Steve Dworschak | Susanne Grell | Victoria J. Nikiforova |
Torsten Schaub | Joachim Selbig

Modeling Biological Networks by Action Languages via Answer Set Programming

Suggested citation referring to the original publication:
Constraints 13 (2008) 21–65
DOI https://doi.org/10.1007/s10601-007-9031-y
ISSN (print) 1383-7133
ISSN (online) 1572-9354

Constraints (2008) 13:21–65
DOI 10.1007/s10601-007-9031-y

Modeling Biological Networks by Action Languages via Answer Set Programming

**Steve Dworschak · Susanne Grell ·
Victoria J. Nikiforova · Torsten Schaub ·
Joachim Selbig**

Published online: 25 March 2008

Abstract We describe an approach to modeling biological networks by action languages via answer set programming. To this end, we propose an action language for modeling biological networks, building on previous work by Baral et al. (Proceedings of the twelfth international conference on intelligent systems for molecular biology/third European conference on computational biology (ISMB'04/ECCB'04) (pp. 15–22), 2004). We introduce its syntax and semantics along with a translation into answer set programming, an efficient Boolean Constraint Programming Paradigm. 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.

Keywords Biological network model · Action language · Answer set programming

S. Dworschak (✉) · S. Grell · T. Schaub · J. Selbig
Institut für Informatik, Universität Potsdam,
August-Bebel-Str. 89, 14482 Potsdam, Germany
e-mail: steve@cs.uni-potsdam.de

S. Grell · V. J. Nikiforova · J. Selbig
Max-Planck-Institut für molekulare Pflanzenphysiologie,
14424 Potsdam, Germany

J. Selbig
Institut für Biologie/Biochemie, Universität Potsdam,
Postfach 900327, 14439 Potsdam, Germany

V. J. Nikiforova
Timiryazev Institute of Plant Physiology, Russian Academy of Sciences,
Moscow 127276, Russia

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 modeling complex biological systems and their behavior. To this end, we elaborate upon qualitative methods and tools that allow for dealing with biological and biochemical networks. Since these networks are very large, a biologist can manually only deal with a small part of it at once. Among the more traditional qualitative formalisms, we find e.g. Petri Nets [39, 40], Flux Balance Analysis [5] or Boolean Networks [42]. As detailed in Baral et al. [4], 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 [4, 48]. A comprehensive account of this approach is given in Tran [47]. Action languages were introduced in the 1990s by Gelfond and Lifschitz (cf. [20]) as a declarative syntactical means for describing transition systems expressing causal relationships. By now, there exists a large variety of action languages, like the most basic language $\mathcal{A}$ and its extensions [21] as well as more expressive action languages like $\mathcal{B}$ [21], $\mathcal{C}$ [22] or $\mathcal{K}$ [10, 11], and variations thereof. 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 Baral et al. [4] by proposing *trigger* and *inhibition rules* as an addition to the basic action language $\mathcal{A}$; the resulting language is referred to as $\mathcal{A}^0_T$. A further extension, allowing knowledge about event ordering, is introduced in Tran et al. [49].

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

- We get a simple model. 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 can help 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 formal model.
- The approach is elaboration tolerant[1] because it allows to easily extend a part of the model without requiring to change the rest of it.

[1] As put forward in McCarthy [31]: *"A formalism is elaboration tolerant to the extent that it is convenient to modify a set of facts expressed in the formalism to take into account new phenomena or changed circumstances"*.

We start by introducing our action language $\mathcal{C}_{TAID}$ by building on language $\mathcal{A}_T^0$ [4, 48] and $\mathcal{C}$ [22]. $\mathcal{C}_{TAID}$ extends $\mathcal{C}$ by adding biologically relevant concepts from $\mathcal{A}_T^0$ such as triggers and it augments $\mathcal{A}_T^0$, as defined in Tran and Baral [48], by providing static causal laws for modeling background knowledge.[2] Moreover, fluents (that is, propositions changing their value over time; see below) are no longer inertial by definition and the concurrent execution of actions can be restricted. Besides $\mathcal{C}$, similar features are for instance provided by $\mathcal{B}$ and $\mathcal{K}$.

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 characterized by a high degree of incomplete knowledge about the dependencies among different components 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 slow reactions. 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. For implementing $\mathcal{C}_{TAID}$, we compile specifications in $\mathcal{C}_{TAID}$ into logic programs under *answer set semantics* [2]. This approach, also referred to as answer set programming (ASP) [4], is besides satisfiability checking (SAT), the most popular approach to Boolean constraint solving. Both approaches offer high-performance solvers, which are able to solve problems with millions of variables. The two major differences between them are (i) that ASP is more expressive than SAT[3] and therefore problem representations are more succinct and (ii) that ASP has a rich input language due to its root in knowledge representation. Although our compilation maps specifications into a Boolean setting, our approach is able to deal with multi-valued fluents as well. However, up to now, no such requirement was found in our application scenarios. Hence, we also confine our formal development to the Boolean case.

The overall approach has been implemented in Java and used meanwhile in several different application scenarios at the Max Planck Institute for Molecular Plant Physiology. Here we present the smallest application, namely the sulfur starvation response-pathway of the model plant *Arabidopsis thaliana*.

[2]To be precise, static causal laws were already informally used in Baral et al. [4].

[3]To be precise, ASP cannot be translated into SAT in a modular way, while the inverse is possible [33]. Although SAT as well as ASP are basically NP-complete, a language-preserving translation of ASP into SAT leads to an exponential blow-up in the worst case [26].

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 *action names* A and a set of *fluent names* F.[4] Informally, fluents describe changing 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 sub-languages 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.

2.1 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:

$$(a \textbf{ causes } f_1, \dots, f_n \textbf{ if } g_1, \dots, g_m) \quad (1)$$

$$(f_1, \dots, f_n \textbf{ if } g_1, \dots, g_m) \quad (2)$$

$$(f_1, \dots, f_n \textbf{ triggers } a) \quad (3)$$

$$(f_1, \dots, f_n \textbf{ allows } a) \quad (4)$$

$$(f_1, \dots, f_n \textbf{ inhibits } a) \quad (5)$$

$$(\textbf{noconcurrency } a_1, \dots, a_n) \quad (6)$$

$$(\textbf{default } f) \quad (7)$$

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

Note that $\mathcal{A}_T^0$, as defined in Tran and Baral [48], consists of expressions of form (1), (3), and (5) only.

A *dynamic causal law* is a rule of form (1), stating that $f_1, \dots, f_n$ hold after the occurrence of action a, provided that $g_1, \dots, g_m$ hold when a occurs. If there are no preconditions of the form $g_1, \dots, g_m$, the if-part can be omitted. Rule (2) is a *static causal law*, used to express immediate dependencies between fluents; it guarantees that $f_1, \dots, f_n$ hold whenever $g_1, \dots, g_m$ 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 $f_1, \dots, f_n$ hold, unless it is inhibited. An *allowance rule* of form (4) states that action a can but need not occur if the preconditions $f_1, \dots, f_n$ hold. An action for which either triggering or allowance rules are specified can only occur if one of its triggering or allowance rules, respectively, is satisfied. An *inhibition rule* of form (5) can be used to express that action a cannot occur if $f_1, \dots, f_n$ hold. A rule of the form (6) is a no-concurrency

[4]For simplicity, we use in what follows the terms *action* and *fluent* rather than *action name* and *fluent name*, respectively.

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.

The latter makes us distinguish two kinds of fluents: *inertial* and *non-inertial* fluents. Inertial fluents 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. (See end of this section, for a detailed example.) 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. If there are no allowance or triggering rules declared for an action occurring in the knowledge-base, it is considered to be an exogenous action, being external to the model. Such an exogenous action can occur at all times as long as it is not inhibited. Otherwise, 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 [21]. An *interpretation* I over 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 over 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 $s \models f_1 \wedge \cdots \wedge f_n$, 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 $s \models f_1 \wedge \cdots \wedge f_n$ 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 $s \models f_1 \wedge \cdots \wedge f_n$ 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 $s \models g_1 \wedge \cdots \wedge g_n$.
5. A static causal law $(f_1, \ldots, f_n$ **if** $g_1, \ldots, g_n)$ is applicable in s, if $s \models g_1 \wedge \cdots \wedge g_n$.

Observe that point two and three of the definition express that an action is activated (and thus has to occur) or may become activated as long as there is one active triggering or allowance rule respectively. A non-exogenous 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, respectively.

The effects of an action are determined by the applicable dynamic causal laws defined for this action. Following [21], 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 a resulting state $s' \in S$ after executing all actions $B \subseteq A$ in state $s \in S$ as follows:

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

where

$$\begin{aligned}\Delta(s') = \quad & \{\ \ f \mid (\textbf{default} \quad f) \in D(A, F), \neg f \notin E(B, s) \cup L(s')\} \\ \cup \ & \{\neg f \mid (\textbf{default} \neg f) \in D(A, F), \quad f \notin E(B, s) \cup L(s')\}\end{aligned}$$

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 sets of actions $A_i \subseteq A$ and states s_i of $D(A, F)$ satisfying the following conditions for $0 \leq i < n$:

1. $(s_i, A_{i+1}, s_{i+1}) \in \Phi$
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. $|A_{i+1} \cap B| \leq 1$ for all (**noconcurrency** B) $\in D(A, F)$.

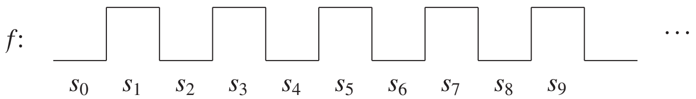
A trajectory assures that there always is a reason why an action occurs or why it does not occur. The second and third point of the definition make sure that the actions of all active triggering rules are included in the set of actions and that no action for which all triggering rules are passive is 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.[5] The definition does not include assertions about active allowance rules or about the occurrence of exogenous actions, 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.) The last point of the definition assures that at most one of the actions occurring in a no-concurrency constraint can occur at each point of time.

For illustrating the interaction of non-inertial fluents with trigger and allowance rules, let us consider the following three domain descriptions.

1. (**default** $\neg f$) ($\neg f$ **triggers** a) (a **causes** f)
   has trajectory model

   $$\{\neg f\},\ \{a\},\ \{f\},\ \emptyset,\ \{\neg f\},\ \{a\},\ \{f\},\ \emptyset,\ \{\neg f\}\ \ldots$$

   The behavior of fluent f in this trajectory model can be visualized as follows:

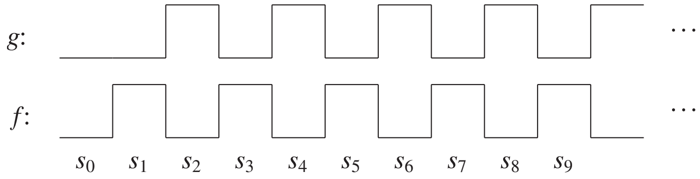
   

   The oscillation of f is caused by the fact that it keeps returning to its default state after each execution of action a.
2. (**default** $\neg f$) ($\neg f$ **triggers** a) (a **causes** f)
   (**default** $\neg g$) (f **triggers** b) (b **causes** g)
   has trajectory model

   $$\{\neg f, \neg g\},\ \{a\},\ \{f, \neg g\}\ \{b\},\ \{\neg f, g\},\ \{a\},\ \{f, \neg g\}\ \{b\},\ \{\neg f, g\}\ \ldots$$

[5] 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 non-modular compilation (see Section 3).

The behavior of fluent f and g in this trajectory model can be visualized as follows:

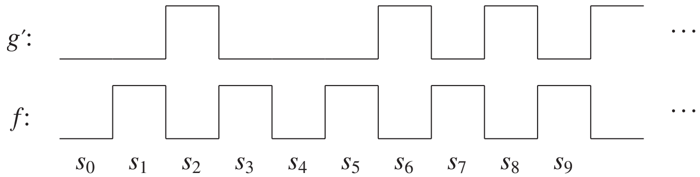


As above, f and g are oscillating, yet in a complementary fashion.

3. (**default** $\neg f$) ($\neg f$ **triggers** a) (a **causes** f)
(**default** $\neg g'$) (f **allows** b) (b **causes** g')
has trajectory model

$$\{\neg f, \neg g'\},\ \{a\},\ \{f, \neg g'\}\ \{b\},\ \{\neg f, g'\},\ \{a\},\ \{f, \neg g'\},\ \emptyset,\ \{\neg f, g'\}\ \ldots$$

The behavior of fluent f and g' in this trajectory model can be visualized as follows:



Unlike b above, b is merely allowed to happen and not automatically triggered; as a result, g' remains false at s_4.

2.2 Action Observation Language

The action observation language provides expressions to describe particular states and occurrences of actions:

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

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 **at** t_i) is 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 **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 underlying system. 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:

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

The problem that arises here is to distinguish biologically meaningful trajectory models. In other domains,[6] often, only certain optimal 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. Moreover, redundancy is a common phenomenon of biological systems and it is hence impossible to simply exclude trajectory models bearing putatively redundant information. So, 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 or objective functions, which are subject to future work.

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

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

- (D, O) entails fluent observation $(f$ **at** $t_i)$, written $(D, O) \models (f$ **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 $(a$ **occurs_at** $t_i)$, written $(D, O) \models (a$ **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).

2.3 Action Query Language

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

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

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

[6]For instance in planning, one is usually interested in shortest or least expensive trajectories.

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).[7] 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 $s_p \models f_1 \wedge \cdots \wedge f_n$.
- 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 $s_p \models f_1 \wedge \cdots \wedge f_n$.

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 behavior, 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 types of queries introduced in the next sections.

2.3.1 Planning

In planning, we try to find possibilities to influence a system in a certain way. Neither the initial state (viz. s_0) nor the goal state (viz. s_n in Definition 5) have to be completely specified by fluent observations. A plan is then 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 action theory such that O_{init} contains only fluent observations about the initial state and let Q be a query of form (9).

If $(D, O_{init}) \models_b Q$ wrt some trajectory model $s_0, A'_1, s_1, A'_2, \ldots, A'_p, s_p$ of (D, O), 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.

2.3.2 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 behavior of a system than in finding a plan to cause certain behavior of the system.

[7] Parameters m and n are taken as defined in (9); the same applies to fluent literals $f_1, \ldots, f_n$, sets of actions $A_1, \ldots, A_m$, and time points $t_1, \ldots, t_m$.

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

If $(D, O) \models_b Q$ wrt some trajectory model $s_0, A'_1, s_1, A'_2, \ldots, A'_p, s_p$ of (D, O), then $E = \{(A'_1$ **occurs_at** $t_1), \ldots, (A'_m$ **occurs_at** $t_m)\}$ is an explanation for the set of observations O.

When explaining observed behavior it is neither necessary to completely define the initial state, nor the final state. The less information is provided the more possible explanations 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.

2.3.3 Prediction

Prediction is mainly used to determine the influence of actions on the system; it tries to answer questions about the development of the biological 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. [2, 19]). 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-shelf answer set solvers like `smodels` [43] or `clasp` [17], respectively, 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 ASP (cf. [2, 19]).

Our translation builds upon and extends the one in Lifschitz and Turner [28] and Tran and Baral [48]. We adapt the translation of the language $\mathcal{A}_T^0$ to include new language constructs and we extend the compilation scheme 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 [43], using lowercase strings for predicate, function, and constant symbols and uppercase strings for variables. A rule is of the form

$$\mathtt{h} :- \mathtt{g}_1, ..., \mathtt{g}_n$$

which means that h is derivable **if** all sub-goals $\mathtt{g}_1, ..., \mathtt{g}_n$ are derivable. Facts have no such goals and are simply denoted by `h.` Integrity constraints have no head on the left, viz.

$$:- \mathtt{g}_1, ..., \mathtt{g}_n$$

meaning that the $\mathtt{g}_1, ..., \mathtt{g}_n$ cannot jointly hold. Furthermore, we (once) make use of `smodels`'s basic cardinality constraints, having the form $k\ \{l_1, \ldots, l_m : t_1, \ldots, t_n\}$ and meaning that at least k literals among $\{l_1, \ldots, l_m\}$ must be contained in an answer set; the remaining literals $t_1, \ldots, t_n$ are used for restricting the instantiation of variables in $l_1, \ldots, l_m$.

3.1 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 $a \in A$, respectively, by adding a fact of the form `fluent(f)`, and `action(a)`. We use continuously a variable `T`, representing a time point where $0 \leq \mathtt{T} \leq t_{max}$, where t_{max} is an upper time bound. 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).
```

An atom like `holds(l,T)` expresses that fluent literal `l` is true at time point `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)), default(f), 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),time(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).
```

Once the preconditions of the triggering rule are satisfied, the occurrence of action *a* is enforced *unless* `holds(ab(occurs(a)),T)` is generated by the compilation of an inhibition rule (see below).

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)}.
```

3.2 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, for $i > 0$:

```
:- not holds(f,ti),fluent(f),time(ti)
```

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),ti).` 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:

$$\texttt{:- holds(neg(occurs(}a\texttt{)),}t_i\texttt{),action(}a\texttt{),time(}t_i\texttt{).}$$

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)

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 (see Appendix).

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*

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

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* $0 \le k \le m$
   - (a) $\texttt{holds}(\texttt{f},k) \in X$, *if* $s_k \models f$ *and*
   - (b) $\texttt{holds}(\texttt{neg}(\texttt{f}),k) \in X$, *if* $s_k \models \neg f$.
   - (c) $\texttt{holds}(\texttt{occurs}(\texttt{a}),k) \in X$, *if* $a \in A_{k+1}$
   - (d) $\texttt{holds}(\texttt{neg}(\texttt{occurs}(\texttt{a})),k) \in X$, *if* $a \notin A_{k+1}$
2. *If* X *is an answer set of logic program* $\mathcal{T}(D, O_{init} \cup A_Q)$ *and for* $0 \le k \le m$
   - (a) $s_k = \{f \mid \texttt{holds}(\texttt{f},k) \in X\} \cup \{\neg f \mid \texttt{holds}(\texttt{neg}(\texttt{f}),k) \in X\}$
   - (b) $A_{k+1} = \{a \mid \texttt{holds}(\texttt{occurs}(\texttt{a}),k) \in X\}$

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

3.3 Action Query Language

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

3.3.1 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(f1,T),...,holds(fn,T),
   achieved(T+1),fluent(f1),...,fluent(fn),time(T),time(T+1).
achieved(tmax) :- holds(f1,tmax),...,holds(fn,tmax),
   fluent(f1),...,fluent(fn).
```

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 **occurs_at** t_i) $\in P$ if `holds(occurs(a),i)`$\in X$.

3.3.2 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 **occurs_at** t_i) $\in E$ if `holds(occurs(a),i)`$\in X$.

3.3.3 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 behavior. 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(f1,T), ..., holds(fn,T),
   fluent(f1),...,fluent(fn),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 application scenarios at the Max-Planck Institute for Molecular Plant Physiology for modeling metabolic as well as signal transduction networks. For illustration, we describe below 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 normalize its sulfur level. However, if this strategy is not successful the plant channels 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 modeling 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. A surplus of the auxin flux 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 *inter alia* to the expression of IAA28, a gene which is involved in the inhibition of lateral root growth. If the sulfur level is still low and IAA28 is expressed, other processes result in a different physiological endpoint, the production of seeds [34, 36].

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

$$A = \{sulfur\_depletion, sulfur\_repletion, enhanced\_lateral\_root\_formation,$$
$$iaa28\_expression, rapid\_seed\_production\}$$

$$F = \{normal\_sulfur, depleted\_sulfur, enhanced\_lateral\_roots, expressed\_iaa28, seeds\}$$

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**¬*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), (¬ *enhanced_lateral_roots at 0*),
(¬ *expressed_iaa28 at* 0), (¬ *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 behavior:

O_1 = O ∪ {(*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 behavior 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 stabilizes 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.

4.1 Applying $\mathcal{C}_{TAID}$ to the Biological Example

Let us now show how $\mathcal{C}_{TAID}$ can be applied to an extended biological example and explain different aspects of its reasoning capacities.

To build the model of the tested biological example of *Arabidopsis* plants responding to hypo-sulfur stress, we compiled the available data about the behavior of the particular system elements and on their mutual coherence in form of interactions of distinct nature (Fig. 1). These data were translated into the model using

1. formalization of the states of individual fluents and actions (in a binary form of a fluent f either holding or not holding and an action a either occurring or not occurring) at the discrete time points of response development, and
2. known reasons for the changes in these states (as causally directed connections between fluents and actions).

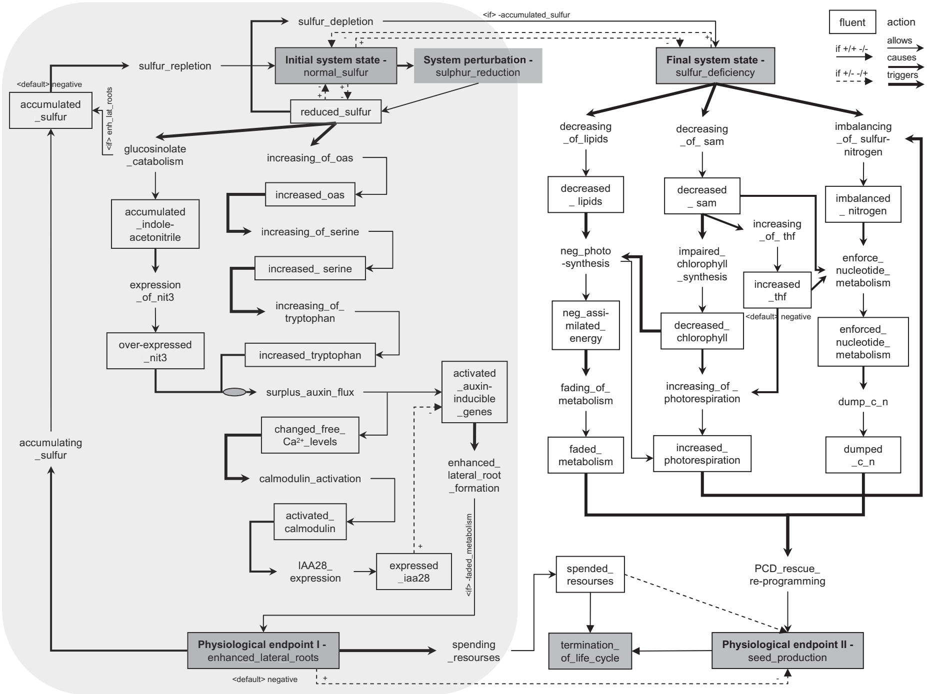


Fig. 1 A network of causal influences between elements of the system in *Arabidopsis* plants in response to hypo-sulfur stress, a compilation on which a model of an extended biological example was built and translated into $\mathcal{C}_{TAID}$. A subset of interactions covered by the *shadowed area* is compiled into a small biological example, used for the verification of the model

Fluents were represented by genes (expressed_iaa28), metabolites (increased_ tryptophan), or more complex phenotypical traits (enhanced_lateral_roots). Actions corresponded to particular cellular processes (glucosinolate_catabolism). In such a way, a systems' state is described in a query by a combination of fluent and action observations.

4.1.1 Analysis of the Initial State by using the Biological Model as Verification

To adjust the model, we tested, whether the final state, which has been observed experimentally, can be achieved by model simulation and how the model can be improved by fine-tuning the domain parameters (especially those for which the exact data are absent) to make the observed final state achievable by the model. Through this verification the unknown initial parameters could be identified.

In spite of the lack of data about the state of some of the fluents at the initial time point, their inclusion into the model is relevant, because their involvement in the response development has been demonstrated experimentally, and the reasoning about them is clear. For example, indole-acetonitrile is not in the list of metabolites, which were detected by applying Gas chromatography – mass spectrometry (GC/MS) techniques, and thus it cannot be estimated whether the fluent 'accumulated_indole_acetonitrile' holds or does not hold from the measurements.

Nevertheless, its accumulation has to be considered in the model, because in sulfur stress response both sulfur-releasing catabolism of glucosinolates and overexpression of Nit3 gene were demonstrated [6, 24], and indole-acetonitrile links these two cellular processes with causal connections, as being simultaneously a direct product of the first and a substrate for the second [23]. Now by simulating the systems' response by including a fluent observation for a particular point of time, which is known experimentally to occur (e.g. 'enhanced_lateral_root_formation occurs_at 6'), a model, where fluent 'accumulated_indole_acetonitrile' is holding or not holding can be tested. For the extended example in Fig. 1, into which behavior of a system in a sulfur-deficient homeostasis is incorporated, more complex fluent traits such as 'imbalanced_nitrogen' or 'neg_assimilated_energy' can be examined in such a way and identified as holding or not holding at initial time points.

4.1.2 Model Verification by Comparing the Estimated and the Modeled Time for a Queried Fluent to Hold

The causal modeling presented here allows hierarchical ordering in a causal consequence of the response events from a dense network of mutual interactions. This ordering is accomplished by assigning the earliest virtual time point to each of the fluents and actions to hold/occur. In the simplified small example, which we used for model adjustment (shadowed area of Fig. 1), such causal hierarchy can be predicted without computing by just visual analysis of the network of causal influences in the system, where the shortest distance from the initial state to a state in which a queried fluent holds, is expressed in a number of fluents to be passed. For our small example such predictions are in full correspondence with the virtual ordering provided by model simulation.

4.1.3 Combinatorial Manner of Functioning of Biosystem Constituents

A comparison of sets of actions of different plans leading to alternative final states shows that some sets of actions they pass are similar, but what is significant to reach either of the final states is the combination of actions and their causal order. This is shown in Table 1. The first column gives the actions in our small bioexample. The second and third column indicate virtual time points in plans of actions for query normal_sulfur and query sulfur_deficiency, respectively.

4.1.4 Synergism in Functioning of Systems Constituents

Analysis of co-occurrence of different actions in the plans leading to alternative states allows reasoning about synergetic influences inside the system. In Table 2 the sets of actions leading to the alternative states of normal sulfur or sulfur deficiency are analyzed by their co-occurrence at the same virtual time points.

Here, in the plan to get back to normal sulfur the action 'sulfur_reduction' co-occurs at time point 2 simultaneously with two groups of actions,

(i) 'glucosinolate_catabolism', 'increasing_of_oas' and 'activation_of_auxin_inducible_genes' and
(ii) 'expression_of_nit3' and 'increasing_of_serine'

Table 1 Comparison of the occurrence of sets of consecutive actions to achieve two different final states (normal sulfur or sulfur deficiency)

Actions	Query: normal_sulfur	Query: sulfur_deficiency
sulfur_reduction	0, 1, 2	0, 2, 3
glucosinolate_catabolism	1, 2, 3	1, 3, 4
increasing_of_oas	1, 2, 3	1, 3, 4
activation_of_auxin_inducible_genes	1, 2, 3	1, 3, 4
expression_of_nit3	2, 3	2
increasing_of_serine	2, 3, 4	2, 4
enhanced_lateral_root_formation	2, 3, 4	2, 4
increasing_of_tryptophan	3, 4	3
accumulating_sulfur	3	3
surplus_auxin_flux	–	–
sulfur_repletion	2, 3	2
sulfur_depletion	0, 1, 2, 3	0, 1, 3
calmodulin_activation	–	–
iaa28_expression	–	–

While in a plan leading to an alternative systems state of sulfur deficiency co-occurrence of the action 'sulfur_reduction' with these two sets of actions is scattered between two virtual time points, i.e. time 3 with the set (i) and time 2 with the set (ii).

4.1.5 Essentiality of Causal Hierarchies for the Functioning of the System

Comparative analysis of actions, appearing at a particular time point in exclusively either of the alternative plans, points at their putative essentiality for a particular queried state to be achieved. In Table 3, all similar actions appearing in both alternative plans are filtered out, and only query-specific actions at particular time points are left, indicating which actions have to occur in a specific causal order for a particular final state to be achieved. For example, 'increasing_of_tryptophan' has to occur late in the causal hierarchy to let the system go back to the state of normal sulfur.

4.1.6 By Analysis of Action Essentiality Redundant Side Branches of Informational Flows can be Identified

Among all possible actions those can be identified in the above analysis, which are bypassed by both sets of plans for alternative final states. E.g. among sets of actions, which have to occur for either the recovery of normal-sulfur homeostasis or for the new homeostatic state of sulfur deficiency, three actions are bypassed: surplus_auxin_flux, calmodulin_activation and iaa28_expression (Table 1). These actions constitute one particular branch of causal flow (Fig. 1). Thus, from this comparative analysis the whole branch appeared to be non-essential for the accomplishment of that part of systems response modeled by the small bioexample (shadowed part of the network in Fig. 1). However, it comes into play later and influences the switch between two physiological endpoints, as can be revealed by the simulation of the extended example (the whole Fig. 1).

Table 2 Sets of consecutive actions to occur for two alternative query states (normal sulfur or sulfur deficiency) to be achieved

	Query: normal_sulfur	Query: sulfur_deficiency
0	sulfur_reduction	sulfur_reduction
0	sulfur_depletion	sulfur_depletion
1	sulfur_reduction	–
1	glucosinolate_catabolism	glucosinolate_catabolism
1	increasing_of_oas	increasing_of_oas
1	activation_of_auxin_induciblegenes	activation_of_auxin_induciblegenes
1	sulfur_depletion	sulfur_depletion
2	sulfur_reduction	sulfur_reduction
2	glucosinolate_catabolism	–
2	increasing_of_oas	–
2	activation_of_auxin_induciblegenes	–
2	expression_of_nit3	expression_of_nit3
2	increasing_of_serine	increasing_of_serine
2	enhanced_lateral_root_formation	enhanced_lateral_root_formation
2	sulfur_repletion	sulfur_repletion
2	sulfur_depletion	–
3	–	sulfur_reduction
3	glucosinolate_catabolism	glucosinolate_catabolism
3	increasing_of_oas	increasing_of_oas
3	activation_of_auxin_induciblegenes	activation_of_auxin_induciblegenes
3	expression_of_nit3	–
3	increasing_of_serine	–
3	enhanced_lateral_root_formation	–
3	increasing_of_tryptophan	increasing_of_tryptophan
3	accumulating_sulfur	accumulating_sulfur
3	sulfur_repletion	–
3	sulfur_depletion	–
4	–	glucosinolate_catabolism
4	–	increasing_of_oas
4	–	activation_of_auxin_induciblegenes
4	increasing_of_serine	increasing_of_serine
4	enhanced_lateral_root_formation	enhanced_lateral_root_formation
4	increasing_of_tryptophan	–

4.1.7 Fluent Essentiality can be Estimated by Comparative Simulation of Alternative Models

Fluent essentiality is characterized through comparison of systems behavior modeled for the situations when a certain fluent is altered. In a biological system, such alterations can be obtained experimentally through e.g. gene mutations. Regarding the biological example of sulfur stress response, plants with knocked-out IAA28 gene are incorporated into the experiments, because,

- both enhanced lateral root formation and over-expression of IAA28 gene occur in sulfur-stressed *Arabidopsis* plants [35], and
- IAA28 represents one of the nodes in the network of gene interactions, which regulates the growth of lateral roots [32].

Table 3 Sets of consecutive actions to occur for either of two alternative query states (normal sulfur or sulfur deficiency) to be achieved

	Query: normal_sulfur	Query: sulfur_deficiency
1	sulfur_reduction	–
2	glucosinolate_catabolism	–
2	increasing_of_oas	–
2	activation_of_auxin_inducible_genes	–
2	sulfur_depletion	–
3	–	sulfur_reduction
3	expression_of_nit3	–
3	increasing_of_serine	–
3	enhanced_lateral_root_formation	–
3	sulfur_repletion	–
4	–	glucosinolate_catabolism
4	–	increasing_of_oas
4	–	activation_of_auxin_inducible_genes
4	increasing_of_tryptophan	–

Thus, IAA28 constitutes a causal informational link connecting sulfur stress response to enhanced lateral root formation. Prior to the wet lab biological experiments with IAA28 mutants, we estimated tentative importance of alterations in this gene by comparing a $\mathcal{C}_{TAID}$ model for wild type plants (Fig. 1, small example under the shadowed area) with the model in which IAA28 gene was switched off. In both models, the alternative states of normal sulfur and of sulfur deficiency (i) could be achieved and (ii) at the similar earliest time points. These model simulations, together with non-essentiality of the whole IAA28-containing causal side branch (determined above), point out at putative non-essentiality of IAA28 gene activity for the earlier stages of the response development. In addition, however, as can be seen in Table 4, the number of plans by which the states with different queried fluents can be achieved, is generally lower for the model of IAA28 mutant. To our point of view, this may reflect different levels of systems flexibility in a particular stress response.

Table 4 Number of plans leading to a state with a queried fluent in wild type plants and in plants in which one of the fluents is switched off by a mutation

Queried fluent	IAA28	Mutants
normal_sulfur	120157	98077
accumulated_indoleacetonitrile	92152	75448
increased_oas	92152	75448
active_auxin_inducible_genes	96248	80568
over_expressed_nit3	70848	54720
increased_serine	83688	61800
enhanced_lateral_roots	49704	49704
increased_tryptophan	71872	60736
accumulated_sulfur	106552	85880
sulfur_deficiency	57472	44096

5 Discussion and Related Work

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 Tran and Baral [48]. The latter language provides basic features to define dynamic causal laws, triggering and inhibition rules, which turn to be a fruitful basis but insufficient for modeling 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 biological 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 non-determinism introduced by allowance rules that may yield multiple trajectory models.

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 Tran and Baral [48]. The resulting tool is implemented in Java and freely available at http://bioinformatics.mpimp-golm.mpg.de/projects/own/bionet-reasoning. Technically speaking, our system takes an action description and compiles it into a logic program. This program is then treated by an off-the-shelf grounder, like `lparse` [45] or `gringo` [18]. Similarly, an off-the-shelf answer set solver, like `smodels` [43] or `clasp` [16], is then utilized to compute answer sets, which are passed to the system's back-end for analysis by the biologist. Given the high-performance of these systems, we have so far neither encountered performance nor scalability problems. Most trajectory models are computed in milliseconds. Rather, it is sometimes the huge number of such models and lacking analysis tools that pose a bottle-neck and are thus subject of ongoing work.

Meanwhile, the application of $\mathcal{C}_{TAID}$ has proved to be useful for reconstructing and reasoning about the behavior of biological systems. Of particular biological interest is the possibility to characterize a (biological) system's initial state from the experimentally observable final states by means of explanation. Another outcome was the identification of combinations of different actions, necessary to occur in order to achieve a final state; this allowed for further charting the synergistic influences inside biological systems. It was also possible to estimate the essential fluents, actions and their causal hierarchies for the system's functionality. The approach showed also promise for the in silico probing of putative effects of the mutations on the stability and flexibility of a biological system. We hope to expand this aspect of the analysis in order to characterize state transitions in biological systems.

Further logic-based approaches using rule-based languages have emerged recently. Related work has been conducted in abductive logic programming where abduction was used in Papatheodorou et al. [38] as the principal mode of inference for modeling gene relations from micro-array data. An integration of abduction and induction for modeling metabolic pathways is described in Tamaddoni-Nezhad et al. [46]. Pan et al. [37] investigate the usage of the action language GOLOG [25] for rapid prototyping of applications in evolutionary biology. In Son and Pontelli [44], answer set planning is directly used for planning biochemical pathways.

Boolean constraint processing techniques have also been successfully applied to other biological areas. For instance, [29, 30] report speed-ups of several orders of magnitude by using Boolean satisfiability solvers for Haplotype Inference. Constraint programming as such has already been applied to many biological problems, as best witnessed by the proceedings of the workshop series on *Constraint Based Methods for Bioinformatics* as well as this special issue of the *Constraint Journal* at hand. Among many others, we find Fanchon et al. [14], Backofen et al. [1], and Eveillard et al. [13].

A very sophisticated and rather advanced automated reasoning tool for systems biology can be found in the area of constraint programming, namely the BIOCHAM [7] system. BIOCHAM relies on CTL [8] and is thus particularly strong in modeling temporal aspects of systems biology. Unlike our abstract approach, the constraint-based approach offers fine-grained capacities for modeling biochemical processes, including kinetics and reactions.

The relationship between action languages and more traditional approaches, like Petri nets [39, 40], π-calculus [41], or pathway logic [12], is elaborated upon in detail in Baral et al. [4] and Tran [47]. Basically, all aforementioned approaches are primarily aiming at simulation, that is, prediction in our terminology. Complementary reasoning modes, such as explanation and diagnosis or planning, are usually only addressable in an indirect way. Another difference manifests itself by the rather natural treatment of incomplete information in our framework, which is no intrinsic feature of other approaches. See Baral et al. [4] and Tran [47] for a detailed discussion on this relationship. However, given that these approaches have already proved their value for modeling biological applications, it will be interesting to see how similar domains can be modeled in their and our framework.

Acknowledgements We are grateful to Philippe Veber and Sven Thiele for fruitful discussions and valuable hints on improving our paper.

This work was partially funded by the Max Planck Society and by the Federal Ministry of Education and Research within the GoFORSYS project (http://www.goforsys.org/; grant 0313924).

Appendix

A Compilation of $\mathcal{C}_{TAID}$ to Logic Programs

This section summarizes the translation of $\mathcal{C}_{TAID}$ to logic programs in view of proving Theorem 1.

Given a domain description $D(A, F)$, we define the translation $\mathcal{T}$ as the following collection of rules:

1. For every time point $0 \leq t \leq m$, we have a fact:

$$\texttt{time}(t). \tag{12}$$

2. For every action $a \in A$ and every fluent $f \in F$, we have a fact:

$$\texttt{fluent}(f). \tag{13}$$

$$\texttt{action}(a). \tag{14}$$

3. For every fluent $f \in F$ and every time point $0 \leq t \leq m$, we have a constraint:

$$\texttt{:-}\ \texttt{holds}(f, t), \texttt{holds}(\texttt{neg}(f), t), \texttt{fluent}(f), \texttt{time}(t). \tag{15}$$

4. For every fluent $f \in F$, we have a pair of rules of the form:

$$\texttt{holds}(f, 0)\ \texttt{:-}\ \texttt{not}\ \texttt{holds}(\texttt{neg}(f), 0). \tag{16}$$

$$\texttt{holds}(\texttt{neg}(f), 0)\ \texttt{:-}\ \texttt{not}\ \texttt{holds}(f, 0). \tag{17}$$

5. For every statement (**default** f) $\in D$ and every time point $0 \leq t \leq m$, we have a rule and a fact of the form:

$$\begin{aligned}\texttt{holds}(f, t)\ \texttt{:-}\ &\texttt{not}\ \texttt{holds}(\texttt{neg}(f), t), \texttt{default}(f),\\ &\texttt{fluent}(f), \texttt{time}(t).\end{aligned} \tag{18}$$

$$\texttt{default}(f). \tag{19}$$

   An analogous pair is contained for each statement (**default** $\neg f$) $\in D$.
6. For every (inertial) fluent $f \in F$ such that (**default** f) $\notin D$ and (**default** $\neg f$) $\notin D$, and every time point $0 \leq t < m$, we have two rules of the form:

$$\begin{aligned}\texttt{holds}(f, t+1)\ \texttt{:-}\ &\texttt{holds}(f, t), \texttt{not}\ \texttt{holds}(\texttt{neg}(f), t+1),\\ &\texttt{not}\ \texttt{default}(f), \texttt{fluent}(f), \texttt{time}(t), \texttt{time}(t+1).\end{aligned} \tag{20}$$

$$\begin{aligned}\texttt{holds}(\texttt{neg}(f), t+1)\ \texttt{:-}\ &\texttt{holds}(\texttt{neg}(f), t), \texttt{not}\ \texttt{holds}(f, t+1),\\ &\texttt{not}\ \texttt{default}(f), \texttt{fluent}(f), \texttt{time}(t), \texttt{time}(t+1).\end{aligned} \tag{21}$$

7. For every static causal law ($f_1, \ldots, f_n$ **if** $g_1, \ldots, g_m$) $\in D$, each f_i where $1 \leq i \leq n$, and every time point $0 \leq t \leq m$, we have a rule of the form:

$$\begin{aligned}\texttt{holds}(f_i, t)\ \texttt{:-}\ &\texttt{holds}(g_1, t), \ldots, \texttt{holds}(g_n, t),\\ &\texttt{fluent}(g_1), \ldots, \texttt{fluent}(g_n), \texttt{fluent}(f_i), \texttt{time}(t).\end{aligned} \tag{22}$$

8. For every dynamic causal law (a **causes** $f_1, \ldots, f_n$ **if** $g_1, \ldots, g_m$) $\in D$, each f_i where $1 \leq i \leq n$, and every time point $0 \leq t < m$, we have a rule of the form:

$$\begin{aligned}\texttt{holds}(f_i, t+1) \texttt{ :- } & \texttt{holds(occurs}(a), t),\\ & \texttt{holds}(g_1, t), \ldots, \texttt{holds}(g_n, t),\\ & \texttt{fluent}(g_1), \ldots, \texttt{fluent}(g_n), \texttt{fluent}(f_i),\\ & \texttt{action}(a), \texttt{time}(t), \texttt{time}(t+1).\end{aligned} \tag{23}$$

9. For every allowance rule ($f_1, \ldots, f_n$ **allows** a) $\in D$, each f_i where $1 \leq i \leq n$, and every time point $0 \leq t \leq m$, we have a rule of the form:

$$\begin{aligned}\texttt{holds(allow(occurs}(a)), t) \texttt{ :- } & \texttt{not holds(ab(occurs}(a)), t),\\ & \texttt{holds}(f_1, t), \ldots, \texttt{holds}(f_n, t),\\ & \texttt{fluent}(f_1), \ldots, \texttt{fluent}(f_n),\\ & \texttt{action}(a), \texttt{time}(t).\end{aligned} \tag{24}$$

10. For every exogenous action $a \in A$ and every time point $0 \leq t \leq m \in D$, we have:

$$\texttt{holds(allow(occurs}(a)), t) \texttt{ :- } \texttt{action}(a), \texttt{time}(t). \tag{25}$$

11. For every exogenous or allowed action $a \in A$ and every time point $0 \leq t < m$, we have two rules of the form:

$$\begin{aligned}\texttt{holds(occurs}(a), t) \texttt{ :- } & \texttt{holds(allow(occurs}(a)), t),\\ & \texttt{not holds(ab(occurs}(a)), t),\\ & \texttt{not holds(neg(occurs}(a)), t),\\ & \texttt{action}(a), \texttt{time}(t), t < m.\end{aligned} \tag{26}$$

$$\begin{aligned}\texttt{holds(neg(occurs}(a)), t) \texttt{ :- } & \texttt{not holds(occurs}(a), t),\\ & \texttt{action}(a), \texttt{time}(t), t < m.\end{aligned} \tag{27}$$

12. For every triggering rule ($f_1, \ldots, f_n$ **triggers** a) $\in D$ and every time point $0 \leq t \leq m$, we have a rule of the form:

$$\begin{aligned}\texttt{holds(occurs}(a), t) \texttt{ :- } & \texttt{not holds(ab(occurs}(a)), t),\\ & \texttt{holds}(f_1, t), \ldots, \texttt{holds}(f_n, t),\\ & \texttt{fluent}(f_1), \ldots, \texttt{fluent}(f_n),\\ & \texttt{action}(a), \texttt{time}(t).\end{aligned} \tag{28}$$

13. For every inhibition rule $(f_1, \ldots, f_n$ **inhibits** $a) \in D$ and every time point $0 \leq t \leq m$, we have a rule of the form:

$$\texttt{holds(ab(occurs(}a\texttt{)),}\ t\texttt{)}\ \texttt{:-}\ \texttt{holds(}f_1, t\texttt{)}, \ldots, \texttt{holds(}f_n, t\texttt{)}, \texttt{fluent(}f_1\texttt{)}, \ldots, \texttt{fluent(}f_n\texttt{)}, \texttt{action(}a\texttt{)}, \texttt{time(}t\texttt{)}. \tag{29}$$

14. For every no-concurrency constraints (**noconcurrency** $a_1, \ldots, a_n) \in D$ and every time point $0 \leq t \leq m$, we have a constraint:

$$\texttt{:-}\ \texttt{time(}t\texttt{)}, 2\,\{\texttt{holds(occurs(}a_1\texttt{)}, t\texttt{)} : \texttt{action(}a_1\texttt{)}, \ldots, \texttt{holds(occurs(}a_n\texttt{)}, t\texttt{)} : \texttt{action(}a_n\texttt{)}\}. \tag{30}$$

Next, we address fluent observations about the initial state, given by O_{init}.

1. For every observation $(f$ **at** $0) \in O_{init}$, we have a fact:

$$\texttt{holds(}f, 0\texttt{)}. \tag{31}$$

And finally, we deal with the fluent observations stemming from a query Q as in (9), and collected in A_Q in Theorem 1

1. For every occurrence of $(a$ **occurs_at** $t) \in A_Q$ such that a is an exogenous action, we have a fact:

$$\texttt{holds(occurs(}a\texttt{)}, t\texttt{)}. \tag{32}$$

2. For every occurrence of $(a$ **occurs_at** $t) \in A_Q$ such that a is no exogenous action, we have a constraint:

$$\texttt{:-}\ \texttt{holds(neg(occurs(}a\texttt{))}, t\texttt{)}, \texttt{action(}a\texttt{)}, \texttt{time(}t\texttt{)}. \tag{33}$$

B Auxiliary Definitions and Results

We start with recalling some definitions from answer set theory.

Given a rule

$$p_0 \leftarrow p_1, \ldots, p_m, not\ p_{m+1}, \ldots, not\ p_n\ ,$$

we define

$$\begin{aligned} head(r) &= p_0 \\ body(r) &= \{p_1, \ldots, p_m, not\ p_{m+1}, \ldots, not\ p_n\} \\ body^+(r) &= \{p_1, \ldots, p_m\} \\ body^-(r) &= \{p_{m+1}, \ldots, p_n\} \\ lit(r) &= head(r) \cup body^+(r) \cup body^-(r). \end{aligned}$$

Let Π be a normal logic program and X be a set of atoms. Then, the reduct of Π relative to X is defined as $\Pi^X = \{head(r) \leftarrow body(r)^+ \mid r \in \Pi \text{ and } body^- \cap X = \emptyset\}$. A set X of atoms is an answer set of Π, if X is the $\subseteq$-smallest model of Π^X.

Our proof makes use of an alternative characterization of answer sets relying on *splitting sequences* [27]. This characterization makes use of the concept of a *splitting set*. A splitting set for a program Π is a set of literals U such that[8]

$$\text{if } \{head(r)\} \cap U \neq \emptyset, \text{ then } lit(r) \subseteq U \text{ for each } r \in \Pi .$$

A splitting set U divides the program in two parts, namely, the bottom of Π, defined as

$$b_U(\Pi) = \{r \mid lit(r) \subseteq U\} ,$$

and the top of Π, that is, $\Pi \setminus b_U(\Pi)$.

Note that the bottom $b_U(\Pi)$ does not contain any head atoms from the top $\Pi \setminus b_U(\Pi)$, that is,

$$lit(b_U(\Pi)) \cap head(\Pi \setminus b_U(\Pi)) = \emptyset .$$

This implies according to [15, Theorem 8] that a set X of atoms is an answer set of Π iff there is an answer set[9] Y of $b_U(\Pi)$ such that X is an answer set of $\{p \leftarrow \mid p \in Y\} \cup (\Pi \setminus b_U(\Pi))$.

Given two sets U, X of literals and a program Π, define

$$e_U(\Pi, X) = \left\{ r' \;\middle|\; \begin{array}{r} head(r') = head(r), \\ r \in \Pi^{U,X},\ body^+(r') = body^+(r) \setminus U, \\ body^-(r') = body^-(r) \setminus U \end{array} \right\} \tag{34}$$

where

$$\Pi^{U,X} = \{r \in \Pi \mid (body^+(r) \cap U) \subseteq X \text{ and } (body^-(r) \cap U) \cap X = \emptyset\} . \tag{35}$$

A *solution* to Π wrt a splitting set U is a pair $\langle X_0, X_1 \rangle$ such that

1. X_0 is an answer set of $b_U(\Pi)$
2. X_1 is an answer set of $e_U(\Pi \setminus b_U(\Pi), X_0)$
3. $X_0 \cup X_1$ is consistent

Given a splitting set U for Π, the basic *splitting set theorem* [27] tells us that a set X of literals is a consistent answer set of Π iff $X = X_0 \cup X_1$ for some solution $\langle X_0, X_1 \rangle$ to Π wrt U.

A sequence $\langle U_i \rangle_{i \in I}$ of splitting sets for Π such that $U_i \subset U_j$ whenever $i < j$ and $\bigcup_{i \in I} U_i = lit(\Pi)$ is a *splitting sequence* for Π. The definition of a solution extends to splitting sequences as follows. A solution to Π wrt a splitting sequence $\langle U_i \rangle_{i \in I}$ is a sequence $\langle X_i \rangle_{i \in I}$ of sets of literals such that

1. X_0 is an answer set of $b_{U_0}(\Pi)$
2. X_{i+1} is an answer set of $e_{U_i}(b_{U_{i+1}}(\Pi) \setminus b_{U_i}(\Pi), \bigcup_{j \leq i} X_j)$
3. $\bigcup_{i \in I} X_i$ is consistent

[8] While a normal rule r yields a singleton as $head(r)$, an integrity constraint yields $\emptyset$.

[9] Recall that we deal with finite programs, yielding finite answer sets.

Note that every literal in $b_{U_0}(\Pi)$ belongs to $U_0 \cap lit(\Pi)$, and every literal in $e_{U_i}(b_{U_{i+1}}(\Pi) \setminus b_{U_i}(\Pi), \bigcup_{j\le i} X_j)$ belongs to $(U_{i+1} \setminus U_i) \cap lit(\Pi)$. Accordingly, we have for a solution $\langle X_i \rangle_{i\in I}$ that

$$X_0 \subseteq U_0 \cap lit(\Pi) \tag{36}$$

$$X_{i+1} \subseteq (U_{i+1} \setminus U_i) \cap lit(\Pi) \tag{37}$$

and so $X_i \cap X_j = \emptyset$ for all distinct $i, j \in I$.

In analogy to the basic version, we have according to the *splitting sequence theorem* [27] that given a splitting sequence $\langle U_i \rangle_{i\in I}$ for Π, a set X of literals is a consistent answer set of Π iff $X = \bigcup_{j\le i} X_j$ for some solution $\langle X_i \rangle_{i\in I}$ to Π wrt $\langle U_i \rangle_{i\in I}$.

C Proof of Theorem 1

To begin with, we define a splitting sequence

$$\langle U_b, U_0, U_1, \ldots, U_m \rangle$$

where

$$U_b = \{\mathtt{time}(i) \mid 0 \le i \le m\} \tag{38}$$

$$\cup \{\mathtt{action}(a) \mid a \in A\} \tag{39}$$

$$\cup \{\mathtt{fluent}(f) \mid f \in F\} \tag{40}$$

$$\cup \{\mathtt{default}(f) \mid f \in F\} \tag{41}$$

$$U_0 = U_b \tag{42}$$

$$\cup \{\mathtt{holds}(f, 0), \mathtt{holds}(\mathtt{neg}(f), 0) \mid f \in F\} \tag{43}$$

$$U_i = U_b \cup U_0 \cup \cdots \cup U_{i-1} \tag{44}$$

$$\cup \{\mathtt{holds}(f, i), \mathtt{holds}(\mathtt{neg}(f), i) \mid f \in F\} \tag{45}$$

$$\cup \{\mathtt{holds}(\mathtt{occurs}(a), i-1), \mathtt{holds}(\mathtt{neg}(\mathtt{occurs}(a)), i-1) \mid a \in A\} \tag{46}$$

$$\cup \{\mathtt{holds}(\mathtt{allow}(\mathtt{occurs}(a)), i-1) \mid a \in A\} \tag{47}$$

$$\cup \{\mathtt{holds}(\mathtt{ab}(\mathtt{occurs}(a)), i-1) \mid a \in A\} \qquad \text{for } 1 \le i \le m\,. \tag{48}$$

According to [27], a set X of literals is an answer set of a program Π iff

$$X = X_b \cup X_0 \cup \cdots \cup X_m$$

for some solution $\langle X_b, X_0, \ldots, X_m \rangle$ of Π wrt splitting sequence $\langle U_b, U_0, U_1, \ldots, U_m \rangle$.

Letting Π be $\mathcal{T}(D, O_{init} \cup A_Q)$, this splitting sequence induces the following sequence of programs:

- $\Pi_b = b_{U_b}(\Pi)$
  This program is induced by U_b and contains the definitions of time points, actions, fluents, as well as non-inertial fluents, distinguished by default statements.

- $\Pi_0 = e_{U_b}(b_{U_0}(\Pi) \setminus b_{U_b}(\Pi), X_b)$
  This program is induced by U_0 and, intuitively, contains the rules and constraints expressing information about the initial state. That is, rules and constraints whose literals are indexed with 0.
- $\Pi_i = e_{U_{i-1}}(b_{U_i}(\Pi) \setminus b_{U_{i-1}}(\Pi), X_b \cup \bigcup_{j \leq i-1} X_j)$ for $1 \leq i \leq m$
  This program is induced by U_i and, intuitively, contains all rules and constraints expressing information about the ith state. That is, rules and constraints whose literals are indexed with i.

Finally, we note that any answer set of $\Pi = \mathcal{T}(D, O_{init} \cup A_Q)$ is consistent given that $head(\Pi)$ contains positive literals only (see A).

C.1 Proof of Part 1 of Theorem 1

Let $s_0, A_1, s_1, A_2, \ldots, A_m, s_m$ be a a trajectory model of $(D, O_{init} \cup A_Q)$.
We have to show that $\Pi = \mathcal{T}(D, O_{init} \cup A_Q)$ has an answer set X satisfying

1. $\texttt{holds}(f, k) \in X$, if $s_k \models f$,
2. $\texttt{holds(neg}(f), k) \in X$, if $s_k \models \neg f$,
3. $\texttt{holds(occurs}(a), k) \in X$, if $a \in A_{k+1}$,
4. $\texttt{holds(neg(occurs}(a)), k) \in X$, if $a \notin A_{k+1}$.

for all $f \in F, a \in A$, and $0 \leq k \leq m$.

By the splitting sequence theorem, it is sufficient to show that $X = X_b \cup X_0 \cup \cdots \cup X_m$ for some solution $\langle X_b, X_0, \ldots, X_m \rangle$ of Π wrt splitting sequence $\langle U_b, U_0, U_1, \ldots, U_m \rangle$.

To this end, we let

$$X_b = \{\texttt{time}(i) \mid 0 \leq i \leq m\} \tag{49}$$

$$\cup \{\texttt{action}(a) \mid a \in A\} \tag{50}$$

$$\cup \{\texttt{fluent}(f) \mid f \in F\} \tag{51}$$

$$\cup \{\texttt{default}(f) \mid f \in F, (\textbf{default } f) \in D\} \tag{52}$$

$$X_0 = \{\texttt{holds}(f, 0) \mid f \in F, s_0 \models f\} \tag{53}$$

$$\cup \{\texttt{holds(neg}(f), 0) \mid f \in F, s_0 \models \neg f\} \tag{54}$$

$$X_i = \{\texttt{holds}(f, i) \mid s_i \models f\} \tag{55}$$

$$\cup \{\texttt{holds(neg}(f), i) \mid s_i \models \neg f\} \tag{56}$$

$$\cup \{\texttt{holds(occurs}(a), i-1) \mid a \in A_i\} \tag{57}$$

$$\cup \{\texttt{holds(neg(occurs}(a)), i-1) \mid a \notin A_i\} \tag{58}$$

$$\cup \{\texttt{holds(allow(occurs}(a)), i-1) \mid a \in A_A(s_{i-1})\} \tag{59}$$

$$\cup \{\texttt{holds(allow(occurs}(a)), i-1) \mid a \in A_E\} \tag{60}$$

$$\cup \{\texttt{holds(ab(occurs}(a)), i-1) \mid a \in A_I(s_{i-1})\} \tag{61}$$

where A_E is the set of exogenous actions:

$$A_E = \{a \mid a \in A, \{(f_1, \ldots, f_n \textbf{ allows } a), (f_1, \ldots, f_n \textbf{ triggers } a)\} \cap D = \emptyset)\}.$$

Consider the following three cases:

X_b By construction, Program $\Pi_b = b_{U_b}(\Pi)$ has a unique smallest model, which is X_b.
In other words, X_b is the unique answer set of $\Pi_b = b_{U_b}(\Pi)$.

X_0 Program $\Pi_0 = e_{U_b}(b_{U_0}(\Pi) \setminus b_{U_b}(\Pi), X_b)$ consists of the following types of rules:[10]

1. Simplifications of constraints of form (15) wrt X_b, viz.

$$\texttt{:-}\ \texttt{holds}(f, 0), \texttt{holds}(\texttt{neg}(f), 0).$$

Clearly, this constraint belongs to $\Pi_0^{X_0}$.
Given that s_0 is a state, the construction of X_0 implies that either $\texttt{holds}(f, 0) \in X_0$ or $\texttt{holds}(\texttt{neg}(f), 0) \in X_0$. Hence, this constraint is inapplicable. And so X_0 is trivially closed under this constraint, as is any model of $\Pi_0^{X_0}$ smaller than X_0.

2. Rules of form (16), viz.

$$\texttt{holds}(f, 0)\ \texttt{:-}\ \texttt{not}\ \texttt{holds}(\texttt{neg}(f), 0).$$

$$\texttt{holds}(\texttt{neg}(f), 0)\ \texttt{:-}\ \texttt{not}\ \texttt{holds}(f, 0).$$

Given that s_0 is a state, we distinguish two cases.

(a) If $s_0 \models f$, then $\texttt{holds}(f, 0) \in X_0$ and $\texttt{holds}(\texttt{neg}(f), 0) \notin X_0$.
As a consequence, we have $(\texttt{holds}(f, 0)\ \texttt{:-}) \in \Pi_0^{X_0}$.
As above, X_0 is trivially closed under this fact and $\texttt{holds}(f, 0)$ belongs to X_0 iff it belongs to the smallest model of $\Pi_0^{X_0}$.

(b) If $s_0 \models \neg f$, then $\texttt{holds}(\texttt{neg}(f), 0) \in X_0$ and $\texttt{holds}(f, 0) \notin X_0$.
As a consequence, we have $(\texttt{holds}(\texttt{neg}(f), 0)\ \texttt{:-}) \in \Pi_0^{X_0}$.
As above, X_0 is trivially closed under this fact and $\texttt{holds}(\texttt{neg}(f), 0)$ belongs to X_0 iff it belongs to the smallest model of $\Pi_0^{X_0}$.

3. Rules of form (18) simplified by X_b, viz.

$$\texttt{holds}(f, 0)\ \texttt{:-}\ \texttt{not}\ \texttt{holds}(\texttt{neg}(f), 0).$$

We have to distinguish two cases:

(a) If $s_0 \models f$, then clearly $s_0 \not\models \neg f$ and thus $\texttt{holds}(\texttt{neg}(f), 0) \notin X_0$. As a consequence, $\{\texttt{holds}(f, 0)\texttt{:-}\} \in \Pi_0^{X_0}$. X_0 is trivially closed under this fact.

(b) If $s_0 \models \neg f$, then $\texttt{holds}(\texttt{neg}(f), 0) \in X_0$ and the rule is not contained in $\Pi_0^{X_0}$. Therefore, X_0 is also trivially closed under $\Pi_0^{X_0}$.

The case of (**default** $\neg f$) $\in D$ is dealt with analogously.

4. Simplifications of rules of form (22) wrt X_b, viz.

$$\texttt{holds}(f_i, 0)\ \texttt{:-}\ \texttt{holds}(g_1, 0), \ldots, \texttt{holds}(g_m, 0).$$

Clearly, this rule belongs to $\Pi_0^{X_0}$.

[10] Recall that Π_0 consists of rules from Π that have been "evaluated" wrt X_b.

Note that the above rule stems from a static causal law ($f_1, \ldots, f_n$ **if** $g_1, \ldots, g_m$) in the action description D. Hence, we have $f_i \in s_0$ whenever $\{g_1, \ldots, g_m\} \subseteq s_0$.
By construction of X_0, we also have $\texttt{holds}(f_i, 0) \in X_0$ whenever $\{\texttt{holds}(g_1, 0), \ldots, \texttt{holds}(g_m, 0)\} \subseteq X_0$. Clearly, this holds for any model of $\Pi_0^{X_0}$, in particular, the smallest one.

We have shown that X_0 is closed under $\Pi_0^{X_0}$ and thus a model of $\Pi_0^{X_0}$. Moreover, we have demonstrated that X_0 is closed under $\Pi_0^{X_0}$ iff any smaller model of $\Pi_0^{X_0}$ is. Hence, X_0 is the smallest model of $\Pi_0^{X_0}$. In other words, X_0 is an answer set of $\Pi_0^{X_0}$.

X_i By definition of a trajectory (cf. Definition 5), (s_{i-1}, A_i, s_i) is a valid transition. The program $\Pi_i = e_{U_{i-1}}(b_{U_i}(\Pi) \setminus b_{U_{i-1}}(\Pi), X_b \cup \bigcup_{j \leq i-1} X_j)$ consists of the following rules:

1. Constraints of form (15) simplified by X_b, viz.

$$\texttt{:- holds}(f, i), \texttt{holds(neg}(f), i).$$

   This rule belongs to $\Pi_i^{X_i}$. Since (s_{i-1}, A_i, s_i) is a valid transition, X_i either contains $\texttt{holds}(f, i)$ or $\texttt{holds(neg}(f), i)$. Hence, X_i is trivially closed under this rule since the constraint will not be applicable.
2. Rules of form (18) simplified by X_b, viz.

$$\texttt{holds}(f, i) \texttt{ :- not holds(neg}(f), i).$$

   We have to distinguish two cases:

   (a) If $s_i \models f$, then clearly $s_0 \not\models \neg f$ and thus $\texttt{holds(neg}(f), i) \notin X_i$. As a consequence, $\{\texttt{holds}(f, i) \texttt{:-}\} \in \Pi_i^{X_i}$. X_i is trivially closed under this fact. Since $\texttt{holds}(f, i)$ belongs to every model of $\Pi_i^{X_i}$, it belongs to the smallest one.
   (b) If $s_i \models \neg f$, then $\texttt{holds(neg}(f), i) \in X_i$ and the rule is not contained in $\Pi_i^{X_i}$. Therefore X_i is also trivially closed under $\Pi_i^{X_i}$.

   In case of (**default** $\neg f$) $\in D$ the argumentation is done analogously.
3. Simplifications of rules of form (20) wrt. $X_b \cup X_0 \cup X_{i-1}$, viz.

$$\texttt{holds}(f, i) \texttt{ :- not holds(neg}(f), i).$$

$$\texttt{holds(neg}(f), i) \texttt{ :- not holds}(f, i).$$

   We have to distinguish two cases:

   (a) If $s_i \models f$, then $\texttt{holds}(f, i) \in X_i$ and as a consequence, $\{\texttt{holds}(f, i) \texttt{:-}\} \in \Pi_i^{X_i}$. X_i is trivially closed under this fact.
   (b) If $s_i \models \neg f$, then $\texttt{holds(neg}(f), i) \in X_i$ and as a consequence, $\{\texttt{holds(neg}(f), i) \texttt{:-}\} \in \Pi_i^{X_i}$. X_i is trivially closed under this fact.

   In both cases the rules are reduced to facts. Hence, they are contained in the minimal model of $\Pi_i^{X_i}$.
4. Simplifications of rules of form (22) wrt. X_b. viz.

$$\texttt{holds}(f_k, i) \texttt{ :- holds}(g_1, i), \ldots, \texttt{holds}(g_n, i).$$

We have to distinguish two cases:

(a) If $s_i \models g_l$ for $1 \leq l \leq n$, then due to Definition 2 $s_i \models f_k$. As a consequence, we have $\{\texttt{holds}(g_1, i), \ldots, \texttt{holds}(g_n, i), \texttt{holds}(f_k, i)\} \subset X_i$. That is, X_i is closed under $\Pi_i^{X_i}$. Clearly, this holds for every model of $\Pi_i^{X_i}$, in particular, the smallest one.
(b) If $s_i \not\models g_l$ for $\exists l : 1 \leq l \leq n$ then we have that $\texttt{holds}(\texttt{neg}(g_l), i) \in X_i$. X_i is trivially closed under $\Pi_i^{X_i}$, since the rule is not applicable.

5. If the dynamic law (a **causes** $f_1, \ldots, f_n$ **if** $g_1, \ldots, g_m$) was applicable (Definition 3), we get the following simplified rule of form (23) wrt. $X_b \cup X_0 \cup X_{i-1}$:

$$\texttt{holds}(f_k, i)\ \texttt{:-}\ \texttt{holds}(\texttt{occurs}(a), i-1).$$

We now have to distinguish two cases:

(a) If $a \in A_i$ we have $\texttt{holds}(\texttt{occurs}(a), i-1) \in X_i$. Since the dynamic law was applicable in s_{i-1}, we have $s_i \models f_k$ and as a consequence $\texttt{holds}(f, i) \in X_i$. Therefore X_i is closed under the mentioned rule. Clearly, this holds for every model of $\Pi_i^{X_i}$, in particular, the smallest one.
(b) If $a \notin A_i$ we have $\texttt{holds}(\texttt{occurs}(a), i-1) \notin X_i$. Then the rule is not applicable, and X_i is closed under $\Pi_i^{X_i}$.

If the dynamic law was not applicable in s_{i-1}, then X_i is trivially closed because the above rule is not contained in Π_i.[11]

6. Simplifications of rules of form (24) wrt. $X_b \cup X_0 \cup X_{i-1}$ viz.

$$\texttt{holds}(\texttt{allow}(\texttt{occurs}(a)), i-1)\ \texttt{:-}\ \texttt{not}\ \texttt{holds}(\texttt{ab}(\texttt{occurs}(a)), i-1).$$

We have to distinguish two cases:

(a) If $\texttt{holds}(\texttt{allow}(\texttt{occurs}(a)), i-1) \in X_i$ we have that $a \in A_A(s_{i-1})$. By definition of $A_A(s_{i-1})$ and $A_I(s_{i-1})$ we have that $\texttt{holds}(\texttt{ab}(\texttt{occurs}(a)), i-1) \notin X_i$, otherwise a would be passive in s_{i-1}. Since we have $s_{i-1} \models f_1 \wedge \cdots \wedge f_n$ (Definition 3) if $a \in A_A(s_{i-1})$, we also have $\{\texttt{holds}(f_1, i), \ldots, \texttt{holds}(f_n, i)\} \in X_{i-1}$. As a consequence, X_i is closed under the considered rule. We have $\{\texttt{holds}(\texttt{allow}(\texttt{occurs}(a)), i-1)\,\texttt{:-}\} \in \Pi_i^{X_i}$, thus $\texttt{holds}(\texttt{allow}(\texttt{occurs}(a)), i-1)$ is contained in any model of $\Pi_i^{X_i}$, in particular, the smallest one.
(b) If $\texttt{holds}(\texttt{allow}(\texttt{occurs}(a)), i-1) \notin X_i$ we have that $a \notin A_A(s_{i-1})$. Furthermore, two cases need to be distinguished.

    i. If we have $a \in A_I(s_{i-1})$ then we have $\texttt{holds}(\texttt{allow}(\texttt{occurs}(a)), i-1) \in X_i$ by construction of X_i. Since the rule will not be contained in $\Pi_i^{X_i}$, X_i is trivially closed under this rule.

[11] When constructing Π_i the rule is dropped because at least one of $\texttt{holds}(g_1, i-1), \ldots, \texttt{holds}(g_n, i-1)$ must be false in X_{i-1}.

ii. If we have $a \notin A_I(s_{i-1})$, at least one of $\texttt{holds}(f_1, i), \ldots,$ $\texttt{holds}(f_n, i)$ must be false. Since we have $a \notin A_A(s_{i-1})$, we can conclude $s_{i-1} \not\models f_1 \wedge \cdots \wedge f_n$ (Definition 3). Therefore at least for one f_l for $1 \leq l \leq n$ we have that $\texttt{holds}(f_l, i) \notin X_{i-1}$. Since then the rule is not contained in Π_i, X_i is trivially closed under this rule.

7. Simplifications of rules of form (25) wrt. X_b viz.

$$\texttt{holds(allow(occurs(}a\texttt{)),}\ i-1\texttt{).}$$

Whenever there is no allowance or triggers rule for an action a specified in D, we have $\texttt{holds(allow(occurs(}a\texttt{)),}\ i-1) \in X_i$. The rule is closed under X_i and the fact $\texttt{holds(allow(occurs(}a\texttt{)),}\ i-1)$ belongs to X_i if it belongs to the smallest model of $\Pi_i^{X_i}$.

8. Simplifications of rules of form (26) wrt. X_b viz.

$$\begin{aligned}\texttt{holds(occurs(}a\texttt{),}\ i-1) \ \texttt{:-}\ & \texttt{holds(allow(occurs(}a\texttt{)),}\ i-1),\\ & \texttt{not holds(ab(occurs(}a\texttt{)),}\ i-1),\\ & \texttt{not holds(neg(occurs(}a\texttt{)),}\ i-1).\\ \texttt{holds(neg(occurs(}a\texttt{)),}\ i-1) \ \texttt{:-}\ & \texttt{not holds(occurs(}a\texttt{),}\ i-1).\end{aligned}$$

Since (s_{i-1}, A_i, s_i) is a valid transition, we have to distinguish two cases:

(a) If $a \in A_i$, then we have $\texttt{holds(occurs(}a\texttt{),}\ i-1) \in X_i$ and $\texttt{holds(neg(occurs(}a\texttt{)),}\ i-1) \notin X_i$. Then we can simplify to

$$\begin{aligned}\texttt{holds(occurs(}a\texttt{),}\ i-1) \ \texttt{:-}\ & \texttt{holds(allow(occurs(}a\texttt{)),}\ i-1),\\ & \texttt{not holds(ab(occurs(}a\texttt{)),}\ i-1).\end{aligned}$$

It is clear that we have $\texttt{holds(ab(occurs(}a\texttt{)),}\ i-1) \notin X_i$, otherwise we won't have the given transition because an inhibition rule would be applicable (cf. Definition 5). The same holds for $\texttt{holds(allow(occurs(}a\texttt{)),}\ i-1) \in X_i$, because a must be an exogenous or explicitly allowed action (see Definition of X_i). As a consequence, X_i is closed under the considered rule. We have $\{\texttt{holds(occurs(}a\texttt{),}\ i-1) \ \texttt{:-}\ \texttt{holds(allow(occurs(}a\texttt{)),}\ i-1).\} \in \Pi_i^{X_i}$. Since we just figured out that we have $\texttt{holds(allow}$ $\texttt{(occurs(}a\texttt{)),}\ i-1) \in X_i$, $\texttt{holds(occurs(}a\texttt{),}\ i-1)$ belongs to any model of $\Pi_i^{X_i}$, in particular, the smallest one.

(b) If $a \notin A_i$, then we have $\texttt{holds(occurs(}a\texttt{),}\ i-1) \notin X_i$ and $\texttt{holds(neg(occurs(}a\texttt{)),}\ i-1) \in X_i$. We can reduce to

$$\texttt{holds(neg(occurs(}a\texttt{)),}\ i-1\texttt{).}$$

It is easy to see that X_i is closed under this rule.
Since $\texttt{holds(neg(occurs(}a\texttt{)),}\ i-1)$ belongs to any model of $\Pi_i^{X_i}$, it belongs to the smallest one.

9. If the triggering rule $(f_1, \ldots, f_n$ **triggers** $a) \in D$ was applicable (Definition 3), we get the following simplified rule of form (28) wrt. $X_b \cup X_0 \cup X_{i-1}$ viz.

$$\texttt{holds(occurs(}a\texttt{)}, i-1\texttt{)} \texttt{ :- not holds(ab(occurs(}a\texttt{))}, i-1\texttt{)}.$$

Again, two cases need to be distinguished:

(a) If $a \in A_i$, then we have $\texttt{holds(occurs(}a\texttt{)}, i-1\texttt{)} \in X_i$ and $\texttt{holds(neg(occurs(}a\texttt{))}, i-1\texttt{)} \notin X_i$. Due to the semantics given in Definition 5, we have $\texttt{holds(ab(occurs(}a\texttt{))}, i-1\texttt{)} \notin X_i$, because otherwise a would be blocked by an inhibition rule. That is, X_i is closed under the above rule. We have $\{\texttt{holds(occurs(}a\texttt{)}, i-1\texttt{) :-.}\} \in \Pi_i^{X_i}$. Thus, $\texttt{holds(occurs(}a\texttt{))}$, $i-1\texttt{)}$ belongs to any model of $\Pi_i^{X_i}$, in particular, the smallest one.

(b) If $a \notin A_i$, then we have $\texttt{holds(occurs(}a\texttt{)}, i-1\texttt{)} \notin X_i$ and $\texttt{holds(neg(occurs(}a\texttt{))}, i-1\texttt{)} \in X_i$. We have to show that $\texttt{holds(ab(occurs(}a\texttt{))}, i-1\texttt{)} \in X_i$. Due to the fact that $a \notin A_i$, we must have $a \in \overline{A}_T(s_{i-1})$ (Definition 3). The only case when considering an applicable triggering rule is that there exists an applicable inhibition rule. That means, $a \in A_I(s_{i-1})$. By construction of X_i, we have $\texttt{holds(ab(occurs(}a\texttt{))}, i-1\texttt{)} \in X_i$ and X_i is closed under the considered rule.

If the triggering rule was not applicable in s_{i-1}, then X_i is trivially closed because the above rule is not contained in Π_i.

10. If the inhibition rule $(f_1, \ldots, f_n$ **inhibits** $a) \in D$ was applicable (Definition 3), we get the following simplified rule of form (29) wrt. $X_b \cup X_0 \cup X_{i-1}$ viz.

$$\texttt{holds(ab(occurs(}a\texttt{))}, i-1\texttt{)}.$$

Since we have $a \in A_I(s_{i-1})$, we have $\texttt{holds(ab(occurs(}a\texttt{))}, i-1\texttt{)} \in X_i$. That is, X_i is closed under the considered rule and the fact $\texttt{holds(ab(occurs(}a\texttt{))}, i-1\texttt{)}$ belongs to X_i if it belongs to the smallest model of $\Pi_i^{X_i}$.

If the inhibition rule was not applicable, then the rule is not considered in Π_i at all, because at least one of $\texttt{holds(}f_1, i-1\texttt{)}, \ldots, \texttt{holds(}f_n, i-1\texttt{)}$ must be false in X_{i-1}.

11. No-concurrency constraint of form (30), viz.

$$\texttt{:- } 2\ \{\texttt{holds(occurs(}a_1\texttt{)}, i-1\texttt{)}, \ldots, \texttt{holds(occurs(}a_n\texttt{)}, i-1\texttt{)}\}$$

belongs to $\Pi_i^{X_i}$. The constraint will never be applicable, because otherwise the condition $|A_i \cap B| \leq 1$ in Definition 5 would be violated and we won't have a valid transition.

12. Rules of form (32) remain untouched in $\Pi_i^{X_i}$, viz.

$$\texttt{holds(occurs(}a\texttt{)}, i-1\texttt{)}.$$

Definition 7 states that we must have $a \in A_i$. Hence, we have $\texttt{holds(action(}a\texttt{)}, i-1) \in X_i$. Since the rule is a fact, X_i is trivially closed and the fact belongs to the smallest model of $\Pi_i^{X_i}$.

13. Simplified constraints of form (33) wrt. X_b, viz.

$$\texttt{:- holds(neg(occurs(}a\texttt{))}, i-1). \tag{62}$$

Since we have that $a \in A_i$, we have $\texttt{holds(occurs(}a\texttt{)}, i-1) \in X_i$ and $\texttt{holds(neg(occurs(}a\texttt{))}, i-1) \notin X_i$. Hence, the constraint belongs to $\Pi_i^{X_i}$ and will not be applicable.

We have shown that X_i is closed under $\Pi_i^{X_i}$ and thus a model of $\Pi_i^{X_i}$. Moreover, we have demonstrated that X_i is closed under $\Pi_i^{X_i}$ iff any smaller model of $\Pi_i^{X_i}$ is. Hence, X_i is the smallest model of $\Pi_i^{X_i}$. In other words, X_i is an answer set of $\Pi_i^{X_i}$.

C.2 Proof of Part 2 of Theorem 1

Let X be an answer set of $\Pi = \mathcal{T}(D, O_{init} \cup A_Q)$.

We have to show that there is a trajectory model $s_0, A_1, s_1, A_2, \ldots, A_m, s_m$ of $(D, O_{init} \cup A_Q)$ such that for $0 \le k \le m$

$$s_k = \{f \mid \texttt{holds}(f, k) \in X\} \cup \{\neg f \mid \texttt{holds(neg(}f\texttt{)}, k) \in X\} \tag{63}$$

$$A_{k+1} = \{a \mid \texttt{holds(occurs(}a\texttt{)}, k) \in X\}. \tag{64}$$

By the splitting sequence theorem, we have that $X = X_b \cup X_0 \cup \cdots \cup X_m$ for some solution $\langle X_b, X_0, \ldots, X_m\rangle$ of Π wrt splitting sequence $\langle U_b, U_0, U_1, \ldots, U_m\rangle$.

We distinguish the following two cases.

X_0 Consider $s_0 = \{f \mid \texttt{holds}(f, 0) \in X_0\} \cup \{\neg f \mid \texttt{holds(neg(}f\texttt{)}, 0) \in X_0\}$.

In fact, X_0 is an answer set of Program $\Pi_0 = e_{U_b}(b_{U_0}(\Pi) \setminus b_{U_b}(\Pi), X_b)$, consisting of the following types of rules:[12]

1. Simplifications of constraints of form (15) wrt X_b, viz.

$$\texttt{:- holds}(f, 0), \texttt{holds(neg(}f\texttt{)}, 0).$$

The fact that X_0 is an answer set implies that

$$\{\texttt{holds}(f, 0), \texttt{holds(neg(}f\texttt{)}, 0)\} \not\subseteq X_0. \tag{65}$$

Accordingly, we also have $\{f, \neg f\} \not\subseteq s_0$. In other words, s_0 is consistent.

2. Rules of form (16), viz.

$$\texttt{holds}(f, 0) \texttt{ :- not holds(neg(}f\texttt{)}, 0).$$

$$\texttt{holds(neg(}f\texttt{)}, 0) \texttt{ :- not holds}(f, 0).$$

[12] Recall that Π_0 consists of rules from Π that have been "evaluated" wrt X_b.

Given that (65) holds and that X_0 is an answer set, the definition of an answer set implies that either $\texttt{holds}(f, 0) \in X_0$ or $\texttt{holds}(\texttt{neg}(f), 0) \in X_0$. Accordingly, by the construction of s_0, we also have either $f \in s_0$ or $\neg f \in s_0$. In other words, s_0 is complete.

3. Simplifications of rules of form (22) wrt X_b, viz.

$$\texttt{holds}(f_i, 0)\ \texttt{:-}\ \texttt{holds}(g_1, 0), \ldots, \texttt{holds}(g_m, 0).$$

Again, the fact that X_0 is an answer set implies that it is closed under such rules. By construction of s_0, the same applies to s_0. To be precise, we have $f_i \in s_0$ whenever $\{g_1, \ldots, g_m\} \subseteq s_0$ for $1 \leq i \leq n$. Accordingly, s_0 is closed under the static causal law $(f_1, \ldots, f_n$ **if** $g_1, \ldots, g_m)$.

In all, we have shown that s_0 is a complete and consistent set of fluents being closed under all static laws in action description D.

X_i Consider a transition (s_{i-1}, A_i, s_i) with $s_i = \{f \mid \texttt{holds}(f, i) \in X_i\} \cup \{\neg f \mid \texttt{holds}(\texttt{neg}(f), i) \in X_i\}$ and $A_i = \{a \mid \texttt{holds}(\texttt{occurs}(a), i-1) \in X_i\}$. X_i is an answer set of Program $\Pi_i = e_{U_{i-1}}(b_{U_i}(\Pi) \setminus b_{U_{i-1}}(\Pi), X_{i-1})$.

We have to show the following things: s_i is consistent, complete and closed under the static rules. Furthermore, all conditions in Definition 5 must be satisfied. At first, for every $\texttt{holds}(f, i) \in X_i$ we have to find an explanation due to Definition 4 wrt. its value in s_{i-1}. This means that

$$s_i = E(A_i, s_{i-1}) \cup L(s_i) \cup \Delta(s_i) \cup (s_{i-1} \cap s_i) \tag{66}$$

must be satisfied.

The next part is to show that the remaining conditions in Definition 5 hold:

$$A_T(s_{i-1}) \subseteq A_i \tag{67}$$

$$\overline{A}_T(s_{i-1}) \cap A_i = \emptyset \tag{68}$$

$$\overline{A}_A(s_{i-1}) \cap A_i = \emptyset \tag{69}$$

$$A_I(s_{i-1}) \cap A_i = \emptyset \tag{70}$$

$$|A_i \cap B| \leq 1 \text{ for all } (\textbf{noconcurrency } B) \in D(A, F). \tag{71}$$

The last part is to show that the transition is indeed a part of the trajectory model, that is, the conditions in Definition 7 due to the given observations in A_Q must be satisfied.

1. Π_i contains constraints of form (15) simplified by X_b, viz.

$$\texttt{:-}\ \texttt{holds}(f, i), \texttt{holds}(\texttt{neg}(f), i).$$

Since X_i is an answer set, we have

$$\{\texttt{holds}(f, i), \texttt{holds}(\texttt{neg}(f), i)\} \not\subseteq X_i. \tag{72}$$

Thus we can conclude that we have $\{f, \neg f\} \not\subseteq s_i$ and s_i is consistent.

2. To show that s_i is complete we have to prove that we have at least $\texttt{holds}(f, i) \in X_i$ or $\texttt{holds}(\texttt{neg}(f), i) \in X_i$. We have to distinguish the following cases:

   (a) If we have $\texttt{default}(f) \in X_b$ then Π_i contains a rule of form (18) simplified by X_b, viz.

   $$\texttt{holds}(f, i) \;\texttt{:-}\; \texttt{not}\; \texttt{holds}(\texttt{neg}(f), i).$$

   It is easy to see that X_i contains $\texttt{holds}(f, i)$ if it does not contain $\texttt{holds}(\texttt{neg}(f), i)$. By construction of s_i and given that (72) is satisfied we have either $f \in s_i$ or $\neg f \in s_i$.

   (b) If we have $\texttt{default}(f) \notin X_b$ then Π_i contains rules of form (20). To write down the simplified rules by $X_b \cup X_0 \cup X_{i-1}$ we have to distinguish different cases.

      i. $\texttt{holds}(f, i-1) \in X_{i-1}$:

      $$\texttt{holds}(f, i) \;\texttt{:-}\; \texttt{not}\; \texttt{holds}(\texttt{neg}(f), i).$$

      The definition of an answer set implies that we have at least $\texttt{holds}(f, i) \in X_i$ or $\texttt{holds}(\texttt{neg}(f), i) \in X_i$ when considering only this rule in Π_i. By construction of s_i, and again given that (72) is satisfied, we have either $f \in s_i$ or $\neg f \in s_i$.

      ii. $\texttt{holds}(\texttt{neg}(f), i-1) \in X_{i-1}$:

      $$\texttt{holds}(\texttt{neg}(f), i) \;\texttt{:-}\; \texttt{not}\; \texttt{holds}(f, i).$$

      The argumentation is the same as in the previous case, we have either $f \in s_i$ or $\neg f \in s_i$.

   Since every fluent must be a default or inertial, we now have shown that s_i is complete.

3. Consider the following rule of form (22) wrt. X_b, viz.

   $$\texttt{holds}(f_k, i) \;\texttt{:-}\; \texttt{holds}(g_1, i), \ldots, \texttt{holds}(g_n, i).$$

   Every answer set X_i of Π_i contains $\texttt{holds}(f_k, i)$ if it contains $\texttt{holds}(g_1, i), \ldots, \texttt{holds}(g_n, i)$. Hence, by construction of s_i we have $s_i \models g_l$ for $1 \leq l \leq n$ and as a consequence, $s_i \models f_k$. That is, s_i is closed under the static rules.

4. We now assume that we have $\texttt{holds}(f, i) \in X_i$. Two cases need to be distinguished. To give an intuition, we are looking whether the value of f changes between s_{i-1} and s_i or not. If it does not change, either no static or dynamic law was applicable and therefore the value of f is defined by default or inertia in s_i. If it changes, then f must be a direct effect applying a dynamic law, or indirect effect applying a static rule that was

not applicable in s_{i-1} but in s_i. If none of them can be applied, a default must have been fired.

(a) $\texttt{holds}(f, i-1) \in X_{i-1}$: By construction of s_{i-1} and s_i we have that $f \in s_{i-1}$ and $f \in s_i$. Different cases need to be distinguished, since there are different rules in Π_i with $\texttt{holds}(f, i)$ in their head.

i. Rule of form (18) simplified by X_b, viz.

$$\texttt{holds}(f, i) \texttt{ :- not holds(neg(}f\texttt{)}, i).$$

We have to ensure that no rule with head $\texttt{holds(neg(}f\texttt{)}, i)$ is applicable.[13] For rule (20) this is the case, since we have $\texttt{default}(f) \in X_b$ and it is not contained in Π_i at all.
In Rule (22) at least one of $\texttt{holds}(g_1, i), \ldots, \texttt{holds}(g_n, i)$ needs to be false in X_i, so we have wlog. $\texttt{holds}(g_1, i) \notin X_i$.[14] As a consequence, we have $\neg f \notin L(s_i)$.
Rule (23) requires at least one of $\texttt{holds}(g_1, i-1), \ldots,$ $\texttt{holds}(g_n, i-1)$ to be false in X_{i-1}. Thus, we have wlog. $\texttt{holds}(g_1, i-1) \notin X_{i-1}$ and the rule is not contained in Π_i at all.[15]
We can conclude $\neg f \notin E(A_i, s_{i-1})$, $f \in (s_{i-1} \cap s_i)$ and $f \in \Delta(s_i)$. That is, (66) is satisfied.

ii. Rule of form (20) simplified by $X_b \cup X_0 \cup X_{i-1}$, viz.

$$\texttt{holds}(f, i) \texttt{ :- not holds(neg(}f\texttt{)}, i).$$

At first, we can conclude $\texttt{default}(f) \notin X_B$, otherwise the rule won't be contained in Π_i. That is, we have $f \notin \Delta(s_i)$ since f is no default. As one can see, the resulting rule is exactly the same as in the previous case. Hence, we have $\neg f \notin L(s_i)$, $\neg f \notin E(A_i, s_{i-1})$, $f \in (s_{i-1} \cap s_i)$ and (66) satisfied.

iii. Rules of form (22) wrt. X_b, viz.

$$\texttt{holds}(f_k, i) \texttt{ :- holds}(g_1, i), \ldots, \texttt{holds}(g_n, i).$$

Since X_i is an answer set and the rule was built from a static causal law, we can conclude $f \in L(s_i)$. Given that (72) is satisfied, we can conclude $\neg f \notin E(A_i, s_{i-1})$ from rule (23), $\neg f \notin L(s_i)$ from rule (22) and $\neg f \notin \Delta(s_i)$ from rule (18). $\neg f \notin (s_{i-1} \cap s_i)$ is satisfied since we are in the case that $\texttt{holds}(f, i) \in X_i$, hence $\neg f \notin s_i$. Thus, (66) is satisfied.

iv. Rule of form (23) simplified by $X_b \cup X_0 \cup X_{i-1}$, viz.

$$\texttt{holds}(f_k, i) \texttt{ :- holds(occurs(}a\texttt{)}, i-1).$$

[13] Remember that for rules of form (22) and (23) also rules with negative heads are contained in the encoding if there are dynamic or static laws with negative heads given in the action description D.

[14] This means that no static rule with $\neg f$ in its head was applicable.

[15] This means that no dynamic law with $\neg f$ in its head was applicable.

If this rule is satisfied, we have $a \in A_i$ and $f \in E(A_i, s_{i-1})$. Again, given that (72) is satisfied, we can conclude $\neg f \notin (s_{i-1} \cap s_i)$, $\neg f \notin E(A_i, s_{i-1})$, $\neg f \notin L(s_i)$, $\neg f \notin \Delta(s_i)$. That is, (66) is satisfied.

(b) $\texttt{holds(neg(}f\texttt{)}, i-1) \in X_{i-1}$: We are now concerning the fact that the value of $\texttt{holds(}f, i-1)$ resp. $\texttt{holds(}f, i)$ changes between X_{i-1} and X_i. By construction of s_{i-1} and s_i we have that $\neg f \in s_{i-1}$ and $f \in s_i$. Again, we have to consider all rules with $\texttt{holds(}f, i)$ in their head.

i. Rule of form (18) simplified by X_b, viz.

$$\texttt{holds(}f, i) \texttt{ :- not holds(neg(}f\texttt{)}, i).$$

Since this rule was built from a default rule in D, we can conclude $f \in \Delta(s_i)$. Since (72) is satisfied, we have $\neg f \notin \Delta(s_i)$, $\neg f \notin E(A_i, s_{i-1})$, $\neg f \notin L(s_i)$. By construction of s_i and s_{i-1} we have $\neg f \notin (s_{i-1} \cap s_i)$.[16] Thus, (66) is satisfied.

ii. Rules of form (20) simplified by $X_b \cup X_0 \cup X_{i-1}$. Since we have $\texttt{holds(}f, i-1) \notin X_{i-1}$ and $\texttt{holds(}f, i) \in X_i$, none of the rules is contained in $\Pi_i^{X_i}$ at all. That is, $\texttt{holds(}f, i)$ can't be derived by these rules and we do not have to consider this case.

iii. Rules of form (22) wrt. X_b, viz.

$$\texttt{holds(}f_k, i) \texttt{ :- holds(}g_1, i), \ldots, \texttt{holds(}g_n, i).$$

We have $\texttt{holds(}g_1, i), \ldots, \texttt{holds(}g_n, i) \in X_i$ and therefore $s_i \models g_l$ for $1 \leq l \leq n$. Since this rule was built from a static rule in D, we can conclude $f \in L(s_i)$. Since (72) is satisfied, we have $\neg f \notin L(s_i)$, $\neg f \notin E(A_i, s_{i-1})$, $\neg f \notin \Delta(s_i)$. Thus, (66) is satisfied.

iv. Rule of form (23) simplified by $X_b \cup X_0 \cup X_{i-1}$, viz.

$$\texttt{holds(}f_k, i) \texttt{ :- holds(occurs(}a\texttt{)}, i-1).$$

We can conclude $f \in E(A_i, s_{i-1})$ since the encoded dynamic law must have been applicable in s_{i-1}. By (72) we have $\neg f \notin E(A_i, s_{i-1})$, $\neg f \notin \Delta(s_i)$, $\neg f \notin L(s_i)$ and thus, (66) is satisfied.

5. In case that we have $\texttt{holds(neg(}f\texttt{)}, i) \in X_i$ the argumentation is done analogously.
6. We now show that the equations (67) to (71) are satisfied. At first, we show that (71) is satisfied. The fact that X_i is an answer set of Π_i implies that the constraint (30) is satisfied:

$$\texttt{:- } 2\,\{\texttt{holds(occurs(}a_1\texttt{)}, i-1), \ldots, \texttt{holds(occurs(}a_n\texttt{)}, i-1)\}$$

That is, at most one of $\texttt{holds(occurs(}a_1\texttt{)}, i-1), \ldots, \texttt{holds(occurs(}a_n\texttt{)}, i-1)$ is true in X_i. By construction of A_i, (71) is satisfied.

[16] We will skip this condition in the following cases.

For every $a \in A_i$ we have $\texttt{holds(occurs}(a), i-1) \in X_i$. To show that (67) to (70) are satisfied, we distinguish between the following rules:

(a) Rules of form (26) wrt. X_b viz.

$$\texttt{holds(occurs}(a), i-1) \texttt{ :- holds(allow(occurs}(a)), i-1),$$
$$\texttt{not holds(ab(occurs}(a)), i-1),$$
$$\texttt{not holds(neg(occurs}(a)), i-1).$$

The fact that $\texttt{holds(occurs}(a), i-1) \in X_i$ implies that at least rule (24) or rule (25) was applicable, because these are the rules allowing for deriving $\texttt{holds(allow(occurs}(a)), i-1)$.

i. The case that rule (25) was applicable needs not to be considered, since then we have $a \notin A_T(s_{i-1})$, $a \notin \overline{A}_T(s_{i-1})$, $a \notin \overline{A}_A(s_{i-1})$, $a \notin A_A(s_{i-1})$ and $a \notin A_I(s_{i-1})$.[17]

ii. Recall rule (24) simplified by X_b:

$$\texttt{holds(allow(occurs}(a)), i-1)$$
$$\texttt{:- not holds(ab(occurs}(a)), i-1),$$
$$\texttt{holds}(f_1, i-1), \ldots, \texttt{holds}(f_n, i-1).$$

If this rule is applicable in Π_i, we have $\texttt{holds}(f_1, i-1), \ldots,$ $\texttt{holds}(f_n, i-1) \in X_{i-1}$. We can conclude $s_{i-1} \models f_1 \wedge \cdots \wedge f_n$ and $a \in A_A(s_{i-1})$ (Definition 3) since this rule was built from an allowance rule.[18] We have $\texttt{holds(ab(occurs}(a)), i-1) \notin X_i$ and can conclude $a \notin \overline{A}_A(s_{i-1})$, since rule (29) was not applicable. That is, (69) is satisfied.

We have $\texttt{holds(ab(occurs}(a)), i-1) \notin X_i$. The only rule where $\texttt{holds(ab(occurs}(a)), i-1)$ can be derived is rule (29). Rule (29) was built from an inhibition rule and since the rule was not applicable in Π_i we have $a \notin A_I(s_{i-1})$. By construction of A_i, (70) is satisfied.

(b) Rule of form (28) wrt. $X_b \cup X_0 \cup X_{i-1}$ viz.

$$\texttt{holds(occurs}(a), i-1) \texttt{ :- not holds(ab(occurs}(a)), i-1).$$

Since we have $\{\texttt{holds}(f_1, i-1), \ldots, \texttt{holds}(f_n, i-1)\} \in X_{i-1}$ and this rule was built from a triggering rule, we can conclude $a \in A_T(s_{i-1})$ and because of $\texttt{holds(ab(occurs}(a)), i-1) \notin X_i$ we have $a \notin \overline{A}_T(s_{i-1})$. By construction of A_i, equations (67) and (68) are satisfied.

As in the previous case, we have $\texttt{holds(ab(occurs}(a)), i-1) \notin X_i$. Again, we can conclude $a \notin A_I(s_{i-1})$. Hence, (70) is satisfied.

[17] Recall that rule (25) is only contained in Π_i if a is an exogenous action, that is $\{(f_1, \ldots, f_n \textbf{ allows } a), (f_1, \ldots, f_n \textbf{ triggers } a)\} \cap D = \emptyset$.

[18] We will shorten this argumentation in the next cases by simply saying if the rules were applicable or not applicable.

7. What remains left to show is that in (s_{i-1}, A_i, s_i) the observations in A_Q are satisfied (Definition 7). If a is an exogenous action, the fact that X_i is an answer set of Π_i implies that we have $\texttt{holds(occurs(}a\texttt{)}, i-1) \in X_i$ since rule (32) is a fact:

$$\texttt{holds(occurs(}a\texttt{)}, i-1).$$

By construction of A_i, we have $a \in A_i$.
If a is no exogenous action, the simplified constraint (33) must be satisfied:

$$\texttt{:- holds(neg(occurs(}a\texttt{))}, i-1). \tag{73}$$

Since rule (27) must be satisfied we can conclude $\texttt{holds(neg(occurs(}a\texttt{))}, i-1) \notin X_i$ and $\texttt{holds(occurs(}a\texttt{)}, i-1) \in X_i$, otherwise the constraint (33) won't be satisfied. By construction of A_i, we have $a \in A_i$.

We now have shown that for every answer X set of $\Pi = \mathcal{T}(D, O_{init} \cup A_Q)$ there is a trajectory model $s_0, A_1, s_1, A_2, \ldots, A_m, s_m$ of $(D, O_{init} \cup A_Q)$ such that for $0 \leq k \leq m$ we have

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

$$A_{k+1} = \{a \mid \texttt{holds(occurs(}a\texttt{)}, k) \in X\}.$$

References

1. Backofen, R., Will, S., & Bornberg-Bauer, E. (1999). Application of constraint programming techniques for structure prediction of lattice proteins with extended alphabets. *Bioinformatics, 15*(3), 234–242.
2. Baral, C. (2003). *Knowledge representation, reasoning and declarative problem solving*. Cambridge University Press.
3. Baral, C., Brewka, G., & Schlipf, J. (Eds.) (2007). *Proceedings of the ninth international conference on logic programming and nonmonotonic reasoning (LPNMR'07)*. *Lecture Notes in Artificial Intelligence*, Vol. 4483. Springer-Verlag.
4. Baral, C., Chancellor, K., Tran, N., Tran, N., Joy, A., & 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*) (pp. 15–22). ISCB.
5. Bonarius, H. P. J., Schmid, G., & Tramper, J. (1997). Flux analysis of underdetermined metabolic networks: The quest for the missing constraints. *Trends in Biotechnology, 15*, 308314.
6. Booth, E. J., Walker, K. C., & Griffiths, D. W. (1991). A time-course study of the effect of sulfur on glucosinolates in oilseed rape (brassica napus) from the vegetative stage to maturity. *Journal of the Science of Food and Agriculture, 56*, 479–493.
7. Chabrier-Rivier, N., Fages, F., & Soliman, S. (2004). The biochemical abstract machine biocham. In V. Danos & V. Schächter (Eds.) *Proceedings of the second workshop on computational methods in systems biology* (pp. 172–191). Springer.
8. Clarke, E., Grumberg, O., & Peled, D. (1999). *Model checking*. MIT Press.
9. Danos, V., & Schächter, V. (Eds.) (2005). *Computational methods in systems biology, international conference* (*CMSB 2004*)*, Paris, France, May 26–28, 2004, revised selected papers*. *Lecture Notes in Computer Science*, Vol. 3082. Springer.
10. Eiter, T., Faber, W., Leone, N., Pfeifer, G., & Polleres, A. (2003). A logic programming approach to knowledge-state planning. *Artificial Intelligence, 144*(1–2), 157–211.
11. Eiter, T., Faber, W., Leone, N., Pfeifer, G., & Polleres, A. (2004). A logic programming approach to knowledge-state planning: Semantics and complexity. *ACM Transactions On Computational Logic, 5*(2), 206–263.

12. Eker, S., Knapp, M., Laderoute, K., Lincoln, P., & Talcott, C. L. (2002). Pathway logic: Executable models of biological networks. *Electr. Notes Theor. Comp. Sci., 71*.
13. Eveillard, D., Ropers, D., de Jong, H., Branlant, C., & Bockmayr, A. (2004). A multi-scale constraint programming model of alternative splicing regulation. *Theoretical Computer Science, 325*(1), 3–24.
14. Fanchon, E., Corblin, F., Trilling, L., Hermant, B., & Gulino, D. (2004). Modeling the molecular network controlling adhesion between human endothelial cells: Inference and simulation using constraint logic programming. In V. Danos & V. Schächter (Eds.), *Lecture Notes in Bioinformatics—Computational Methods in Systems Biology* (pp. 104–118).
15. Ferraris, P., & Lifschitz, V. (2005). Mathematical foundations of answer set programming. In S. Artëmov, H. Barringer, A. d'Avila Garcez, L. Lamb & J. Woods (Eds.), *We will show them! Essays in honour of dov gabbay*, (Vol. 1, pp. 615–664). College Publications.
16. Gebser, M., Kaufmann, B., Neumann, A., & Schaub, T. (2007). Clasp: A conflict-driven answer set solver. In C. Baral, G. Brewka & J. Schlipf (Eds.), *Proceedings of the ninth international conference on logic programming and nonmonotonic reasoning* (*LPNMR'07*). *Lecture Notes in Artificial Intelligence*, (Vol. 4483, pp. 260–265). Springer-Verlag.
17. Gebser, M., Kaufmann, B., Neumann, A., & Schaub, T. (2007). Conflict-driven answer set solving. In M. Veloso (Ed.), *Proceedings of the twentieth international joint conference on artificial intelligence* (*IJCAI'07*) (pp. 386–392). AAAI Press/The MIT Press. Available at http://www.ijcai.org/papers07/contents.php.
18. Gebser, M., Schaub, T., & Thiele, S. (2007). GrinGo: A new grounder for answer set programming. In C. Baral, G. Brewka & J. Schlipf (Eds.), *Proceedings of the ninth international conference on logic programming and nonmonotonic reasoning* (*LPNMR'07*). *Lecture Notes in Artificial Intelligence*, (Vol. 4483, pp. 266–271). Springer-Verlag.
19. Gelfond, M., & Lifschitz, V. (1991). Classical negation in logic programs and disjunctive databases. *New Generation Computing, 9*, 365–385.
20. Gelfond, M., & Lifschitz, V. (1993). Representing action and change by logic programs. *Journal of Logic Programming, 17*(2–4), 301–321.
21. Gelfond, M., & Lifschitz, V. (1998). Action languages. *Electronic Transactions on Artifical Intelligence, 3*(6), 193–210.
22. Giunchiglia, E., & Lifschitz, V. (1998). An action language based on causal explanation: Preliminary report. In *Proceedings of the national conference on artificial intelligence* (*AAAI*) (pp. 623–630). AAAI.
23. Halkier, B. A., & Gershenzon, J. (2006). Biology and biochemistry of glucosinolates. *Annual Review of Plant Biology, 57*, 303–333.
24. Kutz, A., Müller, A., Hennig, P., Kaiser, W. M., Piotrowskiand, M., & Weiler, E. W. (2002). A role for nitrilase 3 in the regulation of root morphology in sulphur-starving arabidopsis thaliana. *Plant Journal, 30*, 95–106.
25. Levesque, H., Reiter, R., Lespérance, Y., Lin, F., & Scherl, R. (1997). Golog: A logic programming language for dynamic domains. *Journal of Logic Programming, 31*(1–3), 59–83.
26. Lifschitz, V., & Razborov, A. (2006). Why are there so many loop formulas? *ACM Transactions on Computational Logic, 7*(2), 261–268.
27. Lifschitz, V., & Turner, H. (1994). Splitting a logic program. In *Proceedings of the eleventh international conference on logic programming* (pp. 23–37). MIT Press.
28. Lifschitz, V., & Turner, H. (1999) Representing transition systems by logic programs. In M. Gelfond, N. Leone & G. Pfeifer (Eds.), *Logic programming and non-monotonic reasoning*. *Lecture Notes in Artificial Intelligence*, (Vol. 1730, pp. 92–106). Springer-Verlag.
29. Lynce, I., & Marques-Silva, J. (2006). Efficient haplotype inference with boolean satisfiability. In Y. Gil & R. Mooney (Eds.), *Proceedings of the twenty-first national conference on artificial intelligence* (*AAAI'06*). AAAI Press.
30. Lynce, I., & Marques-Silva, J. (2006). Sat in bioinformatics: Making the case with haplotype inference. In A. Biere & C. Gomes (Eds.) *Proceedings of the ninth international conference on theory and applications of satisfiability testing* (*SAT'06*). *Lecture Notes in Computer Science*, (Vol. 4121, pp. 136–141). Springer-Verlag.
31. McCarthy, J. (1998). Elaboration tolerance. http://www.formal.stanford.edu/jmc/elaboration.html.
32. Montiel, G., Gantet, P., Jay-Allemand, C., & Breton, C. (2004). Transcription factor networks. Pathways to the knowledge of root development. *Plant Physiology, 136*, 3478–3485.
33. Niemelä, I. (1999). Logic programs with stable model semantics as a constraint programming paradigm. *Annals of Mathematics and Artificial Intelligence, 25*(3–4), 241–273.

34. Nikiforova, V. J., Daub, C. O., Hesse, H., Willmitzer, L., & Hoefgen, R. (2005). Integrative gene-metabolite network with implemented causality deciphers informational fluxes of sulfur stress response. *Journal of Experimental Botany, 56*, 1887–1896.
35. Nikiforova, V. J., Freitag, J., Kempa, S., Adamik, M., Hesse, H., & Hoefgen, R. (2003). Transcriptome analysis of sulfur depletion in arabidopsis thaliana: Interlacing of biosynthetic pathways provides response specificity. *Plant Journal, 33*, 633–650.
36. Nikiforova, V. J., Kopka, J., Tolstikov, V., Fiehn, O., Hopkins, L., Hawkesford, M. J., et al. (2005). Systems re-balancing of metabolism in response to sulfur deprivation, as revealed by metabolome analysis of arabidopsis plants. *Plant Physiology, 138*, 304–318.
37. Pan, Y., Tu, P., Pontelli, E., & Son, T. (2004). Construction of an agent-based framework for evolutionary biology: A progress report. In J. Leite, A. Omicini, P. Torroni & P. Yolum (Eds.), *Proceedings of the second international workshop on declarative agent languages and technologies* (*DALT'04*). *Lecture Notes in Computer Science*, (Vol. 3476, pp. 92–111). Springer-Verlag.
38. Papatheodorou, I., Kakas, A., & Sergot, M. (2005). Inference of gene relations from microarray data by abduction. In C. Baral, G. Greco, N. Leone & G. Terracina (Eds.), *Proceedings of the eighth international conference on logic programming and nonmonotonic reasoning* (*LPNMR'05*). *Lecture Notes in Artificial Intelligence*, (Vol. 3662, pp. 389–393). Springer-Verlag.
39. Pinney, J. W., Westhead, D. R., & McConkey, G. A. (2003). Petri net representations in systems biology. *Biochemical Society Transactions, 31*(Pt 6), 1513–1515.
40. Reddy, V. N., Mavrovouniotis, M. L., & Liebman, M. N. (1993). Petri net representations in metabolic pathways. In *Proc. First ISMB* (pp. 328–336).
41. Regev, A., Panina, E. M., Silverman, W., Cardelli, L., & Shapiro, E. Y. (2004). Bioambients: An abstraction for biological compartments. *Theoretical Computer Science, 325*(1), 141–167.
42. Shmulevich, I., Dougherty, E. R., Kim, S., & Zhang, W. (2002). Probabilistic boolean networks: A rule-based uncertainty model for gene regulatory networks. *Bioinformatics, 18*(2), 261–274.
43. Simons, P., Niemelä, I., & Soininen, T. (2002). Extending and implementing the stable model semantics. *Artificial Intelligence, 138*(1–2), 181–234.
44. Son, T., & Pontelli, E. (2007). Planning for biochemical pathways: A case study of answer set planning in large planning problem instances. In M. De Vos & T. Schaub (Eds.), *Proceedings of the workshop on software engineering for answer set programming* (*SEA'07*), number CSBU-2007-05 in Department of Computer Science, University of Bath, Technical Report Series, (pp. 116–130). ISSN 1740-9497.
45. Syrjänen, T. (1998). Lparse 1.0 user's manual. http://www.tcs.hut.fi/Software/smodels/lparse.ps.gz.
46. Tamaddoni-Nezhad, A., Chaleil, R., Kakas, A., & Muggleton, S. (2006). Application of abductive ilp to learning metabolic network inhibition from temporal data. *Machine Learning, 64*(1–3), 209–230.
47. Tran, N. (2006). *Reasoning and hypothesing about signaling networks*. Ph.D. thesis, Arizona State University.
48. Tran, N., & Baral, C. (2004). Reasoning about triggered actions in AnsProlog and its application to molecular interactions in cells. In D. Dubois, C. Welty & M. Williams (Eds.), *Proceedings of the ninth international conference on principles of knowledge representation and reasoning* (*KR'04*) (pp. 554–564). AAAI Press.
49. Tran, N., Baral, C., & Shankland, C. (2005). Issues in reasoning about interaction networks in cells: Necessity of event ordering knowledge. In M. Veloso & S. Kambhampati (Eds.), *Proceedings of the twentieth national conference on artificial intelligence* (*AAAI'05*) (pp. 676–681). AAAI Press.