Modelling Biological Networks by Action Languages Via Answer Set Programming

Susanne Grell[1], Torsten Schaub[1,⋆], and Joachim Selbig[1,2]

[1] Institut für Informatik, Universität Potsdam, Postfach 900327, D-14439 Potsdam, Germany
[2] Institut für Biologie, Universität Potsdam, Postfach 900327, D-14439 Potsdam, Germany

Abstract. We describe an approach to modelling biological networks by action languages via answer set programming. To this end, we propose an action language for modelling biological networks, building on previous work by Baral et al. We introduce its syntax and semantics along with a translation into answer set programming. Finally, we describe one of its applications, namely, the sulfur starvation response-pathway of the model plant Arabidopsis thaliana and sketch the functionality of our system and its usage.

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 data than was procurable before. For turning such huge amounts of data into knowledge, one needs appropriate and powerful knowledge representation tools that allow for modelling complex biological systems and their behaviour. Of particular interest are qualitative tools that allow for dealing with biological and biochemical networks. Since these networks are very large, a biologist can manually deal with a small part of it at once. Among the more traditional qualitative formalisms, we find e.g. Petri Nets [1,2], Flux Balance Analysis [3] or Boolean Networks [4]. As detailed in [5], these approaches lack sufficiently expressive reasoning capacities.

Groundbreaking work addressing this deficiency was recently done by Chitta Baral and colleagues who developed a first *action language* for representing and reasoning about biological networks [6,5]. Action languages were introduced in the 1990s by Gelfond and Lifschitz (cf. [7]). By now, there exists a large variety of action languages, like the most basic language $\mathcal{A}$ and its extensions [8] as well as more expressive action languages like $\mathcal{C}$ [9] or $\mathcal{K}$ [10]. Traditionally, action languages are designed for applications in autonomous agents, planning, diagnosis, etc, in which the explicit applicability of actions plays a dominant role. This is slightly different in biological systems where *reactions* are a major concern. For instance, while an agent usually has the choice to execute an action or not, a biological reaction is often simply triggered by its application conditions. This is addressed in [5] by proposing *trigger* and *inhibition rules* as an addition to the basic action language $\mathcal{A}$; the resulting language is referred to as $\mathcal{A}_T^0$. A further extension, allowing knowledge about event ordering, is introduced in [11].

⋆ Affiliated with the School of Computing Science at Simon Fraser University, Burnaby, Canada.

S. Etalle and M. Truszczyński (Eds.): ICLP 2006, LNCS 4079, pp. 285–299, 2006.

The advantages of action languages for modelling biological systems are manifold:

- We get a simplified model. It is not necessary to have any kinetic parameters. The approach can thus already be used in a very early state to verify whether the proposed model of the biological system can or cannot hold.
- Different kinds of reasoning can be used to plan and support experiments. This helps to reduce the number of expensive experiments.
- Further reasoning modes allow for prediction of consequences and explanation of observations.
- The usage of static causal laws allows to easily include background knowledge like environmental conditions, which play an important role for the development of a biological system but are usually difficult to include in the model.
- The approach is elaboration tolerant because it allows to easily extend the model without requiring to change the rest of the model.

We start by introducing our action language $\mathcal{C}_{TAID}$ by building on language $\mathcal{A}_T^0$ [5] and $\mathcal{C}$ [9]. $\mathcal{C}_{TAID}$ extends $\mathcal{C}$ by adding biologically relevant concepts from $\mathcal{A}_T^0$ such as triggers and it augments $\mathcal{A}_T^0$ by providing static causal laws for modelling background knowledge. Moreover, fluents are no longer inertial by definition and the concurrent execution of actions can be restricted. A feature distinguishing $\mathcal{C}_{TAID}$ from its predecessors is its concept of *allowance*, which was motivated by our biological applications. The corresponding *allowance rules* let us express that an action can occur under certain conditions but does not have to occur. In fact, biological systems are characterised by a high degree of incomplete knowledge about the dependencies among different component and the actual reasons for their interaction. If the dependencies are well understood, they can be expressed using triggering rules. However, if the dependencies are only partly known or not part of the model, e.g. environmental conditions, they cannot be expressed appropriately using triggering rules. The concept of allowance permits actions to take place or not, as long as they are allowed (and not inhibited). This introduces a certain non-determinism that is used to model alternative paths, actions for which the preconditions are not yet fully understood, and low reaction rates. Of course, such a non-deterministic construct increases the number of solutions. However, this is a desired feature since we pursue an exploratory approach to bioinformatics that allows the biologist to browse through the possible models of its application.

We introduce the syntax and semantics of $\mathcal{C}_{TAID}$ and give a soundness and completeness result, proved in [12]. For implementing $\mathcal{C}_{TAID}$, we compile specifications in $\mathcal{C}_{TAID}$ into logic programs under answer set semantics [13]. This has been implemented in Java and was used meanwhile in ten different application scenarios at the Max-Planck Institute for Molecular Plant Physiology for modelling metabolic as well as signal transduction networks. Among them we present the smallest application, namely the sulfur starvation response-pathway of the model plant Arabidopsis thaliana.

2 Action Language $\mathcal{C}_{TAID}$

The alphabet of our action language $\mathcal{C}_{TAID}$ consists of two nonempty disjoint sets of symbols: a set of *actions* A and a set of *fluents* F. Informally, fluents describe chang-

ing properties of a world and actions can influence fluents. We deal with propositional fluents that are either *true* or *false*. A *fluent literal* is a fluent f possibly preceded by $\neg$.

We distinguish three sublanguages of $\mathcal{C}_{TAID}$: The *action description language* is used to describe the general knowledge about the system, the *action observation language* is used to express knowledge about particular points of time and the *action query language* is used to reason about the described system.

Action Description Language. To begin with, we fix the syntax of $\mathcal{C}_{TAID}$'s action description language:

Definition 1. *A domain description $D(A, F)$ in $\mathcal{C}_{TAID}$ consists of expressions of the following form:*

$$(a \textbf{ causes } f_1, \ldots, f_n \textbf{ if } g_1, \ldots, g_m) \quad (1)$$
$$(f_1, \ldots, f_n \textbf{ if } g_1, \ldots, g_m) \quad (2)$$
$$(f_1, \ldots, f_n \textbf{ triggers } a) \quad (3)$$
$$(f_1, \ldots, f_n \textbf{ allows } a) \quad (4)$$
$$(f_1, \ldots, f_n \textbf{ inhibits } a) \quad (5)$$
$$(\textbf{noconcurrency } a_1, \ldots, a_n) \quad (6)$$
$$(\textbf{default } f) \quad (7)$$

where $a, a_1, \ldots, a_n$ are actions and $f, f_1, \ldots, f_n, g_1, \ldots, g_m$ are fluent literals.

Note that $\mathcal{A}_T^0$ consists of expressions of form (1), (3), and (5) only.

A *dynamic causal law* is a rule of form (1), stating that $f_1, \ldots, f_n$ hold after the occurrence of action a if $g_1, \ldots, g_m$ hold when a occurs. If there are no preconditions of the form $g_1, \ldots, g_m$, the if-part can be omitted. Rule (2) is a *static causal law*, used to express immediate dependencies between fluents; it guarantees that $f_1, \ldots, f_n$ hold whenever $g_1, \ldots, g_m$ hold. Rules (3) to (6) can be used to express whether and when an action can or cannot occur. A *triggering rule* (3) is used to state that action a occurs immediately if the preconditions $f_1, \ldots, f_n$ hold, unless it is inhibited. An *allowance rule* of form (4) states that action a can but need not occur if the preconditions $f_1, \ldots, f_n$ hold. An action for which triggering or allowance rules are specified can only occur if one of its triggering or allowance rules, resp., is satisfied. An *inhibition rule* of form (5) can be used to express that action a cannot occur if $f_1, \ldots, f_n$ hold. A rule of the form (6) is a no-concurrency constraint. Actions included in such a constraint cannot occur at the same time. Rule (7) is a *default rule*, which is used to define a default value for a fluent. This makes us distinguish two kinds of fluents: inertial and non-inertial fluents. Inertial fluent change their value only if they are affected by dynamic or static causal laws. Non-inertial fluents on the other hand have the value, specified by a default rule, unless they are affected by a dynamic or static causal law. Every fluent that has no default value is regarded to be inertial. Additionally, we distinguish three groups of actions depending on the rules defined for them. An action can either be a triggered, an allowed or an exogenous action. That means, for one action there can be several triggering or several allowance rules but not both.

As usual, the semantics of a domain description $D(A, F)$ is defined in terms of transition systems. An *interpretation* I of F is a complete and consistent set of fluents.

Definition 2 (State). *A state $s \in S$ of the domain description $D(A, F)$ is an interpretation of F such that for every static causal law $(f_1, \ldots, f_n \textbf{ if } g_1, \ldots, g_n) \in D(A, F)$, we have $\{f_1, \ldots, f_n\} \subseteq s$ whenever $\{g_1, \ldots, g_n\} \subseteq s$.*

Hence, we are only interested in sets of fluents satisfying all static causal laws, i.e. correctly model the dependencies between the fluents.

Depending on the state, it is possible to decide which actions can or cannot occur. Therefore we define the notion of active, passive and applicable rules.

Definition 3. *Let $D(A, F)$ be a domain description and s a state of $D(A, F)$.*

1. *An inhibition rule ($f_1, \ldots, f_n$ **inhibits** a) is active in s, if $s \models f_1 \wedge \ldots \wedge f_n$, otherwise the inhibition rule is passive. The set $A_I(s)$ is the set of actions for which there exists at least one active inhibition rule in s.*
2. *A triggering rule ($f_1, \ldots, f_n$ **triggers** a) is active in s, if $s \models f_1 \wedge \ldots \wedge f_n$ and all inhibition rules of action a are passive in s, otherwise the triggering rule is passive in s. The set $A_T(s)$ is the set of actions for which there exists at least one active triggering rule in s. The set $\overline{A}_T(s)$ is the set of actions for which there exists at least one triggering rule and all triggering rules are passive in s.*
3. *An allowance rule ($f_1, \ldots, f_n$ **allows** a) is active in s, if $s \models f_1 \wedge \ldots \wedge f_n$ and all inhibition rules of action a are passive in s, otherwise the allowance rule is passive in s. The set $A_A(s)$ is the set of actions for which there exists at least one active allowance rule in s. The set $\overline{A}_A(s)$ is the set of actions for which there exists at least one allowance rule and all allowance rules are passive in s.*
4. *A dynamic causal law (a **causes** $f_1, \ldots, f_n$ **if** $g_1, \ldots, g_n$) is applicable in s, if $s \models g_1 \wedge \ldots \wedge g_n$.*
5. *A static causal law ($f_1, \ldots, f_n$ **if** $g_1, \ldots, g_n$) is applicable in s, if $s \models g_1 \wedge \ldots \wedge g_n$.*

Observe that point two and three of the definition express that an action has to occur or may occur as long as there is one active triggering or allowance rule respectively. An action cannot occur if either an inhibition rule for the action is active or if all triggering or allowance rules for the action are passive.

The effects of an action are determined by the applicable dynamic causal laws defined for this action. Following [8], the effects of an action a in a state s of domain description $D(A, F)$ are defined as follows:

$$E(a, s) = \{f_1, \ldots, f_n \mid (a \textbf{ causes } f_1, \ldots, f_n \textbf{ if } g_1, \ldots, g_m) \text{ is applicable in } s\}$$

The effects of a set of actions A is defined as the union of the effects of the single actions: $E(A, s) = \bigcup_{a \in A} E(a, s)$. Besides the direct effects of actions, a domain description also defines the consequences of static relationships between fluents. For a set of static causal laws in a domain description $D(A, F)$ and a state s, the set

$$L(s) = \{f_1, \ldots, f_n \mid (f_1, \ldots, f_n \textbf{ if } g_1, \ldots, g_m) \text{ is applicable in } s\}$$

contains the heads of all static causal laws whose preconditions hold in s.

Finally, the way the world evolves according to a domain description is captured by a *transition relation*; it defines to which state the execution of a set of actions leads.

Definition 4. *Let $D(A, F)$ be a domain description and S be the set of states of $D(A, F)$. Then, the transition relation $\Phi \subseteq S \times 2^A \times S$ determines the resulting state $s' \in S$ after executing all actions $B \subseteq A$ in state $s \in S$ as follows:*

$$(s, B, s') \in \Phi \textit{ for } \quad s' = \{(s \cap s') \cup E(B, s) \cup L(s') \cup \Delta(s')\}$$

$$\textit{where} \qquad \Delta(s') = \quad \{\ f \mid (\textbf{default} \quad f) \in D(A, F), \neg f \notin E(B, s) \cup L(s')\}$$
$$\cup \{\neg f \mid (\textbf{default}\ \neg f) \in D(A, F), \quad f \notin E(B, s) \cup L(s')\}$$

Even if no actions are performed, there can nevertheless be a change of state due to the default values defined by the domain description. Intuitively, if actions occur, the next state is determined by taking all effects of the applicable dynamic and static causal laws and adding the default values of fluents not affected by these actions. The values of all fluents that are not affected by these actions or by default values remain unchanged.

The transition relation determines the resulting state when an action is executed, but it cannot be used to decide whether the action happens at all, since it does not consider triggering, allowance or inhibition rules. This is accomplished by the concept of a *trajectory*, which is a sequence of states and actions that takes all rules in the domain description into account.

Definition 5 (Trajectory). *Let $D(A, F)$ be a domain description.*

A trajectory $s_0, A_1, s_1, \ldots, A_n, s_n$ of $D(A, F)$ is a sequence of actions $A_i \subseteq A$ and states s_i satisfying the following conditions for $0 \leq i < n$:

1. $(s_i, A, s_{i+1}) \in \Phi$
2. $A_T(s_i) \subseteq A_{i+1}$
3. $\overline{A}_T(s_i) \cap A_{i+1} = \emptyset$
4. $\overline{A}_A(s_i) \cap A_{i+1} = \emptyset$
5. $A_I(s_i) \cap A_{i+1} = \emptyset$
6. $|A_i \cap B| \leq 1$
   for all $(\textbf{noconcurrency}\ B) \in D(A, F)$.

A trajectory assures that there is a reason why an action occurs or why it does not occur. The second and third point of the definition make sure that the actions of all active triggering rules are included in the set of actions and that no action for which all triggering rules are passive is included in the set of actions. Point four and five assure that no actions for which all allowance rules are passive and no inhibited actions are included in the set of actions. [1] The definition does not include assertions about the active allowance rules, because they can be, but not necessarily have to be, included in the set of actions. (As detailed above, this is motivated by our biological application.) Point two to four imply that for an action there can either be only triggering rules or only allowance rules defined. The last point of the definition assures that all no-concurrency constraints are correctly applied.

Action Observation Language. The action observation language provides expressions to describe particular states and occurrences of actions:

$$(f\ \textbf{at}\ t_i) \qquad\qquad (a\ \textbf{occurs\_at}\ t_i) \tag{8}$$

where f is a fluent literal, a is an action and t_i is a point of time. The initial point of time is t_0. For a set of actions $A' = \{a_1, \ldots, a_k\}$ we write *(A'* **occurs_at** *t_i)* to abbreviate *(a_1* **occurs_at** *t_i), ..., (a_k* **occurs_at** *t_i)*. Intuitively, an expression of form $(f\ \textbf{at}\ t_i)$ is

[1] Allowance rules can be rewritten as inhibition rules, if the corresponding action is declared to be exogenous. But this is inadequate in view of our biological application and results in a significant blow-up in the number of rules obtained after compilation.

used to state that a fluent f is *true* or present at time t_i. If the fluent f is preceded by $\neg$ it states that f is *false* at t_i. An observation of form $(a \ \mathbf{occurs\_at} \ t_i)$ says that action a occurs at time t_i. It is possible that action a is preceded by $\neg$ to express that a does not occur at time t_i.

A domain description specifies how the system can evolve over time. By including observations the possibilities of this evolution are restricted. So only when all information, the domain description and the observations, is taken into account, we get an appropriate picture of the world. The combination of domain description and observations is called an *action theory*.

Definition 6 (Action theory). *Let D be a domain description and O be a set of observations. The pair (D, O) is called an action theory.*

Intuitively, trajectories specify possible evolutions of the system with respect to the given domain description. However, not all trajectories satisfy the observations given by an action theory. Trajectories satisfying both, the domain description as well as given observations, are called *trajectory models*:

Definition 7 (Trajectory model). *Let (D, O) be an action theory.*

A trajectory $s_0, A_1, s_1, A_2, \ldots, A_n, s_n$ of D is a trajectory model of (D, O), if it satisfies all observations in O in the following way:

- *if (f* **at** *t) $\in O$, then $f \in s_t$*
- *if (a* **occurs_at** *t) $\in O$, then $a \in A_{t+1}$.*

The problem that arises here is to find biologically meaningful models. Obviously, such trajectory models often include redundant information, but since this is a common phenomena of biological systems it is not possible to simply exclude such trajectory models. Often, only the minimal trajectories are considered to be of interest, but this is not appropriate for biological systems, since we are not only interested in the shortest path through the transition system, but also in, possibly longer, alternative paths and just as well in models which include the concurrent execution of actions. To decide which actions are redundant is thus a rather difficult problem and the question whether a model is biologically meaningful can only be answered by a biologist, not by an automated reasoner. One way to include additional information which may be derived from data on measurement could be the use of preferences, which is subject to future work.

A question we can already answer is that abut logical consequence of observations.

Definition 8. *Let (D, O) be an action theory. Then,*

- *(D, O) entails fluent observation $(f \ \mathbf{at} \ t_i)$, written $(D, O) \models (f \ \mathbf{at} \ t_i)$, if $f \in s_i$ for all trajectory models $s_0, A_1, \ldots, s_i, A_{i+1}, \ldots, A_n, s_n$ of (D, O),*
- *(D, O) entails action observation $(a \ \mathbf{occurs\_at} \ t_i)$, written $(D, O) \models (a \ \mathbf{occurs\_at} \ t_i)$, if $a \in A_{i+1}$ for all trajectory models $s_0, A_1, \ldots, s_i, A_{i+1}, \ldots, A_n, s_n$ of (D, O).*

Action Query Language. Queries are about the evolution of the biological system, i.e. about trajectories. In general, a query is of the form:

$$(f_1, \ldots, f_n \ \mathbf{after} \ A_1 \ \mathbf{occurs\_at} \ t_1, \ldots, A_m \ \mathbf{occurs\_at} \ t_m) \tag{9}$$

where $f_1, ..., f_n$ are fluent literals, $A_1, ..., A_m$ sets of actions, and $t_1, ..., t_m$ time points.

For queries the most prominent question is the notion of logical consequence. Under which circumstances entails an action theory or a single trajectory model a query.

Definition 9. *Let (D, O) be an action theory and Q be a query of form (9).*[2] *Then,*

- *Q is cautiously entailed by (D, O), written $(D, O) \models_c Q$, if every trajectory model $s_0, A'_1, s_1, A'_2, \ldots, A'_p, s_p$ of (D, O) satisfies $A_i \subseteq A'_i$ for $0 < i \leq m \leq p$ and $s_p \models f_1 \wedge \ldots \wedge f_n$.*
- *Q is bravely entailed by (D, O), written $(D, O) \models_b Q$, if some trajectory model $s_0, A'_1, s_1, A'_2, \ldots, A'_p, s_p$ of (D, O) satisfies $A_i \subseteq A'_i$ for $0 < i \leq m \leq p$ and $s_p \models f_1 \wedge \ldots \wedge f_n$.*

While cautiously entailed queries are supported by all models, bravely entailed queries can be used for checking the possible hypotheses.

We want to use the knowledge given as an action theory to reason about the corresponding biological system. Reasoning includes explaining observed behaviour, but also predicting the future development of the system or how the system may be influenced in a particular way. The above notion of entailment is used to verify the different queries introduced in the next sections.

Planning. In planning, we try to find possibilities to influence a system in a certain way. Neither the initial state nor the goal state have to be completely specified by fluent observations. A plan is thus a sequence of actions starting from one possible initial state and ending at one possible goal state. There are usually several plans, taking into account different paths but also different initial and goal states.

Definition 10 (Plan). *Let (D, O_{init}) be an a action theory such that O_{init} contains only fluent observations about the initial state and let Q be a query of form (9).*

If $(D, O_{init}) \models_b Q$, then $P = \{(A_1 \textbf{ occurs\_at } t_1), \ldots, (A_m \textbf{ occurs\_at } t_m)\}$ is a plan for $f_1, \ldots, f_n$.

Note that a plan is always derived from the corresponding trajectory model.

Explanation. Usually, there are not only observations about the initial state but also about other time points and we are more interested in understanding the observed behaviour of a system than in finding a plan to cause certain behaviour of the system.

Definition 11 (Explanation). *Let (D, O) be an action theory and let Q be a query of form (9) where $f_1 \wedge \ldots \wedge f_n \equiv true$.*

If $(D, O) \models_b Q$, then $E = \{(A_1 \textbf{ occurs\_at } t_1), \ldots, (A_m \textbf{ occurs\_at } t_m)\}$ is an explanation for the set of observations O.

When explaining observed behaviour it is neither necessary to completely define the initial state, nor the final state. The less information is provided the more possible explanation there are, because an explanation is one path from one possible initial state to one possible final state, via some possible intermediate partially defined states given by the observations. The initial state and the explanation are induced by the underlying trajectory model.

[2] Parameters m and n are taken as defined in (9).

Prediction is mainly used to determine the influence of actions on the system; it tries to answer questions about the possible evolution of the system. A query answers the question whether, starting at the current state and executing a given sequence of actions, fluents will hold or not hold after a certain time.

Definition 12 (Prediction). *Let (D, O) be an action theory and let Q be a query of form (9).*

- *If $(D, O) \models_c Q$, then $f_1, \ldots, f_n$ are cautiously predicted,*
- *If $(D, O) \models_b Q$, then $f_1, \ldots, f_n$ are bravely predicted.*

All of the above reasoning modes are implemented in our tool and used in our biological applications. Before describing its usage, we first detail how it is implemented.

3 Compilation

We implemented our action language by means of a compiler mapping $\mathcal{C}_{TAID}$ onto logic programs under *answer set semantics* (cf. [14,13]). This semantics associates with a logic program a set of distinguished models, referred to as *answer sets*. This model-based approach to logic programming is different from the traditional one, like Prolog, insofar as solutions are read off issuing answer sets rather than proofs of posed queries. Our compiler uses efficient off-the-self answer set solvers as a back-end, whose purpose is to compute answer sets from the result of our compilation. Since we do not elaborate upon theoretical aspects of this, we refer the reader to the literature for a formal introduction to answer set programming (cf. [14,13]).

Our translation builds upon and extends the one in [6]. We adapt the translation of the language $\mathcal{A}_T^0$ to include new language constructs and we extend the compilation of $\mathcal{A}_T^0$ in order to capture the semantics of static causal laws, allowance and default rules, and of no-concurrency constraints. In what follows, we stick to the syntax of the `smodels` system [15].

Action Description Language. The expressions defined in a domain description $D(A, F)$ have to be composed of symbols from A an F. When constructing the logic program for $D(A, F)$, we first have to define the alphabet. We declare every fluent $f \in F$ and action $a \in A$, resp., by adding a fact of the form `fluent(f)`, and `action(a)`. We use continuously a variable `T`, representing a time point where $0 \leq$ `T`$\leq t_{max}$. This range is encoded by the `smodels` construct `time(0..`t_{max}`)`, standing for the facts `time(0)`,...,`time(`t_{max}`)`. Furthermore, it is necessary to add constraints expressing that f and $\neg f$ are contradictory.

```
:- holds(f,T), holds(neg(f),T), fluent(f), time(T).
```

Whenever clear from the context, we only give translations for positive fluent literals $f \in F$ and omit the dual rule for the negative fluent, viz. $\neg f$ represented as `neg(f)`.

For each inertial fluent $f \in F$, we include rules expressing that f has the same value at t_{i+1} as at t_i, unless it is known otherwise:

```
holds(f,T+1) :- holds(f,T),not holds(neg(f,T+1)),not default(f),
   fluent(f),time(T),time(T+1).
```

For each non-inertial fluent $f \in F$, we add the fact `default(f)` and include for the default value $true$:

```
holds(f,T) :- not holds(neg(f),T), fluent(f), time(T).
```

For each dynamic causal law (1) in $D(A, F)$ and each fluent $f_i \in F$, we include:

```
holds(fi,T+1) :- holds(occurs(a),T),holds(g1,T),...,holds(gn,T),
   fluent(g1),...,fluent(gn),fluent(fi),action(a),time(T;T+1).
```

For each static causal law (2) in $D(A, F)$ and each fluent $f_i \in F$, we include:

```
holds(fi,T) :- holds(g1,T),...,holds(gn,T),
   fluent(g1), ..., fluent(gn),fluent(fi), time(T).
```

Every triggering rule (3) in $D(A, F)$ is translated as:

```
holds(occurs(a),T) :- not holds(ab(occurs(a)),T),
   holds(f1,T),...,holds(fn,T),
   fluent(f1),...,fluent(fn),action(a),time(T).
```

For each allowance rule (4) in $D(A, F)$, we include:

```
holds(allow(occurs(a)),T) :- not holds(ab(occurs(a)),T),
   holds(f1,T),...,holds(fn,T),
   fluent(f1),...,fluent(fn),action(a),time(T).
```

For every exogenous action $a \in A$, the translation includes a rule, stating that this action can always occur.

```
holds(allow(occurs(a)),T) :- action(a), time(T).
```

Every inhibition rule (5) in $D(A, F)$ is translated as:

```
holds(ab(occurs(a)),T) :- holds(f1,T),...,holds(fn,T),
   action(a),fluent(f1),...,fluent(fn), time(T).
```

For each no-concurrency constraint (6) in $D(A, F)$, we include an integrity constraint assuring that at most one of the respective actions can hold at time t:

```
:- time(T), 2 {holds(occurs(a1),T):action(a1),...,
                              holds(occurs(an),T):action(an)}.
```

Action Observation Language. There are two different kinds of fluent observations. Those about the initial state, *(f* **at** t_0*)*, and the fluent observations about all other states, *(f* **at** t_i*)* for $i > 0$. Fluent observations about the initial state are simply translated as facts: `holds(f,0).` Because they are just assumed to be true and need no further justification. All other fluent observations however need a justification. Due to this, fluent observations about all states except the initial state are translated into integrity constraints of the form: `:- not holds(f,T),fluent(f),time(T).`

The initial state can be partially specified by fluent observations. In fact, only the translation of the (initial) fluent observations must be given. All possible completions of the initial state are then generated by adding for every fluent $f \in F$ the rules:

```
holds(f,0):- not holds(neg(f),0).
holds(neg(f),0):- not holds(f,0).
```
(10)

When translating action observations of form (8) the different kinds of actions have to be considered. Exogenous actions can always occur and need no further justification. Such an exogenous action observation is translated as a fact: `holds(occurs(a),T).` Unlike this, observations about triggered or allowed actions must have a reason, e.g. an active triggering or allowance rule, to occur. To assure this justification, the action observation is translated using constraints of the form:

```
:- holds(neg(occurs(a)),T),action(a),time(T).
```

assuring that every answer set must satisfy the observation *(*a **occurs_at** t_i*)*.

Apart from planning (see below), we also have to generate possible combinations of occurrences of actions, for all states. To this effect, the translation includes two rules for every exogenous and allowed action.

```
holds(occurs(a),T) :- holds(allow(occurs(a)),T),
   not holds(ab(occurs(a)),T), not
       holds(neg(occurs(a)),T),
   action(a), time(T), T<t_max.
holds(neg(occurs(a)),T) :- not holds(occurs(a),T),
   action(a), time(T), T<t_max.
```
(11)

Basic correctness and completeness result. The following result provides a basic correctness and completeness result; corresponding results for the specific reasoning modes are either obtained as corollaries or adaptions of its proof.

Theorem 1. *Let* (D, O_{init}) *be an action theory such that* O_{init} *contains only fluent observations about the initial state. Let* Q *be a query as in (9) and let*

$$A_Q = \{(a \textbf{ occurs\_at } t_i) \mid a \in A_i, 1 \leq i \leq m\} .$$

Let $\mathcal{T}$ *denote the translation of* $\mathcal{C}_{TAID}$ *into logic programs, described above.*

Then, we have the following results.

1. *If* $s_0, A_1, s_1, A_2, \ldots, A_m, s_m$ *is a trajectory model of* $(D, O_{init} \cup A_Q)$*, then there is an answer set* X *of logic program* $\mathcal{T}(D, O_{init} \cup A_Q)$ *such that we have for all* $f \in F$ *and* $0 \leq k \leq m$
   (a) `holds(f,k)` $\in X$*, if* $s_k \models f$ *and*
   (b) `holds(neg(f),k)` $\in X$*, if* $s_k \models \neg f$.
2. *If* X *is an answer set of logic program* $\mathcal{T}(D, O_{init} \cup A_Q)$ *and for* $0 \leq k \leq m$

$$s_k = \{f \mid \texttt{holds(f,k)} \in X\} \cup \{\neg f \mid \texttt{holds(neg(f),k)} \in X\}$$

   then there is a trajectory model $s_0, A_1, s_1, A_2, \ldots, A_m, s_m$ *of* $(D, O_{init} \cup A_Q)$.

Action Query Language. In the following t_{max} is the upper time bound, which has to be provided when the answer sets are computed.

Planning. Recall that the initial state can be partially specified; it is then completed by the rules in (10) for taking into account all possible initial states. A plan for $f_1, \ldots, f_n$ (cf. Definition 10) is translated using the predicate "achieved". It ensures that the goal holds in the final state of every answer set for the query.

```
:- not achieved.
achieved :- achieved(0).
achieved :- achieved(T+1),not achieved(T),time(T),time(T+1).
achieved(T) :- holds(f_1,T),...,holds(f_n,T),
   achieved(T+1),fluent(f_1),...,fluent(f_n),time(T),time(T+1).
achieved(n) :- holds(f_1,T),...,holds(f_n,T),
   fluent(f_1),...,fluent(f_n),T = t_max.
```

Constant t_{max} is the maximum number of steps in which the goals $f_1, \ldots, f_n$ should be achieved. The proposition `achieved(T)` represents the earliest point of time `T` at which the plan is successfully achieved. Once the query is satisfied only triggered actions can occur, all other actions should not occur since that might invalidate the plan. That is why `achieved(T)` occurs in the translation of every allowed and exogenous action.

```
holds(occurs(a),T) :- holds(allow(occurs(a)),T),not achieved(T),
   not holds(ab(occurs(a)),T), not holds(neg(occurs(a)),T),
   action(a),time(T).
holds(neg(occurs(a)),T) :- not holds(occurs(a),T),
   action(a),time(T).
```

These rules are used to generate all possible combinations of occurrences of non-triggered actions. Such actions can only occur as long as the goal is not yet achieved and if they are not inhibited. If there is an answer set X for the planning problem, then we have for a plan P (cf. Definition 10) that $(a \textbf{ occurs\_at } t_i) \in P$ if `holds(occurs(a),i)`$\in X$.

Explanation. The translation of an explanation contains the translation of all action and fluent observations in O, as described above. Since the observations about the initial state are often incomplete the translation contains the rules in (10) to generate all initial states which do not contradict the observations. Also, we have to generate possible combinations of occurrences of actions for all states. To this effect, the translation includes for every exogenous and allowed action the rules in (11). If there exists an answer set X for the explanation problem, then for an explanation E as in Definition 11 we have $(a \textbf{ occurs\_at } t_i) \in E$ if `holds(occurs(a),i)`$\in X$.

Prediction. The translation includes all fluent and action observations in O, as described above. As in explanation, we have to fill in missing information, which is necessary to justify the observed behaviour. That means we have to include for every fluent f two rules of form (10) to generate possible initial states. Moreover the translation includes for every non-triggered action two rules similar to those of an explanation of form (11). The actual prediction for $f_1, \ldots, f_n$ (cf. Definition 12) is translated as:

```
predicted :- holds(f_1,T), ..., holds(f_n,T),
   fluent(f_1),...,fluent(f_n),time(T),T >= i.
```

where i is the time of the latest observation. If the atom `predicted` is included in all (some) answer sets, it is a cautious (brave) prediction.

4 Application

Meanwhile, we have used $\mathcal{C}_{TAID}$ in several different application scenarios at the Max-Planck Institute for Molecular Plant Physiology for modelling metabolic as well as

signal transduction networks. For illustration, we describe below the smallest such application, namely the sulfur starvation response-pathway of the model plant Arabidopsis thaliana. Sulfur is essential for the plant. If the amount of sulfur it can access is not sufficient to allow a normal development of the plant, the plant follows a complex strategy. First the plant forms additional lateral roots to access additional sources of sulfur and to normalise its sulfur level. However, if this strategy is not successful the plant uses its remaining resources to form seeds.

Normally, the amount of sulfur in a plant is sufficient, but due to external, e.g. environmental conditions, the amount of sulfur can be reduced. A problem, when modelling this network are such environmental conditions, which are not and cannot be part of a model and which might or might not lead to the reduction of sulfur. Once the level of sulfur in the plant is decreased, complex interactions of different compounds are triggered. Genes are activated, which induce the generation of auxin, a plant hormone, playing a key role as a signal in coordinating the development of the plant. This eventually leads to the formation of additional lateral roots. Since this consumes the scarce resources, the development should be stopped, when it becomes apparent that it is not successful (i.e. it takes too long and consumes too many of the plant's resources). This "emergency stop" is triggered by complex interactions that lead, via a surplus of the auxin flux, to the expression of IAA28, a gene which is subject to current research. If IAA28 is expressed and the sulfur level is still low, other processes result in a different physiological endpoint, the production of seeds.

We now show how this biological network can be represented as a domain description $D(A, F)$ in $\mathcal{C}_{TAID}$.

$$A = \{\textit{sulfur\_depletion}, \textit{sulfur\_repletion}, \textit{enhanced\_lateral\_root\_formation}, \textit{iaa28\_expression}, \textit{rapid\_seed\_production}\}$$
$$F = \{\textit{normal\_sulfur}, \textit{depleted\_sulfur}, \textit{enhanced\_lateral\_roots}, \textit{expressed\_iaa28}, \textit{seeds}\}$$

The biologist's knowledge about the biological system, gives rise to the following dynamic causal laws.

(sulfur_depletion **causes** *depleted_sulfur* **if** *normal_sulfur)*
(enhanced_lateral_root_formation **causes** *enhanced_lateral_roots)*
(sulfur_repletion **causes** *normal_sulfur)*
(iaa28_expression **causes** *expressed_iaa28)*
(rapid_seed_production **causes** *seeds)*

Additionally, two static causal laws specify the relationship between *normal sulfur* and *depleted sulfur*. They assure that at most one of the fluents is $true$ at all times.

(¬normal_sulfur **if** *depleted_sulfur)*
(¬depleted_sulfur **if** *normal_sulfur)*

For two of the actions, we know all the preconditions that have to be satisfied for the actions to occur.

(depleted_sulfur **triggers** *enhanced_lateral_root_formation)*
(expressed_iaa28 , depleted_sulfur **triggers** *rapid_seed_production)*

For the remaining three actions, it is more difficult to decide whether and when they occur. Whether the action *sulfur depletion* occurs depends on environmental conditions being outside the model. The same holds for the action *sulfur repletion*, which might or might not be successful, depending on the environmental conditions. For the occurrence of action *iaa28 expression* the question is not whether it occurs but when it occurs. The longer it is delayed, the more resources are used to form additional lateral roots.

(normal_sulfur **allows** *sulfur_depletion)*
(depleted_sulfur **allows** *iaa28_expression)*
(enhanced_lateral_roots **allows** *sulfur_repletion)*

There is only one inhibition relation in this example.

(expressed_iaa28 **inhibits** *enhanced_root_formation)*

But only if we add a default value for the fluent *enhanced lateral roots*, the inhibition relation has the desired effect of stopping the formation of additional lateral roots.

(**default** $\neg$*enhanced_lateral_roots)*

The knowledge that the plant either forms additional lateral roots or produces seeds can be expressed by the following no-concurrency constraint:

*(***noconcurrency** *enhanced_lateral_roots_formation , rapid_seed_production)*

After defining the domain description, let us define a set of observations O. The initial state, where we still have a normal level of sulfur can be described by the following fluent observations:

$O =$ {*(normal_sulfur* **at** *0), (*$\neg$ *enhanced_lateral_roots* **at** *0),*
($\neg$ *expressed_iaa28* **at** *0), (*$\neg$ *seeds* **at** *0)*}

Now that we defined our action theory (D, O), we can start to reason about it. Let us first find an explanation for the observed behaviour:

$O_1 = O \cup$ {*(sulfur_depletion* **occurs_at** *0), (normal_sulfur* **at** *3)*}

For a time bound of $t_{max} = 3$ there are already 4 possible explanations. They all have in common that *sulfur depletion* occurs at time point 0, the formation of lateral roots is triggered at time point 1 and the action *sulfur repletion* occurs at time point 2. The explanations differ in whether and when the action *iaa28 expression* and the action *rapid seed production* occurs. One explanation is:

(D,O_1*)* $\models_b$ *(true* **after** *sulfur_depletion* **occurs_at** *0,*
enhanced_lateral_root_formation **occurs_at** *1,*
enhanced_lateral_root_formation **occurs_at** *2, sulfur_repletion* **occurs_at** *2)*

A second explanation is:

(D,O_1*)* $\models_b$ *(true* **after** *sulfur_depletion* **occurs_at** *0,*
enhanced_lateral_root_formation **occurs_at** *1,*
enhanced_lateral_root_formation **occurs_at** *2,*
sulfur_repletion **occurs_at** *2, iaa28_expression* **occurs_at** *2)*

Our next question is whether the given observations are sufficient to predict a certain behaviour of the plant.

(D,O)$\models_c$ *(seeds* **after** *sulfur_depletion* **occurs_at** *0, iaa28_expression* **occurs_at** *1)*
(D,O)$\models_b$ *(normal_sulfur* **after** *sulfur_depletion* **occurs_at** *0, iaa28_expression* **occurs_at** *1)*

Using these predictions, we can say that when sulfur is depleted and IAA28 is expressed the plant grows seeds, but it is still possible that it also stabilises its sulfur level.

Finally, we want to find a plan for the action theory (D, O) that results in the production of seeds. For time bound $t_{max} = 3$, there are 4 plans. One possible plan is:

(D,O) $\models_b$ *(seeds* **after** *sulfur_depletion* **occurs_at** *0, iaa28_expression* **occurs_at** *1, enhanced_lateral_root_formation* **occurs_at** *1, rapid_seed_production* **occurs_at** *2, rapid_seed_production* **occurs_at** *3)*

The number of plans and explanations depend on the number of allowance rules, since the different possibilities for the occurrence of such an allowed action is reflected by different answer sets.

5 Discussion

We proposed the action language $\mathcal{C}_{TAID}$ and showed how it can be used to represent and reason about biological networks. $\mathcal{C}_{TAID}$ is based on the action language $\mathcal{A}_T^0$ introduced in [6]. The latter language provides only minimal features to define dynamic causal laws, triggering and inhibition rules, which turn to be a fruitful basis but insufficient for modelling our biological applications. Moreover, our exploratory approach made us propose the concept of allowance that enables the experimenter to investigate alternative models "in silico". As a consequence, we extended $\mathcal{A}_T^0$ by static causal laws, allowance rules, default rules and no-concurrency constraint which furnish a more appropriate representation of our biological networks. Especially static causal laws and default rules can be used to include background knowledge and other dependencies like environmental conditions which influence the biological system, but are not part of the actual model. Allowance rules are mainly used to express incomplete knowledge about the reasons why an action occurs. This missing information is a common problem for biologists due to the immanent complexity of biological systems.

We fixed the semantics of $\mathcal{C}_{TAID}$ in the standard way by means of transition relations, trajectories and trajectory models. In contrast to $\mathcal{A}_T^0$, for example, default values can enable state changes without the occurrence of an action. Also, Baral et al. guarantee a unique trajectory model and a unique answer set, if the initial state is completely defined by a set of observations. This is not the case in $\mathcal{C}_{TAID}$ because of the nondeterminism introduced by allowance rules that may yield multiple answer sets.

We implemented our action language by means of a compiler mapping $\mathcal{C}_{TAID}$ onto logic programs under answer set semantics. Our translation builds upon and extends the one given in [6]. The resulting tool is implemented in Java and freely available at [16]. Meanwhile, it has been used in ten different application scenarios at the Max-Planck Institute for Molecular Plant Physiology for modelling metabolic as well as signal transduction networks. For illustration, we described the smallest such application, namely, a part of the sulfur starvation response-pathway of model plant Arabidopsis thaliana.

Beyond the traditional approaches mentioned in the introductory section, further logic-based approaches using rule-based languages have emerged recently: Closely related work has been conducted in abductive logic programming where abduction was used in [17] as the principal mode of inference for modelling gene relations from microarray data. A very sophisticated and much more advanced automated reasoning tool for systems biology can be found in the area of constraint programming, namely the BIOCHAM [18] system. BIOCHAM relies on CTL [19] and is thus particularly strong in modelling temporal aspects of systems biology. Unlike our abstract approach, the constraint-based approach offers fine-grained capacities for modelling biochemical processes, including kinetics and reactions.

References

1. Reddy, V., Mavrovouniotis, M., Liebman, M.: Petri net representations in metabolic pathways. Proc. First ISMB (1993) 328–336
2. Pinney, J.W., Westhead, D.R., McConkey, G.A.: Petri net representations in systems biology. Biochem Soc Trans **31**(Pt 6) (2003) 1513–1515
3. Bonarius, H.P.J., Schmid, G., Tramper, J.: Flux analysis of underdetermined metabolic networks: The quest for the missing constraints. Trends Biotechnol **15** (1997) 308314
4. Shmulevich, I., Dougherty, E., Kim, S., Zhang, W.: Probabilistic boolean networks: A rule-based uncertainty model for gene regulatory networks. Bioinformatics **18(2)** (2002) 261–274
5. Baral, C., Chancellor, K., Tran, N., Tran, N., Joy, A., Berens, M.: A knowledge based approach for representing and reasoning about signaling networks. In: ISMB. (2004) 15–22
6. Tran, N., Baral, C.: Reasoning about triggered actions in ansprolog and its application to molecular interactions in cells. In: KR. (2004) 554–564
7. Gelfond, M., Lifschitz, V.: Representing action and change by logic programs. Journal of Logic Programming **17** (1993) 301–321
8. Gelfond, M., Lifschitz, V.: Action languages. Electron. Trans. Artif. Intell. **2** (1998) 193–210
9. Giunchiglia, E., Lifschitz, V.: An action language based on causal explanation: preliminary report. In: AAAI/IAAI, AAAI Press (1998) 623–630
10. Eiter, T., Faber, W., Leone, N., Pfeifer, G., Polleres, A.: Planning under incomplete knowledge. In: CL, Springer (2000) 807–821
11. Tran, N., Baral, C., Shankland, C.: Issues in reasoning about interaction networks in cells: Necessity of event ordering knowledge. In: AAAI, AAAI Press (2005) 676–681
12. Grell, S.: Investigation and analysis of new approaches for representing and reasoning about biological networks using action languages. Diploma thesis, University of Potsdam and Max Planck Institute of Molecular Plant Physiology (2006)
13. Baral, C.: Knowledge Representation, Reasoning and Declarative Problem Solving. Cambridge University Press (2003)
14. Gelfond, M., Lifschitz, V.: Classical negation in logic programs and disjunctive databases. New Generation Computing **9** (1991) 365–385
15. Simons, P., Niemelä, I., Soininen, T.: Extending and implementing the stable model semantics. Artificial Intelligence **138**(1-2) (2002) 181–234
16. (http://bioinformatics.mpimp-golm.mpg.de/bionetreasoning/)
17. Papatheodorou, I., Kakas, A.C., Sergot, M.J.: Inference of gene relations from microarray data by abduction. In: LPNMR, Springer (2005) 389–393
18. Chabrier-Rivier, N., Fages, F., Soliman, S.: The biochemical abstract machine biocham. In: CMSB, Springer (2004) 172–191
19. Clarke, E., Grumberg, O., Peled, D.: Model checking. MIT Press (1999)