The details here are NOT EXAMINABLE. They are provided for background for the action language $C^+$, and for general interest.


Nonmonotonic causal theories (or just ‘causal theories’ for short) is a general purpose non-monotonic representation formalism. A causal theory is a set of causal rules of the form

$$F \Leftarrow G$$

where $F$ and $G$ are formulas (defined on next page). Informally, this is to be read as saying that $F$ is ‘caused’ if $G$ is true (which is not the same as saying that $G$ is the cause of $F$). We don’t need to rely on this informal reading to use the formalism.

‘Causal theories’ are also called ‘the logic of causal explanation’ in [2]. Turner [3] has a more general formalism which he calls the ‘logic of universal causation’.

The main interest for us is that causal theories are intimately connected to the action language $C^+$ which we will be looking at later.

An implementation (of both causal theories and $C^+$) is available in the form of the Causal Calculator (CCALC). See: [http://www.cs.utexas.edu/users/tag/cc](http://www.cs.utexas.edu/users/tag/cc).

We also have our own (re-)implementation of CCALC, called tCCALC, which has some additional features.

‘Causal theories’ (and $C^+$) allows multi-valued propositions as well as Boolean ones. This is actually a very minor, but useful, extension. We won’t need it in these notes but it is useful when we look at $C^+$, so I might as well introduce it now.

### Multi-valued signatures

A multi-valued propositional signature $\sigma$ consists of:

- a set of symbols called constants,
- for each constant $c$, a non-empty set $\text{dom}(c)$ of values, called the domain of $c$. For simplicity we will assume that there are at least two distinct values in every $\text{dom}(c)$ (otherwise $c$ is trivial — meaningful but causes some minor technical complications in definitions and so on, which I want to avoid).

An atom of a signature $\sigma$ is an expression of the form $c=v$ where $c$ is a constant in $\sigma$ and $v \in \text{dom}(c)$.

A formula $\phi$ of signature $\sigma$ is any truth-functional compound of atoms of $\sigma$.

$$\phi := \bot | T | \text{any atom } c=v | \neg \phi | \phi \land \psi | \phi \lor \psi | \phi \rightarrow \psi.$$  

($\bot$ and $T$, representing ‘false’ and ‘true’, and all atoms are formulas. If $\phi$ is a formula so is $\neg \phi$. If $\phi$ and $\psi$ are formulas, then so are $\phi \land \psi$ and $\phi \lor \psi$ and $\phi \rightarrow \psi$.)

A Boolean constant is one whose domain is the set of truth values $\{t,f\}$. If $p$ is a Boolean constant, $p$ is shorthand for the atom $p=t$ and $\neg p$ for the atom $p=f$.

An interpretation $I$ of $\sigma$ is a function that maps every constant in $\sigma$ to an element of its domain. An interpretation $I$ satisfies an atom $c=v$, written $I \models c=v$, if $I(c)=v$. The satisfaction relation $\models$ is extended from atoms to formulas in accordance with the standard truth tables for the propositional connectives. $I(\phi)$ stands for the set of all interpretations of $\phi$. As usual, when $X$ is a set of formulas, $I \models X$ signifies that $I$ is a model of $X$, i.e., that $I \models \phi$ for every $\phi \in X$.

It is often convenient to represent an interpretation $I$ by the set of atoms satisfied by $I$.

### Reduction to Boolean signatures

Multi-valued signatures are for convenience. As long as the set of constants is finite, and the domain of every constant is finite, a multi-valued signature can be translated to an equivalent Boolean signature.

An atom $c=v$ can be viewed as a classical, propositional atom. Then add the following set of additional formulas:

$$\bigvee_{v} (c=v) \wedge \bigwedge_{v \neq w} \neg (c=v \land c=w)$$

for all $c \in \sigma$.

There are various optimisations — Details omitted.
Nonmonotonic causal theories

Syntax  A causal theory of signature $\sigma$ is a set of expressions (‘causal rules’) of the form

$$F \iff G$$

where $F$ and $G$ are formulas of signature $\sigma$. A rule of this form is to be read as saying that $F$ is ‘caused’ if $G$ is true (which is not the same as saying that $G$ is the cause of $F$).

Semantics  Let $\Gamma$ be a causal theory and let $X$ be an interpretation of its signature.

$\Gamma^X$ is the set of heads of all rules of $\Gamma$ whose bodies are satisfied by the interpretation $X$:

$$\Gamma^X = \{ F \mid F \iff G \text{ is a rule in } \Gamma \text{ and } X \models G \}$$

By definition: $X$ is a model of $\Gamma$ iff $X$ is the unique (classical) model of $\Gamma^X$.

Examples below.

Definite clausal theories  A causal theory $\Gamma$ is definite if

- the head of every rule of $\Gamma$ is an atom or $\bot$, and
- no atom is the head of infinitely many rules of $\Gamma$.

Note that this is not the same definition of ‘definite’ as in ‘definite clauses’ or ‘definite logic programs’.

Translation to (classical) propositional logic

Definite causal theories (defined above) can be translated via the process of ‘literal completion’ into expressions of (classical) propositional logic. The process is analogous to the Clark completion that provides the original semantics for negation by failure in logic programs.

Literal completion

For a definite causal theory $\Gamma$, translate to set of (classical) formulas $\text{comp}(\Gamma)$:

$$\begin{align*}
F &\iff G_1 \\
\vdots \\
F &\iff G_n
\end{align*}$$

becomes

$$F \iff G_1 \lor \cdots \lor G_n$$

If $F$ is an atom and there are no causal rules with $F$ as the head then $F \iff \bot$ (which is logically equivalent to $\neg F$).

A causal rule $\bot \iff G$ becomes $\neg G$.

Models of $\Gamma$ are the (classical) models of the formulas $\text{comp}(\Gamma)$.

(As already observed, for multi-valued signatures the task of finding a model of the completion $\text{comp}(\Gamma)$ of a definite causal theory $\Gamma$ can be reduced to the task of finding a model of a set of classical (Boolean) propositional formulas. Details omitted.)

Example  Signature: $p, q$ (Boolean)

Interpretations will be represented by the set of atoms that they satisfy. So the interpretations are: $\{p, q\}$, $\{p, \neg q\}$, $\{\neg p, q\}$, $\{\neg p, \neg q\}$. ($\neg p$ and $\neg q$ are being used as shorthand for the atoms $p=\bot$ and $q=\bot$, respectively.)

$$\Gamma_1 = \{ p \iff q, \ q \iff q, \ \neg q \iff \neg q \}$$

Compute the models of $\Gamma_1$:

$$\begin{align*}
X_1 = \{ p, q \}: \quad & \Gamma_1^{X_1} = \{ p, q \} \quad \text{which has one (classical) model, } X_1.
X_2 = \{ p, \neg q \}: \quad & \Gamma_1^{X_2} = \{ \neg q \} \quad \text{which has two (classical) models, }\{ p, \neg q \} \text{ and } \{ \neg p, \neg q \}.
X_3 = \{ \neg p, q \}: \quad & \Gamma_1^{X_3} = \{ p, q \} \quad \text{which has one (classical) model } X_1 \neq X_3.
X_4 = \{ \neg p, \neg q \}: \quad & \Gamma_1^{X_4} = \{ \neg q \} \quad \text{which has two (classical) models, } \{ p, \neg q \} \text{ and } \{ \neg p, \neg q \}.
\end{align*}$$

So $\Gamma_1$ has one model: $\{ p, q \}$.

The literal completion $\text{comp}(\Gamma_1) = \{ p \iff q, \neg p \iff \bot, q \iff q, \neg q \iff \neg q \} \equiv \{ p \iff q, p \}$. This has one model. Correct.
### Example

**Signature:** \( p, q \) (Boolean)

\[ \Gamma_2 = \{ p \leftarrow q, \ q \leftarrow q, \ \neg q \leftarrow \neg q, \ \neg p \leftarrow \neg p \} \]

Compute the models of \( \Gamma_2 \):

\( X_1 = \{ p, q \} : \Gamma_{X_1} \)

\[ \begin{array}{c|c|c|c}
\{c=1\} & \{c=2\} & \{c=3\} \\
\hline
\{g=b\} & \{g=b\} & \{g=b\} \\
\hline
\text{yes} & \text{no} & \text{no}
\end{array} \]

So \( \Gamma_2 \) has two models: \( \{ p, q \} \) and \( \{ \neg p, \neg q \} \).

The literal completion \( \text{comp}(\Gamma_2) = \{ p \leftrightarrow q, q \leftrightarrow q, \neg p \leftrightarrow \neg p, \neg q \leftrightarrow \neg q \} \equiv \{ p \leftrightarrow q \} \). This has two models. Correct.

### Example

**Signature:** \( p, q \) (Boolean)

Careful: the causal theory \( \{ p \leftarrow q \} \) has no models. The completion of this theory is \textit{not} \( \{ p \leftrightarrow q \} \), which has models, but \( \{ p \leftrightarrow q, \neg p \leftrightarrow \bot, q \leftrightarrow \bot, \neg q \leftrightarrow \bot \} \), which has no models.

### Example

**Signature:** constants \( c, g \) with \( \text{dom}(c) = \{ 1, 2, 3 \} \) and \( \text{dom}(g) = \{ a, b \} \).

\[
\begin{align*}
c = 1 & \leftarrow c = 1 \\
c = 2 & \leftarrow g = a \\
c = 3 & \leftarrow g = b \\
g = a & \leftarrow g = a \\
g = b & \leftarrow g = b
\end{align*}
\]

There are obviously six possible interpretations:

<table>
<thead>
<tr>
<th>interpretation X</th>
<th>reduct ( \Gamma^X )</th>
<th>models of ( \Gamma^X )</th>
<th>X model of ( \Gamma )</th>
</tr>
</thead>
<tbody>
<tr>
<td>{c=1,g=a}</td>
<td>{c=1,c=2,g=a}</td>
<td>none</td>
<td>no</td>
</tr>
<tr>
<td>{c=2,g=a}</td>
<td>{c=2,g=a}</td>
<td>{c=2,g=a}</td>
<td>yes</td>
</tr>
<tr>
<td>{c=3,g=a}</td>
<td>{c=2,g=a}</td>
<td>{c=2,g=a}</td>
<td>no</td>
</tr>
<tr>
<td>{c=1,g=b}</td>
<td>{c=1,c=3,g=b}</td>
<td>none</td>
<td>no</td>
</tr>
<tr>
<td>{c=2,g=b}</td>
<td>{c=3,g=b}</td>
<td>{c=3,g=b}</td>
<td>no</td>
</tr>
<tr>
<td>{c=3,g=b}</td>
<td>{c=3,g=b}</td>
<td>{c=3,g=b}</td>
<td>yes</td>
</tr>
</tbody>
</table>

### Defaults

In causal theories, a causal rule of the form

\[ F \leftarrow F \]

effectively says ‘\( F \) holds by default’.

To see why, do the tutorial exercises.

It might help to know that the causal rule \( F \leftarrow G \) is (nearly exactly) equivalent to the Reiter default rule \( \frac{G}{F} \). So \( F \leftarrow F \) corresponds to default rule \( \frac{\top}{F} \). It is consistent that \( F \), then \( F \), or ‘\( F \) holds by default’.

There is also a corresponding translation to (extended) logic programs. For an atom \( p \),

\[ p \leftarrow p \]

translates to

\[ p \leftarrow \neg \neg p \]

More details later.

When \( c \) is a constant and \( v \) is a value in its domain

\[ c = v \leftarrow c = v \]

says that the default value of \( c \) is \( v \).

For Boolean constant \( p \):

\[ p \leftarrow p \]

represents that \( p \) is true by default. (Because \( p \) is shorthand for \( p = t \leftarrow p = t \).)

\[ \neg p \leftarrow \neg p \]

represents that \( p \) is false by default. (Because \( \neg p \) is shorthand for \( p = f \leftarrow p = f \).)
Example (‘Nixon diamond’)

Signature: Boolean constants pacifist(X), quaker(X), republican(X) for X ranging over alan, bill.

\[
\begin{align*}
\text{pacifist}(X) & \iff \text{pacifist}(X) \land \text{quaker}(X) \\
\text{¬pacifist}(X) & \iff \text{¬pacifist}(X) \land \text{republican}(X)
\end{align*}
\]

By default, X is not a Quaker:

\[
\text{¬quaker}(X) \iff \text{¬quaker}(X)
\]

By default, X is not a Republican:

\[
\text{¬republican}(X) \iff \text{¬republican}(X)
\]

And some facts:

\[
\begin{align*}
\text{quaker}(alan) & \iff \top \\
\text{republican}(bill) & \iff \top
\end{align*}
\]

This causal theory has one model – (check as an exercise):

\[
\{\text{quaker}(alan), \text{¬quaker}(bill), \text{republican}(bill), \text{¬republican}(alan), \\
\text{pacifist}(alan), \text{¬pacifist}(bill)\}
\]

Suppose we add another fact: pacifist(bill) \iff \top.

This causal theory has a different model (check!):

\[
\{\text{quaker}(alan), \text{¬quaker}(bill), \text{republican}(bill), \text{¬republican}(alan), \\
\text{pacifist}(alan), \text{pacifist}(bill)\}
\]

What about Nixon?

Signature: same as above, but with X ranging over nixon as well.

Let’s add two facts (to the original)

\[
\begin{align*}
\text{quaker}(nixon) & \iff \top \\
\text{republican}(nixon) & \iff \top
\end{align*}
\]

This causal theory has two models (check!):

\[
\{\text{quaker}(alan), \text{republican}(bill), \text{quaker}(nixon), \text{republican}(nixon), \\
\text{pacifist}(alan), \text{pacifist}(bill), \text{pacifist}(nixon)\}
\]

\[
\{\text{quaker}(alan), \text{republican}(bill), \text{quaker}(nixon), \text{republican}(nixon), \\
\text{pacifist}(alan), \text{pacifist}(bill), \text{¬pacifist}(nixon)\}
\]

How do we interpret multiple models?

Usual story:

- cautious/sceptical \( \Gamma \vdash \alpha \) if \( \alpha \) satisfied in all models of \( \Gamma \)
- brave/credulous \( \Gamma \vdash \text{credulous } \alpha \) if \( \alpha \) satisfied in some model of \( \Gamma \)

Finally, what if there are no Republicans who are Quakers?

Forget about Nixon for a minute. How would we express that no-one is both a Quaker and a Republican?

By adding a causal rule like this:

\[
\bot \iff \text{quaker}(X) \land \text{republican}(X)
\]

The ‘Causal Calculator’ CCalc

CCalc works with causal theories and also with action descriptions in the action language C+ to be described later.

Given a definite causal theory \( \Gamma \) of some specified signature, CCalc

- constructs \( \text{comp}(\Gamma) \).
- invokes a standard propositional sat-solver to find (classical) models of \( \text{comp}(\Gamma) \), and then
- post-processes the sat-solver output to show the models obtained.

In practice, the first two steps may be combined into one, possibly with some additional optimisations to simplify the set of formulas passed to the sat-solver.

(http://www.cs.utexas.edu/users/tag/cc)

In addition, CCalc provides

- a language for specifying the signature (sorts, variables, various shorthand notations)
- various other syntactic shorthands, which we will ignore.

I won’t show the details.
Relationship to other formalisms

Reiter default logic (You can ignore the details!!)

A causal theory of a Boolean signature can be viewed as a Reiter default theory.

Translate causal rule \( F \leftarrow G \) to the default rule \( \frac{G}{F} \).

More precisely: Let \( \Gamma \) be a causal theory of Boolean signature. Let \( D(\Gamma) \) be the set of default rules obtained by translating every rule in \( \Gamma \) as described above.

An interpretation \( I \) is a model of \( \Gamma \) iff \( th(I) \) is an extension of the default theory \( (D(\Gamma), \emptyset) \). \( th(I) \) stands for the set of all formulas true in \( I \).

Extended logic programs (IMPORTANT)

Suppose that a causal theory \( \Gamma \) has a Boolean signature, and moreover that the head of every rule in \( \Gamma \) is of the form \( c \) or \( \neg c \) (these are shorthands for the atoms \( c = t \) and \( c = f \)) or \( \bot \).

Every such causal theory can be written equivalently as a set of causal rules of the form

\[
L \leftarrow L_1 \land \cdots \land L_n \quad (n \geq 0)
\]

or

\[
\bot \leftarrow L_1 \land \cdots \land L_n \quad (n \geq 0)
\]

where \( L \) and every \( L_i \) is a Boolean atom, i.e., of the form \( c \) or \( \neg c \) (\( c = t \) and \( c = f \)). Translate each such rule to an extended logic programming clause:

\[
L \leftarrow \neg L_1, \ldots, \neg L_n
\]

where as usual \( L_i \) stands for the literal complementary to \( L_i \). (Translate \( L \leftarrow \top \) to \( L \leftarrow \).)

Translate

\[
\bot \leftarrow L_1 \land \cdots \land L_n \quad (n \geq 0)
\]

to the constraint

\[
\neg L_1, \ldots, \neg L_n
\]

Call this the program \( lp(\Gamma) \).

Now identify an interpretation \( I \) with the set of literals satisfied by \( I \) (as usual).

An interpretation \( I \) is a model of \( \Gamma \) iff \( I \) is an answer set of \( lp(\Gamma) \).

Notice: the above does not say that every answer set of an extended logic program \( lp(\Gamma) \) is a model of \( \Gamma \). It says that every interpretation \( I \) — every consistent and complete set of atoms of the signature of \( \Gamma \) — is a model of \( \Gamma \) iff it is an answer set of \( lp(\Gamma) \).

Example (a big person is strong by default; a small (not big) person is not strong by default):

\[
\begin{align*}
\text{strong} &\leftarrow \text{big} \land \text{strong} \\
\neg \text{strong} &\leftarrow \neg \text{big} \land \neg \text{strong}
\end{align*}
\]

translates to the extended logic program

\[
\begin{align*}
\text{strong} &\leftarrow \neg \text{big}, \neg \text{strong} \\
\neg \text{strong} &\leftarrow \text{big}, \neg \text{strong}
\end{align*}
\]

Notice that

\[
\neg \text{big} \leftarrow \neg \text{big} \quad (\neg \text{big} \text{ unless one can show } \text{big})
\]

is the translation of

\[
\neg \text{big} \leftarrow \neg \text{big} \quad (\neg \text{big} \text{ by default; or, the default value of } \text{big} \text{ is } f)
\]

And similarly

\[
\text{big} \leftarrow \neg \text{big} \quad (\text{big} \text{ unless one can show } \neg \text{big})
\]

is the translation of

\[
\text{big} \leftarrow \text{big} \quad (\text{big} \text{ by default; or, the default value of } \text{big} \text{ is } t)
\]

In certain circumstances it is possible to translate to a simpler extended logic program \( lp'(\Gamma) \) that has fewer occurrences of negation-by-failure.

For instance, it turns out that the extended logic program

\[
\begin{align*}
\text{strong} &\leftarrow \neg \text{big}, \neg \text{strong} \\
\neg \text{strong} &\leftarrow \text{big}, \neg \text{strong}
\end{align*}
\]

is equivalent to (has the same answer sets as):

\[
\begin{align*}
\text{strong} &\leftarrow \text{big}, \neg \text{strong} \\
\neg \text{strong} &\leftarrow \neg \text{big}, \neg \text{strong}
\end{align*}
\]

What are these circumstances? They are similar to conditions for splitting sets. I will omit the details. They are not of any practical significance because the simplification does not seem to affect performance of answer set solver (in clingo anyway).
Multi-valued signatures

For multi-valued signatures it is still the case that every definite causal theory can be written equivalently as a set of causal rules of the forms:

\[ L \iff L_1 \land \cdots \land L_n \quad (n \geq 0) \]
\[ \bot \iff L_1 \land \cdots \land L_n \quad (n \geq 0) \]

where \( L \) and every \( L_i \) is an atom, except that now it is not necessarily Boolean. In other words, these rules will look like this:

\[ c = v \iff c_1 = v_1 \land \cdots \land c_n = v_n \quad (n \geq 0) \]
\[ \bot \iff c_1 = v_1 \land \cdots \land c_n = v_n \quad (n \geq 0) \]

I will write the logic program \( lp(\Gamma) \) in clingo notation to try to make things clearer. Let the propositional atom \( \text{val}(c,v) \) in the logic program represent the causal atom \( c = v \).

Translate the causal rules to logic program clauses as before:

\[ \text{val}(c,v) :- \neg \text{-val}(c_1,v_1), \ldots, \neg \text{-val}(c_n,v_n) \]
\[ -\text{-val}(c_1,v_1) \text{ represents } \neg(c_1 = v_1), \text{etc.} \]

Constraints are translated similarly:

\[ :\neg \text{-val}(c_1,v_1), \ldots, \neg \text{-val}(c_n,v_n) \]

Now we have to add logic program clauses to define \( -\text{-val}(c,v) \) for every constant \( c \) and every \( v \in \text{dom}(c) \). Assuming \( \text{dom}(c) = \{v_1, \ldots, v_n\} \), these rules will be (in clingo notation):

\[ -\text{-val}(c,v) :- \text{val}(c,v_1). \]
\[ : \quad (v_i \neq v) \]
\[ -\text{-val}(c,v) :- \text{val}(c,v_n). \]

There are other ways to get the same effect but this is the simplest. It also has other advantages (see below).

Note that we don't need the constraints

\[ :\text{val}(c,v), -\text{val}(c,v) \]

because they are automatically satisfied by the way that the \( -\text{-val}(c,v) \) literals are defined.

Example

Here is the example earlier.

Signature: constants \( c, g \) with \( \text{dom}(c) = \{1,2,3\} \) and \( \text{dom}(g) = \{a,b\} \).

\[ c = 1 \iff c = 1 \]
\[ c = 2 \iff g = a \]
\[ c = 3 \iff g = b \]
\[ g = a \iff g = a \]
\[ g = b \iff g = b \]

Translates to the logic program:

\[ \text{val}(c,1) :- \neg \text{-val}(c,1). \]
\[ \text{val}(c,2) :- \neg \text{-val}(g,a). \]
\[ \text{val}(c,3) :- \neg \text{-val}(g,b). \]
\[ \text{val}(g,a) :- \neg \text{-val}(g,a). \]
\[ \text{val}(g,b) :- \neg \text{-val}(g,b). \]
\[ -\text{-val}(c,1) :- \text{val}(c,2). \]
\[ -\text{-val}(c,1) :- \text{val}(c,3). \]
\[ -\text{-val}(c,2) :- \text{val}(c,1). \% \text{not used in this example} \]
\[ -\text{-val}(c,2) :- \text{val}(c,3). \% " \]
\[ -\text{-val}(c,3) :- \text{val}(c,1). \% " \]
\[ -\text{-val}(c,3) :- \text{val}(c,2). \% " \]
\[ -\text{-val}(g,a) :- \text{val}(g,b). \]
\[ -\text{-val}(g,b) :- \text{val}(g,a). \]

Two-valued constants, such as \( g \) in this example, can be represented more concisely by means of a single Boolean atom in the logic program. We could represent \( g = a \) and \( g = a \) by means of \( g(a) \) and \( \neg g(a) \) respectively. That would save a couple of clauses but is harder to read. It is a detail, and does not save much.

There are other possible translations that give the same effect. For example, notice that (in this example)

\[ \neg(c = 1) \iff c = 2 \lor c = 3 \]

So, instead of defining \( -\text{-val}(c,1) \) etc., we could also translate

\[ c = 1 \iff c = 1 \]

into

\[ \text{val}(c,1) :- \neg \text{val}(c,2), \neg \text{val}(c,3) \]

And similarly for the other values of \( c \) and for \( g \).

The method using explicit \( -\text{-val}/2 \) rules has other advantages.
In particular: every definite causal theory can also be written equivalently as a set of causal rules of the form

\[ L \leftarrow L_1 \land \cdots \land L_k \land \lnot L_{k+1} \land \cdots \land L_n \quad (k \geq 0, n \geq k) \]

or

\[ \bot \leftarrow L_1 \land \cdots \land L_k \land \lnot L_{k+1} \land \cdots \land \lnot L_n \quad (k \geq 0, n \geq k) \]

where \( L \) and every \( L_i \) is an atom (not necessarily Boolean). In other words, in this form:

\[ c = v \leftarrow c_1 = v_1 \land \cdots \land c_k = v_k \land \lnot (c_{k+1} = v_{k+1}) \land \cdots \land \lnot (c_n = v_n) \quad (k \geq 0, n \geq k) \]

or

\[ \bot \leftarrow c_1 = v_1 \land \cdots \land c_k = v_k \land \lnot (c_{k+1} = v_{k+1}) \land \cdots \land \lnot (c_n = v_n) \quad (k \geq 0, n \geq k) \]

These causal rules are translated, respectively, to the following logic program clauses

\[ L \leftarrow \lnot L_1, \ldots, \lnot L_k, \lnot L_{k+1}, \ldots, \lnot L_n \]

and constraints

\[ \lnot c_1 = v_1, \ldots, \lnot c_k = v_k, \lnot (c_{k+1} = v_{k+1}), \ldots, \lnot (c_n = v_n) \quad (k \geq 0, n \geq k) \]

In clingo notation, logic program clauses:

\[ \text{val}(c,v) := \lnot \text{val}(c_1,v_1), \ldots, \lnot \text{val}(c_k,v_k), \]
\[ \text{not val}(c_{k+1},v_{k+1}), \ldots, \text{not val}(c_n,v_n). \]

and constraints:

\[ \lnot \text{val}(c_1,v_1), \ldots, \lnot \text{val}(c_k,v_k), \]
\[ \text{not val}(c_{k+1},v_{k+1}), \ldots, \text{not val}(c_n,v_n). \]

\[ \lnot \text{val}(c,v) \text{ etc are defined as before.} \]

This is essentially the translation used in iCCALC.

\[ \text{Note} \quad \text{There is nothing here that guarantees that the answer sets of lp(\Gamma) will be interpretations — complete and consistent sets of val}(c,v) \text{ and } \lnot \text{val}(c,v) \text{ literals. They will be consistent but not necessarily complete.} \]

How to make them complete? We have to add some more constraints. I won’t go into detail because we are primarily interested in \( C^+ \) not in causal theories per se. In \( C^+ \) translations the necessary constraints will be done in a special way.

One last little note

The very observant might have noticed that the causal rule

\[ L \leftarrow L_1 \land \cdots \land L_n \]

translates to the Reiter default

\[ \text{L} \leftarrow L_1 \land \cdots \land L_n \]

Whereas the logic program clause

\[ L \leftarrow \lnot L_1, \ldots, \lnot L_n \]

translates to the Reiter default

\[ \text{L} \leftarrow L_1, \ldots, L_n \]

These are not equivalent, in general. Compare, for example

\[ \begin{array}{c}
\frac{p \land q}{r} \\
\frac{p, q}{r} \quad \text{and} \quad \frac{\lnot p \lor \lnot q}{r}
\end{array} \]

But they are equivalent when causal rules have only atoms in the head, i.e., when they can be translated to a logic program.

References