Non-Monotonic Reasoning

Steffen Hölldobler
International Center for Computational Logic
Technische Universität Dresden
Germany

- Introduction
- Closed World Assumption
- Completion
- Circumscription
- Default Logic
- Answer Set Programming

"Logic is everywhere ..."
Introduction – The Missionaries and Cannibals Puzzle

Puzzle  Three missionaries and three cannibals come to a river. A rowboat that seats two is available. If the cannibals ever outnumber the missionaries on either bank of the river, the missionaries will be eaten. How shall the missionaries and the cannibals cross the river?

Representation  \( MCB \), where

- \( M \) denotes the number of missionaries on the left bank of the river,
- \( C \) denotes the number of cannibals on the left bank of the river, and
- \( B \) denotes the number of rowboats on the left bank of the river.

Solution

\[(331, 220, 321, 300, 311, 110, 221, 020, 031, 010, 021, 000)\]

Can it be derived as a logical consequence of a first order formalization?
Problems

- Unless it can be deduced that an object is present, we conjecture that it is not present.
- Unless there is something wrong with the boat or something else prevents the boat from using it, it can be used to cross the river.
Non-Monotonic Logics

- A logic $\langle A, \mathcal{L}, \models \rangle$ is said to be **non-monotonic** iff there exist $\mathcal{K}$, $\mathcal{K}'$ and $G$ such that

  $$\mathcal{K} \models G \text{ and } \mathcal{K} \cup \mathcal{K}' \nmid G,$$

  where $\mathcal{K}$ and $\mathcal{K}'$ are sets of formulas in $\mathcal{L}$ and $G$ is a formula in $\mathcal{L}$.

- Propositional and first order logic are monotonic $\iff$ Exercise
Closed World Assumption

► **Open world assumption (OWS)**
the only answers given to a query are those that can be obtained from proofs of the query, given the knowledge base.

► **Closed world assumption (CWS)**
certain additional answers are admitted as a result of a failure to prove a result.

► **Example**

\[ \mathcal{K} = \{ \text{lectures(steffen, cl1), lectures(steffen, cl5), lectures(michael, cl2), lectures(michael, cl5), lectures(heiko, cl4), lectures(horst, cl3)} \} \]

<table>
<thead>
<tr>
<th>query</th>
<th>OWS</th>
<th>CWS</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \mathcal{K} \models (\exists X) \text{lectures(steffen, X)} )</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>( \mathcal{K} \models \neg \text{lectures(michael, cl6)} )</td>
<td>no</td>
<td>yes</td>
</tr>
</tbody>
</table>
The Formal Theory

- Let \( \langle A, L, \models \rangle \) be a first order logic.
- Let \( \mathcal{K} \subseteq L \) be a satisfiable set of formulas.
- \( C(\mathcal{K}) = \{ G \mid \mathcal{K} \models G \} \) is the theory or closure of \( \mathcal{K} \).
- Let \( \mathcal{K}_{CWA} = \{ \neg A \mid A \text{ is a ground atom in } L \text{ and } \mathcal{K} \not\models A \} \).
- \( C_{CWA}(\mathcal{K}) = C(\mathcal{K} \cup \mathcal{K}_{CWA}) \) is the theory of \( \mathcal{K} \) under the closed world assumption.
Satisfiability

▶ Is $C_{CWA}(\mathcal{K})$ satisfiable?

▶ Consider $\mathcal{K} = \{\text{leakyValve} \lor \text{puncturedTube}\}$

▶ $\mathcal{K} \not\models \text{leakyValve}$

▶ $\mathcal{K} \not\models \text{puncturedTube}$

▶ $\{\neg\text{leakyValve}, \neg\text{puncturedTube}\} \subseteq \mathcal{K}_{CWA}$

▶ $\mathcal{K} \cup \mathcal{K}_{CWA} \supseteq \{\text{leakyValve} \lor \text{puncturedTube}, \neg\text{leakyValve}, \neg\text{puncturedTube}\}$

▶ $\mathcal{K} \cup \mathcal{K}_{CWA}$ is unsatisfiable!

▶ Theorem Let $\mathcal{K}$ be a satisfiable set of formulas in Skolem normal form. $C_{CWA}(\mathcal{K})$ is satisfiable iff $\mathcal{K}$ admits a least Herbrand model.
Models and the Closed World Assumption

- Let $M = (\mathcal{D}, \cdot^I)$ and $M' = (\mathcal{D}', \cdot'^I)$ be two models of $\mathcal{K}$.
- Let $\mathcal{R}$ be the set of relation symbols in $\mathcal{L}$ and $\mathcal{P} \subseteq \mathcal{R}$.
- $M$ is a submodel of $M'$ wrt $\mathcal{P}$ ($M \preceq_{\mathcal{P}} M'$) iff $\mathcal{D} = \mathcal{D}'$ and $\cdot^I, \cdot'^I$ are identical except that for all $q \in \mathcal{P}$ we find $q^I \subseteq q'^I$.
  \(\triangleright\) ($q^I$ is often called the extension of $q$ under $I$)
- If $\mathcal{P} = \mathcal{R}$ then we write $M \preceq M'$ instead of $M \preceq_{\mathcal{P}} M'$.
- A model $M$ of $\mathcal{K}$ is minimal iff for all models $M'$ of $\mathcal{K}$ we find that $M' \preceq M$ implies $M = M'$.
- $M \prec M'$ iff $M \preceq M'$ and $M \neq M'$.
- A model $M$ of $\mathcal{K}$ is the least model of $\mathcal{K}$ iff for all models $M'$ of $\mathcal{K}$ we find $M \neq M'$ implies $M \prec M'$.
- The closed world assumption eliminates non-least models!
Remarks

- $\mathcal{K} \not\models A$ cannot be decided in first-order logic!
- There are several extensions of the closed world assumption.
Can we add more complex formulas than negative ground atoms to a knowledge base?

Consider $\mathcal{F} = \{\text{tweedy, john}\}$ and $\mathcal{R} = \{\text{penguin}\}$.

Let $\mathcal{K} = \{\text{penguin(tweedy)}\}$.

Models for $\mathcal{K}$

- $M_1 = \{\text{penguin(tweedy)}\}$,
- $M_2 = \{\text{penguin(tweedy), penguin(john)}\}$.

$M_1 \prec M_2$.

How can the least model be computed?

Another example $\mathcal{K} = \{\text{penguin(tweedy), penguin(john)}\}$.

And another one $\mathcal{K} = \{(\forall X) (\neg \text{fly}(X) \rightarrow \text{fly}(X))\}$. 
Solitary Clauses

An occurrence of an $n$-ary predicate symbol $p$ in a clause $C$ is said to be

- positive iff we find terms $t_i$, $1 \leq i \leq n$, such that $p(t_1, \ldots, t_n) \in C$,
- negative iff we find terms $t_i$, $1 \leq i \leq n$, such that $\neg p(t_1, \ldots, t_n) \in C$.

A set $\mathcal{K}$ of clauses is said to be solitary wrt $p$

iff for each clause $C \in \mathcal{K}$ we find that if $C$ contains a positive occurrence of $p$ then $C$ does not contain another occurrence of $p$.

The clause

$$[\neg fly(tweedy), \neg fly(john), penguin(tweedy), \neg penguin(john)]$$

is solitary in $fly$ and
not solitary in $penguin$. 
The Completion Algorithm

Input  A set $\mathcal{K}$ of clauses and a predicate symbol $p$.

Output  The completion formula $C_{\mathcal{K},p}$ of $\mathcal{K}$ with respect to $p$.

1  Replace each clause $[L_1, \ldots, L_n, p(t_1, \ldots, t_m)] \in \mathcal{K}$ by

$$(\forall \bar{X})((\exists \bar{Y}) (X_1 \approx t_1 \land \ldots \land X_m \approx t_m \land \neg L_1 \land \ldots \land \neg L_n) \rightarrow p(\bar{X})), $$

where

$\bar{X} = X_1, \ldots, X_m$ is a sequence of ‘new’ variables,

$\bar{Y}$ is a sequence of those variables which occur in the clause, and

all occurrences of double negation are removed.

2  Let

$$\{(\forall \bar{X}) (C_i \rightarrow p(\bar{X})) \mid 1 \leq i \leq k\}$$

be the set of all formulas where $p$ occurs in the conclusion.

Return the completion formula

$$C_{\mathcal{K},p} = (\forall \bar{X}) (C_1 \lor \ldots \lor C_k \leftarrow p(\bar{X})).$$
An Example

\[ \mathcal{K} = \left\{ \neg\text{penguin}(Y) \lor \text{bird}(Y), \right. \]
\[ \text{bird(tweedy)}, \]
\[ \left. \neg\text{penguin}(john) \right\} \]

\[ \mathcal{K}_1 = \left\{ (\forall X)((\exists Y)(X \approx Y \land \text{penguin}(Y)) \rightarrow \text{bird}(X)), \right. \]
\[ (\forall X)(X \approx \text{tweedy} \rightarrow \text{bird}(X)), \]
\[ \neg\text{penguin}(john) \left\} \right. \]

\[ (\forall X)((\exists Y)(X \approx Y \land \text{penguin}(Y)) \lor X \approx \text{tweedy}) \rightarrow \text{bird}(X)) \]

\[ C_{\mathcal{K}, \text{bird}} = (\forall X)((\exists Y)(X \approx Y \land \text{penguin}(Y) \lor X \approx \text{tweedy}) \leftarrow \text{bird}(X)) \]
The Equational System $\mathcal{E}_C$

Let

$$\mathcal{E}_C = \{(\forall X, Y) f(X) \not\approx g(Y) \mid f, g \in \mathcal{F} \text{ and } f \neq g\} \cup \{(\forall X) t[X] \not\approx X \mid t \neq X\} \cup \{(\forall X, Y) (\bigvee_{i=1}^{n} X_i \not\approx Y_i \rightarrow f(X) \not\approx f(Y)) \mid n\text{-ary } f \in \mathcal{F}\} \cup \{(\forall X) X \approx X\} \cup \{(\forall X, Y) (\bigwedge_{i=1}^{n} X_i \approx Y_i \rightarrow f(X) \approx f(Y)) \mid n\text{-ary } f \in \mathcal{F}\} \cup \{\forall (\bigwedge_{i=1}^{n} X_i \approx Y_i \land p(X) \rightarrow p(Y)) \mid n\text{-ary } p \in \mathcal{R}\}$$
Predicate Completion

- Let $\mathcal{K}$ be a set of formulas, which is solitary in $p$.
- The predicate completion $C_C(\mathcal{K}, p)$ of $p$ is defined as
  $$C_C(\mathcal{K}, p) = \{ G \mid \mathcal{K} \cup \{C_{\mathcal{K},p}\} \cup \mathcal{E}_C \models G \}.$$
- **Theorem** Let $\mathcal{K}$ be a set of clauses which is solitary in $p$. If $\mathcal{K}$ is satisfiable, then so is $C_C(\mathcal{K}, p)$. 
Parallel Completion and Logic Programming

- \( \mathcal{K} = \{ \text{bird(tweedy)}, (\forall X) (\neg \text{bird}(X) \lor \text{ab}(X) \lor \text{fly}(X)) \} \)

- **Normal program clauses** \( p(t) \leftarrow A_1 \land \ldots \land A_m \land \neg A_{m+1} \land \ldots \land \neg A_n \).

- **A normal logic program** is a set of normal program clauses.

- **\( p \)** is defined in the logic program \( \mathcal{K} \) iff \( \mathcal{K} \) contains a clause with \( p \) occurring in its head.

- Let \( \mathcal{R}_D \) be the set of defined predicate symbols.

- **The completion** \( C_C(\mathcal{K}) \) of a normal logic program \( \mathcal{K} \) with defined predicate symbols \( \mathcal{R}_D \) is defined as

\[
C_C(\mathcal{K}) = \{ G \mid \mathcal{K} \cup \{ C_{\mathcal{K},p} \mid p \in \mathcal{R}_D \} \cup \mathcal{E}_C \cup \{ (\forall X) \neg p(X) \mid p \in \mathcal{R} \setminus \mathcal{R}_D \} \models G \}.
\]
Stratified Logic Programs

Consider $\mathcal{R}$, $\mathcal{F}$ and $\mathcal{V}$.

A level mapping is a total mapping $\operatorname{lvl} : \mathcal{R} \to \mathbb{N}$.

$\operatorname{lvl}(p)$ is the level of $p$.

A normal logic program $\mathcal{K}$ is stratified iff in each clause of the form

$$p(\bar{t}) \leftarrow p_1(\bar{s}_1) \land \ldots \land p_m(\bar{s}_m) \land \neg p_{m+1}(\bar{s}_{m+1}) \land \ldots \land \neg p_n(\bar{s}_n)$$

of $\mathcal{K}$ we find $\operatorname{lvl}(p) \geq \operatorname{lvl}(p_i)$, $1 \leq i \leq m$, and $\operatorname{lvl}(p) > \operatorname{lvl}(p_j)$, $m < j \leq n$.

Theorem

Let $\mathcal{K}$ be a stratified normal logic program. Then $\mathcal{C}_C(\mathcal{K})$ is satisfiable.

Exercise

Find non-stratifiable programs $\mathcal{K}_1$ and $\mathcal{K}_2$ such that

$\mathcal{C}_C(\mathcal{K}_1)$ is satisfiable and

$\mathcal{C}_C(\mathcal{K}_2)$ is unsatisfiable.
Negation as Failure

- We do not want to compute with the “only-if” parts and $\mathcal{E}_C$!

- $\{\neg A \mid \neg A \in C(\mathcal{K})\} \neq \{\neg A \mid \neg A \in C_C(\mathcal{K})\}

- Replace $\neg$ by $\sim$ called negation as failure.

- $\mathcal{K} = \{\text{bird(tweedy)}, \text{fly}(X) \leftarrow \text{bird}(X) \land \sim \text{ab}(X)\}$
Finitely Failed Search Trees

- A search tree is **finitely failed** iff it is finite and each leaf is labelled as a failure.

\[
\mathcal{K}' = \{ \begin{align*}
    ab(X) & \leftarrow \text{brokenWing}(X), \\
    ab(X) & \leftarrow \text{ratite}(X), \\
    \text{ratite}(X) & \leftarrow \text{ostrich}(X), \\
    \text{ratite}(X) & \leftarrow \text{emu}(X), \\
    \text{ratite}(X) & \leftarrow \text{kiwi}(X)
\end{align*} \}
\]

\begin{tikzpicture}
    \node {\text{\small \textnormal{\textit{ab}}(tweedy)}};
    \node[below left] {\text{\small \textnormal{\textit{brokenWing}}(tweedy)}}; \node[below right] {\text{\small \textnormal{\textit{ratite}}}(tweedy)};
    \node[below left] {\text{\small \textnormal{\textit{kiwi}}}(tweedy)}; \node[below right] {\text{\small \textnormal{\textit{emu}}}(tweedy)}; \node[below] {\text{\small \textnormal{\textit{ostrich}}}(tweedy)};
\end{tikzpicture}
SLDNF-Resolution

Let $G$ be a goal clause consisting of positive and negative literals, $\mathcal{K}$ a normal logic program, $L$ be the selected literal in $G$ and $A$ be a ground atom.

- If $L$ is positive, then each SLD-resolvent of $G$ using $L$ and some new variant of a clause in $\mathcal{K}$ is also an SLDNF-resolvent.
- If $L$ is a ground negative literal, i.e. $L = \sim A$, and the query $\leftarrow A$ finitely fails with respect to $\mathcal{K}$ and SLDNF-resolution, then the SLDNF-resolvent of $G$ is obtained from $G$ by deleting $L$.
- If $L$ is a ground negative literal, i.e. $L = \sim A$, and the query $\leftarrow A$ succeeds with respect to $\mathcal{K}$ and SLDNF-resolution, then the SLDNF-derivation of $G$ fails.
- If $L$ is negative and non-ground, then without loss of generality we may assume that each literal in $G$ is negative and non-ground. In this case $G$ is said to be blocked.

Theorem

Let $\mathcal{K}$ be a normal logic program. SLDNF-resolution is sound with respect to the completion of $\mathcal{K}$.
Classical Negation vs. Negation as Failure

- $\text{cross} \leftarrow \sim \text{train}$
- $\text{cross} \leftarrow \neg \text{train}$
Circumscription

- All approaches mentioned so far cannot handle disjunctions or $(\exists X)\ green(X)$.
- How can we compute their minimal models?
- Idea We want to conjecture that the tuples $(X_1, \ldots, X_m)$ which can be shown to satisfy an $m$-ary relation $p$ are all the tuples satisfying $p$.
  - According to McCarthy, we want to circumscribe $p$.
- $G\{p \mapsto Q\}$ is the string obtained from the formula $G$ by replacing each occurrence of $p$ by the predicate variable $Q$.
- The circumscription of $p$ in $G$:
  \[
  \text{Circ}(G, p) = (G\{p \mapsto Q\} \land (\forall X) (Q(X) \rightarrow p(X))) \rightarrow (\forall X)(p(X) \rightarrow Q(X))
  \]
- Note $\text{Circ}(G, p)$ is a schema.
Example 1

▶ Let $G = isblock(a) \land isblock(b) \land isblock(c)$.

▶ Then

$$Circ(G, isblock) = ((Q(a) \land Q(b) \land Q(c)) \land (\forall X) (Q(X) \rightarrow isblock(X))) \rightarrow (\forall X) (isblock(X) \rightarrow Q(X)).$$

▶ This schema can be instantiated replacing $Q(X)$ by $(X \approx a \lor X \approx b \lor X \approx c)$, where $\approx$ denotes syntactic equality.

▶ Let $G'$ be the corresponding instance of $Circ(G, isblock)$.

▶ Note

$$Q(a) = a \approx a \lor a \approx b \lor a \approx c \equiv \langle \rangle$$
$$Q(b) = b \approx a \lor b \approx b \lor b \approx c \equiv \langle \rangle$$
$$Q(c) = c \approx a \lor c \approx b \lor c \approx c \equiv \langle \rangle$$
Example 1 Continued

▸ Remember

\[ G = \text{isblock}(a) \land \text{isblock}(b) \land \text{isblock}(c) \]
\[ G' \equiv (\forall X) (X \approx a \lor X \approx b \lor X \approx c \rightarrow \text{isblock}(X)) \]
\[ \rightarrow (\forall X) (\text{isblock}(X) \rightarrow X \approx a \lor X \approx b \lor X \approx c) \]

▸ Let \( I \) be a Herbrand-interpretation (which interprets \( \approx \) as syntactic equality) such that \( I \models \{ G, G' \} \).

▸ for all \( t \in T(\mathcal{F}) \) we find \( [t \approx a \lor t \approx b \lor t \approx c \rightarrow \text{isblock}(t)]^I = \top. \) (\(*\))

▸ case \( t \in \{ a, b, c \} \)

\[ [t \approx a \lor t \approx b \lor t \approx c]^I = \top \] and \([\text{isblock}(t)]^I = \top, \) thus, (\(*\)) holds.

▸ case \( t \not\in \{ a, b, c \} \)

\[ [t \approx a \lor t \approx b \lor t \approx c]^I = \bot, \] thus, (\(*\)) holds.

▸ Therefore, we obtain

\[ \{ G, G' \} \models (\forall X) (\text{isblock}(X) \rightarrow (X \approx a \lor X \approx b \lor X \approx c)). \]
Non-monotonicity

- Circumscription is non-monotonic.
  - Reconsider Example 1.
  - \( \{G, G'\} \models \neg isblock(d) \).
  - Now let \( H = G \land isblock(d) \).
  - Then \( \{H, Circ(H, isblock)\} \not\models \neg isblock(d) \).
Example 2

Let $G = p(a) \lor p(b)$.

Then

$$\text{Circ}(G, p) = ((Q(a) \lor Q(b)) \land (\forall X) (Q(X) \rightarrow p(X)))$$

$$\rightarrow (\forall X) (p(X) \rightarrow Q(X)).$$

This schema can be instantiated replacing $Q(X)$ by $X \approx a$, where $\approx$ denotes syntactic equality.

We obtain

$$G_1 = ((a \approx a \lor b \approx a) \land (\forall X) (X \approx a \rightarrow p(X)))$$

$$\rightarrow (\forall X) (p(X) \rightarrow X \approx a)$$

$$\equiv (\forall X) (X \approx a \rightarrow p(X)) \rightarrow (\forall X) (p(X) \rightarrow X \approx a)$$

$$\equiv p(a) \rightarrow (\forall X) (p(X) \rightarrow X \approx a).$$
Example 2 Continued

▶ Remember

\[ G = p(a) \lor p(b) \]

and

\[ G_1 \equiv p(a) \rightarrow (\forall X) (p(X) \rightarrow X \approx a). \]

▶ \textit{Circ}(G, p) can also be instantiated replacing \( Q(X) \) by \( X \approx b \) and we obtain

\[ G_2 \equiv p(b) \rightarrow (\forall X) (p(X) \rightarrow X \approx b). \]

▶ Note

\[ \{ H_1 \lor H_2, H_1 \rightarrow H_1', H_2 \rightarrow H_2' \} \models H_1' \lor H_2'. \]

▶ Hence

\[ \{ G, G_1, G_2 \} \models (\forall X) (p(X) \rightarrow X \approx a) \lor (\forall X) (p(X) \rightarrow X \approx b). \]
The Main Result

Theorem
Let $G'$ be an instance of $\text{Circ}(G, p)$. $G'$ holds in all models of $G$ which are minimal in $\{p\}$.

Proof
Consider $(G\{p \mapsto Q\} \land (\forall X) (Q(X) \rightarrow p(X))) \rightarrow (\forall X)(p(X) \rightarrow Q(X))$.

Let $I$ be a model for $G$ which is minimal in $\{p\}$.

Let $p'$ be a predicate symbol such that $[G\{p \mapsto p'\} \land (\forall X) (p'(X) \rightarrow p(X))]^I = \top$.

To show $[(\forall X)(p(X) \rightarrow p'(X))]^I = \top$.

Suppose $[(\forall X)(p(X) \rightarrow p'(X))]^I = \bot$.

Hence $p^I \subsetneq p'^I$.

Because $[(\forall X) (p'(X) \rightarrow p(X))]^I = \top$ we find $p'^I \subseteq p^I$.

We conclude $p'^I \subseteq p^I$.

Because $[G\{p \mapsto p'\}]^I = \top$ we can construct a model $I'$ for $G$ which is identical to $I$ for all predicate letters different from $p$ and with $p'' = p'^I$.

Because $p'' = p'^I \subseteq p^I$ this contradicts the minimality of $I$ in $\{p\}$. 
Remarks

- **G follows minimally from** $\mathcal{K}$ with respect to $p$, written $\mathcal{K} \models \{p\} \ G$, iff $G$ holds in all models of $\mathcal{K}$ which are minimal in $\{p\}$.

- **Corollary** Let $G'$ be an instance of $\text{Circ}(G, p)$.
  If $\{G, G'\} \models H$ then $\{G\} \models \{p\} \ H$.

- Circumscribing a predicate may lead to an unsatisfiable theory.

- Under certain circumstances circumscription can be reduced to first order reasoning.

- Many extensions are known.
Default Logic

Most objects of sort $s$ have property $p$. Object $o$ is of sort $s$.

Does object $o$ have property $p$?

Most birds are flying. Tweedy is a bird.

Does Tweedy fly?

A first order formalization:

$$(\forall X) \ (bird(X) \land \neg penguin(X) \land \neg ostrich(X) \land \ldots \rightarrow fly(X)).$$

Problems

We do not know all exceptions.

We cannot conclude that Tweedy does not belong to one of the exceptions.

Idea We would like to conclude the Tweedy flies by default.
Default Reasoning

► Unless any information to the contrary is known we assume that
  ▶ exceptions are not logical consequences (CWA),
  ▶ we finitely failed to prove exceptions (NAF),
  ▶ it is consistent to assume that . . . (Default Logic).

► Default rules  \( \text{bird}(X) : \text{fly}(X) / \text{fly}(X) \).

► Exceptions  \{  (\forall X) (\text{penguin}(X) \rightarrow \neg \text{fly}(X)),
              (\forall X) (\text{ostrich}(X) \rightarrow \neg \text{fly}(X)),
              \ldots \} \)

► But how is consistency defined?

► Few objects of sort \( s \) have property \( p \):

\[ \text{man}(X) : \neg \text{moon}(X) / \neg \text{moon}(X) . \]
Default Rules

- Let $\langle A, \mathcal{L}, \models \rangle$ be a first order logic.
- A **default rule** is any expression of the form $G : G_1, \ldots, G_n / H$ or

$$G : G_1, \ldots, G_n \ \frac{H}{\circ}.$$ 

- $\circ$ is called prerequisite,
- $G_1, \ldots, G_n$ are called justifications,
- $\circ$ is called consequent.

- A default rule is said to be **closed** iff all formulas occurring in it are closed.
- It is said to be **open** iff it is not closed.

- It is a scheme representing the set of its ground instances.
If $G$ is missing, then $G \equiv \langle \rangle$.

If $n = 0$, then this is a rule in $\langle \mathcal{A}, \mathcal{L}, \models \rangle$.

If $n = 1$ and $G_1 = H$, then the default rule is said to be normal.

If $n = 1$ and $G_1 = H \land H'$, then the default rule is said to be semi-normal.
Default Knowledge Bases

A default knowledge base is a pair \( \langle \mathcal{K}_D, \mathcal{K}_W \rangle \), where

- \( \mathcal{K}_D \) is a set of at most countably many default rules and
- \( \mathcal{K}_W \) is a set of at most countably many closed first order formulas over \( \mathcal{A} \).

A default knowledge base is said to be closed iff all default rules occurring in it are closed.

It is said to be open iff it is not closed.

Example

\( \mathcal{K}_D : \)  
spouse(\( X, Y \)) \land htown(\( Y \)) \approx Z : htown(\( X \)) \approx Z / htown(\( X \)) \approx Z,  
employer(\( X, Y \)) \land location(\( Y \)) \approx Z : htown(\( X \)) \approx Z / htown(\( X \)) \approx Z.

\( \mathcal{K}_W : \)  
spouse(jane, john),  
htown(john) \approx munich,  
employer(jane, tud),  
location(tud) \approx dresden,  
(\( \forall X, Y, Z \) (htown(\( X \)) \approx Y \land htown(\( X \)) \approx Z \rightarrow Y \approx Z). \)
Extensions

► Let $\mathcal{K}$ be a set of closed first-order formulas and $\langle \mathcal{K}_D, \mathcal{K}_W \rangle$ a closed default knowledge base.

► Intuitively, an extension $\mathcal{K}$ of $\langle \mathcal{K}_D, \mathcal{K}_W \rangle$ should have the properties:
  ▶ $\mathcal{K}_W \subseteq \mathcal{K}$,
  ▶ $\mathcal{C}(\mathcal{K}) = \mathcal{K}$,
  ▶ $\mathcal{K}$ should be closed under the application of default rules.

► Let $\Gamma(\mathcal{K})$ be the smallest set satisfying the following properties:
  1 $\mathcal{K}_W \subseteq \Gamma(\mathcal{K})$.
  2 $\mathcal{C}(\Gamma(\mathcal{K})) = \Gamma(\mathcal{K})$.
  3 If $G : G_1, \ldots, G_n / H \in \mathcal{K}_D$, $G \in \Gamma(\mathcal{K})$ and for all $1 \leq j \leq n$ we find $\neg G_j \not\in \mathcal{K}$ then $H \in \Gamma(\mathcal{K})$.

$\mathcal{K}$ is said to be an extension of $\langle \mathcal{K}_D, \mathcal{K}_W \rangle$ iff $\Gamma(\mathcal{K}) = \mathcal{K}$.

► The set of extensions of $\langle \mathcal{K}_D, \mathcal{K}_W \rangle$ is a subset of the set of models for $\mathcal{K}_W$. 
Another Characterization of Extensions

**Theorem** Let \( \langle \mathcal{K}_D, \mathcal{K}_W \rangle \) be a closed default knowledge base and \( \mathcal{K} \) be a set of sentences. Define

\[
\mathcal{K}_0 = \mathcal{K}_W
\]

and for \( i \geq 1 \):

\[
\mathcal{K}_{i+1} = C(\mathcal{K}_i) \cup \{ H \mid G : G_1, \ldots, G_n / H \in \mathcal{K}_D, G \in \mathcal{K}_i \text{ and } \neg G_j \not\in \mathcal{K} \text{ for all } 1 \leq j \leq n \}.
\]

Then, \( \mathcal{K} \) is an extension of \( \langle \mathcal{K}_D, \mathcal{K}_W \rangle \) iff \( \mathcal{K} = \bigcup_{i=0}^{\infty} \mathcal{K}_i \).

**We have to guess extensions!**

\[
\mathcal{K}_D = \{ \text{bird}(X) : \text{fly}(X) \} \quad \mathcal{K}_W = \{ \text{bird(tweedy)} \}
\]

\( \mathcal{K} = C(\{ \text{bird(tweedy)}, \text{fly(tweedy)} \}) \) is an extension.
Another Example

**K_D**: $\text{spouse}(X, Y) \land \text{htown}(Y) \approx Z : \text{htown}(X) \approx Z / \text{htown}(X) \approx Z,$

$\text{employer}(X, Y) \land \text{location}(Y) \approx Z : \text{htown}(X) \approx Z / \text{htown}(X) \approx Z.$

**K_W**: $\text{spouse}(\text{jane}, \text{john}),$

$\text{htown}(\text{john}) \approx \text{munich},$

$\text{employer}(\text{jane}, \text{tud}),$

$\text{location}(\text{tud}) \approx \text{dresden},$

$(\forall X, Y, Z) (\text{htown}(X) \approx Y \land \text{htown}(X) \approx Z \rightarrow Y \approx Z).$

**Its extensions are**

$$C(K_W \cup \{\text{htown}(\text{jane}) \approx \text{munich}\})$$

and

$$C(K_W \cup \{\text{htown}(\text{jane}) \approx \text{dresden}\}).$$
Credolous vs. Sceptical Reasoning

► G follows credolously from \( \langle \mathcal{K}_D, \mathcal{K}_W \rangle \), in symbols \( \langle \mathcal{K}_D, \mathcal{K}_W \rangle \models_c G \),

iff there exists an extension \( \mathcal{K} \) of \( \langle \mathcal{K}_D, \mathcal{K}_W \rangle \) such that \( G \in \mathcal{K} \).

► G follows sceptically from \( \langle \mathcal{K}_D, \mathcal{K}_W \rangle \), in symbols \( \langle \mathcal{K}_D, \mathcal{K}_W \rangle \models_s G \),

iff for all extensions \( \mathcal{K} \) of \( \langle \mathcal{K}_D, \mathcal{K}_W \rangle \) we find \( G \in \mathcal{K} \).
Remarks

- Default logic is non-monotonic.
- Extensions are always satisfiable.
- Extensions may contain counter-intuitive facts.

\[
\mathcal{K}_W = \{ \text{broken(left arm)} \lor \text{broken(right arm)} \}
\]

\[
\mathcal{K}_D = \{ : \text{usable}(X) \land \neg \text{broken}(X) / \text{usable}(X) \}
\]

- There are many approaches extending default logic.
Answer Set Programming

Example

- Every student with a GPA of at least 3.8 is eligible.
- Every minority student with a GPA of at least 3.6 is eligible.
- Students with a GPA under 3.8 who do not belong to a minority are not eligible.
- The students whose eligibility is not determined by these rules are interviewed by the scholarship committee.

\[ \mathcal{K}_1 = \{ \text{eligible}(X) \leftarrow \text{highGPA}(X), \text{eligible}(X) \leftarrow \text{minority}(X) \land \text{fairGPA}(X), \neg \text{eligible}(X) \leftarrow \neg \text{highGPA}(X) \land \neg \text{minority}(X), \text{interview}(X) \leftarrow \sim \text{eligible}(X) \land \sim \neg \text{eligible}(X) \} \]

\[ \mathcal{K}_2 = \{ \text{fairGPA}(john) \leftarrow, \neg \text{highGPA}(john) \leftarrow \}. \]

What happens with John?
Rules and Programs

▶ Rules

\[ L_1 \lor \ldots \lor L_k \lor \neg L_{k+1} \lor \ldots \lor \neg L_l \leftarrow L_{l+1} \land \ldots \land L_m \land \neg L_{m+1} \land \ldots \land \neg L_n \]

▷ \( L_i \) are propositional literals.
▷ \( 0 \leq k \leq l \leq m \leq n \).
▷ If \( k = l = 0 \) then rules are called constraints.

▶ A program is a set of rules.
▷ \( \mathcal{K}_1 \cup \mathcal{K}_2 \).
Answer Sets

▶ Remember rules:

\[ L_1 \lor \ldots \lor L_k \lor \sim L_{k+1} \lor \ldots \lor \sim L_l \leftarrow L_{l+1} \land \ldots \land L_m \land \sim L_{m+1} \land \ldots \land \sim L_n \]

▶ Let \( \mathcal{M} \) be a satisfiable set of literals
and \( \mathcal{K} \) be a program where \( k = l \) and \( n = m \), i.e., rules are of the form

\[ L_1 \lor \ldots \lor L_k \leftarrow L_{l+1} \land \ldots \land L_m. \]

\( \mathcal{M} \) is said to be closed under \( \mathcal{K} \) if for every rule of \( \mathcal{K} \) we find that
\( \{L_1, \ldots, L_k\} \cap \mathcal{M} \neq \emptyset \) whenever \( \{L_{l+1}, \ldots, L_m\} \subseteq \mathcal{M} \).

\( \mathcal{M} \) is said to be an answer set for \( \mathcal{K} \)
if \( \mathcal{M} \) is minimal among the sets closed under \( \mathcal{K} \).

▶ Example \( \mathcal{K}_3 = \{s \lor r \leftarrow, \neg b \leftarrow r\} \)

▶ What are the answer sets of \( \mathcal{K}_3 \)?
▶ What happens if we add the constraint \( \leftarrow s \)?
Reducts and Answer Sets

- Let $\mathcal{K}$ be a program and $\mathcal{M}$ a satisfiable set of literals.
- The reduct $\mathcal{K}|\mathcal{M}$ of $\mathcal{K}$ relative to $\mathcal{M}$ is the set of rules
  \[ L_1 \lor \ldots \lor L_k \leftarrow L_{l+1} \land \ldots \land L_m \]
  such that
  \[ L_1 \lor \ldots \lor L_k \lor \sim L_{k+1} \lor \ldots \lor \sim L_l \leftarrow L_{l+1} \land \ldots \land L_m \land \sim L_{m+1} \land \ldots \land \sim L_n \]
  occurs in $\mathcal{K}$, \( \{L_{k+1}, \ldots, L_l\} \subseteq \mathcal{M} \) and \( \{L_{m+1}, \ldots, L_n\} \cap \mathcal{M} = \emptyset \).
- In $\mathcal{K}|\mathcal{M}$ the symbol $\sim$ does not occur anymore.
- $\mathcal{M}$ is said to be an answer set for $\mathcal{K}$ iff $\mathcal{M}$ is an answer set for $\mathcal{K}|\mathcal{M}$.
- Examples
  \[ \{p \leftarrow \sim q\}, \]
  \[ \{\neg p \leftarrow \sim p\}, \]
  \[ \{p \leftarrow \sim \neg p\}, \]
  \[ \{q \leftarrow p \land \sim q, p \leftarrow, q \leftarrow\} \]
  What happens if we delete $q \leftarrow$ from the last example?
Predicate Symbols, Constants and Variables

- We allow n-ary predicate symbols ranging over constants and variables.
- We view rules containing variable occurrences as schemas.

\[ \mathcal{K}_1 = \{ \begin{align*} 
\text{eligible}(X) & \leftarrow \text{highGPA}(X), \\
\text{eligible}(X) & \leftarrow \text{minority}(X) \land \text{fairGPA}(X), \\
\neg\text{eligible}(X) & \leftarrow \neg\text{highGPA}(X) \land \neg\text{minority}(X), \\
\text{interview}(X) & \leftarrow \sim\text{eligible}(X) \land \sim\neg\text{eligible}(X) 
\end{align*} \} \]

\[ \mathcal{K}_2 = \{ \begin{align*} 
\text{fairGPA}(\text{john}) & \leftarrow, \\
\neg\text{highGPA}(\text{john}) & \leftarrow \}. 
\]

- Its only answer set is:

\[ \{ \text{fairGPA}(\text{john}), \neg\text{highGPA}(\text{john}), \text{interview}(\text{john}) \}. \]

- What happens if we add \( \neg\text{minority}(\text{john}) \leftarrow ? \)
- Answer set programming is non-monotonic!
Programming with Answer Sets

► A Hamiltonian cycle is a cyclic tour through a graph visiting each vertex exactly once.

► The problem of finding a Hamiltonian cycle is known to be NP-complete.

► Let $G$ be a graph with vertices $0, \ldots, n$.

► Consider an alphabet with
  ▶ $\mathcal{F} = \{0, \ldots, n\}$ and
  ▶ $\mathcal{R} = \{reachable, in\}$.

► Idea
  ▶ WLOG let 0 be the starting vertex of the tour.
  ▶ $reachable(i)$ represents the fact that vertex $i$ is reachable from 0.
  ▶ $in(i, j)$ represents the fact that the edge from $i$ to $j$ is in the cycle.
  ▶ Specify a program such that for each answer set $\mathcal{M}$ we find:
    \[ \{ \langle u, v \rangle \mid in(u, v) \in \mathcal{M} \} \] is the set of edges in the Hamiltonian cycle.
Computing Hamiltonian Cycles

▶ Program

▷ \{ \text{in}(u, v) \lor \neg \text{in}(u, v) \leftarrow \mid \langle u, v \rangle \in G \}\}

▷ \{ \leftarrow \text{in}(u, v) \land \text{in}(u, w) \mid \langle u, v \rangle, \langle u, w \rangle \in G \text{ and } v \not\approx w \}\}

▷ \{ \leftarrow \text{in}(v, u) \land \text{in}(w, u) \mid \langle v, u \rangle, \langle w, u \rangle \in G \text{ and } v \not\approx w \}\}

▷ \{ \text{reachable}(u) \leftarrow \text{in}(0, u) \mid \langle 0, u \rangle \in G \}\}

▷ \{ \text{reachable}(v) \leftarrow \text{reachable}(u) \land \text{in}(u, v) \mid \langle u, v \rangle \in G \}\}

▷ \{ \leftarrow \sim \text{reachable}(u) \mid 0 \leq u \leq n \}\}

▶ You have to show that the answer sets of this program correspond to Hamiltonian cycles!
Computing Answer Sets

- **Paradigm shift**
  - Logic and constraint programming $\leadsto$ answer substitution.
  - Answer set programming $\leadsto$ model, i.e., answer set.
- **Quite successful in recent years.**
- **Systems**
  - Smodels
  - DLv
  - DeReS
  - Clasp