Non-Monotonic Reasoning

1 Introduction
2 Closed World Assumption
3 Completion
4 Circumscription
5 Default Logic
6 Answer Set Programming
Introduction – The Missionaries and Cannibals 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 they cross 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 $\langle \mathcal{A}, \mathcal{L}, \models \rangle$ is said to be non-monotonic iff there exist $\mathcal{F}$, $\mathcal{F}'$ and $\mathcal{G}$ such that

$$\mathcal{F} \models \mathcal{G} \text{ and } \mathcal{F} \cup \mathcal{F}' \not\models \mathcal{G},$$

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

Propositional and first order logic are monotonic.
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 database.

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

**Example**

\[ \mathcal{F} = \{ \text{lectures}(\text{steffen}, \text{cl001}) , \ \text{lectures}(\text{steffen}, \text{cl005}) , \ \text{lectures}(\text{michael}, \text{cl002}) , \ \text{lectures}(\text{michael}, \text{cl005}) , \ \text{lectures}(\text{heiko}, \text{cl004}) , \ \text{lectures}(\text{horst}, \text{cl003}) \}. \]

<table>
<thead>
<tr>
<th>query</th>
<th>OWS</th>
<th>CWS</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \mathcal{F} \models (\exists X) \text{lectures}(\text{steffen}, X) )</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>( \mathcal{F} \models \neg \text{lectures}(\text{michael}, \text{cl006}) )</td>
<td>no</td>
<td>yes</td>
</tr>
</tbody>
</table>
The Formal Theory

- Let $\langle A, \mathcal{L}, \models \rangle$ be a first order logic.

- $\mathcal{T}(\mathcal{F}) = \{ G \mid \mathcal{F} \models G \}$ is the theory of a satisfiable set $\mathcal{F}$ of formulas.

- Let $\overline{\mathcal{F}} = \{ \neg A \mid A$ is a ground atom in $\mathcal{L}$ and $\mathcal{F} \not\models A \}$.

- $\mathcal{T}_{CWA}(\mathcal{F}) = \mathcal{T}(\mathcal{F} \cup \overline{\mathcal{F}})$ is the theory of $\mathcal{F}$ under the closed world assumption.
Entailment under the Closed World Assumption

Let \( M_0 = \mathcal{T}(\mathcal{F}) \cup \overline{\mathcal{F}} \),
\( M_i = \{ H \mid \text{there exists } G \in M_{i-1} \text{ such that } \mathcal{F} \cup \{ G \} \models H \} \)
for all \( i \geq 1 \),
\( M = \bigcup_{i \geq 0} M_i \).

\( \mathcal{F} \models_{CWA} G \iff G \in M. \)

Theorem 11.1 \( \mathcal{T}_{CWA}(\mathcal{F}) = \{ G \mid \mathcal{F} \models_{CWA} G \} \).

Computing under the closed world assumption:

Extend \( \vdash /2 \) by the rule

\[
\text{if } \nexists A \text{ then conclude } \neg A,
\]

where \( A \) is a ground atom in \( \mathcal{L} \).
Satisfiability

▶ Is $\mathcal{I}_{CWA}(\mathcal{F})$ satisfiable?

▶ Consider $\mathcal{F} = \{leaky-valve \lor punctured-tube\}$

▶ $\mathcal{F} \not\models leaky-valve$
▶ $\mathcal{F} \not\models punctured-tube$
▶ $\{\neg leaky-valve, \neg punctured-tube\} \subseteq \overline{\mathcal{F}}$
▶ $\mathcal{F} \cup \overline{\mathcal{F}} \supseteq \{leaky-valve \lor punctured-tube, \neg leaky-valve, \neg punctured-tube\}$ is unsatisfiable!

▶ **Theorem 11.2** Let $\mathcal{F}$ be a satisfiable set of formulas. $\mathcal{I}_{CWA}(\mathcal{F})$ is satisfiable iff $\mathcal{F}$ admits a least Herbrand model.
Models and the Closed World Assumption

- Let $M = (D, I)$ and $M' = (D', I')$ be two models of $\mathcal{F}$.

- $M$ is a submodel of $M'$ wrt a set $P$ of predicate symbols ($M \preceq_P M'$) iff $D = D'$ and $I$ and $I'$ are identical except that for all $q \in P$ we find $q^I \subseteq q^{I'}$.

- **Notation** If $P = A_R$ then we write $M \preceq M'$ instead of $M \preceq_P M'$.

- A model $M$ of $\mathcal{F}$ is **minimal** iff for all models $M'$ of $\mathcal{F}$ we find that $M' \preceq M$ implies $M = M'$.

- $M \prec M'$ iff $M \preceq M'$ and $M \neq M'$.

- A model $M$ of $\mathcal{F}$ is the **least model** of $\mathcal{F}$ iff for all models $M'$ of $\mathcal{F}$ we find $M \neq M'$ implies $M \prec M'$.

- The closed world assumption eliminates non-least models!
Remarks

- $\mathcal{F} \not\models A$ cannot be decided!

- There are several extensions of the closed world assumption.
Completion

- Can we add more complex formulas than negative ground atoms to a knowledge base?

- $\mathcal{A}_F = \{\text{tweedy, john}\}$, $\mathcal{A}_R = \{\text{penguin}\}$, $\mathcal{F} = \{\text{penguin(tweedy)}\}$.

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

- $M_1 \prec M_2$.

- How can the minimal model be computed?

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

- And another one: $(\forall X) (\neg \text{fly}(X) \rightarrow \text{fly}(X))$. 
Solitary Clauses

- An occurrence of a predicate symbol \( p/n \) 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 \( F \) of clauses is said to be solitary wrt \( p/n \) iff for each clause \( C \in F \) we find that if \( C \) contains a positive occurrence of \( p/n \) then \( C \) does not contain another occurrence of \( p/n \).

- \( \{ \neg \text{fly(tweedy)}, \neg \text{fly(john)}, \text{penguin(tweedy)}, \neg \text{penguin(john)} \} \)
  - solitary in \( \text{fly/1} \).
  - not solitary in \( \text{penguin/1} \).
The Completion Algorithm

Input  A set $\mathcal{F}$ of clauses and a predicate symbol $p/m$.
Output  The completion formula $C_{\mathcal{F},p}$ of $\mathcal{F}$ with respect to $p$.

1 Replace each clause of the form $\{\neg L_1, \ldots, \neg L_n, p(t_1, \ldots, t_m)\}$ occurring in $\mathcal{F}$ by
\[ L_1 \land \ldots \land L_n \rightarrow p(t_1, \ldots, t_m). \] (1)

2 Replace each clause of the form (1) occurring in $\mathcal{F}$ by
\[ (\forall X)[(\exists Y)\ (X_1 \approx t_1 \land \ldots \land X_m \approx t_m \land L_1 \land \ldots \land L_n) \rightarrow p(X)], \] (2)
where $\overline{X} = X_1, \ldots, X_m$ is a sequence of ‘new’ variables and $\overline{Y}$ is a sequence of those variables which occur in (1).

3 Let
\[ \{(\forall X)\ (C_i \rightarrow p(X)) \mid 1 \leq i \leq k\} \]
be the set of clauses having the form (2). Return the completion formula
\[ C_{\mathcal{F},p} = (\forall X)\ (C_1 \lor \ldots \lor C_k \leftarrow p(X)). \]
An Example

- \( \mathcal{F} = \{ \neg \text{penguin}(Y) \lor \text{bird}(Y), \text{bird}(\text{tweedy}), \neg \text{penguin}(\text{john}) \} \)
- \( \mathcal{F}_1 = \{ \text{penguin}(Y) \rightarrow \text{bird}(Y), \text{bird}(\text{tweedy}), \neg \text{penguin}(\text{john}) \} \)
- \( \mathcal{F}_2 = \{ (\forall X)[(\exists Y)(X \approx Y \land \text{penguin}(Y)) \rightarrow \text{bird}(X)], (\forall X)(X \approx \text{tweedy} \rightarrow \text{bird}(X)), \neg \text{penguin}(\text{john}) \} \)
- \( (\forall X)((\exists Y)(X \approx Y \land \text{penguin}(Y)) \lor X \approx \text{tweedy} \rightarrow \text{bird}(X)) \)
- \( \text{C}_{\mathcal{F}},\text{bird} = (\forall X)((\exists Y)(X \approx Y \land \text{penguin}(Y)) \lor X \approx \text{tweedy} \leftarrow \text{bird}(X)) \)
The Equational System $\mathcal{F}_C$

$\mathcal{F}_C = \{ \begin{align*}
& (\forall X, Y) \ f(X) \not\approx g(Y), \\
& \text{for each pair } f/n, g/m \text{ of different function symbols occurring in } \mathcal{A}_F, \\
& (\forall X) \ t[X] \not\approx X, \\
& \text{for each term } t \text{ which is different from } X \text{ but contains an occurrence of } X, \\
& (\forall X, Y) \ (\bigvee_{i=1}^n X_i \not\approx Y_i \rightarrow f(X) \not\approx f(Y)), \\
& \text{for each function symbol } f/n \text{ occurring in } \mathcal{A}_F, \\
& (\forall X) \ X \approx X, \\
& (\forall X, Y) \ (\bigwedge_{x=1}^n X_i \approx Y_i \rightarrow f(X) \approx f(Y)), \\
& \text{for each function symbol } f/n \text{ occurring in } \mathcal{A}_F, \\
& \forall \ (\bigwedge_{x=1}^n X_i \approx Y_i \land p(X) \rightarrow p(Y)), \\
& \text{for each predicate symbol } p/n \text{ occurring in } \mathcal{A}_F
\}
Predicate Completion

- Let $\mathcal{F}$ be a set of formulas, which is solitary in $p$.
- The predicate completion $\mathcal{T}_C(\mathcal{F}, p)$ of $p$ is defined as

$$\mathcal{T}_C(\mathcal{F}, p) = \{G \mid \mathcal{F} \cup \mathcal{C}_{\mathcal{F}, p} \cup \mathcal{F}_C \models G\}.$$ 

- **Theorem 11.3** Let $\mathcal{F}$ be a set of formulas which is solitary in $p$. If $\mathcal{F}$ is satisfiable, then so is $\mathcal{T}_C(\mathcal{F}, p)$. 

Parallel Completion and Logic Programming

- \( F = \{ \text{bird(tweedy)}, (\forall X) (\text{bird}(X) \land \neg \text{abnormal}(X) \rightarrow \text{fly}(X)) \} \)
- 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.
- \( p/n \) is defined in the logic program \( F \) iff \( F \) contains a clause with \( p/n \) occurring in its head.
- \( A_D \) is the set of defined predicate symbols.
- The completion \( T_C(F) \) of a normal logic program \( F \) with defined predicate symbols \( A_D \) is defined as

\[
T_C(F) = \{ G \mid F \cup \{ C_{F,p} \mid p \in A_D \} \cup F_C \cup \{ (\forall X) \neg p(X) \mid p \in A_P \setminus A_D \} \models G \}.
\]
Stratified Logic Programs

- Let $\mathcal{A}$ be an alphabet.
- A level mapping is a total mapping $l$ from $\mathcal{A}_P$ to $\mathbb{N}$.
- $l(p)$ is the level of $p$.
- A normal logic program $F$ is stratified iff in each clause of the form

  $$
p(t) \leftarrow p_1(s_1) \land \ldots \land p_m(s_m) \land \neg p_{m+1}(s_{m+1}) \land \ldots \land \neg p_n(s_n)
  $$

  of $F$ we find $l(p) \geq l(p_i), 1 \leq i \leq m$, and $l(p) > l(p_j), m < j \leq n$.

- Theorem 11.4 Let $F$ be a stratified normal logic programs. Then $\mathcal{T}_C(F)$ is satisfiable.
Negation as Failure

- We do not want to compute with the “only-if” parts and $\mathcal{F}_C$!
- $\{\neg A \mid \neg A \in \mathcal{T}(\mathcal{F})\} \neq \{\neg A \mid \neg A \in \mathcal{T}_C(\mathcal{F})\}$
- Replace $\neg / 1$ by not / 1 called negation as failure.
- $\mathcal{F} = \{\text{bird(tweedy)}, \text{fly}(X) \leftarrow \text{bird}(X) \land \text{not abnormal}(X)\}$
Finitely Failed Search Trees

- A search tree is **finitely failed** iff it is finite and each leaf is labelled as a failure.
- \[ \mathcal{F}' = \{ \text{abnormal}(X) \leftarrow \text{broken-wing}(X), \]
  \[ \text{abnormal}(X) \leftarrow \text{ratite}(X), \]
  \[ \text{ratite}(X) \leftarrow \text{ostrich}(X), \]
  \[ \text{ratite}(X) \leftarrow \text{emu}(X), \]
  \[ \text{ratite}(X) \leftarrow \text{kiwi}(X) \} \]

- \[ \leftarrow \text{abnormal(tweedy)} \]
  \[ \leftarrow \text{broken-wing(tweedy)} \quad \leftarrow \text{ratite(tweedy)} \]
  \[ \leftarrow \text{kiwi(tweedy)} \quad \leftarrow \text{emu(tweedy)} \quad \leftarrow \text{ostrich(tweedy)} \]
SLDNF-Resolution

Let $G$ be a goal clause consisting of positive and negative literals, $F$ 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 $F$ is also an SLDNF-resolvent.
- If $L$ is a ground negative literal, i.e. $L = \text{not } A$, and the query $\leftarrow A$ finitely fails with respect to $F$ 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. $L = \text{not } A$, and the query $\leftarrow A$ succeeds with respect to $F$ 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 11.5 Let $F$ be a logic program. SLDNF-resolution is sound with respect to the completion of $F$. 
Classical Negation vs. Negation as Failure

- cross ← not train
- cross ← ¬train
Circumscription

- All approaches mentioned so far cannot handle $p(a) \lor p(b)$ or $(\exists X) \text{green}(X)$.
- How can we compute their minimal models?
- We want to conjecture that the tuples $(X_1, \ldots, X_m)$ which can be shown to satisfy a relation $p/m$ are all the tuples satisfying $p/m$.
- In other words, we want to circumscribe $p/m$.
- $F\{p/\Phi\}$ is the string obtained from $F$ by replacing each occurrence of $p/m$ by $\Phi/m$.
- $\Phi$ is a predicate variable.
- The circumscription of $p$ in $F$:

$$
\text{Circ}(F, p) = (F\{p/\Phi\} \land (\forall X) (\Phi(X) \rightarrow p(X))) \rightarrow (\forall X)(p(X) \rightarrow \Phi(X))
$$

- $\text{Circ}(F, p)$ is a schema.
Example 1

Let $F = \text{isblock}(a) \land \text{isblock}(b) \land \text{isblock}(c)$.

Then $\text{Circ}(F, p) = (\Phi(a) \land \Phi(b) \land \Phi(c) \land (\forall X) (\Phi(X) \rightarrow \text{isblock}(X))) \rightarrow (\forall X) (\text{isblock}(X) \rightarrow \Phi(X))$.

This schema can be instantiated by

$$\Phi(X) \leftrightarrow (X \approx a \lor X \approx b \lor X \approx c).$$

Let $F'$ be the corresponding instance.

We obtain

$$\{F, F'\} \models (\forall X) (\text{isblock}(X) \rightarrow (X \approx a \lor X \approx b \lor X \approx c)).$$

Circumscription is non-monotonic.
Example 2

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

Then $\text{Circ}(F, p) = ((\Phi(a) \lor \Phi(b)) \land (\forall X) (\Phi(X) \rightarrow p(X)))$
$
\rightarrow (\forall X) (p(X) \rightarrow \Phi(X)).$

This schema can be instantiated by

$$\Phi(X) \leftrightarrow X \approx a.$$ 

Let $F_1$ be the corresponding instance.

This schema can also be instantiated by

$$\Phi(X) \leftrightarrow X \approx b.$$ 

Let $F_2$ be the corresponding instance.

We obtain

$$\{F, F_1, F_2\} \models (\forall X) (p(X) \rightarrow X \approx a) \lor (\forall X) (p(X) \rightarrow X \approx b).$$
Results

▶ $G$ follows minimally from $F$ with respect to $p/m$, written $F \models_{\{p\}} G$, iff $G$ holds in all models of $F$ which are minimal in $\{p/m\}$.

▶ Theorem 11.6 Let $F'$ be an instance of $Circ(F, p)$. $F'$ holds in all models of $F$ which are minimal in $\{p/m\}$.

▶ Corollary 11.7 Let $F'$ be an instance of $Circ(F, p)$. If $\{F, F'\} \models G$ then $F \models_{\{p\}} G$.

▶ Remarks

▷ Computing with circumscription is non-monotonic.
▷ 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:

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

- 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 . . .
  - **CWA** Exceptions are not logical consequences.
  - **NAF** We finitely failed to prove exceptions.
  - **Default Logic** It is consistent to assume that . . .

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

$$F : G_1, \ldots, G_n \quad \frac{}{H}.$$

- $F$ is called prerequisite,
- $G_1, \ldots, G_n$ are called justifications,
- $H$ 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.
Default Rules – Special Cases

▶ If $F$ is missing, then $F \equiv \langle \rangle$.
▶ If $n = 0$, then this is a rule in $\langle A, L, \models \rangle$.
▶ 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 \( \langle F_D, F_W \rangle \), where
  - \( F_D \) is a set of at most countably many default rules and
  - \( F_W \) 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.

- \( F_D : \) 
  \[
  \frac{\text{spouse}(X,Y) \land \text{htown}(Y) \approx Z : \text{htown}(X) \approx Z}{\text{htown}(X) \approx Z},
  \frac{\text{employer}(X,Y) \land \text{location}(Y) \approx Z : \text{htown}(X) \approx Z}{\text{htown}(X) \approx Z}
  \]

- \( F_W : \) 
  \[
  \text{spouse}(\text{jane}, \text{john}),
  \text{htown}(\text{john}) \approx \text{munich},
  \text{employer}(\text{jane}, \text{tud}),
  \text{location}(\text{tud}) \approx \text{dresden},
  (\forall X, Y, Z) (\text{htown}(X) \approx Y \land \text{htown}(X) \approx Z \rightarrow Y \approx Z)
  \]
Extensions

- An extension $\mathcal{F}_E$ of $\mathcal{F}$ should have the properties:
  - $\mathcal{F} \subseteq \mathcal{F}_E$,
  - $\mathcal{I}(\mathcal{F}_E) = \mathcal{F}_E$,
  - $\mathcal{F}_E$ should be closed under the application of default rules.

- Let $\Gamma(\mathcal{F})$ be the smallest set satisfying the following properties:
  1. $\mathcal{F}_W \subseteq \Gamma(\mathcal{F})$.
  2. $\mathcal{I}(\Gamma(\mathcal{F})) = \Gamma(\mathcal{F})$.
  3. If $F : G_1, \ldots, G_n / H \in \mathcal{F}_D$, $F \in \Gamma(\mathcal{F})$ and for all $1 \leq j \leq n$ we find that $\neg G_i \notin \mathcal{F}$ then $H \in \Gamma(\mathcal{F})$.

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

- The set of extensions of $\langle \mathcal{F}_D, \mathcal{F}_W \rangle$ is a subset of the set of models for $\mathcal{F}_W$. 

Another Characterization of Extensions

Theorem 11.7  Let $\langle \mathcal{F}_D, \mathcal{F}_W \rangle$ be a default knowledge base and $\mathcal{F}$ be a set of sentences. Define

$$\mathcal{F}_0 = \mathcal{F}_W$$

and for $i \geq 1$:

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

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

- We have to guess extensions!

- $\mathcal{F}_D = \{ \text{bird}(X) : \text{fly}(X) / \text{fly}(X) \}$
- $\mathcal{F}_W = \{ \text{bird(tweedy)} \}$
- $\mathcal{F} = \mathcal{T}(\{\text{bird(tweedy)}, \text{fly(tweedy)}\})$ is an extension.
Another Example

\[ \mathcal{F}_D : \]
\[
\begin{align*}
spouse(X,Y) & \land htown(Y) \approx Z : htown(X) \approx Z, \\
htown(X) & \approx Z \\
employer(X,Y) & \land location(Y) \approx Z : htown(X) \approx Z \\
htown(X) & \approx Z
\end{align*}
\]

\[ \mathcal{F}_W : \]
\[
\text{spouse}(\text{jane}, \text{john}), \\
htown(\text{john}) \approx \text{munich}, \\
employer(\text{jane}, \text{tud}), \\
location(\text{tud}) \approx \text{dresden}, \\
(\forall X, Y, Z) (htown(X) \approx Y \land htown(X) \approx Z \rightarrow Y \approx Z)
\]

\[ \text{Its extensions are:} \]
\[
\mathcal{T}(\{ \text{spouse}(\text{jane}, \text{john}), \ htown(\text{john}) \approx \text{munich}, \ employer(\text{jane}, \text{tud}), \\
location(\text{tud}) \approx \text{dresden}, \ htown(\text{jane}) \approx \text{munich}\})
\]

and
\[
\mathcal{T}(\{ \text{spouse}(\text{jane}, \text{john}), \ htown(\text{john}) \approx \text{munich}, \ employer(\text{jane}, \text{tud}), \\
location(\text{tud}) \approx \text{dresden}, \ htown(\text{jane}) \approx \text{dresden}\})
\]
Credolous vs. Sceptical Reasoning

- **G follows credolously** from $\langle F_D, F_W \rangle$ (in symbols $\langle F_D, F_W \rangle \models_c G$) iff there exists an extension $F$ of $\langle F_D, F_W \rangle$ such that $G \in F$.

- **G follows sceptically** from $\langle F_D, F_W \rangle$ (in symbols $\langle F_D, F_W \rangle \models_s G$) iff for all extensions $F$ of $\langle F_D, F_W \rangle$ we find $G \in F$. 
Remarks

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

\[
\mathcal{F}_W = \{ \text{broken(left-arm)} \lor \text{broken(right-arm)} \}
\]
\[
\mathcal{F}_D = \{ : \text{usable}(X) \land \neg \text{broken}(X) / \text{usable}(X) \}
\]

- 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.
- No student with a GPA under 3.6 is eligible.
- The students whose eligibility is not determined by these rules are inter-viewed by the scholarship committee.

\[ F_1 = \{\text{eligible}(X) \leftarrow \text{highGPA}(X),\]
\[ \text{eligible}(X) \leftarrow \text{minority}(X) \land \text{fairGPA}(X),\]
\[ \neg\text{eligible}(X) \leftarrow \neg\text{fairGPA}(X),\]
\[ \text{interview}(X) \leftarrow \neg\text{eligible}(X) \land \neg\neg\text{eligible}(X) \} \]

\[ F_2 = \{\text{fairGPA}(john) \leftarrow,\]
\[ \neg\text{highGPA}(john) \leftarrow \}. \]

What happens with John?
Rules and Programs

▶ Rules

\[ L_1 \lor \ldots \lor L_k \lor \text{not } L_{k+1} \lor \ldots \lor \text{not } L_l \leftarrow L_{l+1} \land \ldots \land L_m \land \text{not } L_{m+1} \land \ldots \land \text{not } L_n \]

▷ \( L_i \) 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.
Answer Sets

- Remember rules:

\[ L_1 \lor \ldots \lor L_k \lor \text{not } L_{k+1} \lor \ldots \lor \text{not } L_l \leftarrow L_{l+1} \land \ldots \land L_m \land \text{not } L_{m+1} \land \ldots \land \text{not } L_n. \]

- Let \( M \) a satisfiable set of literals and \( F \) be a program where \( k = l \) and \( n = m \), i.e., rules are of the form

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

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

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

- Example \( F_3 = \{s \lor r \leftarrow , \neg b \leftarrow r\}. \)

\( \{s\} \) and \( \{r, \neg b\} \) are answer sets.

- What happens if we add the constraint \( s \)?
Reducts and Answer Sets

- Let \( F \) be a program and \( M \) a satisfiable set of literals.
- The reduct \( F^M \) of \( F \) relative to \( M \) is the set of rules

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

such that

\[
L_1 \lor \ldots \lor L_k \lor \text{not } L_{k+1} \lor \ldots \lor \text{not } L_l \leftarrow L_{l+1} \land \ldots \land L_m \land \text{not } L_{m+1} \land \ldots \land \text{not } L_n
\]

occurs in \( F \), \( \{ L_{k+1}, \ldots, L_l \} \subseteq M \) and \( \{ L_{m+1}, \ldots, L_n \} \cap M = \emptyset \).

- \( F^M \) is a program without \textit{not}/1.
- \( M \) is said to be an \textit{answer set} for \( F \) iff \( M \) is an answer set for \( F^M \).
- Examples

\[
\{ p \leftarrow \text{not } q \}, \quad \{ \neg p \leftarrow \text{not } p \}, \quad \{ p \leftarrow \text{not } \neg p \}
\]
Predicate Symbols, Constants and Variables

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

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

\[ \mathcal{F}_2 = \{ \text{fairGPA}(\text{john}) \leftarrow, \]
\[ \lnot \text{highGPA}(\text{john}) \leftarrow \} \]

- Its only answer set is:

\{ \text{fairGPA}(\text{john}), \lnot \text{highGPA}(\text{john}), \text{interview}(\text{john}) \}.

- What happens if we add \lnot \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 $0, \ldots, n$
- Let $\mathcal{A}$ be an alphabet with
  - constants $0, \ldots, n$ and
  - predicate symbols $\text{reachable}/1$ and $\text{in}/2$.

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

Program

- \{ \text{in}(u, v) \lor \neg\text{in}(u, v) \leftarrow | \langle u, v \rangle \in G \}\}
- \{ \leftarrow \text{in}(u, v) \land \text{in}(u, w) | \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) | \langle v, u \rangle, \langle w, u \rangle \in G \text{ and } v \not\approx w \}\}
- \{ \text{reachable}(u) \leftarrow \text{in}(0, u) | \langle 0, u \rangle \in G \}\}
- \{ \text{reachable}(v) \leftarrow \text{reachable}(u) \land \text{in}(u, v) | \langle u, v \rangle \in G \}\}
- \{ \leftarrow \text{not reachable}(u) | 0 \leq u \leq n \}\}
Answer Set Planning

Program:

\[\begin{align*}
&\text{on}(B, L, 0) \lor \neg \text{on}(B, L, 0) \leftarrow \\
&\text{move}(B, L, T) \lor \neg \text{move}(B, L, T) \leftarrow \\
&\text{on}(B, L, T + 1) \leftarrow \text{move}(B, L, T) \\
&\text{on}(B, L, T + 1) \leftarrow \text{on}(B, L, T) \land \text{not} \neg \text{on}(B, L, T + 1) \\
&\neg \text{on}(B, L, T) \leftarrow \text{on}(B, L', T) \ (L \neq L') \\
&\leftarrow \text{move}(B, L, T) \land \text{on}(B', B, T) \ (B \neq B') \\
&\leftarrow \text{move}(B, B', T) \land \text{move}(B', L, T) \ (B \neq B') \\
&\leftarrow \text{move}(B, L, T) \land \text{move}(B', L', T) \land \text{move}(B'', L'', T) \\
&\quad \langle B, L \rangle \neq \langle B', L' \rangle, \langle B, L \rangle \neq \langle B'', L'' \rangle, \langle B', L' \rangle \neq \langle B'', L'' \rangle \\
&\leftarrow \text{on}(B, B'', T) \land \text{on}(B', B'', T) \ (B \neq B') \\
&\text{supported}(B, T) \leftarrow \text{on}(B, \text{table}, T) \\
&\text{supported}(B, T) \leftarrow \text{on}(B, B', T) \land \text{supported}(B', T) \\
&\leftarrow \text{not supported}(B, T)
\end{align*}\]
Computing Answer Sets

- Smodels
- Dlv
- DeReS