Handling Exceptions in Knowledge Representation: A Brief Introduction to Nonmonotonic Reasoning

Gerhard Brewka

Computer Science Institute
University of Leipzig
brewka@informatik.uni-leipzig.de
Outline

Lecture I: Background and Simple Forms of Nonmon

1 Background and Motivation
2 Closed World Assumption
3 Argumentation Frameworks

Lecture II: The Big Three and ASP

4 Preferences Among Formulas: Poole and Beyond
5 Preferences Among Models: Circumscription
6 Nonstandard Inference Rules: Default Logic
7 Answer Set Programming
Lecture I: Background and Simple Forms of Nonmon

1 Background and Motivation
2 Closed World Assumption
3 Argumentation Frameworks

Lecture II: The Big Three and ASP

4 Preferences Among Formulas: Poole and Beyond
5 Preferences Among Models: Circumscription
6 Nonstandard Inference Rules: Default Logic
7 Answer Set Programming
The Big Three and ASP
4. Preferences Among Formulas: Poole and Beyond

- Treat defaults as classical formulas with lower priority.
- Partition KB into (consistent) strict part $F$ and defeasible part $W$.
- In case of a conflict give up formulas from the latter set, that is consider “scenarios” (Poole) of the form

$$F \cup W'$$

where $W'$ is a maximal $F$-consistent subset of $W$.

**Example**

$F = \{ \text{bird(tweety)}, \text{bird(fritz)}, \neg \text{flies(fritz)} \}$

$W = \{ \text{bird(tweety)} \rightarrow \text{flies(tweety)}, \text{bird(fritz)} \rightarrow \text{flies(fritz)} \}$

Scenario: $F \cup \{ \text{bird(tweety)} \rightarrow \text{flies(tweety)} \}$

Conclude $\text{flies(tweety)}$ from single scenario.
4. Preferences Among Formulas: Poole and Beyond

- Treat defaults as classical formulas with lower priority.
- Partition KB into (consistent) strict part $F$ and defeasible part $W$.
- In case of a conflict give up formulas from the latter set, that is consider “scenarios” (Poole) of the form

$$F \cup W'$$

where $W'$ is a maximal $F$-consistent subset of $W$.

Example

$F = \{\text{bird}(\text{tweety}), \text{bird}(\text{fritz}), \neg\text{flies}(\text{fritz})\}$

$W = \{\text{bird}(\text{tweety}) \rightarrow \text{flies}(\text{tweety}), \text{bird}(\text{fritz}) \rightarrow \text{flies}(\text{fritz})\}$

Scenario: $F \cup \{\text{bird}(\text{tweety}) \rightarrow \text{flies}(\text{tweety})\}$

Conclude $\text{flies}(\text{tweety})$ from single scenario.
• May get multiple scenarios.
• Skeptical vs. credulous reasoning: \( p \) follows from all scenarios vs. \( p \) follows from some scenario.

Example

\[ F = \{bird(tweety), peng(tweety)\} \]
\[ W = \{bird(tweety) \rightarrow flies(tweety), peng(tweety) \rightarrow \neg flies(tweety)\} \]
Scenario 1: \( F \cup \{bird(tweety) \rightarrow flies(tweety)\} \)
Scenario 2: \( F \cup \{peng(tweety) \rightarrow \neg flies(tweety)\} \)
Neither \( flies(tweety) \) nor \( \neg flies(tweety) \) follows skeptically.

• Important to represent instances of \( Birds\ fly \), not universal formula (otherwise single nonflying bird eliminates the default).
• Example suggests generalization: defaults preferred to others.
May get multiple scenarios.

Skeptical vs. credulous reasoning: \( p \) follows from all scenarios vs. \( p \) follows from some scenario.

**Example**

\[
F = \{ \text{bird(tweety)}, \text{peng(tweety)} \} \\
W = \{ \text{bird(tweety)} \rightarrow \text{flies(tweety)}, \text{peng(tweety)} \rightarrow \neg \text{flies(tweety)} \} \\
\]

Scenario 1: \( F \cup \{ \text{bird(tweety)} \rightarrow \text{flies(tweety)} \} \)

Scenario 2: \( F \cup \{ \text{peng(tweety)} \rightarrow \neg \text{flies(tweety)} \} \)

neither \( \text{flies(tweety)} \) nor \( \neg \text{flies(tweety)} \) follows skeptically.

Important to represent instances of *Birds fly*, not universal formula (otherwise single nonflying bird eliminates the default).

Example suggests generalization: defaults preferred to others.
• May get multiple scenarios.
• Skeptical vs. credulous reasoning: $p$ follows from all scenarios vs. $p$ follows from some scenario.

Example

$F = \{\text{bird(tweety)}, \text{peng(tweety)}\}$

$W = \{\text{bird(tweety)} \rightarrow \text{flies(tweety)}, \text{peng(tweety)} \rightarrow \neg \text{flies(tweety)}\}$

Scenario 1: $F \cup \{\text{bird(tweety)} \rightarrow \text{flies(tweety)}\}$

Scenario 2: $F \cup \{\text{peng(tweety)} \rightarrow \neg \text{flies(tweety)}\}$

neither $\text{flies(tweety)}$ nor $\neg \text{flies(tweety)}$ follows skeptically.

• Important to represent instances of *Birds fly*, not universal formula (otherwise single nonflying bird eliminates the default).
• Example suggests generalization: defaults preferred to others.
Preferred subtheories

- Basic idea: introduce arbitrary preference levels.
- Rather than $(F, W)$ use partition $KB = (F_1, \ldots, F_n)$; $F_1$ most reliable formulas, $F_2$ second best, etc.
- Preferred subtheory: maxi-consistent subset $S$ of $F_1 \cup \ldots \cup F_n$ containing maxi-consistent subset of $F_1 \cup \ldots \cup F_i$ for each $i \leq n$.
- Intuition: pick maxi-consistent subset of $F_1$, extend it maximally with formulas from $F_2$, etc.
Example

\[ F_1 = \{ \text{bird(tweety), penguin(tweety)} \} \]
\[ F_2 = \{ \text{penguin(tweety)} \rightarrow \neg \text{flies(tweety)} \} \]
\[ F_3 = \{ \text{bird(tweety)} \rightarrow \text{flies(tweety)} \} \]

Single preferred subtheory: \( F_1 \cup F_2 \)

\( \neg \text{flies(tweety)} \) follows skeptically
Remarks

- Simple approach reducing default reasoning to inconsistency handling.
- No nonstandard semantics, no nonstandard language constructs.
- Easy handling of preferences.
- Quantitative extensions straightforward, e.g. reliability value for each formula, consistent subsets ranked by sum of values.
- Less expressive than other approaches, e.g. implicit default contraposition.

*A computer scientist normally doesn’t know about nonmon.*

*vs.*

*Who knows about nonmon normally isn’t a computer scientist.*
Remarks

- Simple approach reducing default reasoning to inconsistency handling.
- No nonstandard semantics, no nonstandard language constructs.
- Easy handling of preferences.
- Quantitative extensions straightforward, e.g. reliability value for each formula, consistent subsets ranked by sum of values.
- Less expressive than other approaches, e.g. implicit default contraposition.

A computer scientist normally doesn’t know about nonmon.

vs.

Who knows about nonmon normally isn’t a computer scientist.
Remarks

- Simple approach reducing default reasoning to inconsistency handling.
- No nonstandard semantics, no nonstandard language constructs.
- Easy handling of preferences.
- Quantitative extensions straightforward, e.g. reliability value for each formula, consistent subsets ranked by sum of values.
- Less expressive than other approaches, e.g. implicit default contraposition.

*A computer scientist normally doesn’t know about nonmon.*

*vs.*

*Who knows about nonmon normally isn’t a computer scientist.*
Remarks

- Simple approach reducing default reasoning to inconsistency handling.
- No nonstandard semantics, no nonstandard language constructs.
- Easy handling of preferences.
- Quantitative extensions straightforward, e.g. reliability value for each formula, consistent subsets ranked by sum of values.
- Less expressive than other approaches, e.g. implicit default contraposition.

*A computer scientist normally doesn’t know about nonmon.*

*vs.*

*Who knows about nonmon normally isn’t a computer scientist.*
5. Preferences Among Models: Circumscription

- CWA makes extension of all predicates as small as possible (1st order) or as many atoms false as possible (propositional).
- Let’s do this for selected predicates/atoms only.
- Corresponds to focus on specific minimal models.
- Solves inconsistency problem of CWA.
- Comes with a default representation scheme (ab predicates):

\[ \forall x. \text{Bird}(x) \land \neg \text{Ab}(x) \rightarrow \text{Flies}(x). \]

- Need several Ab predicates, one for each default.
CWA makes extension of all predicates as small as possible (1st order) or as many atoms false as possible (propositional).

Let's do this for selected predicates/atoms only.

Corresponds to focus on specific minimal models.

Solves inconsistency problem of CWA.

Comes with a default representation scheme ($ab$ predicates):

$$\forall x. \text{Bird}(x) \land \neg \text{Ab}(x) \rightarrow \text{Flies}(x).$$

Need several $Ab$ predicates, one for each default.
5. Preferences Among Models: Circumscription

- CWA makes extension of all predicates as small as possible (1st order) or as many atoms false as possible (propositional).
- Let's do this for selected predicates/atoms only.
- Corresponds to focus on specific minimal models.
- Solves inconsistency problem of CWA.
- Comes with a default representation scheme (\(ab\) predicates):
  \[
  \forall x. Bird(x) \land \neg Ab(x) \rightarrow Flies(x).
  \]
- Need several \(Ab\) predicates, one for each default.
5. Preferences Among Models: Circumscription

- CWA makes extension of all predicates as small as possible (1st order) or as many atoms false as possible (propositional).
- Let’s do this for selected predicates/atoms only.
- Corresponds to focus on specific minimal models.
- Solves inconsistency problem of CWA.
- Comes with a default representation scheme (ab predicates):
  \[ \forall x. Bird(x) \land \neg Ab(x) \rightarrow Flies(x). \]
- Need several Ab predicates, one for each default.
CWA makes extension of all predicates as small as possible (1st order) or as many atoms false as possible (propositional).

Let’s do this for selected predicates/atoms only.

Corresponds to focus on specific minimal models.

Solves inconsistency problem of CWA.

Comes with a default representation scheme (ab predicates):

$$\forall x. \text{Bird}(x) \land \neg \text{Ab}(x) \rightarrow \text{Flies}(x).$$

Need several Ab predicates, one for each default.
5. Preferences Among Models: Circumscription

- CWA makes extension of all predicates as small as possible (1st order) or as many atoms false as possible (propositional).
- Let’s do this for selected predicates/atoms only.
- Corresponds to focus on specific minimal models.
- Solves inconsistency problem of CWA.
- Comes with a default representation scheme ($ab$ predicates):
  \[
  \forall x. \text{Bird}(x) \land \neg \text{Ab}(x) \rightarrow \text{Flies}(x).
  \]
- Need several $Ab$ predicates, one for each default.
Example

$KB = \{ \text{bird, bird} \land \neg ab \rightarrow \text{flies} \}$

Models:

$M_1 = \{ \text{bird, ab, flies} \}$, $M_2 = \{ \text{bird, ab, \neg flies} \}$, $M_3 = \{ \text{bird, \neg ab, flies} \}$

- $M_1$ and $M_2$ contain an abnormality.
- Only in $M_3$ nothing is abnormal.
- Focus on models representing most normal situations.
- Accept a formula if it’s true in those models: here $flies$. 
Circumscription, ctd.

Example

$KB = \{\text{bird}, \text{bird} \land \neg ab \rightarrow \text{flies}\}$

Models:

$M_1 = \{\text{bird, ab, flies}\}, M_2 = \{\text{bird, ab, \neg flies}\}, M_3 = \{\text{bird, \neg ab, flies}\}$

- $M_1$ and $M_2$ contain an abnormality.
- Only in $M_3$ nothing is abnormal.
- Focus on models representing most normal situations.
- Accept a formula if it’s true in those models: here $\text{flies}$. 
Circumscription, ctd.

Example

\[ KB = \{ \text{bird, bird} \land \neg ab \rightarrow \text{flies} \} \]

Models:

\[ M_1 = \{ \text{bird, ab, flies} \}, \quad M_2 = \{ \text{bird, ab, \neg flies} \}, \quad M_3 = \{ \text{bird, \neg ab, flies} \} \]

- \( M_1 \) and \( M_2 \) contain an abnormality.
- Only in \( M_3 \) nothing is abnormal.
- Focus on models representing most normal situations.
- Accept a formula if it’s true in those models: here \( \text{flies} \).
Circumscription

- Given two interpretations over the same domain, \( I_1 \) and \( I_2 \). Let
  \[
  I_1 \leq I_2 \text{ iff } I_1[Ab] \subseteq I_2[Ab] \text{ for every } Ab \text{ predicate,}
  \]
  \[
  I_1 < I_2 \text{ iff } I_1 \leq I_2 \text{ but not } I_2 \leq I_1.
  \]

- Define a new version of entailment:
  \[
  KB \models_{\leq} \alpha \text{ iff for every } I,
  \]
  \[
  I \models \alpha \text{ whenever } I \models KB \text{ and for no } I' < I \text{ we have } I' \models KB.
  \]

- So \( \alpha \) must be true in all interpretations satisfying KB that are minimal in abnormalities.
Circumscription

- Given two interpretations over the same domain, $I_1$ and $I_2$. Let
  
  $I_1 \leq I_2$ iff $I_1[Ab] \subseteq I_2[Ab]$ for every Ab predicate,
  
  $I_1 < I_2$ iff $I_1 \leq I_2$ but not $I_2 \leq I_1$.

- Define a new version of entailment:
  
  $KB \models_{\leq} \alpha$ iff for every $I$,
  
  $I \models \alpha$ whenever $I \models KB$ and for no $I' < I$ we have $I' \models KB$.

- So $\alpha$ must be true in all interpretations satisfying $KB$ that are minimal in abnormalities.
Circumscription

- Given two interpretations over the same domain, $I_1$ and $I_2$. Let
  \[ I_1 \leq I_2 \text{ iff } I_1[Ab] \subseteq I_2[Ab] \text{ for every } Ab \text{ predicate}, \]
  \[ I_1 < I_2 \text{ iff } I_1 \leq I_2 \text{ but not } I_2 \leq I_1. \]

- Define a new version of entailment:
  \[ KB \models_{\leq} \alpha \text{ iff for every } I, \]
  \[ I \models \alpha \text{ whenever } I \models KB \text{ and for no } I' < I \text{ we have } I' \models KB. \]

- So $\alpha$ must be true in all interpretations satisfying KB that are minimal in abnormalities.
Circumscription, ctd.

- Why is this nonmonotonic?
- Additional information may eliminate models.
- Must check the most normal among the remaining ones; may have abnormalities.

Example

\[KB = \{\text{bird}, \text{bird} \land \neg \text{ab} \rightarrow \text{flies}, \text{ab}\}\]

Models:

\[M_1 = \{\text{bird}, \text{ab}, \text{flies}\}, \ M_2 = \{\text{bird}, \text{ab}, \neg \text{flies}\}, \ M_3 \text{ no longer a model.}\]

- Both \(M_1\) and \(M_2\) are as normal as possible.
- \(\text{flies}\) no longer in all most normal models.
Circumscription, ctd.

- Why is this nonmonotonic?
- Additional information may eliminate models.
- Must check the most normal among the remaining ones; may have abnormalities.

Example

\[ KB = \{ \text{bird}, \text{bird} \land \neg \text{ab} \rightarrow \text{flies}, \text{ab} \} \]

Models:

\[ M_1 = \{ \text{bird}, \text{ab}, \text{flies} \}, \quad M_2 = \{ \text{bird}, \text{ab}, \neg \text{flies} \}, \quad M_3 \text{ no longer a model.} \]

- Both \( M_1 \) and \( M_2 \) are as normal as possible.
- \( \text{flies} \) no longer in all most normal models.
Circumscription, ctd.

- Why is this nonmonotonic?
- Additional information may eliminate models.
- Must check the most normal among the remaining ones; may have abnormalities.

**Example**

\[ KB = \{bird, bird \land \neg ab \rightarrow flies, ab\} \]

Models:

\[ M_1 = \{bird, ab, flies\}, M_2 = \{bird, ab, \neg flies\}, M_3 \text{ no longer a model.} \]

- Both \( M_1 \) and \( M_2 \) are as normal as possible.
- \( flies \) no longer in all most normal models.
Circumscription: 2nd order characterization

- Circumscription can be represented as a second order formula.

$T(P)$ first order formula containing predicate symbol $P$. $T(p)$ obtained from $T(P)$ by replacing each occurrence of $P$ by variable $p$.

Abbreviations:

$P \leq Q$ for $\forall x. P(x) \rightarrow Q(x)$

$P < Q$ for $P \leq Q$ and not $Q \leq P$

$\text{Circ}(P, T(P))$, the circumscription of $P$ in $T(P)$:

$T(P) \land \neg \exists p. (T(p) \land p < P)$

- Intuition: $T(P)$ and there is no predicate smaller than $P$ satisfying everything $T$ says about $P$.

- Theorem: $T(Ab) \models \leq q$ iff $q$ consequence of $\text{Circ}(Ab, T(Ab))$. 
Remarks

- Circumscription a skeptical approach: conflicting defaults cancel each other.
  - Problem: 2nd order logic not even semi-decidable.
  - Various results about when 2nd order formula has equivalent 1st order representation (Lifschitz).
  - For restricted cases standard theorem provers can be used.
  - Various more flexible variants of circumscription were defined: fixed predicates, preferences, ....
  - They all have corresponding 2nd order formula.
Remarks

- Circumscription a skeptical approach: conflicting defaults cancel each other.

- Problem: 2nd order logic not even semi-decidable.
  - Various results about when 2nd order formula has equivalent 1st order representation (Lifschitz).
  - For restricted cases standard theorem provers can be used.
  - Various more flexible variants of circumscription were defined: fixed predicates, preferences, ....
  - They all have corresponding 2nd order formula.
Remarks

• Circumscription a skeptical approach: conflicting defaults cancel each other.

• Problem: 2nd order logic not even semi-decidable.

• Various results about when 2nd order formula has equivalent 1st order representation (Lifschitz).

• For restricted cases standard theorem provers can be used.

• Various more flexible variants of circumscription were defined: fixed predicates, preferences, ....

• They all have corresponding 2nd order formula.
Remarks

• Circumscription a skeptical approach: conflicting defaults cancel each other.

• Problem: 2nd order logic not even semi-decidable.

• Various results about when 2nd order formula has equivalent 1st order representation (Lifschitz).

• For restricted cases standard theorem provers can be used.

• Various more flexible variants of circumscription were defined: fixed predicates, preferences, ....

• They all have corresponding 2nd order formula.
6. Nonstandard inference rules: default logic

- To represent defaults, Reiter uses rules of the form

\[ A : B_1, \ldots, B_n/C \]

where \( A, B_i, C \) are formulas.

- Intuition: if \( A \) believed and each \( B_i \) consistent with beliefs, then infer \( C \).

- Default theory: \((D, W)\), \( D \) set of defaults, \( W \) set of formulas representing what is known to be true.

- Default theories generate extensions: acceptable sets of beliefs.

- Main problem: cannot apply defaults constructively; consistency condition must hold with respect to final outcome.

- Reiter’s fixpoint solution: guess the final outcome and verify that the guess was good.
6. Nonstandard inference rules: default logic

- To represent defaults, Reiter uses rules of the form
  \[ A : B_1, \ldots, B_n / C \]
  where \( A, B_i, C \) are formulas.

- Intuition: if \( A \) believed and each \( B_i \) consistent with beliefs, then infer \( C \).

- Default theory: \((D, W)\), \( D \) set of defaults, \( W \) set of formulas representing what is known to be true.

- Default theories generate extensions: acceptable sets of beliefs.

- Main problem: cannot apply defaults constructively; consistency condition must hold with respect to final outcome.

- Reiter’s fixpoint solution: guess the final outcome and verify that the guess was good.
To represent defaults, Reiter uses rules of the form

\[ A : B_1, \ldots, B_n / C \]

where \( A, B_i, C \) are formulas.

Intuition: if \( A \) believed and each \( B_i \) consistent with beliefs, then infer \( C \).

Default theory: \((D, W)\), \( D \) set of defaults, \( W \) set of formulas representing what is known to be true.

Default theories generate extensions: acceptable sets of beliefs.

Main problem: cannot apply defaults constructively; consistency condition must hold with respect to final outcome.

Reiter’s fixpoint solution: guess the final outcome and verify that the guess was good.
To represent defaults, Reiter uses rules of the form

\[ A : B_1, \ldots, B_n / C \]

where \( A, B_i, C \) are formulas.

Intuition: if \( A \) believed and each \( B_i \) consistent with beliefs, then infer \( C \).

Default theory: \((D, W)\), \( D \) set of defaults, \( W \) set of formulas representing what is known to be true.

Default theories generate extensions: acceptable sets of beliefs.

Main problem: cannot apply defaults constructively; consistency condition must hold with respect to final outcome.

Reiter’s fixpoint solution: guess the final outcome and verify that the guess was good.
Motivation of fixpoint construction

- Properties an extension $E$ should satisfy
  1. should contain $W$ and be deductively closed,
  2. all defaults applicable wrt. $E$ must have been applied,
  3. no formula in $E$ without reasonable derivation from $W$, possibly using applicable defaults.

- (3) not achieved by considering minimal sets satisfying (1),(2).

Example

$D = \{\text{prof}(x) : \text{teaches}(x)/\text{teaches}(x)\}$

$W = \{\text{prof(gerd)}\}$

$Th(\{\text{prof(gerd)}, \neg \text{teaches(gerd)}\})$ minimal set satisfying (1),(2).

Obviously not intended: $\neg \text{teaches(gerd)}$ out of the blue.
Motivation of fixpoint construction

- Properties an extension $E$ should satisfy
  1. should contain $W$ and be deductively closed,
  2. all defaults applicable wrt. $E$ must have been applied,
  3. no formula in $E$ without reasonable derivation from $W$, possibly using applicable defaults.

- (3) not achieved by considering minimal sets satisfying (1),(2).

Example

$D = \{ \text{prof}(x) : \text{teaches}(x) \}$

$W = \{ \text{prof}(\text{gerd}) \}$

$\text{Th}(\{ \text{prof}(\text{gerd}), \neg \text{teaches}(\text{gerd}) \})$ minimal set satisfying (1),(2).

Obviously not intended: $\neg \text{teaches}(\text{gerd})$ out of the blue.
The problem

- Standard inference: iterative construction of closure; at each step apply inference rule applicable wrt. what was derived so far.
- What is inferred once remains conclusion forever.
- Not so for defaults: consistency at some stage may be lost later.

Example

\[ D = \{ p : q/r, \ p : s/s, \ s : \neg q/\neg q \} \]
\[ w = \{ p \} \]

Sequence of sets generated by applicable defaults and deduction:
\[ E_0 = \{ p \}; \ E_1 = Th(\{ p, r, s \}); \ E_2 = Th(\{ p, r, s, \neg q \}) \]

\[ p : q/r \] applied to construct \( E_1 \); \( q \) inconsistent with \( E_2 \).
The problem

- Standard inference: iterative construction of closure; at each step apply inference rule applicable wrt. what was derived so far.
- What is inferred once remains conclusion forever.
- Not so for defaults: consistency at some stage may be lost later.

Example

\[ D = \{ p : q/r, \ p : s/s, \ s : \neg q/\neg q \} \]
\[ w = \{ p \} \]

Sequence of sets generated by applicable defaults and deduction:
\[ E_0 = \{ p \}; \ E_1 = Th(\{ p, r, s \}); \ E_2 = Th(\{ p, r, s, \neg q \}) \]
\[ p : q/r \text{ applied to construct } E_1; \ q \text{ inconsistent with } E_2. \]
Reiter’s solution

- Guess outcome of inference process; verify it’s justified.
- Define operator assigning to each $S$ the outcome of the construction \textit{when consistency is tested against $S$}.
- Fixpoints of the operator then are what we are looking for.

\textbf{Definition}

Let $\Delta = (D, W)$ be a default theory, $S$ a set of formulas. $\Gamma_\Delta(S)$ is the smallest set of formulas satisfying:

1. $W \subseteq \Gamma_\Delta(S)$,
2. $Th(\Gamma_\Delta(S)) = \Gamma_\Delta(S)$,
3. if $a : b_1, \ldots, b_n / c \in D$, $a \in \Gamma_\Delta(S)$, each $\neg b_i$ not in $S$, then $c \in \Gamma_\Delta(S)$.

$E$ is an extension of $\Delta$ iff $E$ is a fixpoint of $\Gamma_\Delta$. 
Reiter’s solution

- Guess outcome of inference process; verify it’s justified.
- Define operator assigning to each $S$ the outcome of the construction when consistency is tested against $S$.
- Fixpoints of the operator then are what we are looking for.

Definition

Let $\Delta = (D, W)$ be a default theory, $S$ a set of formulas. $\Gamma_\Delta(S)$ is the smallest set of formulas satisfying

1. $W \subseteq \Gamma_\Delta(S)$,
2. $\text{Th}(\Gamma_\Delta(S)) = \Gamma_\Delta(S)$,
3. if $a : b_1, \ldots b_n/c \in D$, $a \in \Gamma_\Delta(S)$, each $\neg b_i$ not in $S$, then $c \in \Gamma_\Delta(S)$.

$E$ is an extension of $\Delta$ iff $E$ is a fixpoint of $\Gamma_\Delta$. 
**Examples**

<table>
<thead>
<tr>
<th>$D$</th>
<th>$W$</th>
<th>Extensions</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>bird : flies/flies</code></td>
<td><code>bird</code></td>
<td>$Th(W \cup {flies})$</td>
</tr>
<tr>
<td><code>bird : flies/flies</code></td>
<td><code>bird, peng</code></td>
<td>$Th(W)$</td>
</tr>
<tr>
<td></td>
<td><code>peng \rightarrow \neg flies</code></td>
<td></td>
</tr>
<tr>
<td><code>bird : flies/flies</code></td>
<td><code>bird, peng</code></td>
<td>$Th(W \cup {flies})$</td>
</tr>
<tr>
<td><code>peng : \neg flies/\neg flies</code></td>
<td><code>bird, peng</code></td>
<td>$Th(W \cup {\neg flies})$</td>
</tr>
<tr>
<td><code>bird : flies \land \neg peng/flies</code></td>
<td><code>bird, peng</code></td>
<td>$Th(W \cup {\neg flies})$</td>
</tr>
</tbody>
</table>
Results

- Extensions may not exist: $\Delta = (\{true : \neg a/a\}, \emptyset)$.

- Types of defaults:
  - Normal: $p : q/q$.
    Normal default theories always have extensions.
  - Supernormal: $true : q/q$.
    Can model Poole systems.
  - Seminormal: $true : p \land q/q$.
    Used to encode preferences. Extensions may not exist.

- Extensions subset minimal: $E_1, E_2$ extensions $\Rightarrow E_1 \not\subseteq E_2$.

- $W$ inconsistent iff set of all formulas single extension.

- Defaults with open variables: usually viewed as schemata.
• Extensions may not exist: $\Delta = (\{true : \neg a/a\}, \emptyset)$.

• Types of defaults:
  • Normal: $p : q/q$. Normal default theories always have extensions.
  • Supernormal: $true : q/q$. Can model Poole systems.
  • Seminormal: $true : p \land q/q$. Used to encode preferences. Extensions may not exist.

• Extensions subset minimal: $E_1, E_2$ extensions $\Rightarrow E_1 \nsubseteq E_2$.

• $\cal W$ inconsistent iff set of all formulas single extension.

• Defaults with open variables: usually viewed as schemata.
• Extensions may not exist: $\Delta = (\{ \text{true} : \neg a/a \}, \emptyset)$.

• Types of defaults:
  • Normal: $p : q/q$.
    Normal default theories always have extensions.
  • Supernormal: $\text{true} : q/q$.
    Can model Poole systems.
  • Seminormal: $\text{true} : p \land q/q$.
    Used to encode preferences. Extensions may not exist.

• Extensions subset minimal: $E_1, E_2$ extensions $\Rightarrow E_1 \nsubseteq E_2$.

• $W$ inconsistent iff set of all formulas single extension.

• Defaults with open variables: usually viewed as schemata.
Answer sets (alias stable models for programs considered here) provide semantics for logic programs with not.

Logic programming initially independent of nonmon.

Default negation not interpreted procedurally: negation as failure.

Problems with cycles.

Example

\[ a \leftarrow \neg b, \quad b \leftarrow \neg a \]

\[ a \ \text{provable iff proof for } b \ \text{fails iff proof of } a \ \text{succeeds iff ...} \]

Solution: bring in ideas from nonmon.

Language restriction basis for highly successful implementations.

Shift from theorems to models basis for ASP paradigm.
Answer Sets

- Answer sets (alias stable models for programs considered here) provide semantics for logic programs with `not`.
- Logic programming initially independent of nonmon.
- Default negation `not` interpreted procedurally: negation as failure.
- Problems with cycles.

**Example**

\[ a \leftarrow \text{not} \ b, \quad b \leftarrow \text{not} \ a \]

\( a \) provable iff proof for \( b \) fails iff proof of \( a \) succeeds iff ...

- Solution: bring in ideas from nonmon.
- Language restriction basis for highly successful implementations.
- Shift from theorems to models basis for ASP paradigm.
7. Answer Sets

- Answer sets (alias stable models for programs considered here) provide semantics for logic programs with not.
- Logic programming initially independent of nonmon.
- Default negation not interpreted procedurally: negation as failure.
- Problems with cycles.

**Example**

\[ a \leftarrow \text{not} \ b, \quad b \leftarrow \text{not} \ a \]

\[ a \text{ provable iff proof for } b \text{ fails iff proof of } a \text{ succeeds iff } \ldots \]

- Solution: bring in ideas from nonmon.
- Language restriction basis for highly successful implementations.
- Shift from theorems to models basis for ASP paradigm.
A (ground) normal logic program $P$ is a collection of rules of the form

$$A \leftarrow B_1, \ldots, B_n, \text{not } C_1, \ldots \text{ not } C_m$$

where $A, B_i, C_j$ are ground atoms. \text{not } C \text{ reads: } C \text{ is not believed.}

- **Answer set:** atoms representing reasonable beliefs based on $P$.
- **Intuition similar to default logic:**
  1. Each applicable rule applied.
  2. No atom without valid derivation.
- **Simplifications:** no set $W$; beliefs fully determined by atoms.
- **Identify rule with default $B_1 \land \ldots \land B_n : \neg C_1, \ldots \neg C_m / A$ and strip unneeded parts off Reiter’s definition $\Rightarrow$ GL-reduct.”
A (ground) normal logic program $P$ is a collection of rules of the form

$$A \leftarrow B_1, \ldots, B_n, \neg C_1, \ldots \neg C_m$$

where $A, B_i, C_j$ are ground atoms. $\neg C$ reads: $C$ is not believed.

- Answer set: atoms representing reasonable beliefs based on $P$.
- Intuition similar to default logic:
  1. Each applicable rule applied.
  2. No atom without valid derivation.
- Simplifications: no set $W$; beliefs fully determined by atoms.
- Identify rule with default $B_1 \land \ldots \land B_n : \neg C_1, \ldots \neg C_m / A$ and strip unneeded parts off Reiter’s definition $\Rightarrow$ GL-reduct.
Gelfond-Lifschitz reduct

Definition

Let $P$ be a (ground) normal logic program, $S$ a set of atoms.

$P^S$ is the program obtained from $P$ by

1. eliminating rules containing $\not C$ for some $C \in S$,
2. eliminating negated literals from the remaining rules.

$S$ is an answer set of $P$ iff $S = Cl(P^S)$.

- $Cl(R)$ denotes the closure of a set of classical inference rules
- Intuition: guess $S$ and evaluate $\not$ wrt. $S$.
  1. Atom $p$ without valid derivation: $p$ will not appear in $Cl(P^S)$.
  2. Applicable rule $r$ not applied: $r$’s conclusion in $Cl(P^S)$.

- Sets of atoms satisfying both intended properties pass the test.
Gelfond-Lifschitz reduct

**Definition**

Let $P$ be a (ground) normal logic program, $S$ a set of atoms. $P^S$ is the program obtained from $P$ by:

1. eliminating rules containing $\text{not } C$ for some $C \in S$,
2. eliminating negated literals from the remaining rules.

$S$ is an answer set of $P$ iff $S = Cl(P^S)$.

- $Cl(R)$ denotes the closure of a set of classical inference rules.
- Intuition: guess $S$ and evaluate $\text{not}$ wrt. $S$.
  
  1. Atom $p$ without valid derivation: $p$ will not appear in $Cl(P^S)$.
  2. Applicable rule $r$ not applied: $r$’s conclusion in $Cl(P^S)$.

- Sets of atoms satisfying both intended properties pass the test.
Answer set programming

- Represent problem such that solutions are (parts of) answer sets.
- Commonly used method: generate and test:
  1. Generate candidate sets of atoms.
  2. Eliminate those not satisfying intended properties.
  3. Elimination via rules without head.
- Observation: if $P$ does not contain $q$, then
  \[ q \leftarrow \text{not } q, \text{body} \]
  eliminates answer sets satisfying body.
- Abbreviation: $\leftarrow \text{body}$. 
Variables in programs

- Definition of answer sets for propositional programs.
- Variables useful for problem descriptions.
- Rule with variables shorthand for all ground instances of the rule.
- ASP system: grounder + solver.
- Grounder produces ground instantiation of program, solver computes its answer sets.
Graph coloring

Example

Description of graph:
node(v₁), ..., node(vₙ), edge(vᵢ, vⱼ), ...

Generate:
\[ \text{col}(X, r) \leftarrow \text{node}(X), \text{not col}(X, b), \text{not col}(X, g) \]
\[ \text{col}(X, b) \leftarrow \text{node}(X), \text{not col}(X, r), \text{not col}(X, g) \]
\[ \text{col}(X, g) \leftarrow \text{node}(X), \text{not col}(X, r), \text{not col}(X, b) \]

Test:
\[ \leftarrow \text{edge}(X, Y), \text{col}(X, Z), \text{col}(Y, Z) \]

Answer sets contain solution to problem!
Meeting scheduling

Example

Problem instance:

- $meeting(m_1), \ldots, meeting(m_n)$
- $time(t_1), \ldots, time(t_s)$
- $room(r_1), \ldots, room(r_m)$
- $person(p_1), \ldots, person(p_k)$
- $par(p_1, m_1), \ldots, par(p_2, m_3), \ldots$

Instance independent part, generate:

- $at(M, T) \leftarrow meeting(M), time(T), not \neg at(M, T)$
- $\neg at(M, T) \leftarrow meeting(M), time(T), not \neg at(M, T)$
- $in(M, R) \leftarrow meeting(M), room(R), not \neg in(M, R)$
- $\neg in(M, R) \leftarrow meeting(M), room(R), not \neg in(M, R)$
Each meeting has assigned time and room:
\[
\text{timeassigned}(M) \leftarrow \text{at}(M, T) \\
\text{roomassigned}(M) \leftarrow \text{in}(M, R)
\]
\[
\leftarrow \text{meeting}(M), \text{not } \text{timeassigned}(M) \\
\leftarrow \text{meeting}(M), \text{not } \text{roomassigned}(M)
\]

No meeting has more than 1 time and room:
\[
\leftarrow \text{meeting}(M), \text{at}(M, T), \text{at}(M, T'), T \neq T' \\
\leftarrow \text{meeting}(M), \text{in}(M, R), \text{in}(M, R'), R \neq R'
\]

Meetings at same time need different rooms:
\[
\leftarrow \text{in}(M, X), \text{in}(M', X), \text{at}(M, T), \text{at}(M', T), M \neq M'
\]

Meetings with same person need different times:
\[
\leftarrow \text{par}(P, M), \text{par}(P, M'), M \neq M', \text{at}(M, T), \text{at}(M', T)
\]
Summary

• Presented some of the major approaches to nonmon.

• Started with motivation and simple forms (CWA, AFs).

• Sketched preferred subtheories, circumscription, default logic.

• Finally presented definition of answer sets, filling a gap in Schaub’s talk.

• Focused on the main underlying ideas.

• Many more approaches (autoepistemic logic, KLM), in particular some with implicit treatment of specificity and explicit preferences.

• Current focus: ASP solvers; argumentation.

• Preferences a natural aspect to bring in quantities.
Summary

- Presented some of the major approaches to nonmon.
- Started with motivation and simple forms (CWA, AFs).
- Sketched preferred subtheories, circumscription, default logic.
- Finally presented definition of answer sets, filling a gap in Schaub’s talk.
- Focused on the main underlying ideas.
- Many more approaches (autoepistemic logic, KLM), in particular some with implicit treatment of specificity and explicit preferences.
- Current focus: ASP solvers; argumentation.
- Preferences a natural aspect to bring in quantities.
Summary

• Presented some of the major approaches to nonmon.

• Started with motivation and simple forms (CWA, AFs).

• Sketched preferred subtheories, circumscription, default logic.

• Finally presented definition of answer sets, filling a gap in Schaub’s talk.

• Focused on the main underlying ideas.

• Many more approaches (autoepistemic logic, KLM), in particular some with implicit treatment of specificity and explicit preferences.

• Current focus: ASP solvers; argumentation.

• Preferences a natural aspect to bring in quantities.
• Presented some of the major approaches to nonmon.
• Started with motivation and simple forms (CWA, AFs).
• Sketched preferred subtheories, circumscription, default logic.
• Finally presented definition of answer sets, filling a gap in Schaub’s talk.
• Focused on the main underlying ideas.
• Many more approaches (autoepistemic logic, KLM), in particular some with implicit treatment of specificity and explicit preferences.
• Current focus: ASP solvers; argumentation.
• Preferences a natural aspect to bring in quantities.
Summary

- Presented some of the major approaches to nonmon.
- Started with motivation and simple forms (CWA, AFs).
- Sketched preferred subtheories, circumscription, default logic.
- Finally presented definition of answer sets, filling a gap in Schaub’s talk.
- Focused on the main underlying ideas.
  - Many more approaches (autoepistemic logic, KLM), in particular some with implicit treatment of specificity and explicit preferences.
  - Current focus: ASP solvers; argumentation.
  - Preferences a natural aspect to bring in quantities.
Summary

• Presented some of the major approaches to nonmon.
• Started with motivation and simple forms (CWA, AFs).
• Sketched preferred subtheories, circumscription, default logic.
• Finally presented definition of answer sets, filling a gap in Schaub’s talk.
• Focused on the main underlying ideas.
• Many more approaches (autoepistemic logic, KLM), in particular some with implicit treatment of specificity and explicit preferences.
• Current focus: ASP solvers; argumentation.
• Preferences a natural aspect to bring in quantities.
• Presented some of the major approaches to nonmon.
• Started with motivation and simple forms (CWA, AFs).
• Sketched preferred subtheories, circumscription, default logic.
• Finally presented definition of answer sets, filling a gap in Schaub’s talk.
• Focused on the main underlying ideas.
• Many more approaches (autoepistemic logic, KLM), in particular some with implicit treatment of specificity and explicit preferences.
• Current focus: ASP solvers; argumentation.
• Preferences a natural aspect to bring in quantities.
Summary

- Presented some of the major approaches to nonmon.
- Started with motivation and simple forms (CWA, AFs).
- Sketched preferred subtheories, circumscription, default logic.
- Finally presented definition of answer sets, filling a gap in Schaub’s talk.
- Focused on the main underlying ideas.
- Many more approaches (autoepistemic logic, KLM), in particular some with implicit treatment of specificity and explicit preferences.
- Current focus: ASP solvers; argumentation.
- Preferences a natural aspect to bring in quantities.
Suggested overview articles/books


THANK YOU!
Suggested overview articles/books


THANK YOU!