491 Knowledge Representation

Default Logic (Reiter)

Marek Sergot
Department of Computing
Imperial College, London

February 2004; February 2007 v1.1; February 2017 v1.1j

Ray Reiter. A logic for default reasoning. *Artificial Intelligence* 13:81–132 (1980).

Very extensively studied. Many variations + generalisations.

Further reading: V.W. Marek, M. Truszczyński. *Nonmonotonic Logic*. Springer-Verlag, 1993. (Ch.2–6)

Notation

$\mathcal{L}$ is some (logical) language, usually propositional or (a fragment of) first-order predicate logic. $\mathcal{L}$ is closed under truth-functional operations. (Thus, if $\alpha \in \mathcal{L}$ then $\neg\alpha \in \mathcal{L}$, and if $\alpha \in \mathcal{L}$ and $\beta \in \mathcal{L}$ then $\alpha \vee \beta \in \mathcal{L}$, $\alpha \wedge \beta \in \mathcal{L}$, $\alpha \rightarrow \beta \in \mathcal{L}$, etc.)
Lower-case Greek letters $\alpha, \beta, \gamma, \ldots$ range over formulas.
Upper case letters $A, B, \ldots, S, \ldots, W, X, Y, \ldots$ represent sets of formulas.

$\mathcal{M}$ is a model for $\mathcal{L}$.

$\mathcal{M} \models \alpha$ means that formula α evaluates to true in model $\mathcal{M}$.
$\models \alpha$ means that formula α evaluates to true in all models $\mathcal{M}$.
$A \models \alpha$ means that α is true in all models of A.

$\mathrm{Th}(A)$ stands for the set of all classical truth-functional consequences of A. $\alpha \in \mathrm{Th}(A)$ when α follows from A in classical propositional logic PL, i.e. when $A \models \alpha$ (α is true in all models of A).

Defaults as inference rules

Default logic = classical logic + default rules of inference
Default rules

$$\frac{\alpha : \beta_1, \ldots, \beta_m}{\gamma} \qquad (m \geq 0)$$

where α, β_1, …, β_m, γ are all *formulas* (not just atoms or literals as in (extended) logic programs).

α: 'prerequisite'
$\beta_1, \ldots, \beta_m$: consistency conditions or 'justification'
γ: 'consequent'

Informally:

If α is derivable and $\beta_1, \ldots, \beta_m$ all consistent, then derive γ.
If α is derivable and $\neg\beta_1, \ldots, \neg\beta_m$ not derivable, then derive γ.

And notice: $\dfrac{\alpha : \beta_1, \ldots, \beta_m}{\gamma}$ is not the same as $\dfrac{\alpha : \beta_1 \wedge \cdots \wedge \beta_m}{\gamma}$.
In the latter case there is just one formula in the consistency condition.

Examples

- $\dfrac{\mathsf{bird}(x) \,:\, \neg\mathsf{ab\_fly}(x)}{\mathsf{fly}(x)}$
  ('Open defaults' — those containing free variables — are a scheme for all their ground instances.)
- $\dfrac{\mathsf{big}(x) \,:\, \mathsf{strong}(x)}{\mathsf{strong}(x)}$
- $\dfrac{\mathsf{quaker}(x) \,:\, \mathsf{pacifist}(x)}{\mathsf{pacifist}(x)}$, $\dfrac{\mathsf{republican}(x) \,:\, \neg\mathsf{pacifist}(x)}{\neg\mathsf{pacifist}(x)}$
- $\dfrac{\mathsf{on}(x,y,t) \wedge \mathsf{next}(t,t') \,:\, \neg\mathsf{moved}(x,t,t')}{\mathsf{on}(x,y,t')}$ (a kind of 'frame axiom')
- $\dfrac{\mathsf{bird}(x) \,:\, \neg\mathsf{penguin}(x), \neg\mathsf{ostrich}(x), \neg\mathsf{wounded}(x)}{\mathsf{fly}(x) \vee \mathsf{dead}(x)}$
- $\dfrac{\mathsf{bird}(x) \;\wedge \forall y\,(\,p(y) \rightarrow q(x,y)\,) : s(x) \wedge q(x,x), \neg\mathsf{dead}(x)}{\forall y\,(\,q(y,y) \rightarrow p(x)\,) \vee \forall y\,\mathsf{bird}(y)}$

I have absolutely no idea what the last rule is supposed to represent. I just wanted to emphasise how general the form of default rules is.
There will be examples of how to use defaults later.

Later we will see that clauses of a (normal or extended) logic program

$$L \leftarrow L_1, \ldots, L_m, \texttt{not}\ L_{m+1}, \ldots, \texttt{not}\ L_n$$

are equivalent to default rules of a certain form.

A default rule is more general. It is like a clause in a logic program of the form:

$$\gamma \leftarrow \alpha_1, \ldots, \alpha_m, \texttt{not}\ \beta_{m+1}, \ldots, \texttt{not}\ \beta_n$$

where γ, α_1, …, α_m, β_{m+1}, …, β_n can be *any formulas*.
The above would be written as a default rule like this:

$$\frac{\alpha_1 \wedge \cdots \wedge \alpha_m \,:\, \neg\beta_{m+1}, \ldots, \neg\beta_n}{\gamma}$$

Informally: $\neg\beta_i$ is consistent when β_i is not derivable, and β_i is not derivable corresponds to negation by failure `not` β_i
But consistent with *what*? That comes next …

Extensions

Default rules are used to create *extensions* of classical theories. Extensions can be regarded as alternative acceptable sets of beliefs given facts W and a set of default rules D.

Default theory (D, W):

W is a set of (first-order) formulas (non-default rules and facts)
D is a (countable) set of default rules

An extension E of (D, W) is the smallest set of formulas containing W, closed under classical consequence Th, and closed under the default rules D that are applicable given E.
It remains to define what 'closed under the default rules D that are applicable given E' means. A formal definition follows presently.

Extensions are minimal: if E is an extension of (D, W) there is no other extension E' of (D, W) such that $E' \subset E$. We will prove this later. (Easy.)

Example

$$\begin{aligned} D &= \{\frac{\mathsf{bird}(x) \,:\, \neg\mathsf{penguin}(x)}{\mathsf{flies}(x)}\} \\ W &= \{\mathsf{bird}(\mathsf{Colin}),\ \mathsf{penguin}(\mathsf{Frank}),\ \forall x\,(\mathsf{penguin}(x) \rightarrow \mathsf{bird}(x)\,)\} \end{aligned}$$

Default theory (D, W) has an extension $\quad \mathrm{Th}(W \cup \{\mathsf{flies}(\mathsf{Colin})\})$.

Example: 'Nixon diamond'

Conflicting defaults induce multiple extensions.

$$\begin{aligned} D &= \{\frac{\mathsf{quaker}(x) \,:\, \mathsf{pacifist}(x)}{\mathsf{pacifist}(x)}\,, \frac{\mathsf{republican}(x) \,:\, \neg\mathsf{pacifist}(x)}{\neg\mathsf{pacifist}(x)}\} \\ W &= \{\mathsf{quaker}(\mathsf{Nixon}),\ \mathsf{republican}(\mathsf{Nixon})\,\} \end{aligned}$$

Default theory (D, W) has two extensions:

$$\begin{aligned} E_1 &= \mathrm{Th}(W \cup \{\mathsf{pacifist}(\mathsf{Nixon})\} \\ E_2 &= \mathrm{Th}(W \cup \{\neg\mathsf{pacifist}(\mathsf{Nixon})\} \end{aligned}$$

This example is commonly called "the Nixon diamond".

So what do we conclude?

- *'brave'* or *'credulous'* reasoning: Nixon is a pacifist and is not a pacifist;
- *'cautious'* or *'sceptical'* reasoning: Nixon pacifist? — no conclusion

In general:

- α follows from (D, W) by *'brave'/'credulous'* reasoning when α in any extension of (D, W): $\quad \alpha \in \bigcup ext(D, W)$;
- α follows from (D, W) by *'cautious'/'sceptical'* reasoning when α in all extensions of (D, W): $\quad \alpha \in \bigcap ext(D, W)$.

Here I'm writing $ext(D, W)$ for the set of extensions of (D, W).

Reiter's original definition of an extension is quite elaborate and rather subtle. It is shown later for reference (no need to memorise). It is much easier to define extensions in terms of *reduct* — exactly the same idea as for stable models (answer sets) of logic programs but much more general.
First we start with monotonic rules of inferences, and closures (of a set of formulas) under ordinary (monotonic) rules.

Closure under (monotonic, non-default) rules

Definition Let S be a set of formulas. Let R be a set of rules $\frac{\alpha}{\gamma}$.

$$\mathrm{T}_R(S) \stackrel{\text{def}}{=} \{\gamma \mid \frac{\alpha}{\gamma} \in R \text{ and } \alpha \in S\}$$

$\mathrm{T}_R(S)$ is the set of formulas that are obtained from S by one application of the rules in R.
Clearly T_R is monotonic: $S_1 \subseteq S_2 \;\Rightarrow\; \mathrm{T}_R(S_1) \subseteq \mathrm{T}_R(S_2)$.
Clearly T_R is also monotonic in R: $R_1 \subseteq R_2 \;\Rightarrow\; \mathrm{T}_{R_1}(S) \subseteq \mathrm{T}_{R_2}(S)$.

A set of formulas S is closed under rules R when, for every $\frac{\alpha}{\gamma} \in R$ and $\alpha \in S$, we have $\gamma \in S$. Or in other words, when $\mathrm{T}_R(S) \subseteq S$.

Note I deliberated long and hard about the choice of notation here. I chose $\mathrm{T}_R(S)$ because of the obvious analogy with the immediate consequence operator T_P of a (definite) logic program P:

$T_P(I) \stackrel{\text{def}}{=} \{A \mid A \leftarrow B_1, \ldots B_m$ is a ground instance of a clause in P and $\{B_1, \ldots, B_m\} \subseteq I\}$

A definite logic program P can be regarded as a set of rules: let P_{rules} be the set of rules obtained by replacing every ground clause $A \leftarrow B_1, \ldots B_m$ in P by the rule $\frac{B_1 \wedge \cdots \wedge B_m}{A}$.
Then $T_P(I) = \mathrm{T}_{P_{\text{rules}}}(I)$.
On the left hand side, T_P is the immediate consequence operator for the logic program P and acts on a set of atoms I. On the right, $\mathrm{T}_{P_{\text{rules}}}$ is the operator that applies rules P_{rules} to sets of formulas, not just sets of atoms.
I hope this does not confuse.

Just for the record: When P is a normal logic program (i.e., where some conditions can be negation by failure literals)

$T_P(I) \stackrel{\text{def}}{=} \{A \mid A \leftarrow B_1, \ldots B_m,$ `not` $B_{m+1}, \ldots$ `not` B_n is a ground instance of a clause in P and $\{B_1, \ldots, B_m\} \subseteq I$ and $B_{m+1} \notin I, \ldots, B_n \notin I\}$

For P a normal logic program, T_P is not monotonic in general. The operator $T'_P(I) \stackrel{\text{def}}{=} I \cup T_P(I)$ used in the iterated fixpoint construction for stratified programs is not monotonic either but is 'progressive'. The stratification of P makes T'_P sufficiently well behaved for construction of the iterated fixpoint. (The proofs were omitted.)

When P is a normal logic program with some occurences of negation by failure `not` , can we also define a set of corresponding rules P_{rules}? Yes, but these rules are Reiter default rules and not ordinary (monotonic) rules as in the case of a definite logic program.

End of note

Definition Let W be a set of formulas and R a set of rules. The *closure* of formulas W under rules R — denoted $\mathrm{Cn}_R(W)$ — is the smallest (set inclusion) set of formulas S such that

1. $W \subseteq S$
2. $\mathrm{Th}(S) \subseteq S$ (S is closed under classical propositional consequence Th)
3. $\mathrm{T}_R(S) \subseteq S$ (S is closed under the rules R, equivalently, under the operator T_R)

The requirement that $\mathrm{Cn}_R(W)$ is the *smallest* such set captures the idea that every formula in $\mathrm{Cn}_R(W)$ is 'grounded in' or 'supported by' W and R, in the sense that it is derivable from W using the rules in R and the inference rules of classical propositional logic (Th).

(Earlier I used the notation $Cl_P(EDB)$ for P a (definite) logic program. In this notation $Cl_P(EDB) = \mathrm{Cn}_{P_{\text{rules}}}(EDB)$.)

How do we know that $\mathrm{Cn}_R(W)$ exists and/or is unique?

Proposition

- $\mathcal{L}$ (the set of all formulas) satisfies the closure conditions (1)–(3) of $\mathrm{Cn}_R(W)$.
- If sets S_1 and S_2 both satisfy the closure conditions (1)–(3) of $\mathrm{Cn}_R(W)$, then so does their intersection $S_1 \cap S_2$.
- $\mathrm{Cn}_R(W)$ is the intersection of all sets S satisfying the closure conditions (1)–(3).

Proof: (i) is obvious. (iii) follows from (ii). (ii) is an easy exercise.

Some properties of $\mathrm{Cn}_R(W)$

Straight from the definition of $\mathrm{Cn}_R(W)$:

- $W \subseteq \mathrm{Cn}_R(W)$
- $\mathrm{Th}(\mathrm{Cn}_R(W)) \subseteq \mathrm{Cn}_R(W)$
- $\mathrm{T}_R(\mathrm{Cn}_R(W)) \subseteq \mathrm{Cn}_R(W)$

And now we can prove (tutorial exercises):

- $W \subseteq \mathrm{Cn}_R(W)$ ('inclusion')
- $\mathrm{Cn}_R(W) \subseteq \mathrm{Cn}_R(W \cup X)$, any set of formulas X (Cn_R is monotonic)
  which is equivalent to $A \subseteq B \;\Rightarrow\; \mathrm{Cn}_R(A) \subseteq \mathrm{Cn}_R(B)$
- $X \subseteq \mathrm{Cn}_R(W) \;\Rightarrow\; \mathrm{Cn}_R(W \cup X) \subseteq \mathrm{Cn}_R(W)$ ('cut' *alias* 'cumulative transitivity')
  which is equivalent to $W \subseteq W' \subseteq \mathrm{Cn}_R(W) \;\Rightarrow\; \mathrm{Cn}_R(W') \subseteq \mathrm{Cn}_R(W)$
- $\mathrm{Cn}_R(\mathrm{Cn}_R(W)) \subseteq \mathrm{Cn}_R(W)$ ('closure')
  (Recall that Cn_R monotonic implies 'cut' is equivalent to 'closure')
- Cn_R is a classical consequence relation
- $\mathrm{Th}(W) \subseteq \mathrm{Cn}_R(W)$ ('supraclassical')
- compactness: if $\alpha \in \mathrm{Cn}_R(W)$ then $\alpha \in \mathrm{Cn}_R(W')$ for some finite subset $W' \subseteq W$.

We can also show:

- $\mathrm{Cn}_R(W)$ is the smallest set of formulas S such that $S = \mathrm{Th}(W \cup \mathrm{T}_R(S))$.
  (This is useful for proving certain properties of default theories.)

Proofs of all the above, except compactness which is a little bit more fiddly, are left as tutorial exercises.

Inductive characterisations

$$\begin{aligned} E_0 &= W \\ E_{n+1} &= \mathrm{Th}(E_n \cup \mathrm{T}_R(E_n)\,) \\ \mathrm{Cn}_R(W) &= \bigcup_{n=0}^{\omega} E_n \end{aligned}$$

(Note: although $E_n \subseteq E_{n+1}$ we still need the big union in the definition of E because we have no other way of expressing that n 'goes up to infinity'.)

Where does this inductive definition come from? It comes from (proof later)

$$\mathrm{Cn}_R(W) = T_R''\uparrow^{\omega}(W)$$

where $T_R''(S) \stackrel{\text{def}}{=} \mathrm{Th}(S \cup \mathrm{T}_R(S)\,)$.

There are some alternative inductive characterisations, as we will see later. The most important of them is the following.

Inductive characterisation: 'base operator'

$\mathrm{Cn}_R(W)$ can also be characterised in terms of the 'base operator':

$$B_R(S) \stackrel{\text{def}}{=} S \cup \mathrm{T}_R(\mathrm{Th}(S)) = S \cup \{\gamma \mid \frac{\alpha}{\gamma} \in R,\, \alpha \in \mathrm{Th}(S)\}$$

The value of the 'base operator' B_R is that when R and W are both *finite* it provides a *finite representation* of the closure $\mathrm{Cn}_R(W)$.
This is important: the key objects in default logic (extensions) are infinitely large sets of formulas (otherwise they would not be closed under Th). It is easier to deal with finite representations of them, if we can.
It can be shown (details later) that

$$\mathrm{Cn}_R(W) = \mathrm{Th}(B_R\uparrow^{\omega}(W)\,)$$

So we get ...

Inductive characterisation

$$\begin{aligned} E_0 &= W \\ E_{n+1} &= B_R(E_n) = E_n \cup \mathrm{T}_R(\mathrm{Th}(E_n)\,) \\ &= E_n \cup \{\gamma \mid \frac{\alpha}{\gamma} \in R,\, \alpha \in \mathrm{Th}(E_n)\} \\ \mathrm{Cn}_R(W) &= \mathrm{Th}(\,\bigcup_{n=0}^{\omega} E_n\,) \end{aligned}$$

The value of this characterisation in terms of the 'base operator' B_R is that if R and W are both *finite* then so is each E_n and so is the union $\bigcup_{n=0}^{\omega} E_n$. We obtain a *finite representation* of the closure $\mathrm{Cn}_R(W)$.

Example

$$R = \{\frac{r}{\neg q},\, \frac{p}{s \vee t}\}, \qquad W = \{p \vee q,\, r\}$$

Let's compute $\mathrm{Cn}_R(W)$:

$$\begin{aligned} E_0 &= \{p \vee q,\, r\} \\ E_1 &= B_R(\{p \vee q,\, r\}) = \{p \vee q,\, r\} \cup \{\neg q\} \\ E_2 &= B_R(\{p \vee q,\, r,\, \neg q\}) = \{p \vee q,\, r,\, \neg q\} \cup \{s \vee t\} \qquad \text{because } p \in \mathrm{Th}(\{p \vee q,\, r,\, \neg q\}) \\ E_3 &= B_R(E_2) = E_2 \end{aligned}$$

So $\mathrm{Cn}_R(W) = \mathrm{Th}(\{p \vee q,\, r,\, \neg q,\, s \vee t\})$.

Notice that $\mathrm{Cn}_R(W)$ can also be written $\mathrm{Th}(\{p,\, r,\, \neg q,\, s \vee t\})$ since in propositional logic $(p \vee q) \wedge \neg q \equiv p \wedge \neg q$.

Algorithm

It's also easy to construct a simple algorithm for computing $B_R\uparrow^{\omega}(W)$. Notice that once a rule $\frac{\alpha}{\gamma}$ has been 'applied' it can be ignored: once the conclusion γ is included it cannot be removed, and so there is no point rechecking the rule $\frac{\alpha}{\gamma}$ any more.

Input: a finite set R of rules and a finite set W of formulas.
Output: $B_R\uparrow^{\omega}(W)$

$E_{\text{in}} := W$
repeat
  $E := E_{\text{in}}$
  $AR := \{\frac{\alpha}{\gamma} \in R \mid \alpha \in \mathrm{Th}(E)\,\}$
  $E_{\text{in}} := E \cup \{\gamma \mid \frac{\alpha}{\gamma} \in AR\}$
  $R := R - AR$
until $AR = \emptyset$
return(E)

Remark: integrity constraints

Recall the so-called *metalevel* or *epistemic* reading of an integrity constraint

$$\text{“if } \alpha \text{ then } \beta\text{”}$$

Informally, this is 'if α is in the database then β is in the database'.
So, if the database content is $\mathrm{Cn}(D)$ (some base D, some notion of consequence Cn) then we are saying:

$$\text{if } \alpha \in \mathrm{Cn}(D) \text{ then } \beta \in \mathrm{Cn}(D)$$

In other words:

$$\mathrm{Cn}(D) \text{ is closed under the rule } \frac{\alpha}{\beta}$$

And then the so-called 'theoremhood' or 'entailment' definition of satisfaction of an integrity constraint "if α then β" is:

$$\mathrm{Cn}(D) \text{ is closed under the rule } \frac{}{\alpha \to \beta}$$

(What about the so-called 'consistency' definition of integrity constraint satisfaction:

$$\neg(\alpha \to \beta) \notin \mathrm{Cn}(D)$$

Not so easy to express in terms of closure under a rule.

Comparison with material implication

Compare rules $\frac{\alpha}{\gamma}$ with truth-functional ('material') implications $\alpha \to \gamma$.
In both cases, given α we can derive γ. But there are differences: properties of material implication ('reasoning by cases', contrapositive, ...) do not hold for rules.

Example (no 'reasoning by cases')

Let $R = \{\frac{\alpha}{\gamma}, \frac{\beta}{\gamma}\}$.
$\gamma \in \mathrm{Cn}_R(\{\alpha\})$. $\gamma \in \mathrm{Cn}_R(\{\beta\})$. But $\gamma \notin \mathrm{Cn}_R(\{\alpha \vee \beta\})$. $\mathrm{Cn}_R(\{\alpha \vee \beta\}) = \mathrm{Th}(\{\alpha \vee \beta\})$.
(In other words, nothing new is obtained from $\mathrm{Th}(\{\alpha \vee \beta\})$ by adding rules $\frac{\alpha}{\gamma}$, $\frac{\beta}{\gamma}$.)

Example (no contrapositive)

Let $R = \{\frac{\alpha}{\gamma}\}$.
$\gamma \in \mathrm{Cn}_R(\{\alpha\})$. But $\neg\alpha \notin \mathrm{Cn}_R(\{\neg\gamma\})$. $\mathrm{Cn}_R(\{\neg\gamma\}) = \mathrm{Th}(\{\neg\gamma\})$.

The following property summarizes the relationship between rules and material implications. (It is not necessary to *memorize* it. Just read what it is saying. It is also instructive to look at how the proof works.)

Proposition (Makinson)
Let $mat(R)$ be the set of 'materialisations' of rules R:

$$mat(R) \stackrel{\text{def}}{=} \{\alpha \to \gamma \mid \frac{\alpha}{\gamma} \in R\}$$

Then

$$\mathrm{Cn}_R(W) \subseteq \mathrm{Th}(mat(R) \cup W)$$

Proof: It is sufficient to show that $\mathrm{Th}(mat(R) \cup W)$ satisfies the closure conditions for $\mathrm{Cn}_R(W)$ because, by definition, $\mathrm{Cn}_R(W)$ is the smallest set that satisfies those conditions.

- $W \subseteq \mathrm{Th}(W \cup mat(R))$. Obvious (Th inclusion).
- $\mathrm{Th}(W \cup mat(R))$ is closed under Th. Obvious (Th 'closure'/'idempotence').
- To show $\mathrm{Th}(W \cup mat(R))$ is closed under T_R, suppose $\gamma \in \mathrm{T}_R(\mathrm{Th}(W \cup mat(R)))$ and show $\gamma \in \mathrm{Th}(W \cup mat(R))$, as follows.
  If $\gamma \in \mathrm{T}_R(\mathrm{Th}(W \cup mat(R)))$ then there is a rule $\frac{\alpha}{\gamma}$ in R such that $\alpha \in \mathrm{Th}(W \cup mat(R))$. But if $\frac{\alpha}{\gamma} \in R$ then $\alpha \to \gamma \in mat(R)$ and hence $\alpha \to \gamma \in \mathrm{Th}(W \cup mat(R))$.
  Now if $\alpha \in \mathrm{Th}(W \cup mat(R))$ and $\alpha \to \gamma \in \mathrm{Th}(W \cup mat(R))$ then $\gamma \in \mathrm{Th}(W \cup mat(R))$, as required.

To show that the result does not hold the other way round, in general, consider either of the two examples above.
Let $R = \{\frac{\alpha}{\gamma}, \frac{\beta}{\gamma}\}$. Then $mat(R) = \{\alpha \to \gamma, \beta \to \gamma\}$.
Let $W = \{\alpha \vee \beta\}$. Then $\gamma \in \mathrm{Th}(\{\alpha \vee \beta\} \cup \{\alpha \to \gamma, \beta \to \gamma\})$ but $\gamma \notin \mathrm{Cn}_R(\{\alpha \vee \beta\})$.
And suppose $R = \{\frac{\alpha}{\gamma}\}$. $mat(R) = \{\alpha \to \gamma\}$.
Let $W = \{\neg\gamma\}$. Then $\neg\alpha \in \mathrm{Th}(\{\neg\gamma\} \cup \{\alpha \to \gamma\})$ but $\neg\alpha \notin \mathrm{Cn}_R(\{\neg\gamma\})$.

Extensions: definitions

Definition Let (D, W) be a default theory. Let E be a set of formulas.
Let D^E denote the (non-default) rules corresponding to the default rules in D that are applicable given E:

$$D^E \stackrel{\text{def}}{=} \{\frac{\alpha}{\gamma} \mid \frac{\alpha : \beta_1, \ldots \beta_m}{\gamma} \in D,\ \neg\beta_i \notin E, \text{for every } i \in 1..m\}$$

D^E is sometimes called the *reduct* of D given E. (No coincidence: see relationships with logic programs later.)

E is an extension of (D, W) when $E = \mathrm{Cn}_{D^E}(W)$.

Example

$$D = \{\frac{\mathsf{bird}(x) : \neg\mathsf{penguin}(x)}{\mathsf{flies}(x)}\}, \qquad W_{\text{penguins}} = \{\forall x\,(\mathsf{penguin}(x) \rightarrow \mathsf{bird}(x))\}$$

Colin is a bird.
Check that $E = \mathrm{Th}(W_{\text{penguins}} \cup \{\mathsf{bird}(\mathsf{Colin}), \mathsf{flies}(\mathsf{Colin})\})$ is an extension of $(D, W_{\text{penguins}} \cup \{\mathsf{bird}(\mathsf{Colin})\})$.
The reduct $D^E = \{\frac{\mathsf{bird}(\mathsf{Colin})}{\mathsf{flies}(\mathsf{Colin})}\}$.
$\mathrm{Cn}_{D^E}(W_{\text{penguins}} \cup \{\mathsf{bird}(\mathsf{Colin})\}) = \mathrm{Th}(W_{\text{penguins}} \cup \{\mathsf{bird}(\mathsf{Colin}), \mathsf{flies}(\mathsf{Colin})\}) = E$. As expected.

How was this obtained? The reduct D^E is just a set of ordinary, monotonic rules, so you can use any of the methods/results of the previous section, such as the 'base operator' and/or the associated algorithm.

Now suppose that Colin is a penguin.
Check that $E' = \mathrm{Th}(W_{\text{penguins}} \cup \{\mathsf{penguin}(\mathsf{Colin}), \mathsf{flies}(\mathsf{Colin})\})$ is not an extension of $(D, W_{\text{penguins}} \cup \{\mathsf{penguin}(\mathsf{Colin})\})$.
The reduct $D^{E'} = \emptyset$. $\mathrm{Cn}_{D^{E'}}(W_{\text{penguins}} \cup \{\mathsf{penguin}(\mathsf{Colin})\}) = \mathrm{Th}(W_{\text{penguins}} \cup \{\mathsf{penguin}(\mathsf{Colin})\}) \neq E'$.

Check that $E'' = \mathrm{Th}(W_{\text{penguins}} \cup \{\mathsf{penguin}(\mathsf{Colin})\})$ is an extension of $(D, W_{\text{penguins}} \cup \{\mathsf{penguin}(\mathsf{Colin})\})$.
The reduct $D^{E''} = \emptyset$. $\mathrm{Cn}_{D^{E''}}(W_{\text{penguins}} \cup \{\mathsf{penguin}(\mathsf{Colin})\}) = \mathrm{Th}(W_{\text{penguins}} \cup \{\mathsf{penguin}(\mathsf{Colin})\}) = E''$.

Example: 'Nixon diamond'

$$D = \{\frac{q : p}{p}\ , \frac{r : \neg p}{\neg p}\}, \qquad W = \{q, r\}$$

Default theory (D, W) has two extensions: $E_1 = \mathrm{Th}(\{q, r, p\})$ and $E_2 = \mathrm{Th}(\{q, r, \neg p\})$

Check: reduct $D^{E_1} = \{\frac{q}{p}\}$. $\mathrm{Cn}_{D^{E_1}}(\{q, r\}) = \mathrm{Th}(\{q, r, p\}) = E_1$.

Check: reduct $D^{E_2} = \{\frac{r}{\neg p}\}$. $\mathrm{Cn}_{D^{E_2}}(\{q, r\}) = \mathrm{Th}(\{q, r, \neg p\}) = E_2$.

Note: Be careful!

E is an extension of (D, W) when $E = \mathrm{Cn}_{D^E}(W)$. By definition, $\mathrm{Cn}_{D^E}(W)$ is the smallest set of formulas containing W, closed under Th, and closed under the reduct D^E, i.e., under the operator T_{D^E}.

The following definition looks equivalent but is NOT.

E is an *extension* of (D, W) when E is the smallest set of formulas such that:

1. $W \subseteq E$
2. $\mathrm{Th}(E) \subseteq E$
3. E is closed under the applicable default rules D given E: $\mathrm{T}_{D^E}(E) \subseteq E$.

To see the difference, consider the following example.
Let $D = \{\frac{p : q}{q}\}$, $W = \{p\}$.
Consider $E = \mathrm{Th}(\{p, \neg q\})$. You can see that E is not an extension of (D, W): $D^E = \emptyset$. And $\mathrm{Cn}_\emptyset(\{p\}) = \mathrm{Th}(\{p\}) \neq E$.

But E satisfies the conditions of the wrong definition above. $E = \mathrm{Th}(\{p, \neg q\})$ contains $W = \{p\}$. E is obviously closed under Th. And E is closed under rules D^E, trivially because $D^E = \emptyset$. Moreover, subsets of $E = \mathrm{Th}(\{p, \neg q\})$ do not satisfy these conditions. In particular, $E' = \mathrm{Th}(\{p\})$ does not, because the reduct $D^{E'} = \{\frac{p}{q}\}$, and E' is not closed under $D^{E'}$.

Where is the difference?

Right definition E is an extension of (D, W) when $D^E = R$ and E is the smallest set of formulas containing W and closed under R and Th.

Wrong definition E is an extension of (D, W) when E is the smallest set of formulas containing W and closed under D^E and Th.

That still looks very similar. Perhaps it is clearer like this:

Right definition $W \subseteq E$, $\mathrm{Th}(E) \subseteq E$, $\mathrm{T}_{D^E}(E) \subseteq E$,
$\neg\exists S[\, S \subset E, W \subseteq S, \mathrm{Th}(S) \subseteq S, \mathrm{T}_{D^E}(S) \subseteq S\,]$

Wrong definition $W \subseteq E$, $\mathrm{Th}(E) \subseteq E$, $\mathrm{T}_{D^E}(E) \subseteq E$,
$\neg\exists S[\, S \subset E, W \subseteq S, \mathrm{Th}(S) \subseteq S, \mathrm{T}_{D^S}(S) \subseteq S\,]$

(The difference is in the very last bit of each.)

My advice? Don't worry about it. Don't use the wrong definition. Just construct the reduct D^E and then the closure $\mathrm{Cn}_{D^E}(W)$.

Some properties

Given E, the reduct D^E is a (possibly empty) set of ordinary, non-default, monotonic rules. So we have available all the properties of (monotonic, non-default) closures.
For example: E is an extension of (D, W) when E is the smallest set of formulas S such that:

$$S \;=\; \mathrm{Th}(W \cup \mathrm{T}_{D^E}(S))$$

And we have various inductive characterisations (see below).

Clearly, if $E_1 \subseteq E_2$, then $D^{E_2} \subseteq D^{E_1}$. This is because any rule that passes the consistency check against E_2 must pass the consistency check against E_1. The following property follows immediately and is very useful.

Proposition $\mathrm{T}_{D^E}(S)$ is monotonic in D and S but anti-monotonic in E:

$$\begin{array}{rcl} D_1 \subseteq D_2 & \Rightarrow & T_{D_1{}^E}(S) \subseteq T_{D_2{}^E}(S) \\ S_1 \subseteq S_2 & \Rightarrow & T_{D^E}(S_1) \subseteq T_{D^E}(S_2) \\ E_1 \subseteq E_2 & \Rightarrow & T_{D^{E_2}}(S) \subseteq T_{D^{E_1}}(S) \end{array}$$

Here's a property of extensions mentioned earlier. Extensions are minimal: if E is an extension of (D, W) then no proper subset of E can be an extension of (D, W). Or put another way ...

Proposition (Extensions are minimal) If E and E' are extensions of (D, W) such that $E' \subseteq E$ then $E' = E$.
Proof: We just need to show $E \subseteq E'$.
$E' \subseteq E$ implies $\mathrm{Cn}_{D^E}(W) \subseteq \mathrm{Cn}_{D^{E'}}(W)$. So we have:

$$E = \mathrm{Cn}_{D^E}(W) \subseteq \mathrm{Cn}_{D^{E'}}(W) = E'$$

and so $E \subseteq E'$.

Inductive characterisation

Now we take the inductive characterisation of the closure Cn_R given earlier but apply to the special case of $\mathrm{Cn}_{D^E}(E)$. We get ...

A set of formulas E is an extension of a default theory (D, W) when:

$$\begin{array}{rcl} E_0 &=& W \\ E_{n+1} &=& \mathrm{Th}(E_n \cup \mathrm{T}_{D^E}(E_n)) \\ E &=& \bigcup_{n=0}^{\omega} E_n \end{array}$$

In the above, you first construct the reduct D^E and then use that to define the inductive characterisation of the closure $\mathrm{Cn}_{D^E}(E)$.

You can omit the separate reduct construction step if you like. Then ...

$$E_0 = W$$
$$E_{n+1} = \mathrm{Th}(E_n \cup \{\gamma \mid \frac{\alpha : \beta_1, \ldots \beta_m}{\gamma} \in D,\ \alpha \in E_n,\ \text{and } \neg\beta_i \notin E, \text{for every } i \in 1..m\})$$

Look *very carefully* at the consistency check in the definition of E_{n+1}. It refers to E not to E_n. So this isn't a *constructive* definition. We need to know the extension E before we start the inductive construction. This feature is sometimes referred to as *end-regulated* induction.

My advice? Construct the reduct first, then compute the closure.

Inductive characterisation (another version)

In case you do some background reading, here is another inductive characterisation. Some books present the following:

$$\begin{array}{rcl} E_0 &=& W \\ E_{n+1} &=& \mathrm{Th}(E_n) \cup \mathrm{T}_{D^E}(E_n) \\ E &=& \bigcup_{n=0}^{\omega} E_n \end{array}$$

This is not the same as the earlier one. But they are equivalent. (Explained why later.)

Inductive characterisation — 'base operator'

Much more important is this version in terms of the 'base operator'. As usual, this has the advantage that if D and W are both *finite* then we obtain a *finite representation* of an extension.
E is an extension of (D, W) when

$$\begin{array}{rcl} E_0 &=& W \\ E_{n+1} = B_{D^E}(E_n) &=& E_n \cup \mathrm{T}_{D^E}(\mathrm{Th}(E_n)) \\ &=& E_n \cup \{\gamma \mid \frac{\alpha}{\gamma} \in D^E \text{ and } \alpha \in \mathrm{Th}(E_n)\} \\ E &=& \mathrm{Th}(\bigcup_{n=0}^{\omega} E_n) \end{array}$$

Again, you can eliminate the separate step of constructing the reduct and do it all in one go if you like:

$$E_0 = W$$
$$E_{n+1} = E_n \cup \{\gamma \mid \frac{\alpha : \beta_1, \ldots \beta_m}{\gamma} \in D,\ \alpha \in \mathrm{Th}(E_n),\ \text{and } \neg\beta_i \notin E, \text{for every } i \in 1..m\}$$

Again, look very carefully at the consistency check. It refers to E not to E_n. This is still end-regulated induction, not a constructive definition.

(It is possible to give constructive inductive definitions but it's quite complicated and I won't cover it in these notes.)

Example (by Henry Prakken)

$$D = \{\frac{d : sp}{sk}, \frac{l : ph}{ph}, \frac{ph : \neg sp}{\neg sp}\}$$

(If a person is Dutch and it's consistent (s)he likes sport, then (s)he can skate.
Logicians typically like philosophy. Persons who like philosophy typically don't like sport.
What about Dutch logicians?)

$$W = \{d, l\}$$

Now if you just casually apply the rules in the order they are written you get $\mathrm{Th}(\{d, l\} \cup \{sk, ph, \neg sp\})$. But this isn't right. This isn't an extension of (D, W).

Check : the reduct is $\{\frac{l}{ph}, \frac{ph}{\neg sp}\}$. Computing the closure gives $\mathrm{Th}(\{d, l\} \cup \{ph, \neg sp\})$, which is different.

$E = \mathrm{Th}(\{d, l\} \cup \{ph, \neg sp\})$ is an extension of (D, W).

Check: the reduct D^E is $\{\frac{l}{ph}, \frac{ph}{\neg sp}\}$.

Compute $\mathrm{Cn}_{D^E}(W)$:

$$\begin{array}{lcl} E_0 & = & \{d, l\} \\ E_1 & = & B_{D^E}(E_0) = \{d, l\} \cup \{ph\} \\ E_2 & = & B_{D^E}(E_1) = \{d, l, ph\} \cup \{\neg sp\} \\ E_3 & = & B_{D^E}(E_2) = E_2 \end{array}$$

So $\mathrm{Cn}_{D^E}(W) = \mathrm{Th}(\{d, l, ph, \neg sp\}) = E$.

Comparison with Reiter's original

This section is just for *reference*. You do *not* have to learn it.
In case you do some background reading, here is Reiter's original definition of an extension of (D, W).

Let (D, W) be a default theory. Let S be a set of formulas.
Let $\Gamma_{D,W}(S)$ be the smallest set of formulas such that:

1. $W \subseteq \Gamma_{D,W}(S)$
2. $\mathrm{Th}(\Gamma_{D,W}(S)) \subseteq \Gamma_{D,W}(S)$
3. if $\frac{\alpha : \beta_1, \ldots \beta_m}{\gamma} \in D$, $\alpha \in \Gamma_{D,W}(S)$, every $\neg\beta_i \notin S$, then $\gamma \in \Gamma_{D,W}(S)$.

E is an extension of (D, W) when $E = \Gamma_{D,W}(E)$.

Notice that expressed in terms of the reduct D^S, condition (3) is

- if $\frac{\alpha}{\gamma} \in D^S$, $\alpha \in \Gamma_{D,W}(S)$, then $\gamma \in \Gamma_{D,W}(S)$, i.e. $\mathrm{T}_{D^S}(\Gamma_{D,W}(S)) \subseteq \Gamma_{D,W}(S)$.

So $\Gamma_{D,W}(S)$ is the smallest set of formulas such that:

- $W \subseteq \Gamma_{D,W}(S)$
- $\mathrm{Th}(\Gamma_{D,W}(S)) \subseteq \Gamma_{D,W}(S)$
- $\mathrm{T}_{D^S}(\Gamma_{D,W}(S)) \subseteq \Gamma_{D,W}(S)$

which means, by definition, that $\Gamma_{D,W}(S) = \mathrm{Cn}_{D^S}(W)$.
So we have E is an extension of (D, W) when $E = \Gamma_{D,W}(E) = \mathrm{Cn}_{D^E}(W)$.

(You don't need to learn the original definition. I included it in case you come across Reiter's definition, for instance at a party, and wonder why it's different from the definition given earlier.)

I think the formulation in terms of the reduct is clearer than the original.

Don't try to memorise the original definition.

Normal, semi-normal, and non-normal default rules

- general form: $\dfrac{\alpha : \beta_1, \ldots, \beta_m}{\gamma}$
- 'normal' default rule: $\dfrac{\alpha : \gamma}{\gamma}$
- 'semi-normal' default rule: $\dfrac{\alpha : \beta}{\gamma}$ where $\forall(\beta \to \gamma)$ is classically valid
- two common forms of semi-normal: $\dfrac{\alpha : \gamma \wedge \delta}{\gamma}$ $\dfrac{\alpha : \gamma}{\gamma \vee \delta}$

It was originally conjectured that normal default rules would be adequate for most, if not all, practical representation problems. But it is clear from even simple examples that the conjecture is false: we need semi-normal and general default rules.

Example (normal default)

$$\begin{aligned} D &= \{\frac{\mathsf{bird}(x) : \mathsf{flies}(x)}{\mathsf{flies}(x)}\} \\ W &= \{\mathsf{bird}(\mathsf{Jim}), \\ &\quad \mathsf{bird}(\mathsf{Frank}),\ \neg\mathsf{flies}(\mathsf{Frank})\} \end{aligned}$$

Example (normal default)

$$\begin{aligned} D &= \{\frac{\mathsf{bird}(x) : \neg\mathsf{baby}(x) \wedge \mathsf{flies}(x)}{\neg\mathsf{baby}(x) \wedge \mathsf{flies}(x)}\} \\ W &= \{\mathsf{bird}(\mathsf{Jim}), \\ &\quad \mathsf{bird}(\mathsf{Keith}),\ \mathsf{baby}(\mathsf{Keith}), \\ &\quad \mathsf{bird}(\mathsf{Alice}),\ \mathsf{baby}(\mathsf{Alice}),\ \mathsf{flies}(\mathsf{Alice}), \\ &\quad \mathsf{bird}(\mathsf{Frank}),\ \neg\mathsf{flies}(\mathsf{Frank})\} \end{aligned}$$

We get one extension $E = \mathrm{Th}(W \cup \{\neg\mathsf{baby}(\mathsf{Jim}),\ \mathsf{flies}(\mathsf{Jim})\}\,)$.

Check! (Tutorial exercise)

Example (semi-normal default)

$$\begin{aligned} D &= \{\frac{\mathsf{bird}(x) : \neg\mathsf{baby}(x) \wedge \mathsf{flies}(x)}{\mathsf{flies}(x)}\} \\ W &= \{\mathsf{bird}(\mathsf{Jim}), \\ &\quad \mathsf{bird}(\mathsf{Keith}),\ \mathsf{baby}(\mathsf{Keith}), \\ &\quad \mathsf{bird}(\mathsf{Francesca}),\ \mathsf{baby}(\mathsf{Francesca}),\ \mathsf{flies}(\mathsf{Francesca}), \\ &\quad \mathsf{bird}(\mathsf{Frank}),\ \neg\mathsf{flies}(\mathsf{Frank})\} \end{aligned}$$

We get one extension $E = \mathrm{Th}(W \cup \{\mathsf{flies}(\mathsf{Jim})\}\,)$.

Example (by Etherington)

$$W = \{\mathsf{has\_motive}(\mathsf{Jim})\}$$

Compare:

$$\begin{aligned} D_1 &= \{\frac{\mathsf{has\_motive}(x) : \mathsf{suspect}(x) \wedge \mathsf{guilty}(x)}{\mathsf{suspect}(x)}\} && \text{semi-normal} \\ D_2 &= \{\frac{\mathsf{has\_motive}(x) : \mathsf{suspect}(x) \wedge \mathsf{guilty}(x)}{\mathsf{suspect}(x) \wedge \mathsf{guilty}(x)}\} && \text{normal} \end{aligned}$$

Example (exceptions)

Here is another example, which we have seen already in various forms.

Birds (typically) fly.
Penguins are birds and (typically) do not fly.

Try this:

$$\begin{aligned} D &= \{\frac{\mathsf{bird}(x) : \mathsf{flies}(x)}{\mathsf{flies}(x)},\ \frac{\mathsf{penguin}(x) : \neg\mathsf{flies}(x)}{\neg\mathsf{flies}(x)}\} \\ W &= \{\forall x\,(\mathsf{penguin}(x) \to \mathsf{bird}(x)\,),\ \mathsf{bird}(\mathsf{Jim})\} \end{aligned}$$

(D, W) has one extension. It contains $\mathsf{flies}(\mathsf{Jim})$. Fine.
But $(D, W \cup \{\mathsf{penguin}(\mathsf{Jim})\})$ has **two** extensions: one in which Jim flies and one in which Jim does not fly. We haven't represented the exception structure.

Solution 1

$$\begin{aligned} D' &= \{\frac{\mathsf{bird}(x) : \mathsf{flies}(x) \wedge \neg\mathsf{penguin}(x)}{\mathsf{flies}(x)},\ \frac{\mathsf{penguin}(x) : \neg\mathsf{flies}(x)}{\neg\mathsf{flies}(x)}\} \\ W' &= W \end{aligned}$$

(The form of the first default rule is chosen so that we can assert, for example, that Jeremy, who is a bird but not a penguin, cannot fly.) OK, but not sufficiently general.

Solution 2 (more general)

$$\begin{aligned} D'' &= \{\frac{\mathsf{bird}(x) : \neg\mathsf{ab\_bird\_fly}(x)}{\mathsf{flies}(x)},\ \frac{\mathsf{penguin}(x) : \neg\mathsf{flies}(x)}{\neg\mathsf{flies}(x)}\} \\ W'' &= W \cup \{\forall x\,(\mathsf{penguin}(x) \to \mathsf{ab\_bird\_fly}(x)\,)\} \end{aligned}$$

Solution 3 (there are many other variations)

$$\begin{aligned} D''' &= \{\frac{\mathsf{bird}(x) : \mathsf{flies}(x) \wedge \neg\mathsf{ab\_bird\_fly}(x)}{\mathsf{flies}(x)},\ \frac{\mathsf{penguin}(x) : \neg\mathsf{flies}(x) \wedge \neg\mathsf{ab\_penguin\_fly}(x)}{\neg\mathsf{flies}(x)}\} \\ W''' &= W'' = W \cup \{\forall x\,(\mathsf{penguin}(x) \to \mathsf{ab\_bird\_fly}(x)\,)\} \end{aligned}$$

OK, but we haven't represented that penguins *typically* do not fly.

Reiter defaults: Problems/inadequacies

- Some default theories have no extensions.

e.g. $W = \emptyset$, $D = \{\frac{:\, p}{\neg p}\}$.

Compare the logic program: $\{p \leftarrow \texttt{not } p\}$.

This isn't really a 'problem' but to some people it is a bad feature of Reiter's definitions that there are default theories with no extension.

Some classes of default theories, e.g. those in which all default rules are normal, always have extensions.

There are variations of Reiter's default logic which guarantee the existence of extensions for all default theories. Details omitted.

- The precondition requirement is too strong (arguably):

$$D = \{\frac{\mathsf{bird}(x) \,:\, \mathsf{flies}(x)}{\mathsf{flies}(x)},\ \frac{\mathsf{bat}(x) \,:\, \mathsf{flies}(x)}{\mathsf{flies}(x)}\}$$
$$W = \{\mathsf{bird}(\mathsf{Alex}) \vee \mathsf{bat}(\mathsf{Alex})\}$$

Shouldn't this imply (defeasibly) that Alex can fly? It doesn't.

- The blocking effect of the consistency check is too weak (arguably):

$$D = \{\frac{\mathsf{suspect}(x) \,:\, \mathsf{innocent}(x)}{\mathsf{innocent}(x)}\}$$
$$W = \{\mathsf{suspect}(\mathsf{Ronnie}),\ \mathsf{suspect}(\mathsf{Reggie}), \neg(\mathsf{innocent}(\mathsf{Ronnie}) \wedge \mathsf{innocent}(\mathsf{Reggie}))\}$$

It says in W that they are not both innocent. But both are innocent by default.

Personally, I don't see anything problematic about this example. There are two extensions. (Tutorial exercise.) One has innocent(Ronnie), and therefore ¬innocent(Reggie). The other has innocent(Reggie), and ¬innocent(Ronnie). That seems fine to me.

- There is no (direct) way to reason about defaults.

> Normally, canaries are yellow.
> Yellow implies not blue.
> *Therefore:*
> Normally, canaries are not blue.

There is no (direct) way to get inferences like these in default logic. The consequences of a default theory are sets of formulas, not rules.

Later developments

- Default logics with guarantee of extensions for all theories
- Default logics with adjusted pre-requisite and/or justification (consistency) conditions
- Semantical characterisations of extensions (i.e., in terms of models)
- Prioritised default logics

⋮ *many others*

Default Logic and (Extended) Logic Programs

The (normal or extended) logic program clause

$$L \leftarrow L_1, \ldots, L_m, \texttt{not } L_{m+1}, \ldots, \texttt{not } L_n$$

has the same meaning as the non-normal default rule:

$$\frac{L_1 \wedge \cdots \wedge L_m \,:\, \overline{L_{m+1}}, \ldots, \overline{L_n}}{L}$$

where $\overline{L_i}$ represents the literal complementary to L_i, as usual.
Any logic program P can be translated to a default theory (D, W) as follows:

- W is the set of 'facts' in P, i.e., clauses in P with empty body.
- D is the set of default rules obtained by translating all other clauses in P to a default rule as shown above.

Now there is 1-1 correspondence between the extensions of (D, W) and the answer sets of P.

- If S is an answer set of P then $\mathrm{Th}(S)$ is an extension of (D, W).
- If E is an extension of (D, W) then $E = \mathrm{Th}(S)$ for exactly one answer set S of P. (In fact, this answer set $S = E \cap \mathit{Lit}(P)$).

Every logic program can be translated to a default theory, but of course not every default theory can be translated to a logic program. Default theories are much more general than logic programs (default theories allow arbitrary formulas in rules and not just literals).

PS. Cf. 'Irritating point of detail' in the notes on Extended Logic Programs.
The Reiter default theory

$$\langle\{\frac{:}{p},\ \frac{:}{\neg p}\},\ \emptyset\rangle$$

has an extension: it is $\mathrm{Th}(\{p, \neg p\}) = \mathcal{L}$. (Easy to check.)
The equivalent extended logic program is $\{p \leftarrow,\ \neg p \leftarrow\}$. That is why (to preserve the equivalence) it is convenient to say that this logic program, by definition, has an answer set, which is $\{p, \neg p\}$ (or *Lit* more generally if there are other literals in the language). (But it's a trivial point.)

Two final remarks: (1)

It should be obvious that, without loss of generality, it is sufficient to consider only Reiter default theories of the form $(D, \emptyset)$, i.e., where the 'facts' (formulas) W are empty.
Why? Because given (D, W) where the formulas W are not empty, we can write them equivalently as (non-default) inference rules: replace every formula α in W by the (non-default) rule $\frac{}{\alpha}$ and add that to the rules D.
Is it obvious that this is equivalent? Surely, but in case it is not, here it is in all its detail.
Given a set W of formulas let

$$R_W \stackrel{\text{def}}{=} \{\frac{}{\alpha} \mid \alpha \in W\}$$

Now, every Reiter default theory (D, W) has the same extensions as $(D \cup R_W, \emptyset)$.
First, the rules R_W are non-default rules with no 'consistency checks' so, for any set E of formulas, the reduct $R_W^E = R_W$.
Now it is easy to show that, for any E, $\mathrm{Cn}_{D^E \cup R_W}(\emptyset) = \mathrm{Cn}_{D^E}(W)$.
And more generally, for any set R of classical (non-default) rules R:

$$\mathrm{Cn}_R(W) = \mathrm{Cn}_{R \cup R_W}(\emptyset)$$

Proof: By definition $\mathrm{Cn}_{R \cup R_W}(\emptyset)$ is the smallest set S of formulas such that:

- $\emptyset \subseteq S$
- $\mathrm{Th}(S) \subseteq S$
- $\mathrm{T}_{R \cup R_W}(S) \subseteq S$

Clearly, $\mathrm{T}_{R \cup R_W}(S) = \mathrm{T}_R(S) \cup \mathrm{T}_{R_W}(S)$, and $\mathrm{T}_{R_W}(S) = W$.
So $\mathrm{Cn}_{R \cup R_W}(\emptyset)$ is the smallest set S of formulas such that:

- $\emptyset \subseteq S$
- $\mathrm{Th}(S) \subseteq S$
- $\mathrm{T}_R(S) \cup W \subseteq S$

or equivalently such that

- $\emptyset \subseteq S$
- $\mathrm{Th}(S) \subseteq S$
- $\mathrm{T}_R(S) \subseteq S$
- $W \subseteq S$

which by definition is $\mathrm{Cn}_R(W)$.

Two final remarks: (2)

Be careful when translating between default theories and logic programs. Do not confuse the rule $\frac{\alpha}{\gamma}$ with the material implication $\alpha \rightarrow \gamma$.
('Material implication' is the name for the simple kind of implication in classical logic whereby $A \rightarrow B \stackrel{\text{def}}{=} \neg A \rightarrow B$.)
I have made this point before. Rules are 'uni-directional'. Material implications in classical logic Th have contrapositives (and other properties).

Example The default theory (D, W)

$$D = \{\frac{:j}{j}, \frac{:k}{k}, \frac{:l}{l}\}, \quad W = \{k \rightarrow j,\ l \rightarrow \neg j\}$$

is equivalent to $(D \cup R_W, \emptyset)$:

$$D \cup R_W = \{\frac{:j}{j}, \frac{:k}{k}, \frac{:l}{l}, \frac{}{k \rightarrow j}, \frac{}{l \rightarrow \neg j}\}$$

(D, W) is *not* equivalent to $(D_{\text{bad}}, \emptyset)$:

$$D_{\text{bad}} = \{\frac{:j}{j}, \frac{:k}{k}, \frac{:l}{l}, \frac{k}{j}, \frac{l}{\neg j}\}$$

D_{bad} represents only one 'direction' of the material implications in W.
Compare the following two logic programs:

$$P_1 \left\{ \begin{array}{rcl} j & \leftarrow & \texttt{not}\ \neg j \\ k & \leftarrow & \texttt{not}\ \neg k \\ l & \leftarrow & \texttt{not}\ \neg l \\ j & \leftarrow & k \\ \neg j & \leftarrow & l \end{array} \right\}$$

no (consistent) answer sets!! (Check it for yourself!)
Informally: there are no rules defining $\neg k$ and $\neg l$. Any answer set must contain k and l, and hence j and $\neg j$.

$$P_2 \left\{ \begin{array}{rcl} j & \leftarrow & \texttt{not}\ \neg j \\ k & \leftarrow & \texttt{not}\ \neg k \\ l & \leftarrow & \texttt{not}\ \neg l \\ j & \leftarrow & k \\ \neg j & \leftarrow & l \\ \neg k & \leftarrow & \neg j \\ \neg l & \leftarrow & j \end{array} \right\}$$

two answer sets: $\{j, k, \neg l\},\ \{\neg j, \neg k, l\}$

P_1 is a translation of $(D_{\text{bad}}, \emptyset)$. P_2 is a translation of $(D \cup R_W, \emptyset)$ and hence of (D, W).
The last two clauses of P_2 are necessary to capture the contrapositives of the formulas in W. (Obviously it is not always so easy in more complicated, bigger examples.)

Appendix: More about operators, closures, and extensions

In the previous sections there were various results about closures, 'base operators', extensions, various inductive characterisations, and so on. In fact they are all (or almost all) properties that follow from elementary properties of 'operators' in general.
Here is a summary of those properties. Nearly everything here was in an earlier Appendix to fixpoint semantics for definite clause programs. Here it is again, with more examples to show how generally applicable these results are.

Operators

Here (again) is a summary of the theory of operators in a set U. The first part is very general. U can be any set. Operators are just mappings of $\wp(U)$ into itself.

Definition

1. An *operator* in U is any mapping $F\colon \wp(U) \to \wp(U)$.
2. An operator F is called *monotonic* (sometimes: monotone) if, for all subsets S_1, S_2 of U, $S_1 \subseteq S_2 \;\Rightarrow\; F(S_1) \subseteq F(S_2)$.
3. An operator F is called *anti-monotonic* (sometimes: anti-monotone) if, for all subsets S_1, S_2 of U, $S_1 \subseteq S_2 \;\Rightarrow\; F(S_2) \subseteq F(S_1)$.
4. An operator F is called *compact* (sometimes: finitizable) if, for all subsets S of U, if $\alpha \in F(S)$ then $\alpha \in F(S')$ for some finite subset $S' \subseteq S$.
5. An operator F is called *progressive* if, for all subsets S of U, $S \subseteq F(S)$.

Examples

- The immediate consequence operator T_P for a definite logic program P is an operator in $atoms(P)$, i.e. $T_P\colon \wp(atoms(P)) \to \wp(atoms(P))$.
The immediate consequence operator T_P is
  - monotonic: $I_1 \subseteq I_2 \;\Rightarrow\; T_P(I_1) \subseteq T_P(I_2)$.
  - not progressive: e.g. $P = \{p \leftarrow q\}$. $T_P(\{q\}) = \{p\}$. $\{p\} \not\subseteq \{q\}$.
  - compact, because the body of every clause has a finite number of conditions. Spelled out in detail: if $A \in T_P(I)$ then there is a ground instance $A \leftarrow B_1, \ldots, B_m$ of a clause in P such that $\{B_1, \ldots, B_m\} \subseteq I$. And so $A \in T_P(I')$ for a finite subset $I' = \{B_1, \ldots, B_m\}$ of I.

  We can also define an operator $T'_P(I) \stackrel{\text{def}}{=} I \cup T_P(I)$. T'_P is obviously monotonic, progressive, and compact (if P has a finite number of atoms).

- When P is a normal logic program (i.e., where some conditions can be negation by failure literals)

$$T_P(I) \stackrel{\text{def}}{=} \{A \mid A \leftarrow B_1, \ldots B_m, \texttt{not}\ B_{m+1}, \ldots \texttt{not}\ B_n \text{ is a ground instance of a clause in } P \text{ and } \{B_1, \ldots, B_m\} \subseteq I \text{ and } B_{m+1} \notin I, \ldots, B_n \notin I\}$$

For P a normal logic program, T_P is not monotonic in general, and is not progressive.
The operator $T'_P(I) \stackrel{\text{def}}{=} I \cup T_P(I)$ is not monotonic either but is 'progressive'.

- For extended logic programs, the corresponding operator (cf. the definition of answer set) is defined on sets of literals $T_P\colon \wp(Lit(P)) \to \wp(Lit(P))$.

$$T_P(S) \stackrel{\text{def}}{=} \{L \mid L \leftarrow L_1, \ldots L_m, \texttt{not}\ L_{m+1}, \ldots \texttt{not}\ L_n \text{ is a ground instance of a clause in } P \text{ and } \{L_1, \ldots, L_m\} \subseteq S \text{ and } L_{m+1} \notin S, \ldots, L_n \notin S\}$$

When P contains no occurrences of negation by failure `not` (but does possibly contain occurrences of $\neg$) T_P is monotonic, not progressive, compact.
As usual, $T'_P(S) \stackrel{\text{def}}{=} S \cup T_P(S)$ is monotonic when P contains no negation by failure literals, progressive, and compact if $Lit(P)$ is finite.

- Let R be a set of (ordinary, monotonic) rules. $\mathrm{T}_R\colon \wp(\mathcal{L}) \to \wp(\mathcal{L})$

$$\mathrm{T}_R(S) \stackrel{\text{def}}{=} \{\gamma \mid \frac{\alpha}{\gamma} \in R,\ \alpha \in S\}$$

is monotonic, not progressive, and compact (since rules have finite prerequisites α).
($\mathcal{L}$ is the set of all formulas of some given language, remember.)

- Classical propositional consequence $\mathrm{Th}\colon \wp(\mathcal{L}) \to \wp(\mathcal{L})$ is monotonic, progressive (since $W \subseteq \mathrm{Th}(W)$), and compact.

- The operator $T''_R\colon \wp(\mathcal{L}) \to \wp(\mathcal{L})$

$$T''_R(S) \stackrel{\text{def}}{=} \mathrm{Th}(S \cup \mathrm{T}_R(S))$$

is monotonic, progressive, and compact.

- The operator $T'_R\colon \wp(\mathcal{L}) \to \wp(\mathcal{L})$

$$T'_R(S) \stackrel{\text{def}}{=} \mathrm{Th}(S) \cup \mathrm{T}_R(S)$$

is monotonic, progressive, and compact.

Note that when we deal with logic programs (sets of clauses P) and sets of atoms/literals I, we define a progressive operator T'_P like this:

$$T'_P(I) \stackrel{\text{def}}{=} I \cup T_P(I)$$

But when we deal with sets of rules R and sets of arbitrary formulas S there are more options for defining a progressive operator. We have used both

$$T'_R(S) \stackrel{\text{def}}{=} \mathrm{Th}(S) \cup T_R(S)$$

and

$$T''_R(S) \stackrel{\text{def}}{=} \mathrm{Th}(S \cup T_R(S))$$

It turns out (conveniently) that the fixpoints of these operators are the same.

Theorem (Knaster-Tarski lemma) Let $F\colon \wp(U) \to \wp(U)$ be a monotonic operator.

1. The operator F possesses a least fixpoint. This least fixpoint is equal to $F{\uparrow}^{\alpha}(\emptyset)$ for some ordinal α.
2. If, in addition, F is compact then the least fixpoint of F is equal to $F{\uparrow}^{\omega}(\emptyset)$.

Theorem Let $F\colon \wp(U) \to \wp(U)$ be a monotonic and progressive operator. Then for every subset $X \subseteq U$, there is a least set S such that $X \subseteq S$ and S is a fixpoint of F. This set S is of the form $F{\uparrow}^{\alpha}(X)$. If, in addition, F is compact then this set S is equal to $F{\uparrow}^{\omega}(X)$.

Examples

- Let P be a definite logic program (i.e., no negation by failure `not` , no negation $\neg$).
$T_P(I) \subseteq I$ means that I is a model of P. It is easy to check that $T_P(I) \subseteq I$ iff $T'_P(I) \subseteq I$. Since $I \subseteq T'_P(I)$ the least (unique, smallest) model of P is also the least fixpoint of T'_P, i.e. (Knaster-Tarski) $T'_P{\uparrow}^{\omega}(\emptyset)$.
The least (unique, smallest) model of P that contains atoms EDB is $T'_P{\uparrow}^{\omega}(EDB)$.
- Recall that for an extended logic program P, the first step in defining an answer set S of literals is to define the answer sets of extended logic programs containing no occurrences of negation by failure `not` . For such a program P, an answer set S is the smallest set of literals closed under T_P, or $Lit(P)$ if S contains complementary literals A and $\neg A$. And so, S is an answer set of P when

$$S = \begin{cases} T'_P{\uparrow}^{\omega}(\emptyset) & \text{if } T'_P{\uparrow}^{\omega}(\emptyset) \text{ contains no pair of complementary literals } A \text{ and } \neg A, \\ Lit(P) & \text{otherwise.} \end{cases}$$

- As we shall see below, the smallest set of formulas closed under the operator T''_R and containing the set of formulas W is $\mathrm{Cn}_R(W)$. So $\mathrm{Cn}_R(W)$ is also the least fixpoint of T''_R, i.e., $\mathrm{Cn}_R(W) = T''_R{\uparrow}^{\omega}(W)$.
- Finally, what is the least fixpoint of Th? Since Th is monotonic, progressive and compact, the least fixpoint is $\mathrm{Th}{\uparrow}^{\omega}(\emptyset)$. But since $\mathrm{Th}(\mathrm{Th}(S)) = \mathrm{Th}(S)$, $\mathrm{Th}{\uparrow}^{\omega}(\emptyset) = \mathrm{Th}(\emptyset)$. So the least fixpoint of Th is $\mathrm{Th}(\emptyset)$ (the set of all propositional tautologies). Exactly as we should expect. And the least fixpoint of Th containing the set of formulas W is just $\mathrm{Th}(W)$. Again, exactly as we would expect.

You've also seen the following before. Here it is as a reminder.

Definition Let F be an operator in U and let X be a subset of U. The closure of X under the operator F — denoted $Cl_F(X)$ — is the smallest subset S of U such that:

$$X \subseteq S \qquad \text{and} \qquad F(S) \subseteq S$$

How do we know that $Cl_F(X)$ exists and/or is unique?

Proposition U satisfies the closure conditions (1)–(2) of $Cl_F(X)$.

Proposition If F is monotonic, then if subsets S_1 and S_2 of U both satisfy the closure conditions (1)–(2) of $Cl_F(W)$, then so does their intersection $S_1 \cap S_2$.

Proposition If F is monotonic, $Cl_F(X)$ is the intersection of all sets S satisfying the closure conditions (1)–(2).

$Cl_F(X)$ is the least set S satisfying $X \subseteq S$ and $F(S) \subseteq S$. If in addition F is progressive, $S \subseteq F(S)$. So then $Cl_F(X)$ is the least fixpoint of F that contains X.

Proposition If F is progressive, then $Cl_F(X)$ is the least fixpoint of F that contains X. If F is monotonic and progressive and compact, then $Cl_F(X) = F{\uparrow}^{\omega}(X)$.

(These examples were to help clarify the defintions and to show how widely applicable the results are.)

Closures under monotonic rules

Now let's consider the special case of operators mapping sets of formulas to sets of formulas.

Let W be a set of formulas and R a set of rules $\frac{\alpha}{\gamma}$. By definition, the *closure* of formulas W under rules R — denoted $\mathrm{Cn}_R(W)$ — is the smallest (set inclusion) set of formulas S such that

- $W \subseteq S$
- $\mathrm{Th}(S) \subseteq S$
- $\mathrm{T}_R(S) \subseteq S$

or equivalently, the smallest set of formulas S such that

- $W \subseteq S$
- $\mathrm{Th}(S) \cup \mathrm{T}_R(S) \subseteq S$

So $\mathrm{Cn}_R(W)$ is the closure of W under the operator

$$T'_R(S) \stackrel{\text{def}}{=} \mathrm{Th}(S) \cup \mathrm{T}_R(S)$$

Note: the terminology. For operator F in an arbitrary set (i.e., not necessarily a set of formulas) I speak of the closure of a subset X under the operator F. This I have written as $Cl_F(X)$.
Now we apply the general results to the special case where the sets are sets of formulas. Of particular interest is the closure of a set of formulas W under both classical consequence Th and (monotonic) rules R, or if you prefer, under the operators Th and T_R. This I am writing as $\mathrm{Cn}_R(W)$. Expressed in terms of the general notation, $\mathrm{Cn}_R(W) = Cl_{T'_R}(W)$.

End of note

Since T_R and Th are both monotonic, so is T'_R.
Since $S \subseteq \mathrm{Th}(S)$, T'_R is progressive.
Since Th is compact, and every rule in R has a finite 'body', then T'_R is also compact. So therefore ...

Proposition

1. $\mathrm{Cn}_R(W)$ is the least (smallest, unique) set S such that $W \subseteq S$ and $S = \mathrm{Th}(S) \cup \mathrm{T}_R(S)$.
2. $\mathrm{Cn}_R(W) = T'_R{\uparrow}^{\omega}(W)$

Inductive characterisation obtained from $T'_R{\uparrow}^{\omega}(W)$:

$$\begin{aligned} E_0 &= W \\ E_{n+1} &= \mathrm{Th}(E_n) \cup \mathrm{T}_R(E_n) \\ \mathrm{Cn}_R(W) &= \bigcup_{i=0}^{\omega} E_i \end{aligned}$$

$\mathrm{Cn}_R(W)$ — Alternative characterisation

$\mathrm{Cn}_R(W)$ is equivalent to the closure of W under the operator $T'_R(S) = \mathrm{Th}(S) \cup \mathrm{T}_R(S)$.
Now we show that it is also equivalent to the closure of W under another operator:

$$T''_R(S) \stackrel{\text{def}}{=} \mathrm{Th}(S \cup \mathrm{T}_R(S))$$

(You can skip the next result. You should certainly not try to memorize the proof. It is provided so you can see the reasoning.)

Proposition Any set S of formulas is closed under $T'_R(S) = \mathrm{Th}(S) \cup \mathrm{T}_R(S)$ iff it is closed under $T''_R(S) = \mathrm{Th}(S \cup \mathrm{T}_R(S))$, i.e., $\mathrm{Th}(S) \cup \mathrm{T}_R(S) \subseteq S$ iff $\mathrm{Th}(S \cup \mathrm{T}_R(S)) \subseteq S$.

Proof: Left-to-right: From LHS we have $\mathrm{T}_R(S) \subseteq S$ and $\mathrm{Th}(S) \subseteq S$.

$$\begin{aligned} \mathrm{T}_R(S) \subseteq S &\Rightarrow S \cup \mathrm{T}_R(S) \subseteq S \\ &\Rightarrow \mathrm{Th}(S \cup \mathrm{T}_R(S)) \subseteq \mathrm{Th}(S) \quad \text{(Th monotonic)} \\ &\Rightarrow \mathrm{Th}(S \cup \mathrm{T}_R(S)) \subseteq S \quad \text{because } \mathrm{Th}(S) \subseteq S \end{aligned}$$

Right-to-left: $S \subseteq S \cup \mathrm{T}_R(S) \Rightarrow \mathrm{Th}(S) \subseteq \mathrm{Th}(S \cup \mathrm{T}_R(S))$ (Th monotonic)
Now $\mathrm{Th}(S) \subseteq \mathrm{Th}(S \cup \mathrm{T}_R(S)) \subseteq S$ by RHS. So $\mathrm{Th}(S) \subseteq S$.
$S \cup \mathrm{T}_R(S) \subseteq \mathrm{Th}(S \cup \mathrm{T}_R(S))$ (Th inclusion), and RHS again gives $S \cup \mathrm{T}_R(S) \subseteq S$, and hence $\mathrm{T}_R(S) \subseteq S$.

So $\mathrm{Cn}_R(W)$ is also equivalent to the closure of W under the operator:

$$T''_R(S) \stackrel{\text{def}}{=} \mathrm{Th}(S \cup \mathrm{T}_R(S))$$

Since T_R and Th are both monotonic, so is T''_R.
Since $S \subseteq \mathrm{Th}(S)$, T''_R is progressive.
Since Th is compact, and every rule in R has a finite 'body', then T''_R is also compact. So therefore ...

Proposition

1. $\mathrm{Cn}_R(W)$ is the least (smallest, unique) set S such that $W \subseteq S$ and $S = \mathrm{Th}(S \cup \mathrm{T}_R(S))$.
2. $\mathrm{Cn}_R(W) = T''_R{\uparrow}^{\omega}(W)$

Inductive characterisation obtained from $T''_R{\uparrow}^{\omega}(W)$:

$$\begin{aligned} E_0 &= W \\ E_{n+1} &= \mathrm{Th}(E_n \cup \mathrm{T}_R(E_n)) \\ \mathrm{Cn}_R(W) &= \bigcup_{n=0}^{\omega} E_n \end{aligned}$$

$\mathrm{Cn}_R(W)$ — One more characterisation ('base operator')

$\mathrm{Cn}_R(W)$ is equivalent to the closure of W under the operators $T'_R(S) \stackrel{\text{def}}{=} \mathrm{Th}(S) \cup \mathrm{T}_R(S)$ and $T''_R(S) \stackrel{\text{def}}{=} \mathrm{Th}(S \cup \mathrm{T}_R(S))$.
Now we show that it can also be characterised in terms of the closure of W under the 'base operator':

$$B_R(S) \stackrel{\text{def}}{=} S \cup \mathrm{T}_R(\mathrm{Th}(S))$$

It's not quite equivalence, but nearly.

B_R is monotonic and obviously progressive.

As usual, don't bother to learn the proofs. They are included to show the reasoning but if you don't find them helpful, ignore them. The first result is just a stepping stone.

Proposition For any set of formulas S, $B_R(S) \subseteq T''_R(\mathrm{Th}(S))$

Proof: $T''_R(\mathrm{Th}(S)) = \mathrm{Th}(\mathrm{Th}(S)) \cup \mathrm{T}_R(\mathrm{Th}(S)) = \mathrm{Th}(S) \cup \mathrm{T}_R(\mathrm{Th}(S))$.
Since $S \subseteq \mathrm{Th}(S)$, $B_R(S) = S \cup \mathrm{T}_R(\mathrm{Th}(S)) \subseteq \mathrm{Th}(S) \cup \mathrm{T}_R(\mathrm{Th}(S)) = T''_R(\mathrm{Th}(S))$.

Here is the main result. (Everything else can be reconstructed from it.)

Proposition

$$\boxed{\mathrm{Cn}_R(W) = T'_R{\uparrow^\omega}(W) = T''_R{\uparrow^\omega}(W) = \mathrm{Th}(B_R{\uparrow^\omega}(W))}$$

Proof: First, observe that $W \subseteq \mathrm{Th}(B_R{\uparrow^\omega}(W))$, and $\mathrm{Th}(B_R{\uparrow^\omega}(W))$ is closed under Th, and $\mathrm{Th}(B_R{\uparrow^\omega}(W))$.
(The first two observations are indeed obvious. M & T simply assert the third one. I don't think it's obvious. I haven't been able to check it.)
By definition, $\mathrm{Cn}_R(W)$ is the least set of formulas satisfying these three conditions, and so $\mathrm{Cn}_R(W) \subseteq \mathrm{Th}(B_R{\uparrow^\omega}(W))$.

To prove the converse inclusion, $\mathrm{Th}(B_R{\uparrow^\omega}(W)) \subseteq \mathrm{Cn}_R(W)$, it is enough to show $B_R{\uparrow^\omega}(W) \subseteq T''_R{\uparrow^\omega}(W)$, because then, by monotonicity of Th, we have $\mathrm{Th}(B_R{\uparrow^\omega}(W)) \subseteq \mathrm{Th}(T''_R{\uparrow^\omega}(W))$, and $T''_R{\uparrow^\omega}(W)$ is closed under Th.

We prove $B_R{\uparrow^n}(W) \subseteq T''_R{\uparrow^n}(W)$ for all n by induction on n.
Base case ($n = 0$): $B_R{\uparrow^n}(W) = T''_R{\uparrow^n}(W) = W$.
Inductive step: assume $B_R{\uparrow^n}(W) \subseteq T''_R{\uparrow^n}(W)$.

$$\begin{aligned}
B_R{\uparrow^{n+1}}(W) &= B_R(B_R{\uparrow^n}(W)) \\
&\subseteq B_R(T''_R{\uparrow^n}(W)) \quad \text{(inductive hypothesis)} \\
&\subseteq T''_R(\mathrm{Th}(T''_R{\uparrow^n}(W))) \quad \text{(we showed } B_R(S) \subseteq T''_R(\mathrm{Th}(S))\text{)} \\
&= T''_R(T''_R{\uparrow^n}(W)) \quad (T''_R{\uparrow^n}(W) \text{ closed under Th}) \\
&= T''_R{\uparrow^{n+1}}(W)
\end{aligned}$$

Inductive characterisation obtained from $B_R{\uparrow^\omega}(W)$

$$\begin{aligned}
E_0 &= W \\
E_{n+1} &= B_R(E_n) = E_n \cup \mathrm{T}_R(\mathrm{Th}(E_n)) \\
\mathrm{Cn}_R(W) &= \mathrm{Th}(\textstyle\bigcup_{n=0}^{\omega} E_n)
\end{aligned}$$

Extensions

Now we just apply all the results above to the special case where the (monotonic) rules are reducts of default rules under some given set of formulas E.

Proposition A set of formulas E is an extension for (D, W) iff

$$E = \mathrm{Cn}_{D^E}(W) = T'_{D^E}{\uparrow^\omega}(W) = T''_{D^E}{\uparrow^\omega}(W) = \mathrm{Th}(B_{D^E}{\uparrow^\omega}(W))$$