Introduction to the action language $\mathcal{C}_{TAID}$ as an approach for representing and reasoning about biological networks

Susanne Grell
University of Potsdam

June 6, 2006

Contents

Chapter 1

Biological Background

1.1 Biological example

In the following chapters, especially in Chapter 2 and 3, we will refer to a biological example to analyse and understand the features and problems of the discussed approaches and translations (for details see [12, 13]).

An overview of this small biological network is given in this section. An extended version of the network is explained in detail in Chapter 4.

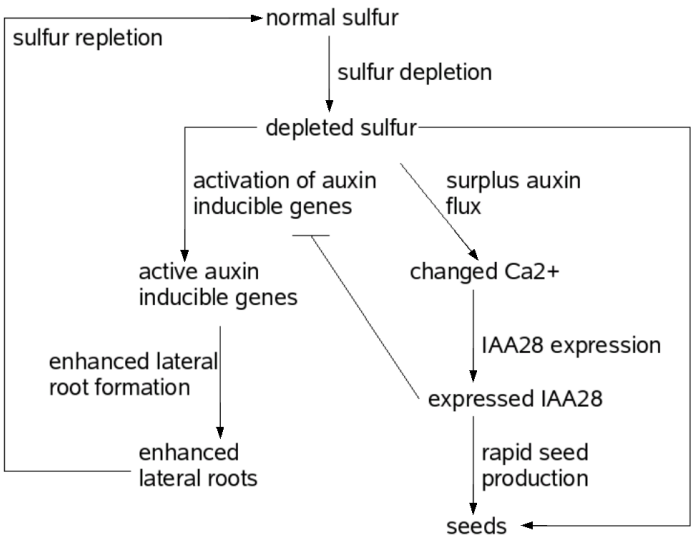

Figure 1.1: Sulfur starvation response-pathway of Arabidopsis thaliana

Figure 1.1 shows a graphical representation of the causal relationships of the sulfur starvation response-pathway of Arabidopsis thaliana.

Sulfur is essential for Arabidopsis thaliana. If the amount of sulfur Arabidopsis thaliana can access is not sufficient to allow a normal development of the plant, the plant follows a complex strategy. First the plant forms additional lateral roots to access additional sources of sulfur and to normalise its sulfur level. However, if this strategy is not successful the plant uses its remaining resources to form seeds. A simplified version of this network is shown in Figure 1.1.

Normally, the amount of sulfur in a plant will be sufficient, but due to external, e.g. environmental conditions, the amount of sulfur can be reduced. A problem, when modelling this network are these environmental conditions, which are not and cannot be part of such 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, which plays a key role as a signal in coordinating the development of the plant. This eventually leads to the formation of additional lateral roots. Since this consumes the scarce resources, this development should be stopped, when it becomes apparent, that it will not be successful, i.e. it takes to long and consumes too many of the plant's resources.

This "emergency stop" is triggered by complex interactions that lead, via a surplus of the auxin flux, to the expression of IAA28. The gene IAA28 is a member of the auxin/IAA family. It appears to prevent or delay the formation of lateral roots [9].

If IAA28 is expressed and the sulfur level is still low, other processes will result in a different physiological endpoint, the production of seeds.

The interactions mentioned here, are very complex and only a small part is included in this model, to keep it concise, which eases the understanding of the following examples.

Chapter 2

Action Languages

In the previous chapters we explained the biological problems we are focussing on and discussed Petri Nets, an approach which is currently used to model biological networks. In this chapter, we propose action languages as an alternative approach. The application of action languages for modelling biological networks was first introduced by Baral and Tran in [11, 2].

We will give an introduction to action languages and present the action language $\mathcal{C}_{TAID}$, which is especially designed for the representation of knowledge about biological networks.

The action language $\mathcal{C}_{TAID}$ consists of three sublanguages. The first sublanguage is the action description language, which is used to describe the general knowledge about a dynamic system. A second sublanguage, called action observation language, allows to specify knowledge about particular situations. The third sublanguage is referred to as action query language. It allows to express queries for planning, explanation and prediction. These queries are used to reason about the dynamic system.

After giving a short historical introduction in Section 2.1, we introduce the different action languages and expressions supported in this thesis. They are explained in the sections 2.2.1, 2.2.2 and 2.2.3. In Section 2.2.5, 2.2.6 and 2.2.7 the different kinds of possible queries are explained in detail, also including small examples.

Formal definitions for interpretations and models are given in Section 2.2.2. An interesting and important question is whether it is possible to decide what a good model is and possibly even what the best model is.

Section 2.2.4 deals with the notion of time and how it is used in the current context.

2.1 History

2.1.1 Fluents and Actions

Action languages are based on the concepts of fluents and actions, a thorough introduction can be found in [10]. Fluents have a value, which can change in time. They can be seen as objects, attributes, conditions or the like, which can have specific values. Every day examples could be: the fluent colour has the value blue, the fluent window has the value open. In the action signature employed in this approach, there are only two possible values: $true$ and $false$, which represent the corresponding boolean truth values.

The value depends on time. That means at different points of time a fluent can have different values. How the value of a fluent can change and how it can be changed, depends on the fluent. Usually, a fluent changes its value as a direct effect of executing an action.

Actions can influence the values of the fluents. Coming back to the every day example, the action "close window" could change the value of the fluent window from open to closed or using fluents with boolean truth values, the fluent "window closed" would change its value from $false$ to $true$ and the fluent "window open" would change its value from $true$ to $false$.

It is possible to change the value of a fluent by executing an action. But the question is, obviously, which action influences which fluent in which way. These questions are discussed in the following sections. One important aspect is the problem of deciding the current value of a fluent, especially if no actions executed before had any influence on the value of a particular fluent. Often it is assumed that the value of a fluent does not normally change, this is referred to as the frame problem.

We can distinguish two kinds of fluents, inertial fluents and non-inertial fluents. Inertial fluents keep the value they had, unless they are affected by an action or by other fluents. Non-inertial fluents can change their value without such a reason.

Since the concept of both kinds of fluents, inertial and non-inertial, are useful for biological systems, both kinds can be expressed in the action description language proposed in Section 2.2.1. In general non-inertial fluents do not need to have a default value. Nevertheless, in the action description language proposed here, there is the restriction, that non-inertial fluents alway require a default value, so that their values change in a predefined way.

A default value is used to express that a fluent has normally a certain value, unless it is known to have a different one.

Especially in biological systems, which are by itself very complex and usually not every aspect is modelled, default values are very useful, for example

when it comes to modelling metabolites that are present in high concentrations or environmental condition which usually do not change.

2.1.2 Action signatures and transition systems

An action signature defines the alphabet, that can be used to describe the knowledge for example about a biological network, i.e. names of fluents and actions. Additionally, it defines the possible values, which can be assigned to the fluents. Obviously the action signature differs for every network. The definitions given in [5] are adapted to the needs of the action language proposed here.

Definition 2.1 (Action signature)**.** *Let A, F and W be non-empty sets of action names, fluent names and value names. An action signature is the triple $\langle A, F, W \rangle$.*

Informally, fluents, which are represented by symbols from F, can be assigned values, which are symbols from W. In the following only propositional action signatures are considered, i.e. $W = \{true, false\}$. The execution of actions, represented by symbols from A, can change the state of the world.

In the following, when talking about fluent names and action names, they will for simplicity be referred to as fluents and actions. A fluent literal is either a fluent f or its negation $\neg f$.

When we talk about a state, we refer to a complete and consistent set of fluents, i.e. a value is assigned to every fluent defined by the action signature.

Definition 2.2 (Transition system)**.** *Let $\langle A, F, W \rangle$ be an action signature, S be a set of states, and $V : F \times S \to W$ and $\Phi : A \times S \to S$ be functions. Then (S, V, Φ) is a transition system for an action signature $\langle A, F, W \rangle$.*

In general a transition system can be seen as a graph, where the states of the transition system are the nodes and the arcs are defined by the transition relation.

There are many different action description languages described in the literature (see [1, 5, 6]).

2.1.3 Action Description Language $\mathcal{A}$

The first high-level action language introduced in the literature, is the action language $\mathcal{A}$ of Gelfond and Lifschitz, (see [4]). It is a very simple language which only supports one sort of expressions, but it is a good start to understand the intuition behind action description languages.

An action description is a set of expressions of the form

$$(A \textbf{ causes } L \textbf{ if } F)$$

Here, A is an elementary action, L and F are sets of fluent literals. The set L is also called the head of the expression. If F is the empty set, the if-part is simply omitted and written as:

$$(A \textbf{ causes } L)$$

Such a causal law describes the effects of an action A on a set of fluents F if the preconditions in L hold. But an important and not easily answered question is: "What happens to those fluents not affected by the causal law?" The solution provided by the action description language $\mathcal{A}$ is very simple, every fluent not affected by the causal law remains unchanged.

In the language $\mathcal{A}$ fluents can only have two possible values, $true$ and $false$, that means $W = \{true, false\}$ So when specifying the transition system for the action description language $\mathcal{A}$, the transition system (S, V, Φ) is defined as follows:

- The set of states S are all possible interpretations of F, i.e. all possible combinations of truth assignments. The number of states is $2^{|F|}$.
- $V(f, s)$ gives the value of fluent f in state s.
- The resulting state s' is defined by the transition relation Φ. If s' exists, $\Phi(A, s)$ has to satisfy the following condition (see [5]):

  $$E(A, s) \subseteq s' \subseteq E(A, s) \cup s$$

  where $E(A, s) = \{L \mid (A \textbf{ causes } L \textbf{ if } F), F \in s\}$

Intuitively, Φ defines s' as the state, which contains all effects of an action A when executed in a state s. All fluents which are not affected by the action A will have the same value in s' as in s. This implies, that all fluents are considered to be inertial.

For every action A and every state s there is at most one state s' which satisfies the transition relation. That means the action language $\mathcal{A}$ is deterministic (see [5] for details).

2.1.4 Action Description Language $\mathcal{C}$

The action language $\mathcal{C}$ is an extension of action language $\mathcal{A}$. It provides additional language expression; besides direct effects of actions it is possible to describe dependencies between fluents. The possible evolution of the world

also changes, because fluents are no longer implicitly inertial, that means they might change their value even if they are not affected by an action.

The major difference, however, is the notion of causality used in $\mathcal{C}$. It clearly distinguishes between assuming that a fact holds and knowing that a fact was caused, (see [5, 6]). Moreover the concurrent execution of actions is possible.

In the following a formula $U = f_1 \wedge \ldots \wedge f_n \wedge a_1 \wedge \ldots \wedge a_m$ is a propositional combination of actions and fluents. The sets F and G are sets of fluents.

Language $\mathcal{C}$ provides two different kinds of expressions. A *static causal law*

*(***caused** F **if** G*)*

expresses that there is a reason or a cause for F to be true if G holds in that state.

A *dynamic causal law*

*(***caused** F **if** G **after** U*)*

describes the effects of the actions contained in U. Informally, it means that F is caused to hold in s' if all fluents in U hold in s, all actions in U occur, and if the successor state s' satisfies G. In both kinds of laws the if part can be omitted, if G is *true*.

Such dynamic causal laws are, for example, used to express inertial behaviour of fluents, to define default values for other fluents or to restrict the executability of actions (for details see [5]).

The transition system of action language $\mathcal{C}$ is more restricted than the one of action language $\mathcal{A}$.

- The set of states S is the set of interpretations of the set of all fluents F which satisfy all static causal laws. That means, whenever the preconditions G of a static causal law hold in a state s also the head F of the static causal law holds in s.
- $V(f, s)$ gives the value of fluent f in state s
- All transitions $\Phi(A, s) = s'$ have to be causally explained. That is, if s' exists, it is the interpretation of F that satisfies all static and dynamic laws and thus satisfies all fluents which are caused in that transition (see [7] for details).

In the context of action language $\mathcal{C}$ the notion of exogenous actions[1] was introduced. Such actions are assumed to have a cause, but this cause is external and thus not modelled by the given propositions.

[1] CCalc Tutorial: http://www.cs.utexas.edu/users/tag/cc/tutorial/toc.html

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

2.2.1 Action Description Language

The action description language presented here provides a framework to describe biological networks, their components and the relationships and interactions between these components. Although there are expressions which are especially useful to express knowledge about a biological network the language itself is not limited to this application.

It adopts the notion of causality of the action language $\mathcal{C}$. That means every fluent needs to have a reason for its current value and every action has to have a cause for its occurrence. The action language $\mathcal{C}_{TAID}$ allows to express triggering (T), allowance (A) and inhibition (I) relationships and it is possible to define default values (D).

Syntax

In the following, the different expressions provided by the action description language are introduced, their syntax and also their intended meaning.

In the following $f, f_1, \ldots, f_n, g_1, \ldots, g_n$ are used as symbols for fluent literals, a is used as an action symbol. A fluent literal is either a fluent (e.g. f) or a negated fluent (e.g. $\neg f$)

As mentioned before, fluents represent the current state of the world and actions can influence this state by changing the values of the fluents. There can be numerous and complex indirect effects which might or might not be modelled or even known. But in general, when it comes to the question how fluents can be influenced, we first think about direct effects of actions. For example during a chemical reaction the reactants will be decreased and the products will be increased as direct effects of this reaction

Example 2.1. Let us consider the small example given in Figure 2.1. The action signature $\langle A, F, W \rangle$ of this example is the following:
$W = \{true, false\}$
$F = \{normal\_sulfur, depleted\_sulfur, active\_auxin\_inducible\_genes,$
$enhanced\_lateral\_roots, changed\_ca2, expressed\_iaa28, seeds\}$
$A = \{sulfur\_depletion, activation\_of\_auxin\_inducible\_genes,$
$enhanced\_lateral\_root\_formation, sulfur\_repletion, surplus\_auxin\_flux,$
$iaa28\_expression, rapid\_seed\_production\}$ •

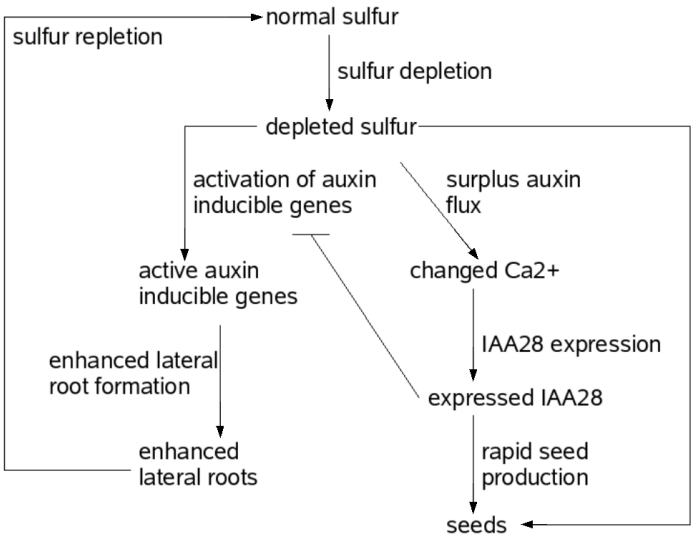

Figure 2.1: Sulfur starvation response-pathway of Arabidopsis thaliana

Dynamic causal laws can be used to express this kind of direct causal relationships. The basic form of a dynamic causal law looks like:

$$(a \textbf{ causes } f_1, \ldots, f_n) \tag{2.1}$$

The action could for example be a chemical reaction or a process in a biological system. Even though, there may exist this direct relationship, the outcome often depends on various additional conditions that could include environmental conditions or the absence or presence of other fluents, for example the energy level has to be high enough or oxygen has to be available or there is no sulfur present in the system.

Although a dynamic causal law cannot be used to express whether or not an action can occur, it can, however, be used to restrict the effects of an action to certain situations. That means, an action may only have an effect if certain preconditions hold. These preconditions are specified in the if-part of the dynamic causal law, which results in an extended dynamic causal law of the form:

$$(a \textbf{ causes } f_1, \ldots, f_n \textbf{ if } g_1, \ldots, g_m) \tag{2.2}$$

It is also possible to specify more than one dynamic causal law for an action. This is very helpful for situations when an action may have different effects depending on the current state of the system. That supersedes additional auxiliary actions.

Example 2.2. For our running example of the sulfur starvation network, we can specify several dynamic causal laws.

(sulfur_depletion **causes** *depleted_sulfur* **if** *normal_sulfur)*
(activation_of_auxin_inducible_genes **causes** *active_auxin_inducible_genes)*
(enhanced_lateral_root_formation **causes** *enhanced_lateral_roots)*
(sulfur_repletion **causes** *normal_sulfur)*
(surplus_auxin_flux **causes** *changed_ca2_level)*
(iaa28_expression **causes** *expressed_iaa28)*
(rapid_seed_production **causes** *seeds)*

•

Static causal laws are used to describe dependencies between fluents, which are expressions of the form:

$$(f_1, \ldots, f_n \textbf{ if } g_1, \ldots, g_m) \tag{2.3}$$

Although we are primarily interested in causal knowledge, there are still different ways to understand such a static causal law (see [8]). Possible meanings are:

1. the fact that $g_1, \ldots, g_m$ holds causes the fact that $f_1, \ldots, f_n$ holds
2. if $g_1, \ldots, g_m$ holds, then the fact that $f_1, \ldots, f_n$ holds is necessarily caused

The first point claims a causal relationship, i.e. $g_1, \ldots, g_m$ are the reason why also $f_1, \ldots, f_n$ hold in this state. That might be the intended meaning but usually the second sentence is more appropriate. It asserts $f_1, \ldots, f_n$ will always hold if $g_1, \ldots, g_m$ hold and that there is indeed a reason for that, but the reason is not specified.

Although the second sentence is weaker than the first one, it often models the rather limited knowledge about a system more adequately, because complex relationships are often not known or are difficult to include in the model. Or simply because it is an external reason which is not part of the model.

The set $f_1, \ldots, f_n$, specified in the static causal law, can also be seen as indirect effects of a set of actions which have the direct effects $g_1, \ldots, g_m$. Such indirect effects could also be specified as direct effects, but that, besides being not very appropriate, might require a large number of additional dynamic causal laws to model the desired effects.

Example 2.3. In the sulfur starvation network, there is an implicit dependency between the fluents *normal_sulfur* and *depleted_sulfur*. Only one of the two fluents can be *true* at a time, though it is possible that both are $false$, in case the level of sulfur is increased. This background knowledge can be modelled, using the following static causal laws.

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

•

Occurrences of actions. Very important to realise is the fact that dynamic causal laws only specify the causal effects of an action they cannot be used to express *if* or *when* an action will occur. To express that an action may only occur under certain conditions, the three expressions, explained below, can be used.

The fact that an action is always *immediately* executed if certain conditions hold can be expressed by using a triggering rule.

$$(f_1, ..., f_n \textbf{ triggers } a) \tag{2.4}$$

This rule says that in a state where $f_1, ..., f_n$ hold and action a is not inhibited by any other rule, action a has to occur.

Example 2.4. In our sulfur starvation network, we have several triggered actions. The last rule includes several preconditions:

(depleted_sulfur **triggers** *activation_of_auxin_inducible_genes)*
(active_auxin_inducible_genes **triggers**
enhanced_lateral_root_formation)
(changed_ca2_level **triggers** *iaa28_expression)*
(expressed_iaa28 , depleted_sulfur **triggers**
rapid_seed_production)

•

The notion of triggered actions was already introduced in [11] and in [2]. Declaring an action to be a triggered action is most useful if the dependencies and the circumstances under which this action occurs are well understood. But often the information is incomplete, either because we do not know, i.e. the knowledge about the system is incomplete, or because we do not care,

i.e. not all information is included in the model. For example, due to the complex relationships of biological systems it might be the case, that certain conditions for an action to occur are known but others are not. Additionally, due to the necessary discretisation, fluents have to be mapped to boolean truth values and all actions are considered to require the same amount of time. That means, for example, that all reaction rates are considered to be equal, but this is obviously not true. Even though it is not possible to include quantitative knowledge, it should still be possible to restrict the situations when an action can occur but at the same time allow delays or a choice for the occurrence of this action.

For this purpose, allowance rules can be used.

$$(f_1, ..., f_n \textbf{ allows } a) \tag{2.5}$$

This rule expresses the fact that action a can - but not necessarily has to - occur in every state where $f_1, ..., f_n$ hold. On the other hand, if the allowance rule is not satisfied the action cannot occur. This allows to elegantly exclude unwanted occurrences of actions.

Example 2.5. In our sulfur starvation network, it is useful to introduce three allowed actions by the following rules:

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

The first rule is necessary, since *normal_sulfur* is a precondition for the action to occur, but it is not the actual reason. The reasons are complex and not included in the network, for example, the soil might not contain a sufficient amount of sulfur.

The second rule is included, to allow an alternative pathway, which leads to a different physiological endpoint, namely the production of seeds. But this should only happen, if the enhanced lateral roots failed to increase the amount of sulfur. The allowance rule is used to model the choice when the action happens.

The third rule is used to model the incomplete knowledge about the reasons, for the success or failure to increase the level of sulfur, by forming additional lateral roots.

•

The expressions so far can be used to state when an action can or has to occur. But there might be situations where an action is not allowed to occur, therefore inhibition rules can be used.

$$(f_1, ..., f_n \textbf{ inhibits } a) \tag{2.6}$$

Such an inhibition rule specifies that the action a will not occur in any state where $f_1, ..., f_n$ hold.

Example 2.6. In our running example, there is only the following inhibition relation:

(expressed_iaa28 **inhibits** *activation_of_auxin_inducible_genes)*

•

Actions for which no triggering or allowance rule is defined, are considered to be exogenous actions. They can happen at any point of time without any further reasons why they happen, as long as they are not inhibited. Nevertheless, these action do have a reason, but this reason is external and not part of the model.

The action a, for which such a triggering, inhibition or allowance rule is defined, is also called the head of the rule. If we want to refer to the set of all triggering (inhibition or allowance) rules with action a as its head, we call it the triggering (inhibition or allowance) rules of action a.

Concurrency. Especially biological networks are characterised by a high degree of concurrency, on the other hand, it is possible that certain actions cannot happen concurrently in a particular situation. That might be due to the fact that these reactions or processes share the same resources or maybe because they require opposing environmental conditions, which are usually not included in the model, for example heat and cold. The no-concurrency rules can be used to express this knowledge.

$$(\textbf{noconcurrency } a_1, ..., a_n) \tag{2.7}$$

The rule lists all actions $a_1, \ldots, a_n$ that cannot happen concurrently, that means, only one action $a_i \in \{a_1, ..., a_n\}$ can occur at one point of time.

Example 2.7. Since resources are scarce, they should be used either for forming roots or to produce seeds, but not both at the same time. This can be expressed by the following constraint:

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

•

Default values. Usually the values of fluents are supposed to change only when they are the effects dynamic or of static causal laws. But especially in biological systems the notion of default values is quite useful. It can be used to say that something has always a specific value, unless explicitly stated otherwise.

$$(\textbf{default } f) \tag{2.8}$$

In this context f is a fluent literal, i.e. a fluent possibly preceded by negation. If the fluent is negated, its default value is $false$ otherwise it is $true$.

Example 2.8. Only if the auxin inducible genes are affected by an action they are active, otherwise they are inactive. In our example we also define a default value for *enhanced_lateral_roots*, in order to efficiently model the inhibition relation and its side effects.

*(***default** ¬*active_auxin_inducible_genes)*
*(***default**¬*enhanced_lateral_roots)*

Once the activation of auxin inducible genes is inhibited, resources are not used to form any additional lateral roots. •

A common phenomena in biological systems is oscillation. Default values can be used to efficiently model this behaviour, for example the continuous increase and decrease of the concentration of some metabolite.

Example 2.9. The following three rules, describe a possible way to model oscillation. This example is not part of our running example.

(¬*high_concentration* **triggers** *increase_concentration)*
(increase_concentration **causes** *high_concentration)*
*(***default** ¬*high_concentration)*

The default value of the fluent is $false$. This value triggers the action *increase_concentration* , which causes the fluent to have the value $true$. Once the value changed to $true$ no action can be executed and in the next state the value of the fluent *high_concentration* falls back to the default value, and the cycle starts again.

The oscillation can be stopped by introducing an inhibition rule for the action. •

Semantics

After defining the syntax and explaining the intended meaning of expressions provided by the action description language we will now formally define its semantics.

Therefore, we first define a domain description, which represents the modelled biological system in the syntax of the action description language, consisting of expressions of the form (2.1) to (2.8).

Definition 2.3 (Domain description)**.** *Let $\langle A, F, \{true, false\}\rangle$ be an action signature.*

A set of rules in the syntax of the action description language $\mathcal{C}_{TAID}$, which uses only symbols from the sets A and F, is called a domain description $D(A, F)$.

When discussing the semantics of this action language some informal assumptions should be kept in mind (see also [3]):

- the only actions that can occur are those given by the language of the domain description,
- actions have only those effects which are defined by the dynamic causal laws of the domain description, and
- all actions are assumed to have the same duration.

The domain description defines a transition system (S, V, Φ), as defined in Definition 2.2. An interpretation of F defines a value for every fluent $f \in F$.

Definition 2.4 (State)**.** *Let (S, V, Φ) be the transition system described by the domain description $D(A, F)$.*

A state $s \in S$ is an interpretation of F such that for every static causal law

$$(f_1, \ldots, f_n \textbf{ if } g_1, \ldots, g_n)$$

defined in $D(A, F)$, we have $\{f_1, \ldots, f_n\} \subseteq s$ whenever $\{g_1, \ldots, g_n\} \subseteq s$.

The definition implies that a state s of the domain description is uniquely determined by a complete and consistent set of fluents.

The states of such a transition system are all possible interpretations of this domain description, which satisfy all static causal laws. The upper bound of the number of states is $2^{|F|}$ since the values of the fluents can only be true or false.

To properly understand what can be expressed and how it can be expressed in this action language, it is necessary to correctly understand under which conditions an action can or has to occur and when it cannot occur.

Definition 2.5. *Let $D(A, F)$ be a domain description and s a state of $D(A, F)$.*

1. *An inhibition rule (f_1, ... , f_n **inhibits** a) is active, if for all fluent literals $s \models f_1 \wedge \ldots \wedge f_n$, otherwise the inhibition rule is passive.*

   The set $A_I(s)$, is the set of those actions for which there exists at least one active inhibition rule in s.

2. *A triggering rule (f_1, ... , f_n **triggers** a) is active, if for all fluent literals $s \models f_1 \wedge \ldots \wedge f_n$, and all inhibition rules of action a are passive, otherwise the triggering rule is passive.*

   The set $A_T(s)$, is the set of those actions for which there exists at least one active triggering rule in s.

   The set $\overline{A}_T(s)$, is the set of those actions for which there exists at least one triggering rule and all triggering rules are passive in s.

3. *An allowance rule (f_1, ... , f_n **allows** a) is active, if for all fluent literals $s \models f_1 \wedge \ldots \wedge f_n$, and all inhibition rules of action a are passive, otherwise the allowance rule is passive.*

   The set $A_A(s)$, is the set of those actions for which there exists at least one active allowance rule in s.

   The set $\overline{A}_A(s)$, is the set of those 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, ... , f_n **if** g_1, ... , g_n) is applicable in a state s, if for all fluent literals $s \models g_1 \wedge \ldots \wedge g_n$.*

5. *A static causal law (f_1, ... , f_n **if** g_1, ... , g_n) is applicable in a state s, if for all fluent literals $s \models g_1 \wedge \ldots \wedge g_n$.*

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

The effects of an action are determined by the applicable dynamic causal laws defined for this action.

As defined in [11], if D is a domain description, the direct effects of an action a are:

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

The effects of a set of actions A is defined as the union of the effects of the single actions.

$$E(A, s) = \bigcup_{a \in A} E(a, s)$$

Obviously, the set of effects can be inconsistent, i.e. contain f and $\neg f$ for some fluent f. Depending on the dynamic causal laws even the effects of a single actions can be inconsistent. That causes problems because the next state cannot be determined.

Besides the direct effects of actions, a domain description also defines the consequences of static relationships between fluents. For a set L of static causal laws given by the domain description D 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.

Recall, that fluents for which a default value is defined, change their value to be the default value, even if they are not affected by a static or dynamic causal law.

Now, we want to know how the world evolves when an action occurs. Therefore, following and extending the definition given in [8], the transition relation is defined, which helps to determine to which state the execution of no, one or several actions leads.

Definition 2.6. *Let $D(A, F)$ be a domain description and S be the set of states of $D(A, F)$. Then, the transition relation $\Phi \subseteq S \times 2^A \times S$ determines the resulting state $s' \in S$ after executing all actions $B \subseteq A$ in state $s \in S$ as follows:*

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

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

If there are no actions 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.

Example 2.10. Consider the following domain description:

(a **causes** *f)*	*(b* **causes** $\neg g$*)*	*(h* **if** *f)*
($\neg f$ **allows** *a)*	*(f* **inhibits** *a)*	*(g* **triggers** *b)*
*(***default** *g)*		

The transition diagram, which is defined by the transition relation Φ, is shown in Figure 2.2. The oscillation, described by g and $\neg g$, is clearly represented in the diagram.

Note that the interpretations $\{f, \neg g, \neg h\}$ and $\{f, g, \neg h\}$ do not satisfy the static causal law and are thus no states of this domain description.

The Δ occurring in the transition diagram, is used to represent, that no action occurs and that the change of state is due to the application of default rules.

•

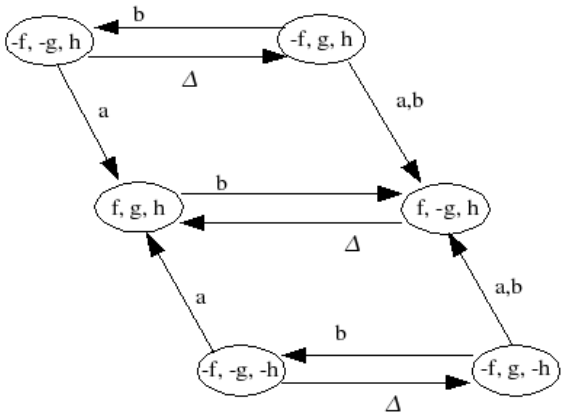

Figure 2.2: State transition diagram of Example 2.10.

The transition relation determines the resulting state when an action is executed, but it cannot be used to decide whether the action can happen at all, since it does not consider triggering, allowance or inhibition rules. A trajectory is a sequence of states and actions that takes all rules, laws and constraints given by the domain description correctly into account.

Recall Definition 2.5 for the definition of the sets $A_T(s_i)$, $\overline{A}_T(s_i)$, $\overline{A}_A(s_i)$ and $A_I(s_i)$ and the definition of active and passive triggering, allowance and inhibition rules.

Definition 2.7 (Trajectory)**.** *Let $D(A, F)$ be a domain description. A trajectory $\tau = s_0, A_1, s_1, \ldots, A_n, s_n$ of the domain description $D(A, F)$ is a sequence of actions and states where $A_i \subseteq A$ $(1 \leq i \leq n)$ and the following conditions are satisfied for $0 \leq i < n$:*

1. $(s_i, A, s_{i+1}) \in \Phi$
2. $A_T(s_i) \subseteq A_{i+1}$
3. $\overline{A}_T(s_i) \cap A_{i+1} = \emptyset$
4. $\overline{A}_A(s_i) \cap A_{i+1} = \emptyset$
5. $A_I(s_i) \cap A_{i+1} = \emptyset$
6. *all actions in A_i can occur concurrently, i.e. for all*

   *(***noconcurrency** *A)*

   it holds that $|A \cap A_i| \leq 1$

A trajectory assures that there is a reason why an action occurs or why it does not occur. The second and third point of the definition make sure that the actions of all active triggering rules are included in the set of actions and that no action for which all triggering rules are passive is included in the set of actions. Point three and four assure that no actions for which all allowance rules are passive and no inhibited actions are included in the set of actions. The definition does not include assertions about the active allowance rules, because they can be all, but not necessarily have to be all, included in the set of actions.

The last point of the definition assures, that all non-concurrency constraints are correctly applied.

Example 2.11. For the domain description given in Example 2.10 we have an infinite number of possible trajectories, which is due to the oscillation.

We have the following states:

$s_1 = \{\neg f, \neg g, h\}$ $s_2 = \{\neg f, g, h\}$ $s_2 = \{f, g, \neg h\}$ $s_4 = \{f, g, h\}$
$s_5 = \{f, \neg g, h\}$ $s_6 = \{f, \neg g, \neg h\}$ $s_7 = \{\neg f, \neg g, \neg h\}$ $s_8 = \{\neg f, g, \neg h\}$

Some possible trajectories are:

$\tau_1 = s_1, \{\}, s_2, \{b\}, s_1, \{a\}, s_4, \{b\}, s_5, \{\}, s_4, \{b\}, s_5, \{\}, \ldots$
$\tau_1 = s_8, \{a, b\}, s_5, \{\}, s_4, \{b\}, s_5, \{\}, s_4, \{b\}, \ldots$

•

2.2.2 Action Observation Language

The action description language defined in the previous section is used to define the general knowledge about a biological system. It describes the fluents, e.g. metabolites, genes or environmental conditions, and the relationships between them. But it does not include expressions to describe the current state of the system, to describe what fluents are present and just as important which ones are not. Nor is there a way to describe observed actions or actions that should occur.

Syntax

The action observation language provides a framework to express observations and assumptions. Observations can be made about fluents and about occurrences of actions. But unlike the general knowledge defined in the knowledge base, observations are associated with points of time.

Examples of such observations could be "at the beginning there was enough sulfur", "although the serine level was normal at t_i it was observed to be increased at t_j" or "in the final state additional lateral roots where formed".

The following rule is used to express observations about fluents.

$$(f \textbf{ at } t_i) \tag{2.9}$$

In this rule f is a fluent literal, i.e. a fluent possibly preceded by negation to be able to express the knowledge that a fluent is not present in the state, i.e. has the value $false$, at time t_i. The initial point of time is referred to as t_0.

How "time" in this context should be understood is explained in more detail in Section 2.2.4.

Observations about occurrences of actions are expressed using the following rule:

$$(a \textbf{ occurs\_at } t_i) \tag{2.10}$$

For t_i holds the same as explained above. The symbol a stands for an action, this action might be preceded by a negation symbol. Stating that an action does not occur is usually not very useful, because it contains only very limited information, especially if there are a lot of possible actions. But on the other hand, it might be used selectively, to eliminate unwanted solutions.

Example 2.12. Observations about our sulfur starvation network could include the following observation about a fluent and the occurrence of an action.

(normal_sulfur **at** t_0*)*
(sulfur_depletion **occurs_at** t_0*)*

•

Semantics

A set of observations about a state might include information about the values of all fluents or only about the values of some fluents at a certain point of time. Obviously, if such a set contains only observations about some fluents for one point of time, it is possible that the same set describes several states, which differ only from the fluents not included in the observations.

It is often useful if the initial state is completely described by observations. This idea was for example specified in [11] or [1].

The initial state is here referred to by the symbol s_0.

Definition 2.8. *Let $D(A, F)$ be a domain description. A set O of observations containing either (f* **at** *t_0) or ($\neg f$* **at** *t_0) for every $f \in F$ is called initial state complete.*

The initial state described by a set of initial state complete observations is the state where:

- *$f \in s_0$ iff (f* **at** *t_0) $\in O$*
- *$\neg f \in s_0$ iff ($\neg f$* **at** *t_0) $\in O$*

The idea is to use the action description language to describe the general knowledge about the system or network. This description specifies how the system can evolve over time. By including observations the possibilities of this evolution are restricted. So only when all information, the domain description and the observations, is taken into account, we get an appropriate picture of the world. The combination of domain description and observations is called an action theory.

Definition 2.9 (Action theory)**.** *Let D be a domain description and O be a set of observations. The pair (D, O) is called an action theory. If the set of observations O is initial state complete the action theory is initial state complete.*

We already introduced trajectories. Intuitivelly, they specify possible evolutions of the system with respect to the given domain description. Usually not all trajectories satisfy the observations given by an action theory. Now we are looking only for those trajectories which satisfy both, the domain description and the observations given by an action theory. These trajectories are called trajectory models of the action theory.

Definition 2.10 (Trajectory model)**.** *Let (D, O) be an action theory and $\tau = s_0, A_1, s_1, A_2, \ldots, A_n, s_n$ be a trajectory of the domain description D that satisfies all observations in O in the following way:*

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

Then τ is a trajectory model of the action theory (D, O).

Such a model describes correctly the possible evolution of the world. Because, since it is a trajectory, it satisfies, among others, all dynamic and static causal laws and all triggering and allowance rules under consideration of the given observations. Often, there are several different models for an action theory and a set of observations. They describe different paths through the transition system, contain actions in different orders and give different possible ways, how the world could evolve. The number of such models can become exceedingly large, whcih is appropriate due to the complexity inherent in biological systems. But many of these models contain redundant information. Minimal models are a subclass of these models, where the number of redundant actions is minimised. The occurrence of an action in a state is considered to be redundant for example if there are no applicable dynamic causal laws for this action in this state or if the effects of all applicable dynamic causal laws for this action already hold in this state.

The problem that arises here, is to find biologically meaningful models. So we are not only interested in the shortest path through the transition system, but also in, possibly longer, alternative paths and just as well in models which include the concurrent execution of actions. To decide which actions are redundant is thus a rather difficult problem and the question whether a model is biologically meaningful can only be answered by a biologist, not by an automated reasoner.

Despite these problems Tran and Baral proposed in [11] an ordering of trajectories which allows to descide which of the trajectory models are minimal. This ordering allows to compare different trajectory models which have the same initial state. A trajectory model is minimal if it is minimal in the number of states and the sets of actions are minimal with respect to set minimality.

This notion of minimality might exclude biologically meaninful and possibly very interesting models. Especially in biological systems the shortest path might not always be the best or the most likely one. That means, we first have to find an answer to the question what a good model of a biological network is, before we can define it in terms of minimal trajectory models. Finding a solution for this problem is not part of this thesis.

A question we can already answer is the question of logical consequence of observations. That means, given the explicit knowledge specified by an action theory what information is implicitly given by it.

Definition 2.11. *Let (D, O) be an action theory. If $f \in s_i$ holds for all trajectory models $\tau = s_0, A_1, \ldots, s_i, A_{i+1}, \ldots, A_n, s_n$, then (D, O) entails the fluent observation (f* **at** *t_i), written $(D, O) \models$ (f* **at** *t_i).*

If $a \in A_{i+1}$ holds for all trajectory models $\tau = s_0, A_1, \ldots, s_i, A_{i+1}, \ldots, A_n, s_n$, then (D, O) entails the action observation (a **occurs_at** *t_i), written $(D, O) \models$ (a* **occurs_at** *t_i).*

Given a particular trajectory model it can be completely described using fluent and action observations. But even if we have several trajectory models, it is possible that some observations are entailed by all of them. This notion of entailment is used to verify the different queries introduced in the next sections.

Example 2.13. Consider the observations:

(normal_sulfur **at** *t_i)*
(sulfur_depletion **occurs_at** *t_i)*

Given the dynamic causal law *(sulfur_depletion* **causes** *¬normal_sulfur)*, a logical consequence is the observation *(¬normal_sulfur* **at** *t_{i+1})* •

2.2.3 Action Query Language

Now after specifying the knowledge about the system, we are trying to model, we want to use this knowledge to reason about the system. Reasoning includes explaining of observed behaviour, but also predicting the futur development of the system or how the system may be influenced in a particular way. There are numerous questions one might want to find an answer for.

Syntax

Queries are about the evolution of the world, i.e. about trajectories. What queries are possible and how they can be transcribed is explained in more detail in sections 2.2.5, 2.2.6, and 2.2.7

In general, a query is of the form:

$$(f_1, \ldots, f_n \textbf{ after } A_1 \textbf{ at } t_1, \ldots, A_m \textbf{ at } t_m) \tag{2.11}$$

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

Semantics

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

Definition 2.12. *Let (D, O) be an action theory and Q be the query*

$$Q = (f_1, \ldots, f_n \textbf{ after } A_1 \textbf{ at } t_1, \ldots, A_m \textbf{ at } t_m)$$

If every trajectory model $\tau = s_0, A'_1, s_1, A'_2, \ldots, A'_p, s_p$ of (D, O) satisfies $A_i \subseteq A'_i$ $(0 < i \leq m)$ and $s_p \models f_1 \wedge \ldots \wedge f_n$. Then $f_1 \wedge \ldots \wedge f_n$ are cautiously entailed, written $(D, O) \models_c Q$.

If there is at least one model which satisfies these conditions, then $f_1 \wedge \ldots \wedge f_n$ are bravely entailed, written $(D, O) \models_b Q$.

Observe that the fluents $f_1, \ldots, f_n$ specified by the query do not have to hold immediately at time t_m, they only have to hold in the last state of the trajectory model. For example, there could be some triggering rules which have to be applied in state s_m and this might consequently lead to the desired result.

Both kinds of reasoning, brave reasoning and cautious reasoning, can be useful depending on the situation. When verifying whether a certain state is reachable given the current knowledge cautious reasoning is the most appropriate method. But on the other hand, brave reasoning allows to check whether a particular assignment for the fluents is possible and the trajectory model would specify how it could be achieved.

2.2.4 Time

The notion of time used in this approach is quite abstract. It is independent of real time points or time intervals. Time is understood more as a means to be able to specify sequences of actions, i.e. to be able to express that an action occurred before or after another action, or to state that actions happened simultaneously.

This is not a disadvantage since duration times of chemical reactions, the time necessary to reach an equilibrium or similar aspects modelled in a biological network, are usually not or only partly known. Considering the lack of information, simulating "real" time would be a very demanding and for this approach unnecessary task.

It is more useful to comprehend time here, as an enumeration of interesting events, which give us a snapshot of the system at different points of

time. An interesting event is the occurrence of one or more actions or the change of a value of a fluent.

Everyone might have a different intuition what time in this context means. The main idea is certainly to model the relative relationships between the actions, not the absolute time points of the events.

2.2.5 Planning

In planning, we try to find possibilities to influence a system in a certain way. Questions that might be answered using planning could be the following: how to make components of a biological system behave in a particular way or how to influence components of a biological system to achieve a particular state.

A plan is a sequence of actions, which have to be executed at certain points of time to reach a goal state, starting from the initial state.

Neither the initial state nor the goal state have to be completely specified by fluent observations. A plan is thus a sequence of actions starting from one possible initial state and ending at one possible goal state. There are usually several plans, taking into account different paths but also different initial and goal states.

More formally, we are trying to find a set of action occurrences satisfying the following problem:

$$(D, O_{init} \cup \{(A_1 \textbf{ occurs\_at } t_1), \ldots, (A_m \textbf{ occurs\_at } t_m)\}) \models f_1, \ldots, f_n$$

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

Definition 2.13 (Plan)**.** *Let (D, O_{init}) be an a action theory such that O_{init} contains only fluent observations about the initial state,*

$$Q = (f_1, \ldots, f_n)$$

be a query and $\tau = s_0, A_1, s_1, A_2, \ldots, A_m, s_m$ be a trajectory model of (D, O_{init}).

Then

$$P = \{(A_1 \textbf{ occurs\_at } t_1), \ldots, (A_m \textbf{ occurs\_at } t_m)\}$$

is a plan for the goal $G = f_1 \wedge \ldots \wedge f_n$, if $s_m \models f_1 \wedge \ldots \wedge f_n$

A plan is always derived from the corresponding trajectory model. The trajectory model assures that all triggering, allowance, inhibition and default rules are applied correctly and that all no-concurrency constraints are taken into account.

Example 2.14. Consider our running example of the sulfur starvation pathway. Assume we have the following initial state complete set of observations:

(¬ normal_sulfur **at** t_0*)* *(depleted_sulfur* **at** t_0*)*
(¬active_auxin_inducible_genes **at** t_0*)* *(¬enhanced_lateral_roots* **at** t_0*)*
(¬normal_sulfur **at** t_0*)* *(¬changed_ca2* **at** t_0*)*
(¬expressed_iaa28 **at** t_0*)* *(¬seeds* **at** t_0*)*

For goal $G = \{normal\_sulfur\}$ and time bound $n = 3$ we find, for example, the solution:

$P = \{$*(activation_of_auxin_inducible_genes* **occurs_at** t_0*),*
(enhanced_lateral_root_formation **occurs_at** t_1*),*
(sulfur_repletion **occurs_at** t_2*)*$\}$

•

2.2.6 Explanation

Usually, we are more interested in understanding the observed behaviour of a system than in finding a plan to cause certain behaviour of the system. These observations usually include values of fluents at the beginning and at the end of an experiment. Of course, they can also include observations about occurrences of actions. These observations could include the knowledge that actions happened concurrently or sequentially.

When explaining such observations, we try to fill the gaps, to find the missing links. An explanation describes the actions that have to occur, actions that might occur and values of fluents of the initial state, which were not specified by fluent observations.

However, when explaining observed behaviour it is not necessary to completely define the initial state, nor the final state. The less information is provided the more possible explanation there are, because an explanation is one path from one possible initial state to one possible final state, via some possible intermediate partially defined states given by the observations.

More formally, a set of action occurrences is an explanation, if it satisfies:

$$(D, O \cup \{(A_1 \textbf{ occurs\_at } t_1), \ldots, (A_m \textbf{ occurs\_at } t_m)\}) \models \textbf{true}$$

In contrast to planning the observations can include observations about fluents and actions at different points of time not only about the initial state. The actions included in the sets $A_1, \ldots, A_m$ have to happen at time points $t_1, \ldots, t_m$.

Definition 2.14 (Explanation)**.** *Let (D, O) be an a action theory, and let $\tau = s_0, A_1, s_1, A_2, \ldots, A_m, s_m$ be a trajectory model of (D, O).*

Then

$$E = \{(A_1 \textbf{ occurs\_at } t_1), \ldots, (A_m \textbf{ occurs\_at } t_m)\}$$

is an explanation for the set of observations O

Such sequences are possible evolutions of the world.

Example 2.15. Considering the sulfur starvation network, we are looking for an explanation of the following two fluent observations:

$$(normal\_sulfur \textbf{ at } t_0)$$
$$(depleted\_sulfur \textbf{ at } t_1)$$

Obviously, an action occurred, since the only action that has *depleted_sulfur* as an effect is *sulfur_depletion* there is only one explanation for these fluent observation.

$$E = \{(sulfur\_depletion \textbf{ occurs\_at } t_0)\}$$

•

2.2.7 Prediction

When predicting a set of fluents, queries are considered to be hypothetical. Such queries ask generally: if some actions were executed at certain points of time would that mean that some fluents will have particular values? Prediction is mainly used to determine the influence of an action or a set of actions on the system.

Formally, a prediction answers the question whether:

$$(D, O) \models f_1, \ldots, f_n$$

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

Prediction is a kind of hypothetical reasoning, it tries to answer questions about the possible evolution of the system. A query answers the question whether, starting at the current state and executing a given sequence of actions, fluents will hold or not hold after a certain time. As in explanation the gaps have to be filled, to justify the assumed observations.

After determining all possible trajectory models that fulfil the observations, it is possible to decide the prediction $P = f_1, \ldots, f_n$. Three different outcomes can be distinguished.

1. The prediction could be true in all possible trajectory models,

2. or it could only be true in some trajectory models,

3. or it could be true in none trajectory model.

The first case means, that no matter which actions are executed, as long as they do not contradict the observations or assumptions, it has no influence on the final values of the fluents of interest. This is usually referred to as cautious reasoning, because it has to hold in all trajectory models.

Another possibility is brave reasoning, where it is enough to find one trajectory model in which P holds to decide that the prediction is true. This might save computation time, because as soon as a solution is found the computation can be stopped. But on the other hand it is a very vague form of prediction, since it could predict both P and $\neg P$.

Definition 2.15 (Prediction)**.** *Let (D, O) be an action theory and*

$$Q = (f_1, \ldots, f_n)$$

be a query. If Q holds in the final states of all trajectory models of the action theory, i.e. $\{f_1, \ldots, f_n\} \in s_m$ holds for all trajectory models $\tau = s_0, A_1, s_1, A_2, \ldots, A_m, s_m$, it is called a cautious prediction.

If Q holds in the final states of at least one trajectory model of the action theory, it is called a brave prediction.

Example 2.16. Consider the following observation about the sulfur starvation network:

$$(\neg \textit{depleted\_sulfur}\ \textbf{at}\ t_1)$$

The prediction of $Q = \{\textit{enhanced\_lateral\_roots}\}$ holds in every trajectory model. On the other hand the prediction $Q = \{\textit{seeds}\}$ does not hold in any trajectory model, since there is no observation about the occurrence of the action *surplus_auxin_flux* or any other action or fluent that requires the occurrence of this action. But that would be necessary to trigger the production of seeds.

•

Chapter 3

Translations

This chapter presents two translations from action language $\mathcal{C}_{TAID}$ into logic programs under the answer set semantics. Translation π_1 is given in Section 3.1 and π_2 in Section 3.2.

The idea of translating action languages into logic programs is not new, there are several papers and books about this topic, see for example [4] or [1].

However, since $\mathcal{C}_{TAID}$ provides a large variety of features, especially designed for modelling biological networks, there is not yet a translation which can capture the semantics correctly. Two solutions for this problem are given in this chapter.

In the following sections the symbols $f, f_1, \ldots, f_n, g_1, \ldots, g_n$ are used as fluents and the symbol a is used to represent an action.

The translations result in normal logic programs with variables, e.g for fluents, actions or points of time.

3.1 Translation π_1

The translation given in this section is based upon the translation given in [11]. It uses a metalanguage, which was adapted to and extended by the expressions provided by $\mathcal{C}_{TAID}$.

3.1.1 Action description language

The translation $\pi_1(D)$ of the domain description includes declarations of fluents and actions, inertial rules, interpretation constraints, to assure for example that a fluent and its negation can never hold at the same time,

and the translation of all rules, laws and constraints defined in the domain description D.

Fluents and actions

The expressions defined by a domain description $D(A, F)$ have to be composed of symbols from A an F. When constructing the logic program we first have to define the alphabet.

We declare every fluent $f \in F$ by using the predicate:

```
fluent(f).
```

Analogously, every action $a \in A$ is declared using the predicate:

```
action(a).
```

But it is not sufficient to simply declare the fluents and actions. It is furthermore necessary to explicitly introduce a relationship between a fluent and its negation. This interpretation constraint is realised by an integrity constraint of the following form:

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

This constraint assures that a fluent and its negation cannot both hold in the same state. In the following we will use the variable `F` as a shorthand for the set of all fluents and the variable `T` as a shorthand for all points of time.

Afterwards, we have to declare which of the fluents are inertial and which are not. An inertial fluent has the same value as in the previous state, unless it is explicitly known that the value changed. This preservation of knowledge is realised by introducing two rules for every fluent.

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

The predicate `default(F)`, used in the body of the rules, assures that only inertial fluents are affected by these rules. Obviously non-inertial fluents have to be treated differently.

Example 3.1. The declaration of one inertial fluent and one action, looks as follows:

```
fluent(normal_sulfur).
holds(normal_sulfur,T+1) :- holds(normal_sulfur,T),
       not holds(neg(normal_sulfur),T+1), not
           default(normal_sulfur),
       fluent(normal_sulfur), time(T;T+1).
holds(neg(normal_sulfur),T+1) :- holds(neg(normal_sulfur),T),
       not holds(normal_sulfur,T+1), not default(normal_sulfur),
       fluent(normal_sulfur), time(T;T+1).
:- holds(normal_sulfur,0), holds(neg(normal_sulfur),0).
action(sulfur_reduction).
```

•

Default rule

In $\mathcal{C}_{TAID}$, non-inertial fluents can only be declared via a default value. That means, a default value has to be provided for every non-inertial fluent. If no default value is specified, the fluent is considered to be inertial.

The default rule

*(***default** *f)*

is translated into a fact and a rule

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

The fact states that there is a default rule for this fluent, specifying the default value *true*. If the default value of a fluent f is $false$ the rule that specifies the default value looks like:

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

It strongly relies on the "negation as failure" operator not, which allows for reasoning in the absence of knowledge, i.e. if it is not explicitly known that f has a particular value, assume it has the default value.

Example 3.2. To illustrate how negation is translated, let us specify default value *(***default** $\neg$*enhanced_lateral_roots)*

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

•

Dynamic causal law

A dynamic causal law describes the effects of an action. If an action occurs, the effects of this action, i.e. truth values for particular fluents, hold at the next point of time, if all preconditions hold. For a dynamic causal law of the general form

$$(a \textbf{ causes } f_1, \ldots, f_n \textbf{ if } g_1, \ldots, g_n)$$

the translation results in one rule for every fluent $f_1, \ldots, f_n$.

```
holds(f_1,T+1) :- holds(occurs(a),T),
       holds(g_1,T), ... , holds(g_n,T),
       fluent(g_1), ... , fluent(g_n),
       fluent(f_1), action(a), time(T;T+1).
       ⋮
holds(f_n,T+1) :- holds(occurs(a),T),
       holds(g_1,T), ... , holds(g_n,T),
       fluent(g_1), ... , fluent(g_n),
       fluent(f_n), action(a), time(T;T+1).
```

Example 3.3. The translation for the dynamic causal law *(sulfur_reduction* **causes** *reduced_sulfur* **if** *normal_sulfur)* is the following:

```
holds(reduced_sulfur,T+1) :- holds(occurs(sulfur_reduction),T),
   holds(normal_sulfur,T), fluent(normal_sulfur),
   fluent(reduced_sulfur), action(sulfur_reduction), time(T;T+1).
```

•

Static causal laws

The value of fluents can also be influenced by static causal laws. In contrast to dynamical causal laws, static causal laws affect the values of the fluents in the current state. If the preconditions of the static causal law

$$(f_1, \ldots, f_n \textbf{ if } g_1, \ldots, g_m)$$

hold in the current state, also the head of the static causal laws has to hold. The translation looks as follows.

```
holds(f1,T) :- holds(g1,T), ... , holds(gn,T),
         fluent(g1), ..., fluent(gn), fluent(f1), time(T).
        ⋮
holds(fn,T) :- holds(g1,T), ... , holds(gn,T),
        fluent(g1), ..., fluent(gn), fluent(fn), time(T).
```

Example 3.4. The static causal law *(¬normal_sulfur* **if** *reduced_sulfur)* is translated into:

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

•

Triggering rule

A triggered action can only occur if all preconditions of the corresponding triggering rule hold and if it is not inhibited or otherwise prevented from occurring. The predicate "ab" stands for "abnormal" and assures this inhibition relation. The translation for a triggering rule of the form:

$$(f_1, \ldots, f_n \textbf{ triggers } a)$$

is the following one:

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

Example 3.5. The translation of the triggering rule *(increased_oas* **triggers** *increasing_of_serine)* is as follows:

```
holds(occurs(increasing_of_serine),T) :- not
   holds(ab(occurs(increasing_of_serine)),T),
   holds(increased_oas,T), fluent(increased_oas),
   action(increasing_of_serine), time(T).
```

•

Allowance rule

The predicate "allow" is introduced to be able to distinguish which actions are allowed in which state. This allowance rule however has no influence on the fact whether the allowed action occurs or not. The allowance rule

(f_1, ... , f_n **allows** *a)*

is translated as:

```
holds(allow(occurs(a)),T) :- not holds(ab(occurs(a)),T),
       holds(f_1,T), ... , holds(f_n,T),
       fluent(f_1), ... , fluent(f_n), action(a), time(T).
```

For all non-triggered actions for which no allowance rule is specified, the translation includes a rule, stating that this action can always occur.

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

Example 3.6. For the allowance rule *(enhanced_lateral_roots* **allows** *sulfur_repletion)*, we have the following translation:

```
holds(allow(occurs(sulfur_repletion)),T) :- not
       holds(ab(occurs(sulfur_repletion)),T),
       holds(enhanced_lateral_roots,T),
          fluent(enhanced_lateral_roots),
       action(sulfur_repletion), time(T).
```

•

Inhibition rule

Since the inhibition rule

(f_1, ... , f_n **inhibits** *a)*

represents abnormal behaviour, in the sense that even though the preconditions e.g. of a triggering rule are satisfied, an action cannot occur, it is translated using the "ab" predicate. The resulting rule is:

```
holds(ab(occurs(a)),T) :- action(a),
       holds(f_1,T), ... , holds(f_n,T),
       fluent(f_1), ... , fluent(f_n), time(T).
```

Example 3.7. Consider the triggering rule *(expressed_iaa28* **inhibits** *activation_of_auxin_inducible_genes)* The translation looks like follows:

```
holds(ab(occurs(activation_of_auxin_inducible_genes)),T) :-
   action(activation_of_auxin_inducible_genes),
   holds(expressed_iaa28,T), fluent(expressed_iaa28), time(T).
```

•

No-concurrency Constraint

The no-concurrency constraint

$$(\textbf{noconcurrency}\ a_1, \ldots, a_n)$$

enumerates a list of actions out of which at most one action can occur at each point of time. It is translated using the `lparse` expression of weight constraints, resulting in the following translation:

```
:- 2 {holds(occurs(a_1),T):action(a_1), ... ,
   holds(occurs(a_n),T):action(a_n)}, time(T).
```

Example 3.8. In the example there are two actions which cannot happen concurrently *(***noconcurrency** *rapid_seed_production, lateral_root_formation)*

```
:- 2 {holds(occurs(rapid_seed_production),T):
   action(rapid_seed_production),
   holds(occurs(enhanced_lateral_root_formation),T):
   action(enhanced_lateral_root_formation)}, time(T).
```

•

3.1.2 Action observation language

Fluent observation

There are two different kinds of fluent observations, on the one hand, those about the initial state and on the other hand the fluent observations about all other states. The fluent observations $(f$ **at** $t_0)$ 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 *(f* **at** t_i*)* however need a justification. Due to this, fluent observations about all states except the initial state are translated into integrity constraints of the form:

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

In this way all solutions which are not of interest, i.e solutions not containing the observations, are eliminated, but it is also assured that all remaining solutions are trajectory models corresponding to Definition 2.10.

Example 3.9. The example specifies two fluent observations, one about the initial state *(*¬*reduced_sulfur* **at** t_0*)*, the other one about a later moment of time *(reduced_sulfur* **at** t_1*)*.

```
holds(neg(reduced_sulfur),0).
:- not holds(reduced_sulfur,1), fluent(reduced_sulfur), time(1).
```

•

Action observation

When translating action observations the different kinds of actions have to be considered. Exogenous actions can always occur and need no further justification, that is why such an exogenous action observation

(a **occurs_at** t_i*)*

is straight forward translated as a fact:

```
holds(occurs(a),i).
```

Unlike this, observations about triggered or allowed actions must have a reason, e.g. an active triggering or allowance rule, to occur. To assure this justification, the action observation is translated using constraints of the form:

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

Example 3.10. There are two action observation in this example, the first one *(sulfur_reduction* **occurs_at** t_0*)* is about an exogenous action, the second one *(sulfur_repletion* **occurs_at** t_4*)* is about an allowed action.

```
holds(occurs(sulfur_reduction),0).
:- holds(occurs(neg(sulfur_repletion)),4),
   action(sulfur_repletion), time(4).
```

•

3.1.3 Action query language

The domain description specified the knowledge about the system and the observations described the actual evolution of the system. Now, The different queries allow us to evaluate and reason about the system using the available information.

When computing the different solutions using answer set solvers, the entire translation of the domain description is taken into account. Which observations are included in this process of evaluation depends on the kind of query. Prediction and explanation considers all observations, in planning, on the other hand, only fluent observations about the initial state are taken into consideration.

Explanation

In explanation, we are looking for a set of actions that explains a set of observations, i.e. we are trying to find initial states and sets of actions that justify the observed behaviour of the system, as defined in Definition 2.14.

The translation of an explanation contains the translation of all action and fluent observations as defined in Section 3.1.2. Since the observations about the initial state are often not complete the translation contains furthermore for every fluent f two rules

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

to generate all initial states which do not contradict the observations.

Similarly, we have to generate possible combinations of occurrences of actions, for all states. 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<n.
holds(neg(occurs(a)),T) :- not holds(occurs(a),T),
    action(a), time(T),T<n.
```

The constant n has to be provided and represents the upper time bound.

Plan

A plan is a sequence of actions, which lead from an initial state to a state where the goal of the plan is satisfied, see Definition 2.13. The initial state is partially or completely specified by fluent observation about the initial state. Only the translation of these initial fluent observations, as defined in Section 3.1.2, is included, when we are computing a plan. As in explanation, we have to include two rules for every fluent f, which generate all possible initial states.

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

A goal G of a plan is a set of fluents $G = f_1, \ldots, f_n$, which is translated using the predicate "achieved". It ensures that the goal holds in the final state of every answer set for query.

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

The constant n is the maximal time bound, i.e. the maximum number of steps in which the goal should be achieved. Additionally, the translation contains for every allowed and exogenous action the two rules:

```
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.

Prediction

In prediction we are interested in the question, whether a set of fluents is a consequence of the observed behaviour of the system, see Definition 2.15. Obviously, the translation includes all fluent and action observations. As in explanation, we have to fill in missing information, which is necessary to justify the observed behaviour. That means we have to include for every fluent f two rules to generate possible initial states.

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

Moreover the translation includes for every non-triggered action two rules similar to those of an explanation. But this time these actions can only occur up to time i. It represents the latest point of time for which there is an observation. Afterwards only triggered actions can occur. This restriction is necessary, since otherwise it is not possible to make any useful prediction, because the system could evolve randomly in any direction.

```
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<n, T<i.
holds(neg(occurs(a)),T) :- not holds(occurs(a),T),
       action(a), time(T), T<n, T<i.
```

The actual prediction, represented as the set of fluents $P = f_1, \ldots, f_n$, is translated as:

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

The value of i is either specified in the query or it is the time of the latest observation.

3.2 Translation π_2

The second translation is more direct than the first one, since it does not involve a meta language

3.2.1 Action description language

Fluents and actions

In general fluent and action names are translated by adding the prefix "fluent" or "action" to the name. This is only done for readability reasons and to ease the evaluation of the answer sets. It is not necessary for the logic program, since the context is sufficient to decide, whether it is a fluent or an action. Since there are no meta language predicates, fluents and actions cannot be directly declared using a simple fact. Fluents are declared indirectly using the inertia rules or by using the default rules, depending on the kind of fluent.

Actions are not explicitly declared in the logic programs, they are introduced when dynamic causal laws, triggering, allowance and inhibition rules are defined.

For an inertial fluent f, i.e. a fluent for which no default value is specified the translation looks as follows:

```
fluent_f(T+1) :- fluent_f(T), not neg_fluent_f(T+1),
    time(T;T+1).
neg_fluent_f(T+1) :- neg_fluent_f(T), not fluent_f(T+1),
    time(T;T+1).
```

In that way the fluent f and its negation $\neg f$ are introduced as `fluent_f` and `neg_fluent_f`

Example 3.11. The rules to define the fluent *normal_sulfur* look as follows:

```
fluent_normal_sulfur(T+1) :- fluent_normal_sulfur(T), not
    neg_fluent_normal_sulfur(T+1), time(T;T+1).
neg_fluent_normal_sulfur(T+1) :- neg_fluent_normal_sulfur(T), not
    fluent_normal_sulfur(T+1), time(T;T+1).
```

•

Additionally, for every action and every fluent, regardless of whether it is an inertial or non-inertial fluent, an integrity constraint is used to realise explicitly the relationship between a fluent and its negation.

```
:- action_a(T), neg_action_a(T), time(T).
:- fluent_f(T), neg_fluent_f(T), time(T).
```

Since the negation of actions and fluents is realised by introducing new names, these names have to be linked and it has to be assured, that a fluent and its negation, respectively an action or its negation, can never hold in the same state.

Default rule

If a default value is defined for a fluent, the fluent has to be declared in a different way. The translation of a default rule

*(***default** *f)*

is used to declare the fluent and to define its default value. To declare the default value *true* the following rule is used:

```
fluent_f(T):- not neg_fluent_f(T), time(T).
```

The default value $false$ is declared analogously:

```
neg_fluent_f(T):- not fluent_f(T), time(T).
```

Example 3.12. For the rule *(***default** $\neg$*enhanced_lateral_roots)*, the translation is:

```
neg_fluent_enhanced_lateral_roots(T):- not
   fluent_enhanced_lateral_roots(T), time(T).
```

•

Dynamic causal law

The translation of a dynamic causal law

(a **causes** $f_1, \ldots, f_n$ **if** $g_1, \ldots, g_n$*)*

defines one rule for every effect of the action. The body of the rule includes the action and all preconditions specified by the dynamic causal law.

```
fluent_f1(T+1) :- action_a(T),
       fluent_g1(T), ... , fluent_gn(T),
           time(T;T+1).
       ⋮
fluent_fn(T+1) :- action_a(T),
       fluent_g1(T), ... , fluent_gn(T),
           time(T;T+1).
```

Example 3.13. The translation of the dynamic causal law without preconditions *(enhanced_lateral_root_formation* **causes** *enhanced_lateral_roots)* is:

```
fluent_enhanced_lateral_roots(T+1) :-
   action_enhanced_lateral_root_formation(T), time(T;T+1).
```

•

Static causal law

The translation of the static causal law

$$(f_1, \ldots, f_n \textbf{ if } g_1, \ldots, g_m)$$

includes one rule for every fluent $f_1, \ldots, f_n$. The preconditions are specified in the bodies of the rules.

```
fluent_f1(T+1) :- fluent_g1(T), ... , fluent_gn(T), time(T).
 ⋮
fluent_fn(T+1) :- fluent_g1(T), ... , fluent_gn(T), time(T).
```

Example 3.14. The static causal law *(¬normal_sulfur* **if** *reduced_sulfur)* is:

```
neg_fluent_normal_sulfur(T) :- fluent_reduced_sulfur(T), time(T).
```

•

Triggering rule

The triggering rule

$$(f_1, \ldots, f_n \textbf{ triggers } a)$$

is split into two parts when it is translated. The first part involves only the preconditions of the rule. If the head of this rule is satisfied, the triggering rule is applicable in the current state. But on the other hand there might be an inhibition rule which can prevent the occurrence of the triggered action. This relationship is ensured by the second rule which defines whether the action occurs or not. If the action occurs, the triggering rule has to be applicable and all inhibition rules are passive.

```
trigger_action_a(T) :- fluent_f1(T),..., fluent_fn(T),
 time(T).
action_a(T) :- trigger_action_a(T), not neg_action_a(T),
 time(T).
```

Here, `neg_action_a(T)` expresses the fact that action a does not occur at time t. It does not provide the information whether this is due to some observations or an inhibition rule or whether it is simply an assumption made by the logic program.

Example 3.15. The rule triggering *(increased_serine* **triggers** *increasing _of_tryptophan)* is translated into:

```
trigger_action_increasing_of_tryptophan(T) :-
    fluent_increased_serine(T), time(T).
```

•

Allowance rule

The allowance rule

$$(f_1, \ldots, f_n \textbf{ allows } a)$$

is translated using the same idea as for triggering rules. But the translation of the knowledge base includes only the first part, which defines whether all preconditions of the allowance rule hold or not. This part of the translation does not take inhibition relationships into account.

```
allow_action_a(T) :- fluent_f1(T), ... , fluent_fn(T), time(T).
```

The second part of the allowance rule, which is used to decide whether the action occurs or not and which is used to generate possible combinations of occurrences of actions, depends on the type of the query. That means the rule that is included when we want to compute possible explanations is different from the rule that is included when we compute a plan. They will be explained in detail in Section 3.2.3.

For every exogenous action a, a rule, saying that the action is always allowed, is included:

```
allow_action_a(T) :- time(T).
```

Example 3.16. The allowance rule *(reduced_sulfur* **allows** *increasing_of_ oas)* is translated as:

```
allow_action_increasing_of_oas(T) :- fluent_reduced_sulfur(T),
    time(T).
```

•

Inhibition rule

The translation of an inhibition rule

(f_1,...,f_n **inhibits** *a)*

is straightforward. If the preconditions of the rule are satisfied the action cannot occur, this fact is expressed by negating the action.

`neg_action_a(T):- fluent_f`$_1$`(T),..., fluent_f`$_n$`(T), time(T).`

Example 3.17. The translation of the inhibition rule *(expressed_iaa28* **inhibits** *activation_of_auxin_inducible_genes)* is:

```
neg_action_activation_of_auxin_inducible_genes(T) :-
   fluent_expressed_iaa28(T), time(T).
```

•

No-concurrency Constraint

The no-concurrency constraint

*(***noconcurrency** a_1,...,a_n*)*

is translated as an integrity constraint using weight constraints:

`:- 2 {action_a`$_1$`(T), ... , action_a`$_2$`(T)}, time(T).`

Example 3.18. The following two actions cannot happen at the same time *(***noconcurrency** *rapid_seed_production, lateral_root_formation)* The translation looks as follows:

```
:- 2 {action_rapid_seed_production(T),
   action_lateral_root_formation(T)}, time(T).
```

•

3.2.2 Action observation language

Fluent observation

When translating fluent observations, we have to distinguish between fluent observations about the initial state or about all other states. A fluent observation *(f* **at** t_0*)* about the initial state is translated as a fact.

```
fluent_f(0).
```

An observation *(f* **at** t_i*)* about fluents in all other states is translated as an integrity constraint:

```
:- not fluent_f(i), time(i).
```

Action observation

Action observations

(a **occurs_at** t_i*)*

can be made about different kinds of actions. Exogenous actions can alway occur. That is the reason why the translation of such an action is a simple fact:

```
action_a(i).
```

Whereas the translation of all other action observations involve integrity constraints, because they need a justification.

```
:- neg_action_a(i), time(i).
```

3.2.3 Action query language

Explanation

The translation of an explanation includes the translation of all actions and fluent observations as specified in Section 3.2.2. Furthermore, it includes for every fluent f two rules

```
fluent_f(0):- not neg_fluent_f(0).
neg_fluent_f(0):- not fluent_f(0).
```

which are used to generate all possible initial states. Additionally, an explanation includes for every exogenous and every allowed action two rules

```
action_a(T) :- allow_action_a(T), not neg_action_a(T),
   time(T), T<n.
neg_action_a(T) :- not action_a(T), time(T), T<n.
```

These rules are used to generate all possible explanations, taking into account that an allowed action can only occur if a corresponding allowance rule is applicable and all inhibition rules are passive.

The upper time bound n has to be provided. It is the maximum length of an explanation.

Plan

A planning problem for the goal $G = f_1, \ldots, f_n$, as defined in Definition 2.13, is translated by including all fluent observations about the initial state. Since the set of initial fluent observations might not include an observation about every fluent, we include for every fluent the two rules

```
fluent_f(0):- not neg_fluent_f(0).
neg_fluent_f(0):- not fluent_f(0).
```

These rules can generate possible initial states, which do not contradict the observations.

Additionally, when evaluating a plan, we have to include for every allowed and exogenous action the following two rules:

```
action_a(T) :- allow_action_a(T), not neg_action_a(T),
   not achieved(T), time(T).
neg_action_a(T) :- not action_a(T), time(T).
```

These rules are used to generate all possible combinations of actions. The literal `not achieved(T)` assures, that allowed and exogenous actions can only occur up to the point of time when the goal of the plan is achieved.

The translation of the goal looks like follows:

```
:- not achieved.
achieved :- achieved(0).
achieved :- achieved(T+1), not achieved(T),
   time(T;T+1).
achieved(n) :- fluent_f_1(T), ..., fluent_f_n(T), T = n.
achieved(T) :- achieved(T+1), fluent_f_1(T), ...,
   fluent_f_n(T), time(T;T+1).
```

The constant n is the maximal time bound. A plan has the maximum length of n, that means there can be no more then $n+1$ (including the initial state) states included in the plan, but a plan can be shorter.

The last rule assures that the atom `achieved(T)` holds at the earliest possible point of time, to prevent unnecessary actions.

Prediction

A prediction of a set of fluents $P = f_1, \ldots, f_n$, as defined in Definition 2.15, includes the translation of all fluent and action observations, as specified in Section 3.2.2.

Similar to an explanation, we have to include for every fluent two rules

```
fluent_f(0):- not neg_fluent_f(0).
neg_fluent_f(0):- not fluent_f(0).
```

which generate all possible initial states. Additionally, we have to find justifications for the observations, that is why we include for every action two rules

```
action_a(T) :- allow_action_a(T), not neg_action_a(T),
   time(T), T<n, T<i.
neg_action_c(T) :- not action_a(T), time(T), T<n, T<i.
```

These rules can generate occurrences of actions up to the latest observation, occurring at time t_i.

The prediction itself is translated as:

predicted :- fluent_f$_1$(T), ..., fluent_f$_n$(T), time(T), T $\geq$ i.

The query does not necessarily need to specify a point of time, if this part is omitted, the translation includes the upper time bound t_n instead of t_i.

3.3 Differences, Advantages and Disadvantages

The major difference between the two translations π_1 and π_2 is in the use of a meta language in translation π_1 and the more direct approach used in translation π_2.

The use of a meta language allows to use variables as placeholders for fluents and actions. That can efficiently be used to reduce the number of rules of the ungrounded logic program, for example, when specifying the rules describing the frame axioms. It might also improve readability of the translation, because it is straightforward to see which actions and fluents are declared and there is not an extensive amount of additional more cryptic rules. Due to the use of a meta language the translation π_1 allows to use brackets in fluent and action names which can be used to parameterise these fluents and actions.

On the other hand, more complex rules like an allowance rule is hardly readable due to the intensive nesting of meta predicates.

The second translation however reduces the nesting and especially the number of variables to a minimum. The only sort of variables left are those representing time, which are necessary to apply the rules to all points of time. This reduction of variables might increase the number of rules of the

ungrounded logic program with variables, because for every fluent and every action the rules have to be written down explicitly. Especially when declaring fluents the translation is more complex than the first one and it is not straight forward to see which actions and fluents are actually declared. This fact might be disadvantageous when errors in the knowledge base, which are due to spelling mistakes, are tried to be resolved.

The main advantage of the second translation is its minimal use of variables. Before the solutions of the translations of the knowledge base and the queries can be computed, the logic program with variables as given by the translation has to be grounded, to obtain a logic program without variables. When using the program `smodels` answer sets can only be computed for programs without variables. This grounding is done by `lparse`.

The grounded programs of both translations, however, will have approximately the same number of rules. That means, the only time saving effects could be observed when the programs are grounded. But since the biological examples are usually not very large, this effect is hardly noticeable.

3.4 BioNetReasoning Tool

Part of this thesis was the implementation of a tool which helps to process the information provided as knowledge base and as observations, to evaluate the queries. The BioNetReasoning Tool can be downloaded from the website of the Max Planck Institute for Molecular Plant Physiology.

This tool provides some keywords which allow to specify expressions of the action language defined in Chapter 2.

The syntax of this tool is explained in the next section. To highlight expressions which can be processed by the tool, a `typewriter font` is used.

It is a command line Java application which was developed under the 1.4.2 Java version. It provides several command line arguments to set the maximal time bound, to choose the type of query, i.e. explanation, prediction or planning, to restrict the number of solutions, to chose the translation, either π_1 or π_2, and to specify one or more files containing the knowledge base, the observations and the queries.

3.4.1 Syntax

The syntax of the knowledge base is very close to the syntax of the action language defined in Chapter 2. But it also has to assure that it can be unambiguously parsed, to allow automated translation into logic programs. Since the syntax of the knowledge base is very clear and brief it is relatively

easy to write and read and to find errors in both syntax and semantics. Table 3.4.1 lists the corresponding expressions of the action language and the expression provided by the tool.

expression	$\mathcal{C}_{TAID}$	Tool
fluent	f	`<fluent> f`
action	a	`<action> a`
dynamic causal law	*(*a **causes** $f_1, \ldots, f_n$ **if** $g_1, \ldots, g_m$*)*	`a <causes> f1,..., fn <if> g1,..., gm`
	(a **causes** $f_1, \ldots, f_n$*)*	`a <causes> f1,..., fn`
static causal law	*(*$f_1, \ldots, f_n$ **if** $g_1, \ldots, g_m$*)*	`f1,..., fn <if> g1,..., gm`
triggering rule	*(*$f_1, \ldots, f_n$ **triggers** a*)*	`f1,..., fn <triggers> a`
allowance rule	*(*$f_1, \ldots, f_n$ **allows** a*)*	`f1,..., fn <allows> a`
inhibition rule	*(*$f_1, \ldots, f_n$ **inhibits** a*)*	`f1,..., fn <inhibits> a`
default rule	*(***default** f*)*	`<default> f`
no-concurrency constraint	*(***noconcurrency** $a_1, \ldots, a_n$*)*	`<noconcurrency> a1,..., an`
fluent observation	*(*f **at** t_i*)*	`f <at> i`
action observation	*(*a **occurs_at** t_i*)*	`a <occurs_at> i`
plan	$(D, O) \models f_1, \ldots, f_n$	`<plan> f1,..., fn`
prediction	$(D, O) \models f_1, \ldots, f_n$	`<predict> f1,..., fn`
explanation	$(D, O) \models$ **true**	

Table 3.1: Correlation between action language $\mathcal{C}_{TAID}$ syntax and the BioNetReasoning tool syntax

In general keywords like “causes”, “if”, “at”, “triggers” and so on are enclosed in angle brackets (”< ... >”). Furthermore are there some restrictions to the characters allowed in names of fluents and actions. Not allowed are among others: spaces, brackets of all kinds and minus signs as part of fluent and action names. When translation π_1 is used round brackets can be part of fluent and action names.

Fluents and actions

Both fluents and actions should be declared before they are used in laws and constraints, although it is not necessary for the correct translation it certainly eases the readability and the process of designing the knowledge base.

3.4.2 Design strategy

The knowledge base has to be carefully and thoroughly designed. Besides the syntax also the semantics have to be understood properly to avoid unwanted side effects.

When modelling a biological system, a strategy which helps to find a model describing the behaviour appropriately, could be the following:

1. Identifying fluents and actions

   - fluents are for example: changing concentrations, a gene that is active or inactive, etc.
   - actions influence fluents, e.g.: chemical reactions, protein transport, etc.

   A fluent could also represent a complete sub-pathway, like a sulfur assimilation pathway, and an action could be used to describe whether it is active or inactive, e.g. the pathway is only active as long as the action is not inhibited

2. Determining the dynamic causal relationships, that is, which action influences which fluents under which conditions.

   As an example let us consider the action "increase_concentration", this action will result in a medium high level of the concentration provided that the level of concentration is low, if on the other hand there is already a medium high level of concentration the action will result in a high level. This could be expressed by using the following two dynamic causal laws:

   ```
   increase_concentration <causes> medium_concentration <if>
      low_concentration
   increase_concentration <causes> high_concentration <if>
      medium_concentration
   ```

3. Determining the causal relationships and dependencies between fluents, which can be expressed using static causal laws.

4. Besides the dynamic causal laws, used to describe *what* the effect of an action is, there are other rules used to express *when* an action can occur.

   If an action has to occur under certain conditions and if the dependencies are known, a triggering rule is used, because a triggering rule states that the action will occur if the conditions hold and it will occur immediately.

   On the other hand if an action can only occur if certain conditions hold, but it does not have to happen immediately or if there is a choice whether it happens or not, an allowance rule is used. This choice might be used to model alternative pathways, or to express incomplete knowledge about the reasons for the occurrence of an action.

   Every action, for which no triggering or allowance rule is specified, can happen at all times.

5. Specifying the behaviour of the fluents is necessary to determine the current value of every fluent. Usually an action changes the value of a fluent, the fluent will maintain this value until another action changes its value again. But there might be exceptions, if there is a fluent with a default value which can only be changed for a short time by an action.

   ```
   increase_energy <causes> high_energy <default> -high_energy
   ```

6. The inhibition of actions is a very important point. Such an inhibition could be considered to be an action, which takes place under certain conditions and has effects on fluents. But more natural and elegantly, it is an effect of a combination of some fluents on the executability of actions, without involving an additional action. This can be expressed using inhibition rules.

   A combination of triggering, default and inhibition rules can for example be used to express the inhibition of a sub-pathway. In the following little example the sulfur assimilation pathway ("assimilated_sulfur") is active at the beginning. While it is active it triggers itself and stays active until the pathway is inhibited. Since now the action "assimilate_sulfur" is stopped the fluent "assimilated_sulfur" falls back to its default value, which causes the whole pathway to be inactive.

   ```
   <default> -assimilated_sulfur
   assimilated_sulfur <at> 0
   assimilated_sulfur <triggers> assimilate_sulfur
   assimilate_sulfur <causes> assimilated_sulfur
   sulfur_deficiency <inhibits> assimilate_sulfur
   ```

7. Concurrency is a main feature of biological networks. If reactions cannot happen concurrently it might be due to shared resources or maybe mutual exclusive environmental conditions. All these combinations of actions that cannot happen concurrently have to be explicitly stated using no-concurrency constraints. If all actions should happen sequentially

   ```
   <noconcurrency> all
   ```

   can be used as a shortcut. Normally this will not be very useful for biological networks, especially when triggers are used, it is often not possible to execute them one after the other.

8. As much additional information as possible should be included. For example observations about the initial state, but also about other states and occurrences of actions. The more information is provided the smaller is the number of possible solutions.

Chapter 4

Biological Example

In this chapter we will present a larger biological network and show how it can be modelled using action languages. It is an extended version of the sulfur starvation response-pathway of Arabidopsis thaliana. The example was taken from [12, 13].

Often, such networks are represented as graphs, which ease the understanding. But graphs of biological network are often ambiguous, e.g. arrows from several compounds to another compound, could represent a complex interaction of several compounds or it could simply mean, that several compounds can influence this particular compound independently. Additional explanations are necessary.

The representation as a domain description is unambiguous, although there are usually several ways to express one thing. A complex interaction is represented by several preconditions in triggering and allowance rules and in the dynamic causal laws, in other cases there might be several independent dynamic causal laws.

Because of its size, the example network is split into two parts, which are shown in Figure 4.1 and Figure 4.2. To ease the understanding of the diagrams fluents are enclosed by rectangles, actions are not. Arrows from actions to fluents represent dynamic causal laws and arrows from fluents to actions represent triggering or allowance rules. Lines with a bar at the end represent an inhibition relation. Additional information about default values or observations is not included in the diagrams.

The first part of the sulfur starvation response-pathway shows how the formation of enhanced lateral roots can be influenced, i.e. how the formation can be induced or inhibited. As already explained in Section 1.1, sulfur is an essential nutrient for plants. If the amount of sulfur is not sufficient to guarantee the plants survival, it can either form additional lateral roots, to access new sources of sulfur or, if this is not successful, it can use its resources

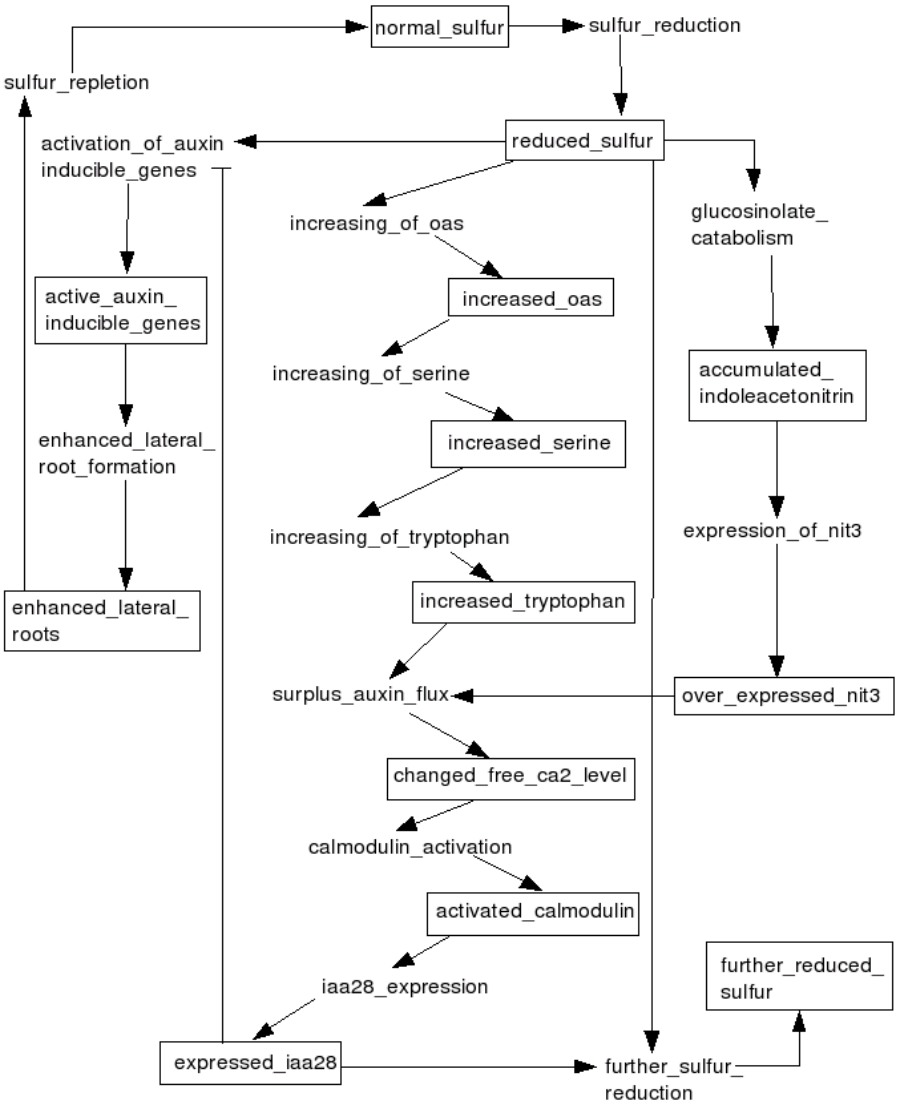

Figure 4.1: Sulfur starvation response-pathway, part I.

for the next generation and produce seeds.

Now we will analyse how this knowledge can be expressed using action description languages. The complete domain description and possible observations and queries are shown in Listing 4.1. The next paragraphs will always refer to this listing.

The first step is to define the alphabet by declaring all fluents and actions, as shown in Line 3 to 29. We introduce the fluents `reduced_sulfur` and `further_reduced_sulfur`, both express the knowledge that the sulfur level is low. They are used here to distinguish the start and the end of the first part of the network. The end of the first part is at the same time the start of the second part of the network.

Afterwards, we can start to express the causal relationships between fluents and actions using dynamic causal laws and static causal laws, see Line 32 to 50. The design decisions were already explained in Chapter 2. Observe, that only the first dynamic causal law includes preconditions. It might be useful to extend some of the dynamic causal laws, for example by the precondition `reduced_sulfur`. But that would only express the fact, that the action has no effect, if it occurs. Usually we want to express, that the action cannot occur at all.

Whether an action can or cannot occur is determined by the allowance, triggering and inhibition rules given in Line 54 to 75. For most of the actions triggering rules should be declared to minimise the number of solutions of the following queries, especially if the preconditions for the occurrence of the action are well known and if there is no choice, e.g. of an alternative pathway. Lines 59 to 69 show the triggering rules declared for this network.

However, if not all preconditions for the occurrence of an action are known or if they are not part of the model, the occurrence of the action should be restricted using allowance rules, as in Line 54 to Line 56. Another situation in which an allowance rule should be used is the choice of an alternative pathway. For example, it is known that the concentration of serine will increase if the level of sulfur is low, but additionally it is possible that the catabolism of glucosinolate is induced. Both pathways eventually induce the expression of IAA28. This is expressed by the allowance rule in Line 55.

In this example there is only one explicit inhibition relation, the one between expressed IAA28 and the genes which induce auxin, see Line 72. All other rules in Line 73 to 75 are implicit, preventing unnecessary occurrences of allowed actions.

Except for the auxin inducing genes and the enhanced lateral roots, all fluents are inertial. However, these two fluents have a default value of $false$, Line 78 and 79, to assure the correct inhibition by IAA28.

Listing 4.1: Part I of the biological example

```
% knowledge base

<fluent> normal_sulfur
<fluent> reduced_sulfur
<fluent> increased_oas
<fluent> increased_serine
<fluent> increased_tryptophan
<fluent> changed_free_ca2_level
<fluent> activated_calmodulin
<fluent> accumulated_indoleacetonitrile
<fluent> over_expressed_nit3
<fluent> active_auxin_inducible_genes
<fluent> enhanced_lateral_roots
<fluent> expressed_iaa28
<fluent> further_reduced_sulfur

<action> sulfur_reduction
<action> glucosinolate_catabolism
<action> expression_of_nit3
<action> increasing_of_oas
<action> increasing_of_serine
<action> increasing_of_tryptophan
<action> surplus_auxin_flux
<action> calmodulin_activation
<action> iaa28_expression
<action> activation_of_auxin_inducible_genes
<action> enhanced_lateral_root_formation
<action> sulfur_repletion
<action> further_sulfur_reduction

% dynamic causal laws
sulfur_reduction <causes> reduced_sulfur <if> normal_sulfur
glucosinolate_catabolism <causes> accumulated_indoleacetonitrile
increasing_of_oas <causes> increased_oas
activation_of_auxin_inducible_genes <causes>
    active_auxin_inducible_genes
expression_of_nit3 <causes> over_expressed_nit3
surplus_auxin_flux <causes> changed_free_ca2_level
increasing_of_serine <causes> increased_serine
increasing_of_tryptophan <causes> increased_tryptophan
calmodulin_activation <causes> activated_calmodulin
iaa28_expression <causes> expressed_iaa28
```

```
42 enhanced_lateral_root_formation <causes> enhanced_lateral_roots
43 sulfur_repletion <causes> normal_sulfur
44 further_sulfur_reduction <causes> further_reduced_sulfur
45
46 % static causal laws
47 -normal_sulfur <if> reduced_sulfur
48 -reduced_sulfur <if> normal_sulfur
49 -normal_sulfur <if> further_reduced_sulfur
50 -further_reduced_sulfur <if> normal_sulfur
51
52
53 % allowance rules
54 normal_sulfur <allows> sulfur_reduction
55 reduced_sulfur <allows> glucosinolate_catabolism
56 enhanced_lateral_roots <allows> sulfur_repletion
57
58 % triggering rules
59 reduced_sulfur <triggers> activation_of_auxin_inducible_genes
60 reduced_sulfur <triggers> increasing_of_oas
61 accumulated_indoleacetonitrile <triggers> expression_of_nit3
62 over_expressed_nit3 <triggers> surplus_auxin_flux
63 increased_oas <triggers> increasing_of_serine
64 increased_serine <triggers> increasing_of_tryptophan
65 increased_tryptophan <triggers> surplus_auxin_flux
66 changed_free_ca2_level <triggers> calmodulin_activation
67 activated_calmodulin <triggers> iaa28_expression
68 active_auxin_inducible_genes <triggers>
       enhanced_lateral_root_formation
69 expressed_iaa28, reduced_sulfur <triggers> further_sulfur_reduction

70
71 % inhibition rules
72 expressed_iaa28 <inhibits> activation_of_auxin_inducible_genes
73 reduced_sulfur <inhibits> sulfur_reduction
74 accumulated_indoleacetonitrile <inhibits> glucosinolate_catabolism
75 normal_sulfur <inhibits> sulfur_repletion
76
77 % default values
78 <default> -active_auxin_inducible_genes
79 <default> -enhanced_lateral_roots
```

If the formation of additional lateral roots was not successful, i.e. the level of sulfur is still low, Arabidopsis thaliana follows a different strategy, it produces seeds. This subnetwork is shown in Figure 4.2.

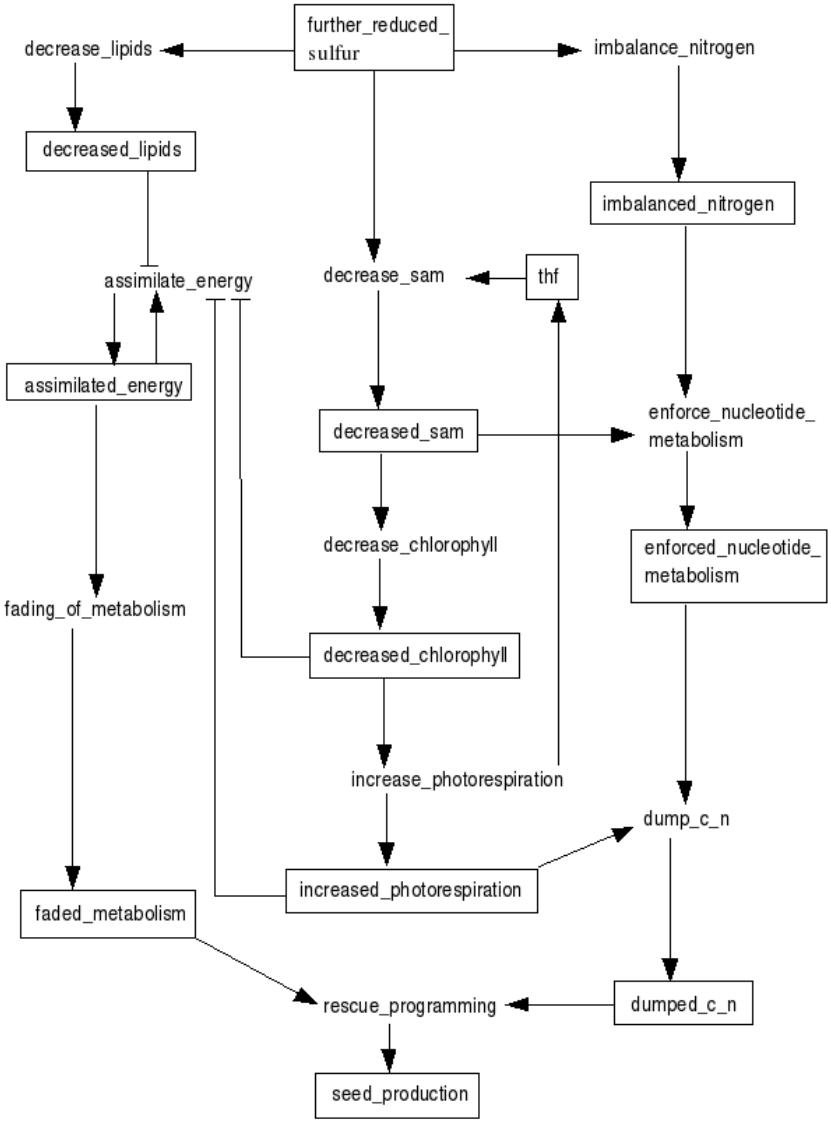

Figure 4.2: Sulfur starvation response-pathway, part II.

This network does not include all details, for example the assimilation of energy is a complex pathway by itself. However, here it is represented by a fluent and an action which can be inhibited. The action is linked to a fluent with a default value, so that its value automatically falls back to the default value once the reaction is inhibited. See the default rule in Line 60, the triggering rule in Line 45, the dynamic causal law in Line 29 and the inhibition rule in Line 54.

In this example, we have one allowance rule, Line 40, which is introduced as a choice when and whether the action occurs. All other actions are considered to be triggered actions. Here, we have triggering rules which are slightly more complex than before, since they include several preconditions, see Line 49 to 51. Such an action is only triggered when all preconditions hold.

In this example we have four inhibition rules. Three of them can independently inhibit the assimilation of energy, Line 54 to 56. The last inhibition rule, Line 57, is included to avoid the unnecessary occurrence of the allowed action.

Listing 4.2: Part II of the biological example

```
1  % knowledge base
2
3  <fluent> further_reduced_sulfur
4  <fluent> decreased_lipids
5  <fluent> assimilated_energy
6  <fluent> faded_metabolism
7  <fluent> seed_production
8  <fluent> decreased_sam
9  <fluent> decreased_chlorophyll
10 <fluent> increased_photorespiration
11 <fluent> imbalanced_nitrogen
12 <fluent> enforced_nucleotide_metabolism
13 <fluent> dumped_c_n
14 <fluent> thf
15
16 <action> decrease_lipids
17 <action> assimilate_energy
18 <action> fading_of_metabolism
19 <action> rescue_reprogramming
20 <action> decrease_sam
21 <action> decrease_chlorophyll
22 <action> increase_photorespiration
23 <action> imbalance_nitrogen
24 <action> enforce_nucleotide_metabolism
```

```
<action> dump_c_n

% dynamic causal laws
decrease_lipids <causes> decreased_lipids
assimilate_energy <causes> assimilated_energy
fading_of_metabolism <causes> faded_metabolism
decrease_sam <causes> decreased_sam
decrease_chlorophyll <causes> decreased_chlorophyll
increase_photorespiration <causes> increased_photorespiration
imbalance_nitrogen <causes> imbalanced_nitrogen
enforce_nucleotide_metabolism <causes>
    enforced_nucleotide_metabolism
dump_c_n <causes> dumped_c_n
rescue_reprogramming <causes> seed_production

% allowance rules
further_reduced_sulfur <allows> imbalance_nitrogen

% triggering rules
further_reduced_sulfur <triggers> decrease_lipids
further_reduced_sulfur <triggers> decrease_sam
assimilated_energy <triggers> assimilate_energy
decreased_sam <triggers> decrease_chlorophyll
decreased_chlorophyll <triggers> increase_photorespiration
-assimilated_energy <triggers> fading_of_metabolism
increased_photorespiration, enforced_nucleotide_metabolism
    <triggers> dump_c_n
faded_metabolism, dumped_c_n <triggers> rescue_reprogramming
imbalanced_nitrogen, decreased_sam <triggers>
    enforce_nucleotide_metabolism

% inhibition rules
decreased_lipids <inhibits> assimilate_energy
decreased_chlorophyll <inhibits> assimilate_energy
increased_photorespiration <inhibits> assimilate_energy
imbalanced_nitrogen <inhibits> imbalance_nitrogen

% default values
<default> -assimilated_energy
<default> thf
```

4.1 Queries of interest

In this section we will use the domain descriptions given in the previous section to evaluate the domain descriptions and to infer knowledge using observations made for example during an experiment. The queries refer to the entire domain description, i.e. to the combination of Listing 4.1 and 4.2.

Query: Prediction 1

Let us consider the following example, which predicts the behaviour of the biological system.

Listing 4.3: Query 1 of the biological example

```
% fluent observations
normal_sulfur <at> 0
-increased_oas <at> 0
-increased_serine <at> 0
-increased_tryptophan <at> 0
-changed_free_ca2_level <at> 0
-activated_calmodulin <at> 0
-expressed_iaa28 <at> 0
-accumulated_indoleacetonitrile <at> 0
-over_expressed_nit3 <at> 0
-active_auxin_inducible_genes <at> 0
-enhanced_lateral_roots <at> 0
-reduced_sulfur <at> 0
-further_reduced_sulfur <at> 0
-decreased_lipids <at> 0
assimilated_energy <at> 0
-faded_metabolism <at> 0
-seed_production <at> 0
-decreased_sam <at> 0
-decreased_chlorophyll <at> 0
-increased_photorespiration <at> 0
-imbalanced_nitrogen <at> 0
-enforced_nucleotide_metabolism <at> 0
-dumped_c_n <at> 0
thf <at> 0

% action observations
sulfur_reduction <occurs_at> 0

% queries
```

```
31 <predict> expressed_iaa28
```

Line 2 to 25 show possible fluent observations about the initial, describing the situation when the sulfur level is still normal. Additionally we include the observation, that the sulfur level was reduced, Line 28. The query in Line 31 is used to ask, whether it can be assumed, that IAA28 is always expressed.

A possible answer to this query could be:

```
1 Time bound: 8
2 Prediction is true in all answer sets: Yes
3 Prediction is true in at least one answer set: Yes
```

This prediction holds in all trajectory models, if the time bound is high enough, i.e. $n > 6$. This is obvious, since once the sulfur level is low, there is a sequence of triggered actions inducing `expressed_iaa28`.

If we choose a time bound of $n = 6$, `expressed_iaa28` can still hold, since there is an alternative pathway via `glucosinolate_catabolism`. This alternative pathway includes a choice, which means the prediction does not hold in all trajectory models.

```
1 Time bound: 6
2 Prediction is true in all answer sets: No
3 Prediction is true in at least one answer set: Yes
```

Query: Explanation 1

In this example we assume, that we observed the expression of IAA28 and we are looking for an explanation for this behaviour. We might formulate our observations in the following way.

Listing 4.4: Query 2 of the biological example

```
1  % fluent observations
2  normal_sulfur <at> 0
3  -increased_oas <at> 0
4  -increased_serine <at> 0
5  -increased_tryptophan <at> 0
6  -changed_free_ca2_level <at> 0
7  -activated_calmodulin <at> 0
8  -expressed_iaa28 <at> 0
9  -accumulated_indoleacetonitrile <at> 0
10 -over_expressed_nit3 <at> 0
11 -active_auxin_inducible_genes <at> 0
12 -enhanced_lateral_roots <at> 0
```

```
13 -reduced_sulfur <at> 0
14 -further_reduced_sulfur <at> 0
15 -decreased_lipids <at> 0
16 assimilated_energy <at> 0
17 -faded_metabolism <at> 0
18 -seed_production <at> 0
19 -decreased_sam <at> 0
20 -decreased_chlorophyll <at> 0
21 -increased_photorespiration <at> 0
22 -imbalanced_nitrogen <at> 0
23 -enforced_nucleotide_metabolism <at> 0
24 -dumped_c_n <at> 0
25 thf <at> 0
26
27 expressed_iaa28 <at> 6
28
29 % action observations
30 sulfur_reduction <occurs_at> 0
```

The fluent observations about the initial state are given in Line 2 to 25. Our observation of the expressed IAA28 is included as a single observation given in Line 27.

Since our last observation is at time $t = 6$, we might use this as an upper time bound for computing possible explanations. One possible explanation is

```
1  sulfur_reduction <occurs_at> 0, 4
2  glucosinolate_catabolism <occurs_at> 1
3  (expression_of_nit3 <occurs_at> 2, 3, 4, 5, 6)
4  (assimilate_energy <occurs_at> 0, 1, 2, 3, 4, 5, 6)
5  (activation_of_auxin_inducible_genes <occurs_at> 1, 2, 3, 5)
6  (enhanced_lateral_root_formation <occurs_at> 2, 3, 4, 6)
7  sulfur_repletion <occurs_at> 3
8  (increasing_of_oas <occurs_at> 1, 2, 3, 5, 6)
9  (increasing_of_serine <occurs_at> 2, 3, 4, 5, 6)
10 (increasing_of_tryptophan <occurs_at> 3, 4, 5, 6)
11 (surplus_auxin_flux <occurs_at> 3, 4, 5, 6)
12 (calmodulin_activation <occurs_at> 4, 5, 6)
13 (further_sulfur_reduction <occurs_at> 6)
14
15 (iaa28_expression <occurs_at> 5, 6)
```

The actions listed in brackets are triggered actions, the actions listed without brackets are allowed actions. To ease the reading of the listing, the

times at which the actions occur are given as a list of time points, instead of writing one line for every occurrence.

The last line of the listing shows, that these action occurrences are indeed an explanation for our fluent observation `expressed_iaa28`.

For the chosen time bound, there are altogether eight possible explanations, only varying in the occurrences of the actions `sulfur_repletion` and `sulfur_reduction`. Obviously, the higher the time bound, the more explanations there are.

Query: Plan 1

Now, we want to find out, how we have to influence the system, to assure that we get `expressed_iaa28`. This can be done by computing a plan. The following listing describes the initial situation, where the sulfur level is still normal.

Listing 4.5: Query 3 of the biological example

```
% fluent observations
normal_sulfur <at> 0
-increased_oas <at> 0
-increased_serine <at> 0
-increased_tryptophan <at> 0
-changed_free_ca2_level <at> 0
-activated_calmodulin <at> 0
-expressed_iaa28 <at> 0
-accumulated_indoleacetonitrile <at> 0
-over_expressed_nit3 <at> 0
-active_auxin_inducible_genes <at> 0
-enhanced_lateral_roots <at> 0
-reduced_sulfur <at> 0
-further_reduced_sulfur <at> 0
-decreased_lipids <at> 0
assimilated_energy <at> 0
-faded_metabolism <at> 0
-seed_production <at> 0
-decreased_sam <at> 0
-decreased_chlorophyll <at> 0
-increased_photorespiration <at> 0
-imbalanced_nitrogen <at> 0
-enforced_nucleotide_metabolism <at> 0
-dumped_c_n <at> 0
thf <at> 0

```

```
27 <plan> expressed_iaa28
```

The smallest time bound for which we find an plan is $n = 6$. For this time bound there are eight possible plans. These plans correspond to the explanations found by the previous query. Usually this is not the case, since we would normally include additional observations when we are looking for an explanation, to restrict which trajectory models are possible.

We find, for example, the following plan:

```
1  sulfur_reduction <occurs_at> 0
2  glucosinolate_catabolism <occurs_at> 1
3  (iaa28_expression <occurs_at> 5, 6)
4  (enhanced_lateral_root_formation <occurs_at> 2, 3, 4, 5, 6)
5  (assimilate_energy <occurs_at> 0, 1, 2, 3, 4, 5, 6)
6  (activation_of_auxin_inducible_genes <occurs_at> 1, 2, 3, 4, 5)
7  (calmodulin_activation <occurs_at> 4, 5, 6)
8  (surplus_auxin_flux <occurs_at> 3, 4, 5, 6)
9  (increasing_of_tryptophan <occurs_at> 3, 4, 5, 6)
10 (increasing_of_serine <occurs_at> 2, 3, 4, 5, 6)
11 (increasing_of_oas <occurs_at> 1, 2, 3, 4, 5, 6)
12 (expression_of_nit3 <occurs_at> 2, 3, 4, 5, 6)
13 (further_sulfur_reduction <occurs_at> 6)
```

Query: Prediction 2

In the following three examples, we will focus on a different fluent, to deepen the understanding of the different kinds of queries.

In this example, we are interested in the question, whether we can predict the production of seeds if we know that IAA28 is expressed. The observations are shown in the following listing.

Listing 4.6: Query 4 of the biological example

```
1  % fluent observations
2  -normal_sulfur <at> 0
3  -increased_oas <at> 0
4  -increased_serine <at> 0
5  -increased_tryptophan <at> 0
6  -changed_free_ca2_level <at> 0
7  -activated_calmodulin <at> 0
8  -expressed_iaa28 <at> 0
9  -accumulated_indoleacetonitrile <at> 0
10 -over_expressed_nit3 <at> 0
11 -active_auxin_inducible_genes <at> 0
12 -enhanced_lateral_roots <at> 0
```

```
13 reduced_sulfur <at> 0
14 -further_reduced_sulfur <at> 0
15 -decreased_lipids <at> 0
16 assimilated_energy <at> 0
17 -faded_metabolism <at> 0
18 -seed_production <at> 0
19 -decreased_sam <at> 0
20 -decreased_chlorophyll <at> 0
21 -increased_photorespiration <at> 0
22 -imbalanced_nitrogen <at> 0
23 -enforced_nucleotide_metabolism <at> 0
24 -dumped_c_n <at> 0
25 thf <at> 0
26
27 expressed_iaa28 <at> 11
28
29 % queries
30 <predict> seed_production
```

Observe the observation given in Line 27, it expresses the knowledge that IAA28 was expressed, it is not the earliest possible point of time for this expression.

The prediction will have the following outcome for a time bound $n > 10$:

```
1 Prediction is true in all answer sets: No
2 Prediction is true in at least one answer set: Yes
```

Obviously, it is possible that the current setting induces the production of seeds, but there are still other possible outcomes, since it may be that the formation of lateral roots helps to normalise the level of sulfur.

Query: Explanation 2

Consider the following observations, for which we are trying to find an explanation:

Listing 4.7: Query 5 of the biological example

```
1 % fluent observations
2 -normal_sulfur <at> 0
3 reduced_sulfur <at> 0
4 -increased_oas <at> 0
5 -increased_serine <at> 0
6 -increased_tryptophan <at> 0
7 -changed_free_ca2_level <at> 0
8 -activated_calmodulin <at> 0
```

```
9  -expressed_iaa28 <at> 0
10 -accumulated_indoleacetonitrile <at> 0
11 -over_expressed_nit3 <at> 0
12 -active_auxin_inducible_genes <at> 0
13 -enhanced_lateral_roots <at> 0
14 -sulfur_deficiency <at> 0
15 -decreased_lipids <at> 0
16 assimilated_energy <at> 0
17 -faded_metabolism <at> 0
18 -seed_production <at> 0
19 assimilated_sulfur <at> 0
20 -decreased_sam <at> 0
21 -decreased_chlorophyll <at> 0
22 -increased_photorespiration <at> 0
23 -imbalanced_nitrogen <at> 0
24 -enforced_nucleotide_metabolism <at> 0
25 -dumped_c_n <at> 0
26 thf <at> 0
27
28 expressed_iaa28 <at> 6
29 faded_metabolism <at> 11
30
31 -sulfur_repletion <occurs_at> 2
32 -sulfur_repletion <occurs_at> 3
33 -sulfur_repletion <occurs_at> 4
34 -sulfur_repletion <occurs_at> 5
```

The fluent observations in Line 2 to 26 specify the same initial state as before. But now we include additional observations. For example we know that sulfur was not repleted, this is expressed by the action observations in Line 31 to 34. Additionally we also observed that IAA28 was expressed at the earliest point of time and that the metabolism faded at a later point of time, Line 28 and 29.

Among others we get the following explanation:

```
1 imbalance_nitrogen <occurs_at> 6
2 (increase_photorespiration <occurs_at> 8, 9,10, 11)
3 (decrease_chlorophyll <occurs_at> 7, 8, 9, 10, 11)
4 (decrease_lipids <occurs_at> 6, 7, 8, 9, 10, 11)
5 glucosinolate_catabolism <occurs_at> 0
6 (iaa28_expression <occurs_at> 4, 5, 6, 7, 8, 9, 10, 11)
7 (further_sulfur_reduction <occurs_at> 5, 6, 7, 8, 9, 10, 11)
8 (enhanced_lateral_root_formation <occurs_at> 1, 2, 3, 4, 5)
9 (decrease_sam <occurs_at> 6, 7, 8, 9, 10, 11)
```

```
10 (dump_c_n <occurs_at> 9, 10, 11)
11 (fading_of_metabolism <occurs_at> 8, 9, 10, 11)
12 (enforce_nucleotide_metabolism <occurs_at> 7, 8, 9, 10, 11)
13 (assimilate_energy <occurs_at> 0, 1, 2, 3, 4, 5, 6)
14 (activation_of_auxin_inducible_genes <occurs_at> 0, 1, 2, 3, 4)
15 (calmodulin_activation <occurs_at> 3, 4, 5, 6, 7, 8, 9, 10, 11)
16 (surplus_auxin_flux <occurs_at> 2, 3, 4, 5, 6, 7, 8, 9, 10, 11)
17 (increasing_of_tryptophan <occurs_at> 2, 3, 4, 5, 6, 7, 8, 9, 10,
       11)
18 (increasing_of_serine <occurs_at> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
       11)
19 (increasing_of_oas <occurs_at> 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
       11)
20 (expression_of_nit3 <occurs_at> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11)
21 (rescue_reprogramming <occurs_at> 10, 11)
```

For a time bound of $n = 11$ there are only two possible explanations. They only vary in the point of time at which the imbalance of nitrogen and consequently also the enforcing of the nucleotide metabolism occurs.

Query: Plan 2

In our last example we will compute a plan, that includes all actions which will lead to the production of seeds, when we already have a reduced level of sulfur. This knowledge is expressed by the fluent observations in the listing below.

Listing 4.8: Query 6 of the biological example

```
1  % fluent observations
2  -normal_sulfur <at> 0
3  reduced_sulfur <at> 0
4  -increased_oas <at> 0
5  -increased_serine <at> 0
6  -increased_tryptophan <at> 0
7  -changed_free_ca2_level <at> 0
8  -activated_calmodulin <at> 0
9  -expressed_iaa28 <at> 0
10 -accumulated_indoleacetonitrile <at> 0
11 -over_expressed_nit3 <at> 0
12 -active_auxin_inducible_genes <at> 0
13 -enhanced_lateral_roots <at> 0
14 -sulfur_deficiency <at> 0
15 -decreased_lipids <at> 0
16 assimilated_energy <at> 0
```

```
17 -faded_metabolism <at> 0
18 -seed_production <at> 0
19 assimilated_sulfur <at> 0
20 -decreased_sam <at> 0
21 -decreased_chlorophyll <at> 0
22 -increased_photorespiration <at> 0
23 -imbalanced_nitrogen <at> 0
24 -enforced_nucleotide_metabolism <at> 0
25 -dumped_c_n <at> 0
26 thf <at> 0
27
28 <plan> seed_production
```

The observations given in Line 2 to 26 describe the initial state as the state in which the level of sulfur is already decreased, but there was not yet a reaction to the change.

The query itself is straight forward and given in Line 28. It is used to express the fact, that we are looking for solutions in which the fluent `seed_production` has the value *true*.

The smallest point of time for which we find a plan is $n = 11$. Even though there are only four allowed actions out of a total of twenty three actions, they allow a huge variety of combinations. For the time bound of $n = 11$ we find only "8" solutions, but already for a time bound of $n = 12$ we get "188" possible plans.

One possible plan that leads to the production of seeds is given below.

```
1  glucosinolate_catabolism <occurs_at> 0
2  (expression_of_nit3 <occurs_at> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11)
3  (increasing_of_oas <occurs_at> 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
       11)
4  (increasing_of_serine <occurs_at> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
       11)
5  (increasing_of_tryptophan <occurs_at> 2, 3, 4, 5, 6, 7, 8, 9, 10,
       11)
6  (activation_of_auxin_inducible_genes <occurs_at> 0, 1, 2)
7  (enhanced_lateral_root_formation <occurs_at> 1, 2, 3)
8  sulfur_repletion <occurs_at> 2
9  sulfur_reduction <occurs_at> 4
10 (surplus_auxin_flux <occurs_at> 2, 3, 4, 5, 6, 7, 8, 9, 10, 11)
11 (calmodulin_activation <occurs_at> 3, 4, 5, 6, 7, 8, 9, 10, 11)
12 (iaa28_expression <occurs_at> 4, 5, 6, 7, 8, 9, 10,11)
13 (further_sulfur_reduction <occurs_at> 5, 6, 7, 8, 9, 10, 11)
14 imbalance_nitrogen <occurs_at> 6
15 (decrease_lipids <occurs_at> 6, 7, 8, 9, 10, 11)
```

```
16 (assimilate_energy <occurs_at> 0, 1, 2, 3, 4, 5, 6)
17 (fading_of_metabolism <occurs_at> 8, 9, 10, 11)
18 (decrease_sam <occurs_at> 6, 7, 8, 9, 10, 11)
19 (decrease_chlorophyll <occurs_at> 7, 8, 9, 10, 11)
20 (increase_photorespiration <occurs_at> 8, 9, 10, 11)
21 (enforce_nucleotide_metabolism <occurs_at> 7, 8, 9, 10, 11)
22 (dump_c_n <occurs_at> 9, 10, 11)
23 (rescue_reprogramming at 10, 11)
```

Analysing the different plans and explanations might help to better understand such a biological system. Obviously, even such a relatively small example allows multiple combinations of fluents and actions. The plans and explanations might show, where this variation comes from.

Bibliography

[1] Chitta Baral. *Knowledge Representation, reasoning and declarative problem solving*. Cambridge University Press, 2003.

[2] Chitta Baral, Karen Chancellor, Nam Tran, Nhan Tran, A. Joy, and M. Berens. A knowledge based approach for representing and reasoning about signaling networks. In *ISMB/ECCB (Supplement of Bioinformatics)*, pages 15–22, 2004.

[3] Chitta Baral, Michael Gelfond, and Alessandro Provetti. Representing actions: Laws, observations and hypotheses. *J. Log. Program.*, 31(1-3):201–243, 1997.

[4] Michael Gelfond and Vladimir Lifschitz. Representing action and change by logic programs. *Journal of Logic Programming*, 17:301–321, 1993.

[5] Michael Gelfond and Vladimir Lifschitz. Action languages. *Electron. Trans. Artif. Intell.*, 2:193–210, 1998.

[6] Enrico Giunchiglia, Joohyung Lee, Vladimir Lifschitz, Norman McCain, and Hudson Turner. Nonmonotonic causal theories. *Artif. Intell.*, 153(1-2):49–104, 2004.

[7] Enrico Giunchiglia and Vladimir Lifschitz. An action language based on causal explanation: preliminary report. In *AAAI '98/IAAI '98: Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence*, pages 623–630. AAAI Press, 1998.

[8] Norman Clayton McCain. *Causality in commonsense reasoning about actions*. PhD thesis, University of Texas at Austin, 1998.

[9] L.E. Rogg, J. Lasswell, and B. Bartel. A gain-of-function mutation in iaa28 suppresses lateral root development. *Plant Cell*, 13:465–480, 2001.

[10] Erik Sandewall. *Features and fluents (vol. 1): the representation of knowledge about dynamical systems*. Oxford University Press, Inc., New York, NY, USA, 1994.

[11] Nam Tran and Chitta Baral. Reasoning about triggered actions in ansprolog and its application to molecular interactions in cells. In *KR*, pages 554–564, 2004.

[12] Nikiforova V.J., Daub C.O., Hesse H., Willmitzer L., and Hoefgen R. Integrative gene-metabolite network with implemented causality deciphers informational fluxes of sulfur stress response. *Journal of Experimental Botany*, 56:1887–1896, 2005.

[13] Nikiforova V.J., Kopka J., Tolstikov V., Fiehn O., Hopkins L., Hawkesford M.J., Hesse H., and Hoefgen R. Systems re-balancing of metabolism in response to sulfur deprivation, as revealed by metabolome analysis of arabidopsis plants. *Plant Physiology*, 138:304–318, 2005.