PROCEEDINGS OF SCIENCE

Drug Design with Answer Set Programming

Dietmar Seipel* [a], **Oleksandr Kovalchuck** [a], **and Thomas Dandekar** [b]
[a] *Department of Computer Science*
[b] *Department of Bioinformatics*
University of Würzburg, Am Hubland, D – 97074 Würzburg, Germany
E–mail: dietmar.seipel@uni-wuerzburg.de | kaval@gmx.net |
dandekar@biozentrum.uni-wuerzburg.de

A *drug target* is a key biomolecule in a metabolic or signalling pathway modifiable by a drug to ameliorate a specific disease condition or pathology [11]. For rational drug design, sufficient information about the biomolecule is required – such as therapeutic value, pathway role, etc. – provided from databases and detailed pharmacological studies. We exploit the Kyoto Encyclopedia of Genes and Genomes (KEGG) database collection, which integrates biological compounds and enzymatic pathways [9]; this collection is part of the Japanese GenomeNet network of database and computational services for genome research and related research areas in biomedical sciences.

We use *answer set programming* (ASP) [1] for deriving suitable minimal sets of drug targets. We encode a metabolic pathway as a disjunctive logic program $\mathscr{P}$ with default negation. Evaluating $\mathscr{P}$ with the answer set programming system DLV [10] obtains alternative sets of enzymes that could be blocked in order to inhibit the production of a target compound.

Due to the problem of *combinatorial explosion*, drug design for metabolic networks is typically done on large *computer grids*, if hundreds of reactions are envolved. The combination with ASP embodies very efficient heuristics for avoiding part the problem, such that larger networks can be handled.

The International Symposium on Grids and Clouds and the Open Grid Forum
ISGC 2012
February 26 – March 02, 2012
Academia Sinica, Taipei, Taiwan

*Speaker.

PoS(ISGC 2012)003

Contents

1. Introduction

A drug target is a key molecule, such as an enzyme, involved in a particular metabolic or signalling pathway that is specific to a disease condition or pathology. Modern drug design does not rely on trial–and–error testing like traditional drug discovery methods. For selecting a biomolecule as a drug target, information about that biomolecule is required, such as its therapeutic value and its role in a certain pathway. Such information requires extensive measurements and drug studies, but then can also be obtained from sources such as chemical and metabolic pathway databases [9].

Different bioinformatical approaches exploit these, starting from more topological and structural approaches, such as static network analysis [7], to more dynamical analytical methods, such as flux balance analysis with its different flavours, e.g., elementary mode analysis and extreme pathways [3, 12]. The YanaSquare system [13] computes elementary

modes for networks of over 200 reactions on a grid (40 CPUs) with runtimes of up to a day. If there is even more kinetic information available, detailed modelling with power law formalisms or ordinary differential equations is possible. We are concerned with the extension of structural approaches by rules implemented in answer set programming. As we will show, this greatly enhances the capabilities of structural analyses without the requirement of kinetic information, and in general without the problems of combinatorial explosion [3, 12].

For example, in the fragment of the metabolic pathway for glycolysis that is shown in Figure 1, blocking the two enzymes 2.7.1.147 and 5.1.3.3 (light red boxes) inhibits the production of the compound β–D–Glucose (middle left). The first enzyme participates in two reactions; it is only essential that the lower one is blocked. Note that this pathway is from the KEGG database, which gives a good overview on all enzyme reactions; but an automated analysis might still give results that do not agree with the biology, since, e.g., enzymes may sometimes be missing or misrepresented in the database. An additional level of complexity is given by spontaneous reactions; e.g., this occurs for the epimerase reaction, in which enzyme 5.1.3.3 participates, albeit with a low reaction rate.

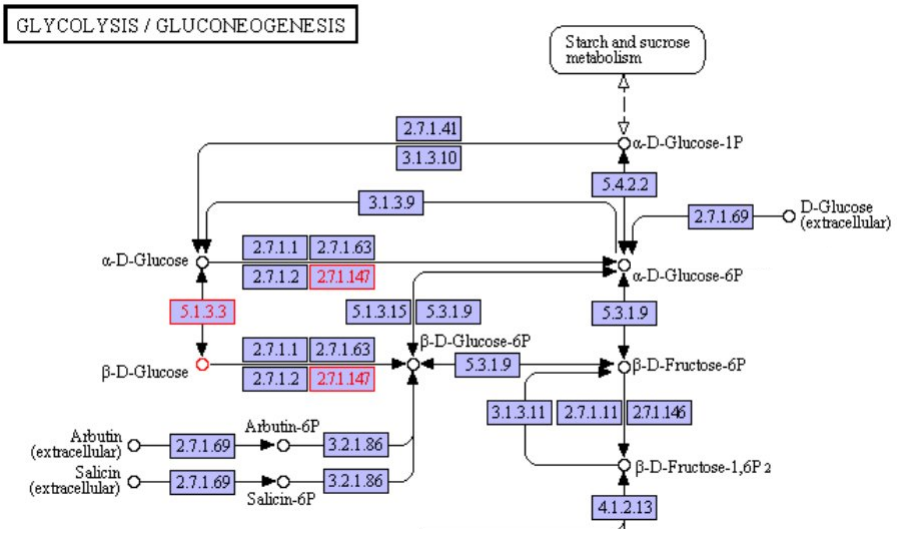

Figure 1: Fragment of a metabolic pathway with two blocked enzymes: 2.7.1.147 and 5.1.3.3

Declarative programming focusses on the semantics and the structure of the solution leaving details of the computation to the underlying engine. Answer set programming (ASP) is a special case [1]. Declarative programming is particularly powerful in software engineering when changes or extensions to the functionality of a system are often performed: Only the logic of the program in the area where the change is required is modified without affecting the rest of the application.

In the ASP system DLV [10], it is possible to minimize the number of blocked enzymes. We derive solutions with higher cardinality iteratively. We use two types of rules for deriving blocked enzymes or metabolites; thus, we can stop the derivation of non–succeeding

branches of the computation early to avoid *combinatorial explosion*. E.g., for each reaction producing a metabolite, a disjunctive rule encodes the fact that inhibiting the production of this metabolite requires that one of the enzymes or compounds involved in the reaction is blocked. Modelling of metabolism involves thus $\mathcal{NP}$–hard problems implying combinatorial explosion and large–scale data, which can advantageously be approached exploiting *grid* and *cloud computing*.

Approaches based on logic or ASP have already been applied to networks in other areas of bioinformatics. The Biocham system [8] offers automated reasoning tools for querying the temporal properties of interaction networks under all possible behaviours using Computation Tree Logic (CTL). Baral et al. [2] use an action–style, knowledge based approach for supporting various tasks of reasoning (including planning, hypothetical reasoning, and explanation) about signaling networks in the presence of incomplete and partial information.

The rest of this paper is organized as follows: In Section 2, we present our encoding of metabolic pathways as disjunctive logic programs. We also give an example, and we briefly explain the semantics, which is given by the stable models. In Section 3, we analyze the encoding. We develop two refinements, the Approches 1 and 2, of our framework, which correctly represent our drug design problem. Readers, who are not so experienced with ASP, might want to skip this technical section. The efficiency of the two approaches is compared for real application examples in Section 4. The application to KEGG pathways is explained in Section 5. Finally, Section 6 gives some conclusions.

2. Encoding of Metabolic Pathways

A fragment of a metabolic pathway for gycolysis is shown in Figure 1. The nodes (circles) represent compounds. The edges represent reactions, and they are labelled by the participating enzymes (boxes). It is sufficient to consider reactions

$$A_1, \ldots, A_n \xrightarrow{E_1,\ldots,E_m} A.$$

producing a compound A from other compounds $A_1, \ldots, A_n$ using the enzymes $E_1, \ldots, E_m$; reactions producing more than one compound can be represented by multiple edges.

A metabolic pathway is mapped to a disjunctive logic program $\mathcal{P}$ with default negation not . Every edge of the pathway – corresponding to a reaction – is mapped to a set of rules. $X^{\otimes}$ means that a compound or enzyme X is blocked.

We use rules for encoding the reachability of nodes, and two types of rules for deriving blocked enzymes, which we call B1– and B2–rules; the importance of using both types of rules is, that we can stop the derivation of non–succeeding branches of the computation early. Moreover, there are generic rules for relating nodes and blocked nodes, and rules for handling the sources and the vital nodes of the pathway.

2.1 Reachability Rules

The compound A can be produced, if the compounds A_i, for $1 \leq i \leq n$, can be produced, and none of the enzymes E_i, for $1 \leq i \leq m$, is blocked:

$$(1) \qquad A \leftarrow A_1 \wedge \ldots \wedge A_n \wedge \mathsf{not}\ E_1^{\otimes} \wedge \ldots \wedge \mathsf{not}\ E_m^{\otimes}.$$

A is called the head, and the conjunction following the implication arrow $\leftarrow$ is called the body of r. In Figure 1, all reactions require just one compound (i.e., $n = 1$), but some reactions require more than one enzyme (i.e., $m \geq 1$).

2.2 Blocking Rules

The production starts with a given set $\mathscr{S}$ of source compounds. The goal is to find a minimal set $\mathscr{E}$ of enzymes to be blocked, such that the production of a given set $\mathscr{I}$ of compounds is inhibited, while all vital compounds can still be produced. Blockings are derived by two types of – possibly recursive and disjunctive – rules:

B1–Rules (Backward Blocking). If the reaction above should be inhibited, then at least one of the compounds A_i or enzymes E_j must be blocked by a drug. This leads to a rule $\alpha_l \leftarrow A^{\otimes}$ with a disjunctive head $\alpha_l = A_1^{\otimes} \vee \ldots \vee A_n^{\otimes} \vee E_1^{\otimes} \vee \ldots \vee E_m^{\otimes}$, which inverts the way of reasoning of the reachability rules by deriving blockings in the body from the blocking of the head; we call these rules B1–rules. If there are k reactions for producing a compound A, then we obtain k independent B1–rules:

$$(2) \qquad \alpha_l \leftarrow A^{\otimes},\ 1 \leq l \leq k.$$

B2–Rules (Forward Blocking). Following the direction of reasoning of the reachability rules, we can also derive $A^{\otimes}$ by the rule

$$(3) \qquad A^{\otimes} \leftarrow \alpha_1 \wedge \ldots \wedge \alpha_k \wedge \mathsf{not}\ \mathit{source}(A).$$

This rule combines the disjunctions $\alpha_l = A_1^{\otimes} \vee \ldots \vee A_n^{\otimes} \vee E_1^{\otimes} \vee \ldots \vee E_m^{\otimes}$ obtained from the reactions producing A. In DLV, the disjunctions α_l are not allowed in rule bodies. By distributive multiplication of $\alpha_1 \wedge \ldots \wedge \alpha_k$ we obtain a disjunction of conjunctions β_j, and we can form suitable B2–rules $A^{\otimes} \leftarrow \beta_j \wedge \mathsf{not}\ \mathit{source}(A)$ for DLV.

2.3 Example for the Encoding

Realistic metabolic pathways from the KEGG database are parts of networks with typically more than 100 nodes. For the toy metabolic pathway with the two edges $a \xrightarrow{1} b$ and $c \xrightarrow{2} b$, the following disjunctive logic program $\mathscr{P}$ is produced. There are only rules for the node b, since the other two nodes a and c have no incoming edges.

Reachability Rules:

$b \leftarrow a \wedge \mathsf{not}\ 1^{\otimes}, \quad b \leftarrow c \wedge \mathsf{not}\ 2^{\otimes}.$

B1–Rules:

$$a^{\otimes} \vee 1^{\otimes} \leftarrow b^{\otimes}, \quad c^{\otimes} \vee 2^{\otimes} \leftarrow b^{\otimes}.$$

B2–Rules:

$$b^{\otimes} \leftarrow a^{\otimes} \wedge c^{\otimes} \wedge \mathsf{not}\ \mathit{source}(a), \quad b^{\otimes} \leftarrow a^{\otimes} \wedge 2^{\otimes} \wedge \mathsf{not}\ \mathit{source}(a),$$
$$b^{\otimes} \leftarrow 1^{\otimes} \wedge c^{\otimes} \wedge \mathsf{not}\ \mathit{source}(a), \quad b^{\otimes} \leftarrow 1^{\otimes} \wedge 2^{\otimes} \wedge \mathsf{not}\ \mathit{source}(a).$$

The B2–rules describe that b is blocked, if both reachability rules cannot fire. They are derived from the disjunctions $\alpha_1 = a^{\otimes} \vee 1^{\otimes}$ and $\alpha_2 = c^{\otimes} \vee 2^{\otimes}$ obtained from the reachability rules supporting b. The corresponding reactions are inhibited, if $\alpha_1 \wedge \alpha_2$ holds; distributive multiplication results in four alternatives $a^{\otimes} \wedge c^{\otimes}$, $a^{\otimes} \wedge 2^{\otimes}$, $1^{\otimes} \wedge c^{\otimes}$, and $1^{\otimes} \wedge 2^{\otimes}$, for the conjunction β_j.

2.4 Generic Rules and Basic Facts

The denial rule (4) states, that a reachable node cannot be blocked. The closed–world rule (5) infers the blocking of non–reachable nodes. Rule (6) expresses, that a source node is reachable, while the denial rules (7) and (8) express, that vital nodes and source nodes should not be blocked. The atoms A range over the nodes of the metabolic pathway.

(4) $\leftarrow A \wedge A^{\otimes}$

(5) $A^{\otimes} \leftarrow \mathsf{not}\ A$

(6) $A \leftarrow \mathit{source}(A)$

(7) $\leftarrow A^{\otimes} \wedge \mathit{vital}(A)$

(8) $\leftarrow A^{\otimes} \wedge \mathit{source}(A)$

The source compounds and the vital compounds of the metabolic pathway are specified in $\mathscr{P}$ by basic facts for the predicate symbols *source*/1 and *vital*/1, respectively, and the compounds A that should be blocked are specified in $\mathscr{P}$ by facts $A^{\otimes}$

2.5 Stable Models of Disjunctive Logic Programs

A disjunctive logic program $\mathscr{P}$ with default negation consists of rules of the form

$$A_1 \vee \ldots \vee A_k \leftarrow B_1 \wedge \ldots \wedge B_m \wedge \mathsf{not}\ C_1 \wedge \ldots \wedge \mathsf{not}\ C_n.$$

The disjunction $A_1 \vee \ldots \vee A_k$ of atoms is called the head of the rule. The conjunction $B_1 \wedge \ldots \wedge B_m$ of atoms is called the positive body, and the conjunction $\mathsf{not}\ C_1 \wedge \ldots \wedge \mathsf{not}\ C_n$ is called the negative body; together, they form the body of the rule. A rule with an empty body, i.e., $n = m = 0$, is called a fact, and it is simply denoted as $A_1 \vee \ldots \vee A_k$. A rule with an empty head, i.e., $k = 0$, is called a denial rule.

An *Herbrand interpretation* of $\mathscr{P}$ assigns a truth value to the ground atoms over the language of $\mathscr{P}$. It can be given by a set I of ground atoms: then, $I(A)$ is true, if and only if $A \in I$, and $I(A)$ is false, otherwise. E.g., for the disjunctive logic program $\mathscr{P} = \{ a \vee b \leftarrow c, c \leftarrow \mathsf{not}\ a \}$ with two rules, a possible Herbrand interpretation is $I = \{ b, c \}$.

In ASP, the models of $\mathscr{P}$ are computed using the Gelfond–Lifschitz transformation $\mathscr{P}^I$. This is necessary, since atoms can depend (recursively) on themselves through default negation. Assume, that we have guessed an Herbrand interpretation I, and that we would like to test, whether I is a stable model of $\mathscr{P}$. Then, we get rid of all default negations in $\mathscr{P}$ by basically evaluating them w.r.t. I. Technically, a ground instance of a rule of the form above is transformed to $A_1 \vee \ldots \vee A_k \leftarrow B_1 \wedge \ldots \wedge B_m \wedge I(\mathsf{not}\, C_1) \wedge \ldots \wedge I(\mathsf{not}\, C_n)$. If all C_i are false in I, then $I(\mathsf{not}\, C_1) \wedge \ldots \wedge I(\mathsf{not}\, C_n)$ is true, and the reduced rule $A_1 \vee \ldots \vee A_k \leftarrow B_1 \wedge \ldots \wedge B_m$ becomes a part of $\mathscr{P}^I$. Then, I is a *stable model* – or answer set – of $\mathscr{P}$, if and only if I is a minimal model of $\mathscr{P}^I$.

As a special case, let us first illustrate logic programs without disjunction in the rule heads, i.e., $k = 1$ head atoms. For instance, the non–recursive logic program $\mathscr{P} = \{a \leftarrow b \wedge \mathsf{not}\, c, b\}$ with two rules has one stable model $I = \{a, b\}$. This is due to the fact that there is no reason for inferring c. The Gelfond–Lifschitz transformation $\mathscr{P}^I = \{a \leftarrow b, b\}$ has exactly the minimal model I. The recursive logic program $\mathscr{P} = \{a \leftarrow b \wedge \mathsf{not}\, c, b, c \leftarrow \mathsf{not}\, a\}$ has two stable models $I_1 = \{a, b\}$ and $I_2 = \{b, c\}$. Now, both a and c are supported by rules. The Gelfond–Lifschitz transformation $\mathscr{P}^{I_1} = \{a \leftarrow b, b\}$ has exactly the minimal model I_1, and $\mathscr{P}^{I_2} = \{b, c\}$ has exactly the minimal model I_2.

The ASP encodings of metabolic pathways in this paper additionally require disjunctive rule heads. The definition of stable models, which we have given above, covers the case of disjunctive rule heads. E.g., for the disjunctive logic program $\mathscr{P} = \{a \vee b \leftarrow c, c \leftarrow \mathsf{not}\, a\}$, obviously, either a or c can be true, but not both. If c is true, then b must also be true. It turns out, that $I = \{b, c\}$ is the unique stable model of $\mathscr{P}$.

Of course, ASP systems do not compute stable models using the described guess–and–check definition. Instead, the stable models are constructed iteratively using highly optimized methods with many techniques; the description of these techniques, however, is beyond the scope of this paper. There exist very efficient ASP systems, such as DLV, which can solve practical problems of considerable size.

For preparing the input to DLV and for further processing the output, we use logic programming in PROLOG [4]. For metabolic pathway problems, there can be several solutions given by the stable models of the corresponding disjunctive logic program. We have also developed a visualization of the sets of blocked enzymes in tree form, cf. Figure 2. Enzymes occuring in several solutions tend to appear higher in the tree.

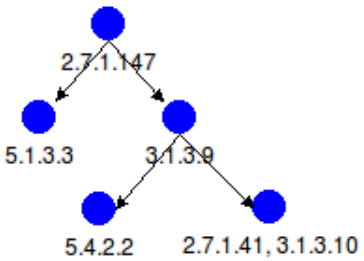

Figure 2: Visualization of the solutions

3. Technical Analysis

The *stable models* of $\mathscr{P}$ help to derive blockings of enzymes to inhibit the production of some given compounds while still permitting the production of the vital compounds. For minimizing *side effects*, we would like to derive minimal sets of blocked enzymes. Thus, for adequately capturing the semantics of a metabolic pathway, we have to analyze the way in which we apply the rules (see approaches A1 and A2 below).

The reachability rule (1) helps to handle cycles in the pathway. The B1–rules (2) cannot be dropped, since then the blocking of enzymes could no longer be inferred. The B2–rules (3) and the closed–world rules (5) could be dropped under certain circumstances. The inverse closed–world rule $A \leftarrow \mathsf{not}\, A^{\otimes}$ should not be included. The generic rule (4) for complementarity of A and $A^{\otimes}$ is necessary to avoid inconsistent models. Finally, the source compounds are reachable (6), no vital compounds should be blocked (7), and no source compounds can be blocked (8).

3.1 Recursion and Stratification

The disjunctive logic program $\mathscr{P}$ is usually *recursive* and *non–stratifiable*. This means, that $\mathscr{P}$ contains recursive cycles involving default negation. As a consequence, standard theorem provers cannot be used for evaluating $\mathscr{P}$; instead, we need answer set solvers, such as DLV, for computing the stable models of $\mathscr{P}$. The following example shows, that $\mathscr{P}$ is not always stratifiable: even a single edge $a \xrightarrow{1} b$ in the metabolic pathway leads to a cycle involving default negation: $b \leftarrow a \wedge \mathsf{not}\, 1^{\otimes}$, $a^{\otimes} \vee 1^{\otimes} \leftarrow b^{\otimes}$, $b^{\otimes} \leftarrow \mathsf{not}\, b$.

3.2 Minimality w.r.t. Blockings

An interpretation I *block–subsumes* another interpretation J, if its restriction $I \cap \mathscr{E}^{\otimes}$ on the set $\mathscr{E}^{\otimes}$ of blocked enzymes is a strict subset of $J \cap \mathscr{E}^{\otimes}$. A stable model M of $\mathscr{P}$ is called block–minimal, if it is not block–subsumed by another stable model of $\mathscr{P}$. For drug design, we are especially interested in the block–minimal stable models of $\mathscr{P}$. However, the metabolic pathway given in Figure 3 shows, that not all stable models of $\mathscr{P}$ are block–minimal.

Example. We consider the metabolic pathway which is visualized in Figure 3.

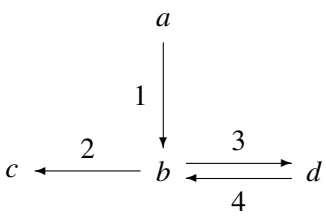

Figure 3: Metabolic pathway with 4 edges

Assume that a is a source and that c should be blocked. Then, the corresponding disjunctive logic program $\mathscr{P}$ has three stable models:

$$M_1 = \{\, 1^{\otimes}, a, b^{\otimes}, c^{\otimes}, d^{\otimes} \,\}, \quad M_2 = \{\, 2^{\otimes}, a, b, c^{\otimes}, d \,\}, \quad M_3 = \{\, 2^{\otimes}, 3^{\otimes}, a, b, c^{\otimes}, d^{\otimes} \,\}.$$

As sets of atoms, they are all minimal; the reason is, that the sets of reachable and blocked atoms are always complementary. But, when we only consider the blocked enzymes, then we can see that the third model M_3 is not block–minimal, since it is block–subsumed by M_2: $M_2 \cap \mathscr{E}^{\otimes} = \{2^{\otimes}\} \subsetneq \{2^{\otimes}, 3^{\otimes}\} = M_3 \cap \mathscr{E}^{\otimes}$.

3.3 Refined Computations

There are two ways out of the dilemma described in the previous subsection.

Iterative Approach A1. We can compute only stable models with a minimal number of blocked enzymes. Above, we would obtain M_1 and M_2. In subsequent interations, we forbid these stable models by denial rules, and thus obtain stable models with a higher cardinality. Thus, stable models that are not block–minimal are no longer computed.

2–Phase Approach A2. We can drop the reachability rules and the closed–world rules $A^{\otimes} \leftarrow \mathsf{not}\, A$, and thus restrict the computation to the blocking of compounds and enzymes. In that case, the computed stables models may not be admissible, since vital compounds may be unreachable. Thus, in a post–processing step, we have to test the computed stable models for admissibility. Moreover, we have to select the block–minimal stable models.

Example. For the metabolic pathway visualized in Figure 4, the first phase of approach A2 produces a non–block–minimal stable model, if we assume the source compound a, the vital compound c, and the compound d to be blocked.

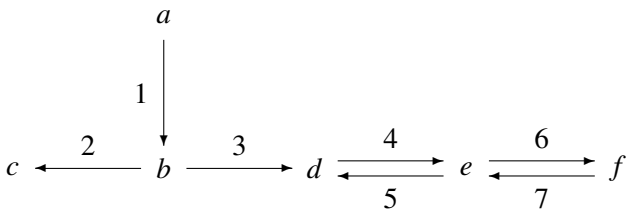

Figure 4: Metabolic pathway with 7 edges

Approach A1 computes the intuitive model M, which only blocks enzyme 3:

$$M = \{3^{\otimes}, a, b, c, d^{\otimes}, e^{\otimes}, f^{\otimes}\}.$$

The first phase of approach A2, however, produces 2 stable models, due to the missing closed–world rule:

$$N_1 = \{3^{\otimes}, d^{\otimes}, e^{\otimes}, f^{\otimes}\}, \quad N_2 = \{3^{\otimes}, 5^{\otimes}, d^{\otimes}\}.$$

The first stable model N_1 of A2 corresponds to the intuitive model M. The second stable model N_2 of A2 additionally blocks the enzyme 5, since it cannot infer that the cycle envolving d is already deactivated by blocking the enzyme 3. N_2 is not block–minimal. In the post–processing phase, N_2 will be filtered out because of N_1. When we look at the completed versions N_1' and N_2' of N_1 and N_2, respectively, then we see that N_1' subsumes N_2', since enzyme 5 is redundantly blocked in N_2':

$$N_1' = \{3^{\otimes}, a, b, c, d^{\otimes}, e^{\otimes}, f^{\otimes}\}, \quad N_2' = \{3^{\otimes}, 5^{\otimes}, a, b, c, d^{\otimes}, e^{\otimes}, f^{\otimes}\},$$

Approach A1 produces the stable model $N_1' = M$, but it does not produce the model N_2', since N_2' is non–minimal.

For the metabolic pathway, which extends the pathway of Figure 4 by an additional loop $c \xrightarrow{7} g$, $g \xrightarrow{8} c$ at the vital compound c, approach A2 would even produce two further stable models blocking the enzyme 1:

$$N_3 = \{1^{\otimes}, 5^{\otimes}, b^{\otimes}, d^{\otimes}\}, \quad N_4 = \{1^{\otimes}, d^{\otimes}, d^{\otimes}, e^{\otimes}, f^{\otimes}\}.$$

Due to the missing reachability and closed–world rules, these stable models cannot infer that c is not reachable. They are not admissible, and they also have to be filtered out in the post–processing phase.

3.4 Further Modifications

A few further modifications of the described approach seem to be worth to be explored. Dropping the B2–rules, or dropping the closed–world rules $A^{\otimes} \leftarrow \mathsf{not}\, A$ (which are only used in approach A1) could result in a speedup of the computation, but they could also produce wrong models. The B1–rules cannot be dropped, since then the blocking of enzymes could no longer be inferred.

Dropping the B2–rules leads to a problem for approach A2, which is working without the reachability rules, since stable models are derived which are not block–minimal. For the metabolic pathway example of Figure 3, the stable model K_3 is not block–minimal:

$$K_1 = \{1^{\otimes}, b^{\otimes}, c^{\otimes}, d^{\otimes}\}, \quad K_2 = \{2^{\otimes}, c^{\otimes}\}, \quad K_3 = \{1^{\otimes}, 4^{\otimes}, b^{\otimes}, c^{\otimes}\}.$$

The two other stable models K_1 and K_2 correspond to M_1 and M_2, respectively, from above; the non–block–minimal stable model K_3 does not correspond to M_3. The approach A1 is not affected from dropping B2–rules, since it uses the reachability and the closed–world rules to derive the correct stable models M_1 and M_2. Dropping both the B2–rules and the closed–world rules may lead to a problem, e.g., if the compound d is vital.

However, we could drop the closed–world rules and merge them into the rules (7) and (8) from above to obtain the denial rules $\leftarrow \mathit{vital}(A) \wedge \mathsf{not}\, A$ and $\leftarrow \mathit{source}(A) \wedge \mathsf{not}\, A$. For approach A1, this does not lead to any problems, even if we drop the B2–rules. Observe, that the resulting disjunctive logic program is stratifiable, which means that simpler concepts of evaluation, namely the *perfect models*, could be used.

4. Efficiency of the Approaches

A large part of our work was on the theoretical side, to make the ASP approach correctly work for solving the drug design problem. ASP can efficiently solve large problems using suitable built–in heuristics. But the modelling of the problem in ASP can only use the concepts provided by ASP: most fundamental is the minimization on the set of derived atoms in a stable model; additionally, it is possible to minimize w.r.t. certain cost functions depending on weights assigned to the derived atoms in a stable model.

ASP is not a general purpose programming language. It is not suitable for all kinds of problems. But, if a problem can be modelled in ASP, then often the heuristics implemented in ASP can efficiently compute solutions. Still, there is some flexibility in modelling a problem is ASP, and some modellings are more suitable than others. In this paper, we have proposed and discussed two alternative approaches to correctly model the drug design problem.

In a comparison of the runtimes of the two methods on a desktop computer, one of the approaches was sufficiently efficient for the real world pathways that we wanted to handle. Often, the runtime of a graph problem mainly depends on the size of the problem; here, this would be the number of nodes and edges of the pathway. However, it turned out, that both approaches for the drug design problem heavily depend on the structure of the pathways. This dependency is hard to characterize. E.g., the runtime heavily depends on the distance between the source nodes of the network, the nodes to be blocked, and the vital nodes, that should not be blocked. These data are not availabe in the KEGG database; for every pathway, the user has to select these source, target and vital nodes depending on the biochemical question in mind. Thus, it is difficult to make a comprehensive analysis of a large collection of pathways, and we could only analyse and try to understand the behaviour (efficiency) of the approaches for selected examples of pathways.

We have investigated connections between the number of solutions, the number of nodes in the pathway, the distance of the sources and targets, and the number of cycles in the pathway. Approach A2 is considerably faster, when there exists only a small number of solutions. It gets slower, when more solutions exist. Approach A1 seems to heavily depend on the number of iterations. We found large pathways (with about 200 reactions) in KEGG with very many solutions, for which approach A2 failed due to limitations of main memory, since very many stable models have been produced simultaneously, before checking reachability. Approach A1, however, always terminated successfully within 30 seconds.

5. Application to KEGG Pathways

ASP rapidly calculates the effect of enzyme deletions including provision of a list of essential and dispensable metabolites for a given metabolic net. In the following, we list some other tasks.

Optimal drug targets can be chosen, i.e., those proteins which should best be targeted by a specific drug. In general, the strategy is here to identify at least one essential metabo-

lite; e.g., antibiotics block the further growth of a pathogen. ASP identifies the optimal enzyme to hit to achieve removal of this metabolite(s). In many medical applications, an opposite strategy should modulate a protein (often a receptor) to ameliorate a pathogenic condition with a *minimum of side–effects*. ASP identifies optimal targets for such modulation without preventing the production of essential metabolites. Furthermore, increased yield in biotechnology, insertion of new pathways, and removal of xenobiotics can be calculated: Each problem is translated into a list of metabolites which should be changed (produced, detoxified, etc.) and others not (e.g., essential core metabolites). With ASP we can calculate optimal enzyme sets for these tasks (yield, production, detoxification).

A number of *alternative approaches* exist including topology analysis of metabolic networks and flux–based approaches calculating the metabolic flux carried by different paths in a given network. ASP extends *structural approaches* (routes from and to a metabolite) by logical connectivity and rules for problem solving. In *metabolic flux calculations*, often several enzymes may be used resulting in an exponential increase of the number of potential paths (combinatorial explosion) leading to calculation time problems, in particular regarding genome–scale networks of more than 200–300 enzymes. However, in ASP the exponential scaling of complexity is only the worst case scenario; for most metabolic networks investigated (e.g., central metabolism of carbohydrates and amino acids), there was power law scaling.

Further applications include essential and non–essential genes in genetics. ASP could *predict essential* and *non–essential genes* for metabolism, however, with strong savings in calculation time compared to metabolic flux calculations [5]. ASP – as outlined above – may also allow new and quick predictions regarding regulatory networks, for instance in proliferation (kinase cascades) and regarding apoptosis and clotting (protease cascades).

Future work will further examine the high potential of our approach in comparison to other metabolic modelling approaches.

6. Conclusions

Our approach results in a fast system for computing alternative drug targets. The system can also be extended flexibly: further conditions which the drug has to fullfil to be applicable for use in therapy might be incorporated. Moreover, the optimization criterion can also be changed: e.g., it is possible to minimize w.r.t. any weighting of the blocked compounds and enzymes – so far, we are using a weight function assigning 0 to compounds and 1 to enzymes.

In the future, we are planning to incorporate additional features into our approach, such as the strenght of the flow through the metabolic network. We hope that the ASP approach is flexible enough to allow such extensions. As we pointed out in the introduction, declarative programming with ASP often allows for adapting the program to the changed needs, and thus can avoid a complete reimplementation, which would be necessary for traditional programming languages. In more complex situations, in particular if we want

to consider regulation or more detailed kinetics, we expect grid computing to be quite helpful in combination with ASP.

ASP is a structural approach including rules and in this way a promising tool to bridge between extensive dynamic approaches of network modelling and those more focussed on structural modelling. Implementation schemes as given here underline the high potential of this approach both for metabolic and for regulatory networks.

References

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

[2] C. Baral, K. Chancellor, N. Tran, N.L. Tran, A. Joy, M. Berens: *A Knowledge Based Approach for Representing and Reasoning about Signaling Networks*. Bioinformatics, 2004, vol. 20 (suppl. 1), pp. 15–22.

[3] J. Behre, L.F. de Figueiredo, S. Schuster, C. Kaleta: *Detecting Structural Invariants in Biological Reaction Networks*. Methods in Molecular Biology, 2012, vol. 804 (3), pp. 377–407.

[4] I. Bratko: PROLOG – *Programming for Artificial Intelligence*. 3rd Ed., Addison–Wesley, 2001.

[5] R.L. Chang, L. Ghamsari, A. Manichaikul, E.F. Hom, S. Balaji, W. Fu, Y. Shen, T. Hao, B. Palsson, K. Salehi–Ashtiani, J.A. Papin: *Metabolic Network Reconstruction of Chlamydomonas Offers Insight Into Light–Driven Algal Metabolism*. Molecular Systems Biology, 2011, vol. 7, article 518.

[6] W. F. Clocksin, C. S. Mellish: *Programming in* PROLOG . 5th Ed., Springer, 2003.

[7] L.F. de Figueiredo, S. Schuster, C. Kaleta, D.A. Fell: *Can Sugars be Produced From Fatty Acids ? A Test Case for Pathway Analysis Tools*. Bioinf., 2009, vol. 25 (1), pp. 152–158.

[8] F. Fages, S. Soliman, N. Chabrier: *Modelling and Querying Interaction Networks in the Biochemical Abstract Machine Biocham*. Journal of Biological Physics and Chemistry, 2004, vol. 4, pp. 64–73.

[9] Kyoto Encyclopedia of Genes and Genomes (KEGG), http://www.genome.jp/kegg.

[10] N. Leone, G. Pfeifer, W. Faber, T. Eiter, G. Gottlob, S. Perri, F. Scarcello: *The DLV System for Knowledge Representation and Reasoning*. ACM Transactions on Computational Logic (TOCL), vol. 7 (3), ACM, 2006.

[11] U. Madsen, P. Krogsgaard–Larsen, T. Liljefors: *Textbook of Drug Design and Discovery.* Taylor and Francis, Washington, DC, 2002.

[12] J.A. Papin, J. Stelling, N.D. Price, S. Klamt, S. Schuster, B. Palsson: *Comparison of Network–Based Pathway Analysis Methods*. Trends in Biotechnology, 2004, vol. 22 (8), pp. 400–405.

[13] R. Schwarz, C. Liang, C. Kaleta, M. Kuhnel, E. Hoffmann, S. Kuznetsov, M. Hecker, G. Griffith, S. Schuster, T. Dandekar: *Integrated Network Reconstruction, Visualization and Analysis using YANAsquare*. BMC Bioinformatics, vol. 8, article 313.