1 Syntax of the BioNetReasoning Tool

The table gives you a quick overview of the syntax of the domain description, the observations and the queries.

fluent	`<fluent> f`
action	`<action> a`
dynamic causal law	`a <causes> f1,..., fn <if> g1, ..., gm`
	`a <causes> f1,..., fn`
static causal law	`f1,..., fn <if> g1,..., gm`
triggering rule	`f1,..., fn <triggers> a`
allowance rule	`f1,..., fn <allows> a`
inhibition rule	`f1,..., fn <inhibits> a`
default rule	`<default> f`
no-concurrency constraint	`<noconcurrency> a1,..., an`
fluent observation	`f <at> i`
action observation	`a <occurs_at> i`
plan	`<plan> f1,..., fn`
prediction	`<predict> f1,..., fn`
explanation	

The following enumeration gives a very brief introduction that explains how to use the different expression.

- dynamic causal laws are used to express causal relationship, i.e. if an action occurs it has the specified effects
- static causal laws desribe dependencies between fluents, i.e. if fluent g has a particular value fluent f is known to always have a certain value

- triggering, allowance and inhibition rules restrict when and whether an action occurs
- if all precoditions for the occurrence of an action are known → use a triggering rule
- if only some precondtions are known, if the reaction rate is very low, if there is a choice whether the action occurs or not → use an allowance rule
- if an action cannot occur under certain conditions → use an inhibition rule
- if fluents are not affected by an action, they keep the value they had in the previous state, if this is not the desired behaviour → use a default rule, it is used to state that a fluent always has a particular value unless it is changed by an action for a particular point of time
- if certain actions can never happen at the same time, e.g. due to environmental conditions → use a no-concurrency constraint
- observed behaviour of the system during an experiment are expressed by fluent or action observations
- a plan is a set of actions that have to occur to derive the given fluents, only the fluent observations about the initial state are taken into account when the plan is computed
- a prediction is either true or false, it expresses whether the specified fluents can be assumed to hold considering the given observations
- an explanation is a set of actions that explains the observed behaviour

2 Usage of the BioNetReasoning Tool

The BioNetReasoning Tool 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.

A call looks like:

```
java -jar BioNetReasoning.jar [options] filenames
```

filenames: can be a list of files but without file extension, e.g. data/kbfile1 data/kbfile2

In the table below the different options are explained.

Option	Explanation
-t=time	set maximal time bound (default = 2)
-ex	translate explanation into a logic program and save it under <filename>.expl
-Ex	as above, evaluate the query and show the different explanations
-noEx=number	set number of explanations to be calculated
-pl	translate plan into a logic program and save it under <filename>.plan
-Pl	as above, evaluate the query and show the different plans
-noPl=number	set number of plans to be calculated
-pr	translate prediction into a logic program and save it under <filename>.pred
-Pr	as above, evaluate the query and show the result
1 — 2	select the translation (default=2)

A typical call looks like:

```
java -jar BioNetReasoning.jar -t=11 -Pl 2 examples/bioexample
```

This call uses the translation π_1 to translate the file bioexample.kb in the subdirectory exampes into a logic program, the maximum time bound is set to 11. The query asks for a plan, all plans are computed and printed to the screen.

3 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.