A Brief Introduction to Nonmonotonic Reasoning

Gerhard Brewka, Stefan Woltran

Computer Science Institute
University of Leipzig
[brewka,woltran]@informatik.uni-leipzig.de

Outline

Lecture I: Background and Simple Forms of Nonmon

1. Background and Motivation
2. Closed World Assumption
3. Argumentation Frameworks $\surd$

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

Outline

Lecture I: Background and Simple Forms of Nonmon

1. Background and Motivation
2. Closed World Assumption
3. Argumentation Frameworks $\surd$

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

Background and Simple Forms of Nonmon

1. Background and Motivation

- Classical logic allows us to represent universal statements:

$$\forall x. PhDstudent(x) \rightarrow Student(x)$$

- Useful, e.g. for concept definitions or in mathematics
- Less useful to represent generic statements which may have exceptions:
  - *Professors teach ... unless they are on sabbatical.*
  - *Birds fly ... unless they are penguins.*
  - *Owls hunt at night ... unless they live in a zoo.*
  - *Students hate theoretical computer science ... unless they are very clever.*
  - *After spending 2 hours in the doctor's waiting room patients get angry ... unless they are close to finishing a proof.*
  - ...

1. Background and Motivation

- Classical logic allows us to represent universal statements:

$$\forall x. PhDstudent(x) \rightarrow Student(x)$$

- Useful, e.g. for concept definitions or in mathematics
- Less useful to represent generic statements which may have exceptions:
  - *Professors teach ... unless they are on sabbatical.*
  - *Birds fly ... unless they are penguins.*
  - *Owls hunt at night ... unless they live in a zoo.*
  - *Students hate theoretical computer science ... unless they are very clever.*
  - *After spending 2 hours in the doctor's waiting room patients get angry ... unless they are close to finishing a proof.*
  - ...

1. Background and Motivation

- Classical logic allows us to represent universal statements:

$$\forall x.PhDstudent(x) \rightarrow Student(x)$$

- Useful, e.g. for concept definitions or in mathematics
- Less useful to represent generic statements which may have exceptions:
  - *Professors teach ... unless they are on sabbatical.*
  - *Birds fly ... unless they are penguins.*
  - *Owls hunt at night ... unless they live in a zoo.*
  - *Students hate theoretical computer science ... unless they are very clever.*
  - *After spending 2 hours in the doctor's waiting room patients get angry ... unless they are close to finishing a proof.*
  - ...

1. Background and Motivation

- Classical logic allows us to represent universal statements:

$$\forall x. PhDstudent(x) \rightarrow Student(x)$$

- Useful, e.g. for concept definitions or in mathematics
- Less useful to represent generic statements which may have exceptions:
  - *Professors teach ... unless they are on sabbatical.*
  - *Birds fly ... unless they are penguins.*
  - *Owls hunt at night ... unless they live in a zoo.*
  - *Students hate theoretical computer science ... unless they are very clever.*
  - *After spending 2 hours in the doctor's waiting room patients get angry ... unless they are close to finishing a proof.*
  - ...

1. Background and Motivation

- Classical logic allows us to represent universal statements:

$$\forall x. PhDstudent(x) \rightarrow Student(x)$$

- Useful, e.g. for concept definitions or in mathematics
- Less useful to represent generic statements which may have exceptions:
  - *Professors teach ... unless they are on sabbatical.*
  - *Birds fly ... unless they are penguins.*
  - *Owls hunt at night ... unless they live in a zoo.*
  - *Students hate theoretical computer science ... unless they are very clever.*
  - *After spending 2 hours in the doctor's waiting room patients get angry ... unless they are close to finishing a proof.*
  - ...

1. Background and Motivation

- Classical logic allows us to represent universal statements:

$$\forall x. PhDstudent(x) \rightarrow Student(x)$$

- Useful, e.g. for concept definitions or in mathematics
- Less useful to represent generic statements which may have exceptions:
  - *Professors teach ... unless they are on sabbatical.*
  - *Birds fly ... unless they are penguins.*
  - *Owls hunt at night ... unless they live in a zoo.*
  - *Students hate theoretical computer science ... unless they are very clever.*
  - *After spending 2 hours in the doctor's waiting room patients get angry ... unless they are close to finishing a proof.*
  - ...

1. Background and Motivation

- Classical logic allows us to represent universal statements:

$$\forall x.PhDstudent(x) \rightarrow Student(x)$$

- Useful, e.g. for concept definitions or in mathematics
- Less useful to represent generic statements which may have exceptions:
    - *Professors teach ... unless they are on sabbatical.*
    - *Birds fly ... unless they are penguins.*
    - *Owls hunt at night ... unless they live in a zoo.*
    - *Students hate theoretical computer science ... unless they are very clever.*
    - *After spending 2 hours in the doctor's waiting room patients get angry ... unless they are close to finishing a proof.*
    - ...

1. Background and Motivation

- Classical logic allows us to represent universal statements:

$$\forall x. PhDstudent(x) \rightarrow Student(x)$$

- Useful, e.g. for concept definitions or in mathematics
- Less useful to represent generic statements which may have exceptions:
  - *Professors teach ... unless they are on sabbatical.*
  - *Birds fly ... unless they are penguins.*
  - *Owls hunt at night ... unless they live in a zoo.*
  - *Students hate theoretical computer science ... unless they are very clever.*
  - *After spending 2 hours in the doctor's waiting room patients get angry ... unless they are close to finishing a proof.*
  - ...

1. Background and Motivation

- Classical logic allows us to represent universal statements:

$$\forall x.PhDstudent(x) \rightarrow Student(x)$$

- Useful, e.g. for concept definitions or in mathematics
- Less useful to represent generic statements which may have exceptions:
  - *Professors teach ... unless they are on sabbatical.*
  - *Birds fly ... unless they are penguins.*
  - *Owls hunt at night ... unless they live in a zoo.*
  - *Students hate theoretical computer science ... unless they are very clever.*
  - *After spending 2 hours in the doctor's waiting room patients get angry ... unless they are close to finishing a proof.*
  - ...

1. Background and Motivation

- Classical logic allows us to represent universal statements:

$$\forall x.PhDstudent(x) \rightarrow Student(x)$$

- Useful, e.g. for concept definitions or in mathematics
- Less useful to represent generic statements which may have exceptions:
  - *Professors teach ... unless they are on sabbatical.*
  - *Birds fly ... unless they are penguins.*
  - *Owls hunt at night ... unless they live in a zoo.*
  - *Students hate theoretical computer science ... unless they are very clever.*
  - *After spending 2 hours in the doctor's waiting room patients get angry ... unless they are close to finishing a proof.*
  - ...

1. Background and Motivation

- Classical logic allows us to represent universal statements:

$$\forall x. PhDstudent(x) \rightarrow Student(x)$$

- Useful, e.g. for concept definitions or in mathematics
- Less useful to represent generic statements which may have exceptions:
  - *Professors teach ... unless they are on sabbatical.*
  - *Birds fly ... unless they are penguins.*
  - *Owls hunt at night ... unless they live in a zoo.*
  - *Students hate theoretical computer science ... unless they are very clever.*
  - *After spending 2 hours in the doctor's waiting room patients get angry ... unless they are close to finishing a proof.*
  - ...

1. Background and Motivation

- Classical logic allows us to represent universal statements:

$$\forall x.PhDstudent(x) \rightarrow Student(x)$$

- Useful, e.g. for concept definitions or in mathematics
- Less useful to represent generic statements which may have exceptions:
    - *Professors teach ... unless they are on sabbatical.*
    - *Birds fly ... unless they are penguins.*
    - *Owls hunt at night ... unless they live in a zoo.*
    - *Students hate theoretical computer science ... unless they are very clever.*
    - *After spending 2 hours in the doctor's waiting room patients get angry ... unless they are close to finishing a proof.*
    - ...

A solution?

- Most of our commonsense knowledge is of this kind
- What can we do to represent it adequately?
- What if instead of $\forall x.Bird(x) \rightarrow Flies(x)$ we use

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

and add

$$\forall x.Ab(x) \leftrightarrow Penguin(x) \lor Ostrich(x) \lor Injured(x) \lor \ldots$$

- Problem 1: no exhaustive list of abnormalities.
- Problem 2: does not give us $Flies(tweety)$ unless $tweety$ is known not to be an exception.

A solution?

- Most of our commonsense knowledge is of this kind
- What can we do to represent it adequately?
- What if instead of $\forall x.Bird(x) \rightarrow Flies(x)$ we use

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

and add

$$\forall x.Ab(x) \leftrightarrow Penguin(x) \lor Ostrich(x) \lor Injured(x) \lor \ldots$$

- Problem 1: no exhaustive list of abnormalities.
- Problem 2: does not give us $Flies(tweety)$ unless $tweety$ is known not to be an exception.

A solution?

- Most of our commonsense knowledge is of this kind
- What can we do to represent it adequately?
- What if instead of $\forall x.Bird(x) \rightarrow Flies(x)$ we use

$$\forall x.Bird(x) \wedge \neg Ab(x) \rightarrow Flies(x)$$

and add

$$\forall x.Ab(x) \leftrightarrow Penguin(x) \vee Ostrich(x) \vee Injured(x) \vee \ldots$$

- Problem 1: no exhaustive list of abnormalities.
- Problem 2: does not give us *Flies*(*tweety*) unless *tweety* is known not to be an exception.

A solution?

- Most of our commonsense knowledge is of this kind
- What can we do to represent it adequately?
- What if instead of $\forall x.Bird(x) \rightarrow Flies(x)$ we use

$$\forall x.Bird(x) \wedge \neg Ab(x) \rightarrow Flies(x)$$

  and add

$$\forall x.Ab(x) \leftrightarrow Penguin(x) \vee Ostrich(x) \vee Injured(x) \vee \dots$$

- Problem 1: no exhaustive list of abnormalities.
- Problem 2: does not give us $Flies(tweety)$ unless $tweety$ is known not to be an exception.

A solution?

- Most of our commonsense knowledge is of this kind
- What can we do to represent it adequately?
- What if instead of $\forall x. Bird(x) \rightarrow Flies(x)$ we use

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

and add

$$\forall x. Ab(x) \leftrightarrow Penguin(x) \lor Ostrich(x) \lor Injured(x) \lor \dots$$

- Problem 1: no exhaustive list of abnormalities.
- Problem 2: does not give us $Flies(tweety)$ unless $tweety$ is known not to be an exception.

How to use generic information

- Want to draw conclusions from generic information *as long as nothing indicates an exception*.
- If additional information tells us something is abnormal, retract former conclusion.

  $\Rightarrow$ *Conclusions do not grow monotonically with premises.*
- Classical logic cannot model this, as it is monotonic:

$$X \subseteq Y \Rightarrow Th(X) \subseteq Th(Y).$$

- Why? q follows from X if q holds in all models of X. Models of Y a subset, thus q holds in all of them as well.
- Observation led to the AI field of nonmonotonic reasoning, active for over 30 years.

How to use generic information

- Want to draw conclusions from generic information *as long as nothing indicates an exception*.
- If additional information tells us something is abnormal, retract former conclusion.

  $\Rightarrow$ *Conclusions do not grow monotonically with premises.*
- Classical logic cannot model this, as it is monotonic:

$$X \subseteq Y \Rightarrow Th(X) \subseteq Th(Y).$$

- Why? q follows from X if q holds in all models of X. Models of Y a subset, thus q holds in all of them as well.
- Observation led to the AI field of nonmonotonic reasoning, active for over 30 years.

How to use generic information

- Want to draw conclusions from generic information *as long as nothing indicates an exception*.
- If additional information tells us something is abnormal, retract former conclusion.

  $\Rightarrow$ *Conclusions do not grow monotonically with premises.*
- Classical logic cannot model this, as it is monotonic:

$$X \subseteq Y \Rightarrow Th(X) \subseteq Th(Y).$$

- Why? q follows from X if q holds in all models of X. Models of Y a subset, thus q holds in all of them as well.
- Observation led to the AI field of nonmonotonic reasoning, active for over 30 years.

How to use generic information

- Want to draw conclusions from generic information *as long as nothing indicates an exception*.
- If additional information tells us something is abnormal, retract former conclusion.

  $\Rightarrow$ *Conclusions do not grow monotonically with premises.*
- Classical logic cannot model this, as it is monotonic:

$$X \subseteq Y \Rightarrow Th(X) \subseteq Th(Y).$$

- Why? q follows from X if q holds in all models of X. Models of Y a subset, thus q holds in all of them as well.
- Observation led to the AI field of nonmonotonic reasoning, active for over 30 years.

Conflicting Defaults

- Defaults may give rise to conflicting conclusions:

  (1) Quakers normally are pacifists.
  (2) Republicans normally are not pacifists.
  (3) Nixon is a quaker and a republican.

- (1) and (2) conflicting.
- Nothing wrong with the defaults!
- Different approaches to deal with this:
  - some apply none of the conflicting defaults,
  - most generate different acceptable belief sets (extensions) leave open whether to use them sceptically (p true in all of them) or credulously (p true in some of them, or in a particular one).

Conflicting Defaults

- Defaults may give rise to conflicting conclusions:

  (1) Quakers normally are pacifists.
  (2) Republicans normally are not pacifists.
  (3) Nixon is a quaker and a republican.

- (1) and (2) conflicting.
- Nothing wrong with the defaults!
- Different approaches to deal with this:
  - some apply none of the conflicting defaults,
  - most generate different acceptable belief sets (extensions) leave open whether to use them sceptically (p true in all of them) or credulously (p true in some of them, or in a particular one).

2. The Closed World Assumption

- Check the course time table
  - *Question: Is the course on Knowledge Representation on Friday?*
  - *Your answer (presumably): No*
- Why is this answer correct?
- Does not follow from the explicit information in the time table
- But: follows from this information *assuming that the list of courses is complete*
- You (presumably) used this assumption, and do so in many everyday contexts

2. The Closed World Assumption

- Check the course time table
  - *Question: Is the course on Knowledge Representation on Friday?*
  - *Your answer (presumably): No*
- Why is this answer correct?
- Does not follow from the explicit information in the time table
- But: follows from this information *assuming that the list of courses is complete*
- You (presumably) used this assumption, and do so in many everyday contexts

2. The Closed World Assumption

- Check the course time table
  - *Question: Is the course on Knowledge Representation on Friday?*
  - *Your answer (presumably): No*
- Why is this answer correct?
- Does not follow from the explicit information in the time table
- But: follows from this information *assuming that the list of courses is complete*
- You (presumably) used this assumption, and do so in many everyday contexts

The Closed World Assumption, ctd.

- In many situations way more negative than positive facts.
- Communication convention: represent the latter only, leave the former implicit.
  - train/flight schedules
  - TV programs
  - library catalogues
  - list of lectures
- Know how to infer negative information based on completeness assumption.

The Closed World Assumption, ctd.

- In many situations way more negative than positive facts.
- Communication convention: represent the latter only, leave the former implicit.
  - train/flight schedules
  - TV programs
  - library catalogues
  - list of lectures
- Know how to infer negative information based on completeness assumption.

The Closed World Assumption, ctd.

- In many situations way more negative than positive facts.
- Communication convention: represent the latter only, leave the former implicit.
  - train/flight schedules
  - TV programs
  - library catalogues
  - list of lectures
- Know how to infer negative information based on completeness assumption.

Reiter's formalization

- Let *KB* be a set of formulas, define new form of entailment under CWA:

$$KB \models_c \alpha \text{ iff } KB \cup Negs \models \alpha$$

  where $Negs = \{\neg p \mid p \text{ atomic and } KB \not\models p\}$

- $\models_c$ nonmonotonic, for instance $\{a\} \models_c \neg b$ whereas $\{a, b\} \not\models_c \neg b$
- CWA makes knowledge complete: for arbitrary α (without quantifiers) we have $KB \models_c \alpha$ or $KB \models_c \neg\alpha$.
- Recursive query evaluation; queries reduced to atomic case.
- Results extend to quantified formulas if we add *domain closure assumption* (each object named by constant) and *unique names assumption* (different constants denote different objects).

Reiter's formalization

- Let *KB* be a set of formulas, define new form of entailment under CWA:

$$KB \models_c \alpha \text{ iff } KB \cup Negs \models \alpha$$

  where $Negs = \{\neg p \mid p \text{ atomic and } KB \not\models p\}$

- $\models_c$ nonmonotonic, for instance $\{a\} \models_c \neg b$ whereas $\{a, b\} \not\models_c \neg b$

- CWA makes knowledge complete: for arbitrary α (without quantifiers) we have $KB \models_c \alpha$ or $KB \models_c \neg\alpha$.

- Recursive query evaluation; queries reduced to atomic case.

- Results extend to quantified formulas if we add *domain closure assumption* (each object named by constant) and *unique names assumption* (different constants denote different objects).

Reiter's formalization

- Let *KB* be a set of formulas, define new form of entailment under CWA:

$$KB \models_c \alpha \text{ iff } KB \cup Negs \models \alpha$$

  where $Negs = \{\neg p \mid p \text{ atomic and } KB \not\models p\}$

- $\models_c$ nonmonotonic, for instance $\{a\} \models_c \neg b$ whereas $\{a, b\} \not\models_c \neg b$
- CWA makes knowledge complete: for arbitrary α (without quantifiers) we have $KB \models_c \alpha$ or $KB \models_c \neg\alpha$.
- Recursive query evaluation; queries reduced to atomic case.
- Results extend to quantified formulas if we add *domain closure assumption* (each object named by constant) and *unique names assumption* (different constants denote different objects).

Reiter's formalization

- Let *KB* be a set of formulas, define new form of entailment under CWA:

$$KB \models_c \alpha \text{ iff } KB \cup Negs \models \alpha$$

  where $Negs = \{\neg p \mid p \text{ atomic and } KB \not\models p\}$

- $\models_c$ nonmonotonic, for instance $\{a\} \models_c \neg b$ whereas $\{a, b\} \not\models_c \neg b$
- CWA makes knowledge complete: for arbitrary α (without quantifiers) we have $KB \models_c \alpha$ or $KB \models_c \neg\alpha$.
- Recursive query evaluation; queries reduced to atomic case.
- Results extend to quantified formulas if we add *domain closure assumption* (each object named by constant) and *unique names assumption* (different constants denote different objects).

A major problem

- Works for simple cases only, e.g. KB a set of atoms.
- Assume $KB \models (p \vee q)$, but $KB \not\models p$ and $KB \not\models q$.
- CWA best viewed as a method for restricted contexts (e.g. databases).

Standard Reference:

Reiter, Raymond (1978). *On Closed World Data Bases*. In Gallaire, H.; Minker, J., Logic and Data Bases. Plenum Press. pp. 119-140.

A major problem

- Works for simple cases only, e.g. KB a set of atoms.
- Assume $KB \models (p \vee q)$, but $KB \not\models p$ and $KB \not\models q$.
- CWA best viewed as a method for restricted contexts (e.g. databases).

Standard Reference:

Reiter, Raymond (1978). *On Closed World Data Bases*. In Gallaire, H.; Minker, J., Logic and Data Bases. Plenum Press. pp. 119-140.

Weaker versions of CWA

- Geneneralized CWA (Minker, 1982):

$$Negs = \{ \neg p \mid p \text{ atomic and for every positive clause } C \text{ with } KB \not\models C,\ KB \not\models C \vee p\}$$

- Extended Generalized CWA (Yahya and Henschen, 1985):

$$Negs = \{ \neg K \mid K \text{ a conjunction of atoms and for every positive clause } C \text{ with } KB \not\models C,\ KB \not\models C \vee K\}$$

- Further refinements partition atoms into different groups (Careful CWA, Extended CWA). Extended CWA is equivalent to cirucmscription for proposotional logic.

Weaker versions of CWA

- Geneneralized CWA (Minker, 1982):

$$Negs = \{ \neg p \mid p \text{ atomic and for every positive clause } C \text{ with } KB \not\models C, KB \not\models C \vee p\}$$

- Extended Generalized CWA (Yahya and Henschen, 1985):

$$Negs = \{ \neg K \mid K \text{ a conjunction of atoms and for every positive clause } C \text{ with } KB \not\models C, KB \not\models C \vee K\}$$

- Further refinements partition atoms into different groups (Careful CWA, Extended CWA). Extended CWA is equivalent to cirucmscription for proposotional logic.

Weaker versions of CWA

- Geneneralized CWA (Minker, 1982):

$$Negs = \{ \neg p \mid p \text{ atomic and for every positive clause } C \text{ with } KB \not\models C,\ KB \not\models C \vee p\}$$

- Extended Generalized CWA (Yahya and Henschen, 1985):

$$Negs = \{ \neg K \mid K \text{ a conjunction of atoms and for every positive clause } C \text{ with } KB \not\models C,\ KB \not\models C \vee K\}$$

- Further refinements partition atoms into different groups (Careful CWA, Extended CWA). Extended CWA is equivalent to cirucmscription for proposotional logic.

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

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

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

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

Conclude $flies(tweety)$ 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

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

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

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

Conclude $flies(tweety)$ from single scenario.

Poole, ctd.

- 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.

Poole, ctd.

- 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.

Poole, ctd.

- 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.

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.

Preferred subtheories, ctd.

Example

$F_1 = \{bird(tweety), penguin(tweety)\}$

$F_2 = \{penguin(tweety) \rightarrow \neg flies(tweety)\}$

$F_3 = \{bird(tweety) \rightarrow flies(tweety)\}$

Single preferred subtheory: $F_1 \cup F_2$

$\neg flies(tweety)$ 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):

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

- 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.

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.

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.

Circumscription, ctd.

Example

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

Models:

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

- 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

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

Models:

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

- 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

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

Models:

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

- 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

- 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 α 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 α 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 α 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

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

Models:

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

- Both M_1 and M_2 are as normal as possible.
- *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 no longer a model.

- Both M_1 and M_2 are as normal as possible.
- *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

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

Models:

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

$$P \leq Q \text{ for } \forall x.P(x) \rightarrow Q(x)$$
$$P < Q \text{ for } P \leq Q \text{ and not } Q \leq P$$

$Circ(P, T(P))$, the circumscription of P in $T(P)$:

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

- 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 $Circ(Ab, T(Ab))$.

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.

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.

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

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

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

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

Obviously not intended: $\neg teaches(gerd)$ 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

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

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

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

Obviously not intended: $\neg teaches(gerd)$ 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:

$E_0 = \{p\}$; $E_1 = Th(\{p, r, s\})$; $E_2 = Th(\{p, r, s, \neg q\})$

$p : q/r$ applied to construct E_1; q 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 *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. $Th(\Gamma_\Delta(S)) = \Gamma_\Delta(S)$,
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 Δ iff E is a fixpoint of Γ_Δ.

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. $Th(\Gamma_\Delta(S)) = \Gamma_\Delta(S)$,
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 Δ iff E is a fixpoint of Γ_Δ.

Examples

D	W	Extensions
$bird : flies/flies$	$bird$	$Th(W \cup \{flies\})$
$bird : flies/flies$	$bird, peng$ $peng \rightarrow \neg flies$	$Th(W)$
$bird : flies/flies$ $peng : \neg flies/\neg flies$	$bird, peng$	$Th(W \cup \{flies\})$ $Th(W \cup \{\neg flies\})$
$bird : flies \wedge \neg peng/flies$ $peng : \neg flies/\neg flies$	$bird, peng$	$Th(W \cup \{\neg flies\})$

Results

- Extensions may not exist: $\Delta = (\{\mathit{true} : \neg a/a\}, \emptyset)$.
- Types of defaults:
  - Normal: $p : q/q$.
    Normal default theories always have extensions.
  - Supernormal: $\mathit{true} : q/q$.
    Can model Poole systems.
  - Seminormal: $\mathit{true} : p \wedge q/q$.
    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.

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: $true : p \wedge q/q$.
    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.

Results

- Extensions may not exist: $\Delta = (\{\textit{true} : \neg a/a\}, \emptyset)$.
- Types of defaults:
  - Normal: $p : q/q$.
    Normal default theories always have extensions.
  - Supernormal: $\textit{true} : q/q$.
    Can model Poole systems.
  - Seminormal: $\textit{true} : p \wedge q/q$.
    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.

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 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

$$a \leftarrow \mathrm{not}\, b, \quad b \leftarrow \mathrm{not}\, a$$

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

$$a \leftarrow \mathrm{not}\, b, \quad b \leftarrow \mathrm{not}\, a$$

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.

Answer Sets, ctd.

Definition

A (ground) normal logic program P is a collection of rules of the form

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

where A, B_i, C_j are ground atoms. $\mathrm{not}\ C$ 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 $B_1 \wedge \ldots \wedge B_n : \neg C_1, \ldots \neg C_m / A$ and strip unneeded parts off Reiter's definition $\Rightarrow$ GL-reduct.

Answer Sets, ctd.

Definition

A (ground) normal logic program P is a collection of rules of the form

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

where A, B_i, C_j are ground atoms. $\mathrm{not}\ C$ 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 $B_1 \wedge \ldots \wedge B_n : \neg C_1, \ldots \neg C_m / A$ 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 form P by

1. eliminating rules containing $\mathrm{not}\ C$ 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 form P by

1. eliminating rules containing $\mathrm{not}\ C$ 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.

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

$$q \leftarrow \mathrm{not}\, q, \mathit{body}$$

  eliminates answer sets satisfying body.
- Abbreviation: $\leftarrow \mathit{body}$.

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:
$node(v_1), ..., node(v_n), edge(v_i, v_j), \ldots$

Generate:
$col(X, r) \leftarrow node(X), \mathrm{not}\ col(X, b), \mathrm{not}\ col(X, g)$
$col(X, b) \leftarrow node(X), \mathrm{not}\ col(X, r), \mathrm{not}\ col(X, g)$
$col(X, g) \leftarrow node(X), \mathrm{not}\ col(X, r), \mathrm{not}\ col(X, b)$

Test:
$\leftarrow edge(X, Y), col(X, Z), col(Y, Z)$

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:

$at(M, T) \leftarrow meeting(M), time(T), \mathrm{not}\, \neg at(M, T)$
$\neg at(M, T) \leftarrow meeting(M), time(T), \mathrm{not}\, at(M, T)$
$in(M, R) \leftarrow meeting(M), room(R), \mathrm{not}\, \neg in(M, R)$
$\neg in(M, R) \leftarrow meeting(M), room(R), \mathrm{not}\, in(M, R)$

Meeting scheduling, test

Example, ctd.

Each meeting has assigned time and room:

$timeassigned(M) \leftarrow at(M, T)$
$roomassigned(M) \leftarrow in(M, R)$
$\leftarrow meeting(M), \text{not } timeassigned(M)$
$\leftarrow meeting(M), \text{not } roomassigned(M)$

No meeting has more than 1 time and room:

$\leftarrow meeting(M), at(M, T), at(M, T'), T \neq T'$
$\leftarrow meeting(M), in(M, R), in(M, R'), R \neq R'$

Meetings at same time need different rooms:

$\leftarrow in(M, X), in(M', X), at(M, T), at(M', T), M \neq M'$

Meetings with same person need different times:

$\leftarrow par(P, M), par(P, M'), M \neq M', at(M, T), at(M', T)$

Summary

- Presented some of the major approaches to nonmon.
- Started with motivation and simple forms.
- Sketched preferred subtheories, circumscription, default logic.
- Finally presented definition of answer sets.
- 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.
- Sketched preferred subtheories, circumscription, default logic.
- Finally presented definition of answer sets.
- 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.
- Sketched preferred subtheories, circumscription, default logic.
- Finally presented definition of answer sets.
- 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.
- Sketched preferred subtheories, circumscription, default logic.
- Finally presented definition of answer sets.
- 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.
- Sketched preferred subtheories, circumscription, default logic.
- Finally presented definition of answer sets.
- 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.
- Sketched preferred subtheories, circumscription, default logic.
- Finally presented definition of answer sets.
- 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.
- Sketched preferred subtheories, circumscription, default logic.
- Finally presented definition of answer sets.
- 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

- W. Marek and M. Truszczynski (1993). Nonmonotonic Logics: Context-Dependent Reasoning. Springer Verlag.
- G. Brewka, J. Dix, K. Konolige (1997). Nonmonotonic Reasoning - An Overview. CSLI publications, Stanford.
- D. Makinson (2005). Bridges from Classical to Nonmonotonic Logic, College Publications.
- G. Brewka, I. Niemelä, M. Truszczynski (2007). Nonmonotonic Reasoning, in: V. Lifschitz, B. Porter, F. van Harmelen (eds.), Handbook of Knowledge Representation, Elsevier, 2007, 239-284
- G. Brewka, T. Eiter, M. Truszczynski (2011). Answer set programming at a glance. Commun. ACM 54(12): 92-103

THANK YOU!

Suggested overview articles/books

- W. Marek and M. Truszczynski (1993). Nonmonotonic Logics: Context-Dependent Reasoning. Springer Verlag.
- G. Brewka, J. Dix, K. Konolige (1997). Nonmonotonic Reasoning - An Overview. CSLI publications, Stanford.
- D. Makinson (2005). Bridges from Classical to Nonmonotonic Logic, College Publications.
- G. Brewka, I. Niemelä, M. Truszczynski (2007). Nonmonotonic Reasoning, in: V. Lifschitz, B. Porter, F. van Harmelen (eds.), Handbook of Knowledge Representation, Elsevier, 2007, 239-284
- G. Brewka, T. Eiter, M. Truszczynski (2011). Answer set programming at a glance. Commun. ACM 54(12): 92-103

THANK YOU!