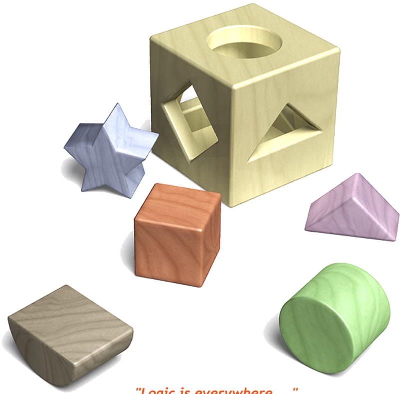
Non-Monotonic Reasoning

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

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

  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 $\rightsquigarrow$ 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**
  - $$\begin{array}{rcl}\mathcal{K} & = & \{\mathit{lectures}(\mathit{steffen}, \mathit{cl1}),\ \mathit{lectures}(\mathit{steffen}, \mathit{cl5}),\\ & & \ \mathit{lectures}(\mathit{michael}, \mathit{cl2}),\ \mathit{lectures}(\mathit{michael}, \mathit{cl5}),\\ & & \ \mathit{lectures}(\mathit{heiko}, \mathit{cl4}),\ \mathit{lectures}(\mathit{horst}, \mathit{cl3})\}\end{array}$$
  -

query	OWS	CWS
$\mathcal{K} \models (\exists X)\ \mathit{lectures}(\mathit{steffen}, X)$	yes	yes
$\mathcal{K} \models \neg \mathit{lectures}(\mathit{michael}, \mathit{cl6})$	no	yes

The Formal Theory

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

Satisfiability

- Is $\mathcal{C}_{CWA}(\mathcal{K})$ satisfiable?
- Consider $\mathcal{K} = \{leakyValve \vee puncturedTube\}$
  - $\mathcal{K} \not\models leakyValve$
  - $\mathcal{K} \not\models puncturedTube$
  - $\{\neg leakyValve, \neg puncturedTube\} \subseteq \mathcal{K}_{CWA}$
  - $\mathcal{K} \cup \mathcal{K}_{CWA} \supseteq \{leakyValve \vee puncturedTube, \ \neg leakyValve, \neg puncturedTube\}$
  - $\mathcal{K} \cup \mathcal{K}_{CWA}$ is unsatisfiable!
- **Theorem** Let $\mathcal{K}$ be a satisfiable set of formulas in Skolem normal form. $\mathcal{C}_{CWA(\mathcal{K})}$ 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 $q^I \subseteq q^{I'}$
  - (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

Completion

- **Can we add more complex formulas than negative ground atoms to a knowledge base?**
- Let $\mathcal{F} = \{tweedy, john\}$ and $\mathcal{R} = \{penguin\}$
- Let $\mathcal{K} = \{penguin(tweedy)\}$
- Models for $\mathcal{K}$
  - $M_1 = \{penguin(tweedy)\}$
  - $M_2 = \{penguin(tweedy), penguin(john)\}$
- $M_1 \prec M_2$
- **How can the least model be computed?**
- Another example $\mathcal{K} = \{penguin(tweedy), penguin(john)\}$
- And another one $\mathcal{K} = \{(\forall X)\,(\neg fly(X) \rightarrow fly(X))\}$

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

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

  - 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

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

where
$\overline{X} = X_1, \ldots, X_m$ is a sequence of 'new' variables
$\overline{Y}$ is a sequence of those variables which occur in the clause and
all occurrences of double negation are removed

2 Let

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

be the set of all formulas where p occurs in the conclusion.
Return the completion formula

$$C_{\mathcal{K},p} = (\forall \overline{X})\,(C_1 \vee \ldots \vee C_k \leftarrow p(\overline{X}))$$

An Example

- $\mathcal{K} = \{\ \neg penguin(Y) \vee bird(Y),\ bird(tweedy),\ \neg penguin(john)\ \}$
- $\mathcal{K}_1 = \{\ (\forall X)((\exists Y)\,(X \approx Y \wedge penguin(Y)) \rightarrow bird(X)),\ (\forall X)\,(X \approx tweedy \rightarrow bird(X)),\ \neg penguin(john)\ \}$
- $(\forall X)\,((\exists Y)\,(X \approx Y \wedge penguin(Y) \vee X \approx tweedy) \rightarrow bird(X))$
- $C_{\mathcal{K},bird} \quad = \quad (\forall X)\,((\exists Y)(X \approx Y \wedge penguin(Y) \vee X \approx tweedy) \leftarrow bird(X))$

The Equational System $\mathcal{E}_C$

- Let

$$
\begin{array}{rcl}
\mathcal{E}_C & = & \{(\forall \overline{X}, \overline{Y})\, f(\overline{X}) \not\approx g(\overline{Y}) \mid f,\ g \in \mathcal{F} \text{ and } f \neq g\} \\
& \cup & \{(\forall X)\, t\lceil X \rceil \not\approx X \mid t \neq X\} \\
& \cup & \{(\forall \overline{X}, \overline{Y})\, (\bigvee_{i=1}^{n} X_i \not\approx Y_i \rightarrow f(\overline{X}) \not\approx f(\overline{Y})) \mid n\text{-ary } f \in \mathcal{F}\} \\
& \cup & \{(\forall X)\, X \approx X\} \\
& \cup & \{(\forall \overline{X}, \overline{Y})\, (\bigwedge_{i=1}^{n} X_i \approx Y_i \rightarrow f(\overline{X}) \approx f(\overline{Y})) \mid n\text{-ary } f \in \mathcal{F}\} \\
& \cup & \{\forall\, (\bigwedge_{i=1}^{n} X_i \approx Y_i \wedge p(\overline{X}) \rightarrow p(\overline{Y})) \mid n\text{-ary } p \in \mathcal{R}\}
\end{array}
$$

Predicate Completion

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

$$\mathcal{C}_C(\mathcal{K}, p) = \{G \mid \mathcal{K} \cup \{C_{\mathcal{K},p}\} \cup \mathcal{E}_C \models G\}$$

- Theorem Let $\mathcal{K}$ be a set of clauses which is solitary in p.
  If $\mathcal{K}$ is satisfiable, then so is $\mathcal{C}_C(\mathcal{K}, p)$.

Parallel Completion and Logic Programming

- $\mathcal{K} = \{bird(tweedy),\ (\forall X)\,(\neg bird(X) \vee ab(X) \vee fly(X))\}$
- Normal program clauses $\quad p(\overline{t}) \leftarrow A_1 \wedge \ldots \wedge A_m \wedge \neg A_{m+1} \wedge \ldots \wedge \neg A_n$
- A normal logic program is a set of normal program clauses
- p is defined in the logic program $\mathcal{K}$
  iff $\quad\mathcal{K}$ contains a clause with p occurring in its head
- Let $\mathcal{R}_D$ be the set of defined predicate symbols
- The completion $\mathcal{C}_C(\mathcal{K})$ of a normal logic program $\mathcal{K}$
  with defined predicate symbols $\mathcal{R}_D$ is defined as

$$\mathcal{C}_C(\mathcal{K}) =$$
$$\{G \mid \mathcal{K} \cup \{C_{\mathcal{K},p} \mid p \in \mathcal{R}_D\} \cup \mathcal{E}_C \cup \{(\forall \overline{X})\, \neg p(\overline{X}) \mid p \in \mathcal{R} \setminus \mathcal{R}_D\} \models G\}$$

Stratified Logic Programs

- Consider $\mathcal{R}$, $\mathcal{F}$ and $\mathcal{V}$
- A level mapping is a total mapping $\mathit{level} : \mathcal{R} \to \mathbb{N}$
- $\mathit{level}(p)$ is the level of p
- A normal logic program $\mathcal{K}$ is stratified iff in each clause of the form

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

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

- **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

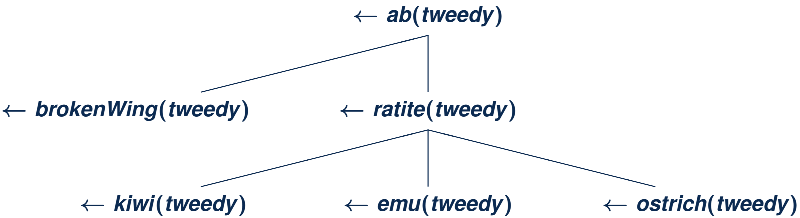
Negation as Failure

- **We do not want to compute with the "only-if" parts and $\mathcal{E}_C$!**
- $\{\neg A \mid \neg A \in \mathcal{C}(\mathcal{K})\} \neq \{\neg A \mid \neg A \in \mathcal{C}_C(\mathcal{K})\}$
- **Replace $\neg$ by $\sim$ called negation as failure**
- $\mathcal{K} = \{bird(tweedy),\ fly(X) \leftarrow bird(X) \wedge \sim ab(X)\}$

Finitely Failed Search Trees

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

- $$\mathcal{K}' = \{ \begin{array}{l} ab(X) \leftarrow brokenWing(X), \\ ab(X) \leftarrow ratite(X), \\ ratite(X) \leftarrow ostrich(X), \\ ratite(X) \leftarrow emu(X), \\ ratite(X) \leftarrow kiwi(X) \end{array} \}$$



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 $\leftarrow A$ suceeds 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

- $cross \leftarrow \sim train$
- $cross \leftarrow \neg train$

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

$$Circ(G, p) = (G\{p \mapsto Q\} \wedge (\forall \overline{X})\,(Q(\overline{X}) \rightarrow p(\overline{X}))) \rightarrow (\forall \overline{X})(p(\overline{X}) \rightarrow Q(\overline{X}))$$

- Note $Circ(G, p)$ is a schema

Example 1

- Let $G = \mathit{isblock}(a) \wedge \mathit{isblock}(b) \wedge \mathit{isblock}(c)$
- Then

$$\begin{array}{rcl} \mathit{Circ}(G, \mathit{isblock}) & = & ((Q(a) \wedge Q(b) \wedge Q(c)) \wedge (\forall X)\,(Q(X) \rightarrow \mathit{isblock}(X))) \\ & & \rightarrow\ (\forall X)\,(\mathit{isblock}(X) \rightarrow Q(X)) \end{array}$$

- This schema can be instantiated replacing $Q(X)$ by $(X \approx a \vee X \approx b \vee X \approx c)$, where $\approx$ denotes syntactic equality
- Let G' be the corresponding instance of $\mathit{Circ}(G, \mathit{isblock})$
- Note

$$\begin{array}{rclcl} Q(a) & = & a \approx a \vee a \approx b \vee a \approx c & \equiv & \langle\rangle \\ Q(b) & = & b \approx a \vee b \approx b \vee b \approx c & \equiv & \langle\rangle \\ Q(c) & = & c \approx a \vee c \approx b \vee c \approx c & \equiv & \langle\rangle \end{array}$$

Example 1 Continued

- Remember

$$\begin{array}{lcl} G & = & \mathit{isblock}(a) \wedge \mathit{isblock}(b) \wedge \mathit{isblock}(c) \\ G' & \equiv & (\forall X)\,(X \approx a \vee X \approx b \vee X \approx c \rightarrow \mathit{isblock}(X)) \\ & & \rightarrow\ (\forall X)\,(\mathit{isblock}(X) \rightarrow X \approx a \vee X \approx b \vee X \approx c) \end{array}$$

- Let I be a Herbrand-interpretation (which interprets $\approx$ as syntactic equality) such that $I \models \{G, G'\}$.
  - $[(\forall X)\,(X \approx a \vee X \approx b \vee X \approx c \rightarrow \mathit{isblock}(X)]^I = \top$
    iff for all $t \in \mathcal{T}(\mathcal{F})$ we find $[t \approx a \vee t \approx b \vee t \approx c \rightarrow \mathit{isblock}(t)]^I = \top \ (*)$
  - case $t \in \{a, b, c\}$
    $[t \approx a \vee t \approx b \vee t \approx c]^I = \top$ and $[\mathit{isblock}(t)]^I = \top$, thus, $(*)$ holds
  - case $t \not\in \{a, b, c\}$
    $[t \approx a \vee t \approx b \vee t \approx c]^I = \bot$, thus, $(*)$ holds
- Therefore, we obtain

$$\{G,\ G'\} \models (\forall X)\,(\mathit{isblock}(X) \rightarrow (X \approx a \vee X \approx b \vee X \approx c))$$

Non-monotonicity

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

Example 2

- Let $G = p(a) \vee p(b)$
- Then

$$\begin{aligned} Circ(G, p) &= ((Q(a) \vee Q(b)) \wedge (\forall X)\,(Q(X) \rightarrow p(X))) \\ &\quad \rightarrow (\forall X)\,(p(X) \rightarrow Q(X)) \end{aligned}$$

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

$$\begin{aligned} G_1 &= ((a \approx a \vee b \approx a) \wedge (\forall X)\,(X \approx a \rightarrow p(X))) \\ &\quad \rightarrow (\forall X)\,(p(X) \rightarrow X \approx a) \\ &\equiv (\forall X)\,(X \approx a \rightarrow p(X)) \rightarrow (\forall X)\,(p(X) \rightarrow X \approx a) \\ &\equiv p(a) \rightarrow (\forall X)\,(p(X) \rightarrow X \approx a) \end{aligned}$$

Example 2 Continued

- **Remember**

$$G = p(a) \vee p(b)$$

and

$$G_1 \equiv p(a) \rightarrow (\forall X)\,(p(X) \rightarrow X \approx a)$$

- $Circ(G, p)$ **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**

$$\{H_1 \vee H_2, H_1 \rightarrow H_1', H_2 \rightarrow H_2'\} \models H_1' \vee H_2'$$

- **Hence**

$$\{G,\ G_1,\ G_2\} \models (\forall X)\,(p(X) \rightarrow X \approx a) \vee (\forall X)\,(p(X) \rightarrow X \approx b)$$

The Main Result

- **Theorem** Let G' be an instance of $Circ(G, p)$.
  G' holds in all models of G which are minimal in $\{p\}$
- **Proof**
  Consider $(G\{p \mapsto Q\} \wedge (\forall \overline{X})\,(Q(\overline{X}) \to p(\overline{X}))) \to (\forall \overline{X})(p(\overline{X}) \to Q(\overline{X}))$
  - Let I be a model for G which is minimal in $\{p\}$
  - Let p' be a predicate symbol such that
    $[G\{p \mapsto p'\} \wedge (\forall \overline{X})\,(p'(\overline{X}) \to p(\overline{X}))]^I = \top$
  - To show $[(\forall \overline{X})(p(\overline{X}) \to p'(\overline{X}))]^I = \top$
    - Suppose $[(\forall \overline{X})(p(\overline{X}) \to p'(\overline{X}))]^I = \bot$
    - Hence $p^I \not\subseteq p'^I$
    - Because $[(\forall \overline{X})\,(p'(\overline{X}) \to p(\overline{X}))]^I = \top$ we find $p'^I \subseteq p^I$
    - We conclude $p'^I \subset p^I$
    - 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 $p^{I'} = p'^I$
    - Because $p^{I'} = p'^I \subset p^I$ 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 $Circ(G, p)$.
  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:**

$$(\forall X)\,(bird(X) \wedge \neg penguin(X) \wedge \neg ostrich(X) \wedge \ldots \rightarrow fly(X))$$

- **Problems**
  - **We do not know all exceptions**
  - **We cannot conlude 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 $bird(X) \; : \; fly(X) \; / \; fly(X)$
- Exceptions $\{ \; (\forall X)\,(penguin(X) \rightarrow \neg fly(X)),\; (\forall X)\,(ostrich(X) \rightarrow \neg fly(X)),\; \ldots \; \}$
- But how is consistency defined?
- Few objects of sort s have property p:

$$man(X) \; : \; \neg moon(X) \; / \; \neg moon(X)$$

Default Rules

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

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

  - G 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 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 $G_1 = H \wedge H'$, 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**

$$\begin{array}{ll}\mathcal{K}_D: & spouse(X, Y) \wedge htown(Y) \approx Z \ : \ htown(X) \approx Z \ / \ htown(X) \approx Z, \\ & employer(X, Y) \wedge location(Y) \approx Z \ : \ htown(X) \approx Z \ / \ htown(X) \approx Z \\ \mathcal{K}_W: & spouse(jane, john), \\ & htown(john) \approx munich, \\ & employer(jane, tud), \\ & location(tud) \approx dresden, \\ & (\forall X, Y, Z)\,(htown(X) \approx Y \wedge htown(X) \approx Z \rightarrow Y \approx Z)\end{array}$$

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 $i \geq 1$

$$\begin{array}{rl} \mathcal{K}_{i+1} = \mathcal{C}(\mathcal{K}_i) \cup \{H \mid & G : G_1, \ldots, G_n \;/\; H \in \mathcal{K}_D, \\ & G \in \mathcal{K}_i \text{ and} \\ & \neg G_j \not\in \mathcal{K} \text{ for all } 1 \leq j \leq n\} \end{array}$$

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!

- $$\begin{array}{lcll} \mathcal{K}_D & = & \{ & bird(X) : fly(X) \;/\; fly(X) \quad \} \\ \mathcal{K}_W & = & \{ & bird(tweedy) \quad \} \end{array}$$

- $\mathcal{K} = \mathcal{C}(\{bird(tweedy),\; fly(tweedy)\})$ is an extension

Another Example

- $\mathcal{K}_D$: $spouse(X, Y) \wedge htown(Y) \approx Z \ : \ htown(X) \approx Z \ / \ htown(X) \approx Z,$
  $employer(X, Y) \wedge location(Y) \approx Z \ : \ htown(X) \approx Z \ / \ htown(X) \approx Z$
  $\mathcal{K}_W$: $spouse(jane, john),$
  $htown(john) \approx munich,$
  $employer(jane, tud),$
  $location(tud) \approx dresden,$
  $(\forall X, Y, Z)\ (htown(X) \approx Y \wedge htown(X) \approx Z \rightarrow Y \approx Z)$
- **Its extensions are**

$$\mathcal{C}(\mathcal{K}_W \cup \{htown(jane) \approx munich\})$$

and

$$\mathcal{C}(\mathcal{K}_W \cup \{htown(jane) \approx dresden\})$$

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

$$\begin{array}{lcl} \mathcal{K}_W & = & \{broken(left - arm) \vee broken(right - arm)\} \\ \mathcal{K}_D & = & \{ : usable(X) \wedge \neg broken(X) \ / \ usable(X)\} \end{array}$$

- 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

- $$\mathcal{K}_1 = \left\{ \begin{array}{lcl} \mathit{eligible}(X) & \leftarrow & \mathit{highGPA}(X), \\ \mathit{eligible}(X) & \leftarrow & \mathit{minority}(X) \wedge \mathit{fairGPA}(X), \\ \neg\mathit{eligible}(X) & \leftarrow & \neg\mathit{highGPA}(X) \wedge \neg\mathit{minority}(X), \\ \mathit{interview}(X) & \leftarrow & \sim \mathit{eligible}(X) \wedge \sim \neg\mathit{eligible}(X) \end{array} \right\}$$

- $$\mathcal{K}_2 = \left\{ \begin{array}{l} \mathit{fairGPA}(\mathit{john}) \leftarrow, \\ \neg\mathit{highGPA}(\mathit{john}) \leftarrow \end{array} \right\}$$

- What happens with John?

Rules and Programs

- Rules

  $$L_1 \vee \ldots \vee L_k \vee \sim L_{k+1} \vee \ldots \vee \sim L_l \leftarrow L_{l+1} \wedge \ldots \wedge L_m \wedge \sim L_{m+1} \wedge \ldots \wedge \sim L_n$$

  - 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**

$$L_1 \vee \ldots \vee L_k \vee \sim L_{k+1} \vee \ldots \vee \sim L_l \leftarrow L_{l+1} \wedge \ldots \wedge L_m \wedge \sim L_{m+1} \wedge \ldots \wedge \sim L_n$$

- **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**

$$L_1 \vee \ldots \vee L_k \leftarrow L_{l+1} \wedge \ldots \wedge L_m$$

  - $\mathcal{M}$ **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** $\mathcal{K}_3 = \{ \; s \vee r \leftarrow \; , \; \neg b \leftarrow r \; \}$

  - **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

$$L_1 \vee \ldots \vee L_k \leftarrow L_{l+1} \wedge \ldots \wedge L_m$$

such that

$$L_1 \vee \ldots \vee L_k \vee \sim L_{k+1} \vee \ldots \vee \sim L_l \leftarrow L_{l+1} \wedge \ldots \wedge L_m \wedge \sim L_{m+1} \wedge \ldots \wedge \sim L_n$$

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** $\{p \leftarrow \sim q\}$

  $\{\neg p \leftarrow \sim p\}$

  $\{p \leftarrow \sim \neg p\}$

  $\{q \leftarrow p \wedge \sim q,\ p \leftarrow,\ q \leftarrow\}$

  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
- $$\mathcal{K}_1 = \{ \begin{array}{lcl} eligible(X) & \leftarrow & highGPA(X), \\ eligible(X) & \leftarrow & minority(X) \wedge fairGPA(X), \\ \neg eligible(X) & \leftarrow & \neg highGPA(X) \wedge \neg minority(X), \\ interview(X) & \leftarrow & \sim eligible(X) \wedge \sim \neg eligible(X) \end{array} \}$$
- $$\mathcal{K}_2 = \{ \begin{array}{l} fairGPA(john) \leftarrow, \\ \neg highGPA(john) \leftarrow \end{array} \}$$
- Its only answer set is:

$$\{fairGPA(john),\ \neg highGPA(john),\ interview(john)\}$$

- What happens if we add $\neg minority(john) \leftarrow$?
- 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
  - $\mathcal{R} = \{\mathit{reachable}, \mathit{in}\}$
- Idea
  - WLOG let 0 be the starting vertex of the tour
  - $\mathit{reachable}(i)$ represents the fact that vertex i is reachable from 0
  - $\mathit{in}(i, j)$ 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:
    $\{\langle u, v\rangle \mid \mathit{in}(u, v) \in \mathcal{M}\}$ is the set of edges in the Hamiltonian cycle

Computing Hamiltonian Cycles

- **Program**
  - $\{in(u, v) \vee \neg in(u, v) \leftarrow \mid \langle u, v\rangle \in G\}$
  - $\{\leftarrow in(u, v) \wedge in(u, w) \mid \langle u, v\rangle, \langle u, w\rangle \in G \text{ and } v \not\approx w\}$
  - $\{\leftarrow in(v, u) \wedge in(w, u) \mid \langle v, u\rangle, \langle w, u\rangle \in G \text{ and } v \not\approx w\}$
  - $\{reachable(u) \leftarrow in(0, u) \mid \langle 0, u\rangle \in G\}$
  - $\{reachable(v) \leftarrow reachable(u) \wedge in(u, v) \mid \langle u, v\rangle \in G\}$
  - $\{\leftarrow \sim reachable(u) \mid 0 \leq u \leq n\}$
- **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**