Network Reconstruction Problem* consists in solving the NRP for all valid extensions of that graph, first, adding a minimum number of additional species and, second, comprising a minimum number of additional changes. For brevity, such extensions are called *minimal valid extensions.*

Figures 2 and 3 show the two[6] valid extensions of the invalid experiment graph in Figure 1. The nodes in the figures are the vectors from Figure 1 extended by the two additional species. Both extensions differ in the values attributed to the two additional species in state x_2 and x_5. The extensions are conformal with the regulatory structures with the networks depicted in Figures 4 and 5, respectively. The additional species are referred to as x and y. Reactions are given as boxes, species as circles. All reaction entries have the capacity $\{-1, 0, 1\}$. An arrow from a species to a reaction stands for a -1 in the reaction vector, one from a reaction to a species stands for $+1$.

Accordingly, the regulatory structure in Figure 4 over species (fr, r, spo, x, y) comprises the following reactions: $r^1 = (-1, 0, 0, 1, 0)$, $r^2 = (0, -1, 0, 0, 0)$, $r^3 = (0, -1, 0, -1, 0)$, $r^4 = (0, 0, 0, -1, 1)$, and $r^5 = (0, 0, 1, 0, -1)$ and the ordering $\prec= \{(r^4, r^3), (r^3, r^2), (r^5, r^2)\}$. The regulatory structure in Figure 5 comprises the reactions: $r^1 = (-1, 0, 0, 1, 1)$, $r^2 = (0, -1, 0, -1, -1)$, $r^3 = (0, -1, 0, 0, 0)$, $r^4 = (0, 0, 1, -1, 0)$, and $r^5 = (0, 0, 0, 0, -1)$ and the ordering $\prec= \{(r^4, r^5), (r^5, r^2), (r^4, r^3), (r^3, r^2)\}$.

[6] Actually, there are four extensions with symmetric behavior on the additional species.

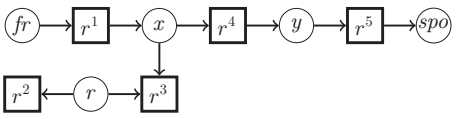

Fig. 4. Regulatory Structure conformal with the extended Experiment Graph in Figure 2.

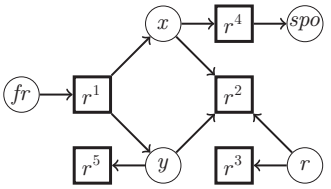

Fig. 5. Regulatory Structure conformal with the extended Experiment Graph in Figure 3.

3 Answer set programming

We rely on the input language of the ASP grounder *gringo* (Gebser *et al.*) (extending the language of *lparse* (Syrjänen)) and introduce only informally the basics of ASP. A comprehensive, formal introduction to ASP can be found in Baral (2003) and Gelfond (2008).

We consider extended logic programs as introduced in Simons *et al.* (2002). A *rule* r is of the following form:

$$H \leftarrow B_1, \ldots, B_m, \sim B_{m+1}, \ldots, \sim B_n.$$

By $head(r) = H$ and $body(r) = \{B_1, \ldots, B_m, \sim B_{m+1}, \ldots, \sim B_n\}$, we denote the *head* and the *body* of r, respectively, where "$\sim$" stands for default negation. The head H is an atom a belonging to some alphabet $\mathscr{A}$, the falsum $\bot$, or a `#sum` constraint $L \,\#\mathtt{sum}[\ell_1 = w_1, \ldots, \ell_k = w_k]\, U$. In the latter, $\ell_i = a_i$ or $\ell_i = \sim a_i$ is a *literal* and w_i a *non-negative* integer *weight* for $a_i \in \mathscr{A}$ and $1 \leqslant i \leqslant k$; L and U are integers providing a lower and an upper bound. Either or both of L and U can be omitted, in which case they are identified with the (trivial) bounds 0 and ∞, respectively. Whenever all weights equal one, the `#sum` constraint $L \,\#\mathtt{sum}\{\ell_1 = 1, \ldots, \ell_k = 1\}\, U$ becomes a "`#count`" constraint and is simply written as $L\{\ell_1, \ldots, \ell_k\}\, U$. A rule r such that $head(r) = \bot$ is an *integrity constraint*. Each body component B_i is either an atom or a `#sum` constraint for $1 \leqslant i \leqslant n$. If $body(r) = \emptyset$, r is called a *fact*, and we skip "$\leftarrow$" when writing facts below. We adhere to the definition of answer sets provided in Simons *et al.* (2002), which applies to logic programs containing extended constructs (`#sum` constraints) under "choice semantics".

In addition to rules, a logic program can contain *#minimize* statements of the form

$$\#minimize\ \{\ \ell_1 = w_1, \ldots, \ell_k = w_k\ \}.$$

Besides literals ℓ_j, a *#minimize* statement includes integer weights w_j for $1 \leqslant j \leqslant k$. A *#minimize* statement distinguishes optimal answer sets of a program as the ones yielding the smallest weighted sum for the true literals among $\ell_1, \ldots, \ell_k$. For a formal introduction, we refer the interested reader to Simons *et al.* (2002).

Similarly, first-order representations, commonly used to encode problems in ASP, are only informally introduced. In fact, *gringo* requires programs to be *safe*, that is, each variable must occur in a positive body literal. Formally, we only rely on the function *ground* to denote the set of all ground instances, $ground(\Pi)$, of a program Π containing first-order variables. Further language constructs of interest, include conditional literals, like "$a:b$", the range and pooling operator ".." and ";" as well as standard arithmetic operations. The ":" connective expands to the list of all instances of its left-hand side such that corresponding instances of literals on the right-hand side hold (Syrjänen ; Gebser *et al.*). Although ".." allows for specifying integer intervals, ";" allows for pooling alternative terms to be used as arguments within an atom. For instance, $p(1..3)$ as well as $p(1;2;3)$ stand for the three facts $p(1)$, $p(2)$, and $p(3)$. Given this, $q(X):p(X)$ results in $q(1), q(2),$ and $q(3)$. See Gebser *et al.* () for a detailed description of the input language of the grounder *gringo*.

4 Declarative automatic network reconstruction

Our approach expresses the ANR Problem in form of a logic program under answer set semantics. In the next sections, we explain the encoding in detail, starting with the representation of the experiment graph.

4.1 Representing experiment instances

As common in ASP, a problem instance is given as facts. We define the instance of an experiment graph $(X, E_P \cup E_R)$ over S with domain D, as a set of facts

$$\begin{aligned}
\mathcal{I}(X, E_P \cup E_R) \;=\; & \{species(s) \mid s \in S\} \\
\cup\; & \{capacity(s_i, d_i) \mid s_i \in S \text{ and } d_i \in D\} \\
\cup\; & \{state(x) \mid x \in X\} \\
\cup\; & \{edge(p, x, x') \mid (x, x') \in E_P\} \\
\cup\; & \{edge(r, x, x') \mid (x, x') \in E_R\} \\
\cup\; & \{terminalState(x') \mid x' = t(x), x \in X\} \\
\cup\; & \{value(x, s_i, x_i) \mid x = (x_1, \ldots, x_n) \in X, 1 \leqslant i \leqslant n\}.
\end{aligned}$$

Predicates $species(s_i)$ and $capacity(s_i, d_i)$ denote the species $s_i \in S$ over their associated capacities $d_i \in D$. Similarly, we use $state(x)$ to denote each state $x \in X$ and mark terminal states with predicate $terminalState(x)$. For the edges of the graph, we use predicate $edge(T, x, x')$, where T can be either p or r to indicate that (x, x') is a perturbation or a response edge, respectively.

As an example, Table 1 gives the specification of the experiment graph in Figure 1. Given the instance in Table 1, the solutions of our final logic program represent all regulatory structures that are conformal with the extensions of the experiment graph. We start with checking the validity of the experiment graph.

4.2 Checking validity

In Section 2, three conditions were specified for validity. Checking these conditions can be done with the following logic program.

Condition **I**, ensuring that each state has only one outgoing arc, is given in (1).

$$\leftarrow edge(r, X_1, X_2), edge(r, X_1, X_3), X_2 \neq X_3. \tag{1}$$

Rules (2)–(5) account for Condition II. We first collect all pairs of states that are not equal and then compute the associated terminal state of each state which can be determined deterministically. Rule (5) ensures that no two equal states lead to unequal terminal states.

$$neq(X_1, X_2) \leftarrow value(X_1, S, V), \sim value(X_2, S, V), state(X_2). \tag{2}$$

$$t(X, T) \leftarrow edge(r, X, T), terminalState(T). \tag{3}$$

$$t(X_1, T) \leftarrow edge(r, X_1, X_2), t(X_2, T). \tag{4}$$

$$\leftarrow \sim neq(X_1, X_2), neq(T_1, T_2), t(X_1, T_1), t(X_2, T_2). \tag{5}$$

The rules in (6) and (7) ensure a decrease in each response as required in Condition III.

$$\begin{aligned} decrease(X_1) \leftarrow\ & edge(r, X_1, X_2), value(X_1, S, V_1), value(X_2, S, V_2), \\ & V_2 - V_1 < 0. \end{aligned} \tag{6}$$

$$\leftarrow \sim decrease(X_1), edge(r, X_1, X_2). \tag{7}$$

The next proposition ensures correctness and completeness of this logic program.

Proposition 1
Let $(X, E_P \cup E_R)$ be the experiment graph and Π_1 be the logic program $ground(\mathscr{I}(X, E_P \cup E_R) \cup \{(1), \ldots, (5)\})$.

Then, the experiment graph $(X, E_P \cup E_R)$ is valid iff there exists an answer set of Π_1.

The proof of this and all following results follow from the construction of the respective logic programs. Recall that the experiment graph in Figure 1 is invalid, so the corresponding program has no answer set.

4.3 Building regulatory structures

We now proceed by defining a finite logic program that allows us to find all regulatory structures conformal with a valid experiment graph.

For guaranteeing the finiteness of the ground program, we need to know the maximum number of reactions that shall be used. As each reaction has to consume at least one species, the number of reactions is bound by the total number of decreases during a response edge. Although there exist better approximations, for simplicity, we define the logic program $\mathscr{B}(X, E_P \cup E_R)$ that results in the single answer set A, such that $maxAdds(n) \in A$, $maxReactions(m) \in A$ and $length(x, x', m) \in A$ for all $(x, x') \in E_R$, where $n, m \in \mathbb{N}_0$ are sufficient bounds. Given these bounds, we now choose a certain number of reactions in (9) to be part of the regulatory structure.

$$r(1). \tag{8}$$

$$\{\ r(N{+}1)\ \} \leftarrow r(N), maxReactions(M), N{<}M. \tag{9}$$

For the resulting reaction vector, we pick out its values in (10). So, for each reaction and each species $s_i \in S$, we choose exactly one value from the set $\{d, -d \mid d \in D_i\}$. This is constrained by (11) ensuring that each reaction has at least one negative entry.

$$1 \{ r(R,S,V) : capacity(S,V), \\ r(R,S,-V) : capacity(S,V) \} 1 \leftarrow r(R), species(S). \tag{10}$$

$$\leftarrow \{ r(R,S,V) : V < 0 \} 0, r(R). \tag{11}$$

Next, we want to define the realizing sequences guessing a partial order $\prec$ of reactions. Therefore, we define the intermediate states of a sequence. Each consecutive intermediate state is built, adding the currently fastest enabled reaction. As with the reactions, we first guess the length of the sequences $\sigma(x, x')$ in (12) and (13), which is bound by the precomputed predicate $length(x, x', .)$.

$$step(X_1, 0) \leftarrow edge(r, X_1, X_2). \tag{12}$$

$$\{ step(X_1, N+1) \} \leftarrow step(X_1, N), length(X_1, X_2, M), N < M. \tag{13}$$

Then, the value of the species is defined for each intermediate step by Rules (14) and (15) by adding the fastest reaction, as stated in Condition IV.

$$interValue(0, X, S, V) \leftarrow value(X, S, V). \tag{14}$$

$$interValue(L+1, X, S, V_{Old} + V_{New}) \leftarrow interValue(L, X, S, V_{Old}), \\ fastest(R, L, X), r(R, S, V_{New}), \\ capacity(S, V_{Old} + V_{New}), \\ step(X, L). \tag{15}$$

Rules (16) and (17) find out which reaction is enabled in which intermediate state.

$$disabled(R, L, X) \leftarrow interValue(L, X, S, V_1), r(R, S, V_2), species(S), \\ \sim capacity(S, V_1 + V_2), step(X, L). \tag{16}$$

$$enabled(R, L, X) \leftarrow \sim disabled(R, L, X), r(R), step(X, L). \tag{17}$$

From the enabled reactions, we freely choose a fastest reaction via Rule (18).

$$\{ fastest(R, L, X) \} \leftarrow enabled(R, L, X). \tag{18}$$

To impose an ordering on the reactions, Rule (19) says that if there is another enabled reaction different from the fastest one, then this one must be slower.

$$slower(R_2, R_1) \leftarrow fastest(R_1, L, X_1), enabled(R_2, L, X_1), R_1 \neq R_2. \tag{19}$$

We just need to add transitivity to the predicate *slower*, and forbid that a reaction is slower than itself to enforce the partial order:

$$slower(R_1, R_3) \leftarrow slower(R_1, R_2), slower(R_2, R_3). \tag{20}$$

$$\leftarrow slower(R, R), r(R). \tag{21}$$

To create a valid realizing sequence, the constraint in (22) assures that the sequence of reactions leads to the next measured state of the system, as stipulated in **V**.

$$\leftarrow interValue(M+1, X_1, S, V), edge(r, X_1, X_2), \sim value(X_2, S, V), \\ step(X, M), \sim step(X, M+1), species(S). \tag{22}$$

Similarly, (23) and (24) enforce that reactions are monotone, as dictated by Condition VI. This is expressed by saying that a reaction applying in a state x must increase/decrease the species into the direction of the next state x', where $(x, x') \in E_R$.

$$\leftarrow fastest(R, L, X_1), r(R, S, V_3), edge(r, X_1, X_2), \\ value(X_1, S, V_1), value(X_2, S, V_2), (V_2 - V_1) * V_3 < 0. \tag{23}$$

$$\leftarrow fastest(R, L, X_1), \sim r(R, S, 0), edge(r, X_1, X_2), \\ value(X_1, S, V), value(X_2, S, V). \tag{24}$$

Condition VII states that for creating a *regulatory structure*, no reaction my be enabled in a terminal state. This is addressed in (25) and (26):

$$disabled(R, X) \leftarrow terminalState(X), r(R, S, V_1), \\ value(X, S, V_2), \sim capacity(S, V_1 + V_2). \tag{25}$$

$$\leftarrow 1 \{ \sim disabled(R, X) : terminalState(X) \}, r(R). \tag{26}$$

To avoid irrelevant solutions, every reaction must be used. This is addressed in conditions VIII and IX. That is, each reaction has to be at least one time the fastest reaction (see (27)) and in each step there must exist at least one fastest reaction (see (28)).

$$\leftarrow r(R), \sim 1 \{ fastest(R, L, X) \}. \tag{27}$$

$$\leftarrow step(X, L), \sim 1 \{ fastest(R, L, X) \}. \tag{28}$$

For formulating our correctness and completeness result, we need the following auxiliary definition.

Definition 1
Let $(s_1, \ldots, s_n)$ be the vector of the species in S. Let A be a set of ground atoms such that $r(x) \in A$ and $r(x, s_i, v_i) \in A$ for $1 \leqslant i \leqslant n$. Then, define $\gamma_A(x) = (v_1, \ldots, v_n)$.

We then get the following result.

Proposition 2
Let $(X, E_P \cup E_R)$ be the experiment graph and Π_2 be the logic program $ground(\mathscr{I}(X, E_P \cup E_R) \cup \mathscr{B}(X, E_P \cup E_R) \cup \{(1), \ldots, (28)\})$.

- If A is an answer set of Π_2, then the regulatory structure $(\mathscr{R}, \prec)$ is conformal with the experiment graph $(X, E_P \cup E_R)$, where $\mathscr{R} = \{\gamma_A(x) \mid r(x) \in A\}$ and $\prec = \{(\gamma_A(x), \gamma_A(x')) \mid slower(x, x') \in A\}$.
- If there exists a regulatory structure $(\mathscr{R}, \prec)$ that is conformal with the experiment graph $(X, E_P \cup E_R)$, then there exists an answer set A of logic program Π_2 such that $\mathscr{R} = \{\gamma_A(x) \mid r(x) \in A\}$ and $\prec = \{(\gamma_A(x), \gamma_A(x')) \mid slower(x, x') \in A\}$.

4.4 Recovering the experiment graph

We now extend our logic program to recover from an invalid experiment graph.

Given that the minimum number of additional species is provided via predicate $maxAdds(M)$, we start by introducing M additional species and their capacities:

$$addSpecies(1..M) \leftarrow maxAdds(M). \tag{29}$$

$$capacity(S, V) \leftarrow addSpecies(S), V = 0..1. \tag{30}$$

We freely choose a value for the additional species in (31). Moreover, we declare them as ordinary species in (32) in order to subject them to all constraints on species.

$$1 \{ value(X, S, V) : capacity(S, V) \} 1 \leftarrow addSpecies(S), state(X). \tag{31}$$

$$species(S) \leftarrow addSpecies(S). \tag{32}$$

In a valid extension, there may be no change in the additional species during a perturbation, according to Condition XI:

$$\leftarrow edge(p, X_1, X_2), value(X_1, S, V), \sim value(X_2, S, V), addSpecies(S). \tag{33}$$

Finally, for minimizing the number of changes in the additional species, captured in (34), we use the minimize statement in (35).

$$\begin{aligned} addChange(X_1) \leftarrow\ & edge(r, X_1, X_2), value(X_1, S, V), \\ & \sim value(X_2, S, V), addSpecies(S). \end{aligned} \tag{34}$$

$$\#minimize\{ addChange(X) \}. \tag{35}$$

As before, we get the following correctness and completeness result for this encoding.

Proposition 3

Let $(X, E_P \cup E_R)$ be the experiment graph and Π_3 the logic program $ground(\mathscr{I}(X, E_P \cup E_R) \cup \mathscr{B}(X, E_P \cup E_R) \cup \{(1), \ldots, (34)\})$.

- If A is an answer set of Π_3 being minimal wrt statement $ground((35))$, the regulatory structure $(\mathscr{R}, \prec)$ is conformal with a minimal valid extension[7] of the experiment graph $(X, E_P \cup E_R)$, where $\mathscr{R} = \{\gamma_A(x) \mid r(x) \in A\}$ and $\prec = \{(\gamma_A(x), \gamma_A(x')) \mid slower(x, x') \in A\}$.
- If $(\mathscr{R}, \prec)$ is a regulatory structure that is conformal with a minimal valid extension of the experiment graph $(X, E_P \cup E_R)$, then there is an answer set A of Π_3 being minimal wrt statement $ground((35))$ such that $\mathscr{R} = \{\gamma_A(x) \mid r(x) \in A\}$ and $\prec = \{(\gamma_A(x), \gamma_A(x')) \mid slower(x, x') \in A\}$.

[7] Recall that such an extension is minimal wrt to the number of species and changes in the additional species.

Given the ground instance I in Table 1, which is the logic representation of the experiment graph in Figure 1. The answer sets of

$$ground(I \cup \{(1), \ldots, (34)\} \cup \{maxReactions(12)\} \cup \{maxAdds(2)\} \cup \{length(x_1, x_2, 1), length(x_2, x_3, 1), length(x_3, x_4, 1), length(x_7, x_8, 2), length(x_6, x_7, 2), length(x_5, x_0, 3)\})$$

minimal wrt to the statement $ground((35))$ do correspond to the regulatory structures shown in Figures 4 and 5. Actually, there are four answer sets with symmetric behavior on the additional species. To avoid these, we use a symmetry breaking technique, that is explained in the next section.

4.5 Symmetry breaking

To avoid symmetric models and to speed up computation, we developed symmetry breaking rules for the additional species and the reactions. The additional species can be freely labeled obeying the constraints. Therefore, producing 2^a extensions of the experiment graph. Given the conditions that must hold for a valid experiment graph and a conformal regulatory structure, the number of extensions is restricted. Despite all that, unnecessary extensions can be created, as for each extension in Figures 2 and 3 a mirrored version exists where the labeling of the additional species is switched between additional species x and y. To overcome this, we define an order on the states.[8] The evolution of the added species s is considered to "precede" that of added species $s+1$, if either s changes and $s+1$ not, or if s decreases more than $s+1$ (increases handled analogously). The omitted models can easily be reconstructed by permuting the added species. This is of course not necessary, as a biologist has to research the "meaning" of the species.

Furthermore, we do symmetry breaking on reactions. As we name reactions by numbers, we enforce them to respect some order.[8] We simply use the reaction vector of each reaction to impose a fixed order. So the reaction with the identifier 1 always has the "smallest" reaction vector. This way we omit models that differ only in the naming of the reactions.

As with the refinements of our encoding, discussed in the next section, the corresponding logic programs can be found at Ostrowski (2011).

4.6 Refinements

The encoding that we have presented above was optimized for readability. An enhanced version optimized for performance can be found on the web (Ostrowski 2011). Also, it contains an optimized possibility to compute the static bounds: the maximum number of reactions, additional species and number of possible intermediate states. A basic approximation for the number of reactions is the number of negative changes of each species, as each reaction has to consume at

[8] Any total order is valid.

least one species. This usually results in a high maximum number of reactions. As this number is crucial for the systems performance, we first solve the problem without checking the partial order of the reactions and maximize the number of reactions that shall be used. This computation can be done much faster and gives good approximations for the maximum number of reactions. For approximating the number of intermediate steps, we have a similar approach, considering only two consecutive states and again maximizing the number of reactions that are enabled in between.

We now describe another interesting optimization, this time for the encoding itself. To reduce the size of the strongly connected components of the positive dependency graph (Lin and Zhao 2004) of the logic program, we are using a complete ordering of the reactions instead of a partial one. To this end, we replace (19) with the following rules:

$$\{\ slower(R_1, R_2)\ \} \leftarrow r(R_1), r(R_2), R_1 \neq R_2. \tag{36}$$

$$\begin{aligned} \leftarrow\ & fastest(R_1, L, X), enabled(R_2, L, X), \\ & R_1 \neq R_2, {\sim}slower(R_2, R_1). \end{aligned} \tag{37}$$

We now freely choose an ordering of the reactions in (36) and then assure in (37) that each other enabled reaction has been chosen to be slower. In this way, we reduce the size of the positive cycles in the dependency graph. Now we are no longer restricted to partial orders and the number of different solutions would increase drastically, as each partial order implies many orderings. To overcome this issue, we project only on the reaction vectors. This means that we compute all (minimal) solutions that differ in the reaction vectors but avoid solutions with the same set of reaction vectors but different orderings. This is a feature of our solver *clasp* and can be done very efficiently as shown in Gebser *et al.* (2009) without enumerating all solutions. Furthermore, different redundant constraints have been added to the encoding. For example, a response may not be enabled in a terminal state, reactions that apply in a state must sum up to the difference vector, etc. Without these optimizations of the encoding, for instance, we were unable to solve the instance "ip3r-1-4-dag" used in the next section.

5 Experiments

To test the feasibility of our approach, we used "in silico"[9] experiments generated from a synthetic bio-chemical network. Several time series are generated and some combinations of them are shown in Table 1. From the time series, the values of two species were removed to simulate experiments not measuring all species. So the data of 14 species is used. We compared our ASP approach to the direct implementation described in Durzinsky *et al.* (2010). Unfortunately, no direct implementation exists that does handle the range of constraints described here. Some approaches additionally handle the creation of catalysators/inhibitors and

[9] The data is confidential and was made anonymous by our industrial partner.

Table 1. *Reconstructing a model using in silico experiments*

Instances	ASP times	unopt. ASP times	Direct impl. times	Add. species	Models	Maximum of reactions	States	Exper-iments
ip3r-1	1.3	4.5	0.1	1	4	7	11	2
ip3r-1-dag	1.1	4.2	0.1	1	4	6	12	3
ip3r-2-dag	0.7	1.5	0.1	1	2	4	11	4
ip3r-3-dag	0.3	0.5	0.1	0	1	4	11	4
ip3r-4-dag	0.5	1.0	0.1	1	8	5	10	4
ip3r-1+4-dag	3.6	16.4	0.1	1	8	8	20	6
ip3r-1+2-dag	34.0	300.9	MEM	2	44	10	21	6
ip3r-1+3-dag	59.0	750.7	MEM	2	128	10	21	6
ip3r	30.2	244.6	MEM	1	2	9	37	11
ip3r-1-4	656.7	3435.3	MEM	2	104	11	37	11
ip3r-1-4-dag	3562.3	TIME	MEM	2	280	12	38	12

others lack the check of the partial ordering on the reactions. The implementation that comes closest to ours does not compute the number of additional species (it has to be given from outside) and does not do the partial order check. But for the feasibility test, we decided to compare with this version, referred to as the "direct implementation". We tested it, giving the maximum number of additional species, as computed by our approach, as input. The benchmark were run single-threaded on an Intel Xeon machine with 32 GB main memory possessing two 3.4 GHz processor with eight cores each; each benchmark was restricted to 2 GB of memory and 1 h runtime. For the times, we show the average of three runs in seconds. MEM indicates that the memory limit was reached. For the ASP approach we use the grounder *gringo* (3.0.4) and the solver *clasp* (1.3.6). We tested the unoptimzed version (denoted by "unopt. asp times") of our encoding as well as the optimzed version with the refinements from Section 4.6 (denoted by "asp times"). The number of models gives the number of different regulatory structures that have been reconstructed by the ASP approach. The number of states (measured time points) and experiments used for the reconstruction is also given. With "maximum of reactions", we refer to the maximum of reactions that are used in the regulatory structures. As it can easily be seen in Table 1, the number of experiments (and therefore the number of states) and additional species increases the difficulty of the problem. Our refined approach was able to solve all of the problems, which means that it is feasible to run it on the shown number of experiments and states, as long as the number of additional species stays low. The direct implementation, also having limited functionality, has severe problems with memory usage. [10] On the other hand, it has less initialization overhead on the small examples.

[10] We also tested it with 3 GB memory restriction, which did not changed any of the results.

6 Discussion

In the area of automatic network reconstruction many different approaches have been developed. They differ in the used techniques and the kind of system they reconstruct. Statistical methods are used, e.g., in Gifford and Jaakkola (2001) and Peer *et al.* (2001) reconstructing wiring diagrams using Bayesian network methods. More descriptive systems are time continuous deterministic dynamical systems. Using ordinary differential equations, Yeung *et al.* (2002) and Laubenbacher and Stigler (2004) infer a network by solving a non-homogeneous system of linear equations, given a set of experiments. The result is a minimal network in means of the structure of the functions. Enumerating algorithms are used in (Liang *et al.* 1998) and (Akutsu *et al.* 1999) to search for the sparsest Boolean model. Boolean networks however are more intuitive but less expressive. Hybrid models seem to compensate the drawbacks of Boolean models. (Repsilber *et al.* 2002) uses a genetic algorithm on gene expression data to produce a hybrid model, including quantitative and qualitative information. Depending on the quality of the available experimental data and the type of the studied models, further approaches have been developed; see (Durzinsky *et al.* 2008; Durzinsky *et al.* 2010; Wagler 2011) for a detailed comparison to our underlying approach in Section 2.

Our approach does not try to find the "best" model, because this approach looses information about important alternatives that only a biologist can decide on. It rather shows *all* possible models conform with the experiments. We use a hybrid model, incorporating several discrete levels of concentrations for the species. This extends the purely Boolean approach, but can of course not keep up with the expressiveness of differential equations. The modeling as a logic program makes it simple to integrate all the various constraints. The approach can easily be extended by further constraints, or constraints can be relaxed. Although we used a state-of-the-art solver for logic programs, we rigorously had to shrink the size of the problem due to different preprocessing steps. In contrast to other approaches, this does not change the problem or the solutions. We still infer all possible explanations for the experiments. Furthermore, our approach benefits from future developments in ASP solving, like parallelization.

From a broader perspective, ASP has already proved its utility for diverse biological applications. Among them, we find (Baral *et al.* 2004; Dworschak *et al.* 2008; Gebser *et al.* 2008; Erdem and Türe 2008; Schaub and Thiele 2009; Erdem 2009; Dal Palù *et al.* 2009; Gebser *et al.* 2010), all of which treat rather different biological problems from what we tackled in the paper at hand. A feature common to many among these approaches is the exploitation of ASP's combinatorial nature in inspecting either all or what is common to all solutions to a biological problem.

7 Summary

We presented a declarative solution to the ANR problem using ASP. We support checking validity of an experiment graph, predicting the behavior of unmeasured species, and reconstructing all possible explanations for a given set of experiments

using a partial order on the used reactions. We showed that the mathematical representation of the problem can be easily translated into a logic program which then can be handled by a state of the art grounder and solver for ASP. As the logic program can easily be split into different parts, also various versions of the problem (adding or relaxing some of the constraints) can be tackled. This is especially useful when the problem is refined to use catalysts or inhibitors as done in (Durzinsky *et al.* 2010) or just has to be changed for special purposes. It can also be used to introduce P-Invariants as described in Durzinsky *et al.* (2010). This avoids rewriting complex programming code and automatically benefits from developments in ASP solving. We have shown that our approach is scalable for a certain class of perturbation experiments and it is already used by the biology research group of Wolfgang Marwan at the Magdeburg Centre for Systems Biology and in the context of the GoFORSYS (goforsys) project. It outperforms the direct implementation of the problem while supporting a broader range of functionality.

In the future we plan to extend our approach in terms of catalysts, as described in Durzinsky *et al.* (2010). We then want to combine it with the ordering of reactions and also the automatic addition of species. Furthermore, we need to improve the approach to be capable of dealing with larger networks, as all currently tested networks are of small to medium size. As time series experiments are usually very costly, we want to investigate how to find optimal experiments to reduce the number of possible regulatory structures.

References

Akutsu, T., Miyano, S., and Kuhara, S. 1999. Identification of genetic networks from a small number of gene expression patterns under the boolean network model. In *Proceedings of the Pacific Symposium on Biocomputing*, vol. 4. World Scientific Press, 17–28.

Baral, C. 2003. *Knowledge Representation, Reasoning and Declarative Problem Solving*. Oxford University Press, Oxford, UK.

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

Dal Palù, A., Dovier, A. and Pontelli, E. 2009. Logic programming techniques in protein structure determination: Methodologies and results. See Erdem *et al.* (2009), 560–566.

Durzinsky, M., Marwan, W. and Wagler, A. 2010. Reconstructing extended petri nets. *Journal of Mathematical Biology*. Preprint series: 10–19. URL: http://www.fma.ovgu.de.

Durzinsky, M., Wagler, A. and Weismantel, R. 2008. A combinatorial approach to reconstruct petri nets from experimental data. *Lecture Notes in Bioinformatics 5307*, 328–346.

Durzinsky, M., Wagler, A. and Weismantel, R. 2010. An algorithmic framework for network reconstruction. *Theoretical Computer Science*. In Press, Corrected Proof. URL: http://www.sciencedirect.com.

Durzinsky, M., Wagler, A., Weismantel, R., and Marwan, W. 2008. Automatic reconstruction of molecular and genetic networks from discrete time series data. *Biosystems 93*(3), 181–190.

Dworschak, S., Grell, S., Nikiforova, V., Schaub, T., and Selbig, J. 2008. Modeling biological networks by action languages via answer set programming. *Constraints 13*(1–2), 21–65.

Erdem, E. 2009. PHYLO-ASP: Phylogenetic systematics with answer set programming. See Erdem *et al.* (2009), 567–572.

Erdem, E., Lin, F. and Schaub, T., Eds. 2009. *Proceedings of the Tenth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'09)*. Lecture Notes in Artificial Intelligence, vol. 5753. Springer-Verlag.

Erdem, E. and Türe, F. 2008. Efficient haplotype inference with answer set programming. In *Proceedings of the Twenty-third National Conference on Artificial Intelligence (AAAI'08)*, D. Fox and C. Gomes, Eds. AAAI, 436–441.

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)*, F. Lin and U. Sattler, Eds. AAAI, 497–507.

Gebser, M., Kaminski, R., Kaufmann, B., Ostrowski, M., Schaub, T., and Thiele, S. A user's guide to `gringo`, `clasp`, `clingo`, and `iclingo`. Accessed 7 June 2011. URl: http://potassco.sourceforge.net.

Gebser, M., Schaub, T., Thiele, S., Usadel, B., and Veber, P. 2008. Detecting inconsistencies in large biological networks with answer set programming. In *Proceedings of the Twenty-fourth International Conference on Logic Programming (ICLP'08)*, M. Garcia de la Banda and E. Pontelli, Eds. Lecture Notes in Computer Science, vol. 5366. Springer-Verlag, Heidelberg, 130–144.

Gebser, M., Kaufmann, B., and Schaub, T. 2009. 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. Lecture Notes in Computer Science, vol. 5547. Springer-Verlag, Heidelberg, 71–86.

Gelfond, M. 2008. Answer sets. In *Handbook of Knowledge Representation*, V. Lifschitz, F. van Hermelen, and B. Porter, Eds. Elsevier, Chapter 7, 285–316.

Gifford, D. and Jaakkola, T. 2001. Using graphical models and genomic expression data to statistically validate models of genetic regulatory networks. In *Proceedings of the Pacific Symposium on Biocomputing* vol. 6. 422–433.

goforsys. Accessed 7 June 2011. URL: http://www.goforsys.de.

Laubenbacher, R. and Stigler, B. 2004. A computational algebra approach to the reverse engineering of gene regulatory networks. *Journal of Theoretical Biology 229*(4), 523–537.

Liang, S., Fuhrman, S. and Somogyi, R. 1998. Reveal, a general reverse engineering algorithm for inference of genetic network architectures. *Pacific Symposium on Biocomputing 3*, 18–29.

Lin, F. and Zhao, Y. 2004. ASSAT: computing answer sets of a logic program by SAT solvers. *Artificial Intelligence 157*(1–2), 115–137.

Marwan, W., Wagler, A. and Weismantel, R. 2008. A mathematical approach to solve the network reconstruction problem. *Mathematical Methods of Operatios Research 67*(1), 117–132.

Ostrowski, M. 2011. Asp encoding for automatic network reconstruction. Accessed 7 June 2011. URL: http://www.cs.uni-potsdam.de/wv/NetworkReconstruction/.

Peer, D., Regev, A., Elidan, G. and Friedman, N. 2001. Inferring subnetworks from perturbed expression profiles. *Bioinformatics 17*(suppl 1), 215–224.

Repsilber, D., Liljenstrm, H. and Andersson, S. 2002. Reverse engineering of regulatory networks: simulation studies on a genetic algorithm approach for ranking hypotheses. *Biosystems 66*(1–2), 31 – 41.

Schaub, T. and Thiele, S. 2009. Metabolic network expansion with ASP. In *Proceedings of the Twenty-fifth International Conference on Logic Programming (ICLP'09)*, P. Hill and D. Warren, Eds. Lecture Notes in Computer Science, vol. 5649. Springer-Verlag, Heidelberg, 312–326.

Simons, P., Niemelä, I. and Soininen, T. 2002. Extending and implementing the stable model semantics. *Artificial Intelligence 138*(1–2), 181–234.

Syrjänen, T. *Lparse 1.0 User's Manual*. Accessed 7 June 2011. URL: http://www.tcs.hut.fi/Software/smodels/lparse.ps.gz.

Wagler, A. 2011. Prediction of network structure. *Modeling in Systems Biology 16*, 307–336.

Yeung, M., Tegn(c)r, J., and Collins, J. 2002. Reverse engineering gene networks using singular value decomposition and robust regression. *Proceedings of the National Academy of Sciences of the United States of America 99*(9), 6163–6168.