

Computational Logic

Davide Martinenghi

Free University of Bozen-Bolzano

Spring 2010

Nonmonotonic Reasoning

- ▶ Propositional logic and FOL both exhibit the **monotonicity property**

if $\mathcal{F} \models \mathbf{G}$ then

for all \mathbf{F} , $\mathcal{F} \cup \{\mathbf{F}\} \models \mathbf{G}$

- ▶ This kind of **exact, deductive** reasoning is very different from **common sense**

Nonmonotonic Reasoning

- ▶ Tweety
 - ▶ Tweety is a bird
 - ▶ Birds fly
 - ▶ so Tweety must fly
 - ▶ I forgot to mention that Tweety is a penguin
 - ▶ what do you make of that?
- ▶ Nixon's diamond
 - ▶ Nixon is a Republican and a Quaker
 - ▶ Quakers are pacifists
 - ▶ Republicans are not pacifists
 - ▶ Is Nixon pacifist or not?
- ▶ Defeasible solutions
 - ▶ Not all the relevant information is known
 - ▶ Yet humans draw plausible conclusions
 - ▶ But they can be invalidated by new information
 - ▶ They are called defeasible
 - ▶ For example, the principle of presumption of innocence is nonmonotonic

Nonmonotonic Reasoning

- ▶ Example: let

$$\mathcal{F} = \{ \text{bird}(\text{tweety}), \\ \forall X(\text{penguin}(X) \rightarrow \text{bird}(X)), \\ \forall X(\text{penguin}(X) \rightarrow \neg \text{fly}(X)), \\ \forall X(\text{bird}(X) \rightarrow \text{fly}(X)) \}$$

- ▶ Then

- ▶ $\mathcal{F} \models \text{fly}(\text{tweety})$, and
- ▶ $\mathcal{F} \cup \{\text{penguin}(\text{tweety})\}$ is inconsistent.

- ▶ We would expect

$$\mathcal{F} \cup \{\text{penguin}(\text{tweety})\} \models \neg \text{fly}(\text{tweety})$$

making $\text{fly}(\text{tweety})$ a **defeasible** consequence.

Nonmonotonic Reasoning

- ▶ Definition of nonmonotonicity (Minsky, 1975)
 - ▶ By nonmonotonic reasoning we understand the drawing of conclusions which may be invalidated in the light of new information
 - ▶ A logical system is nonmonotonic iff its provability relation violates the property of monotonicity.
- ▶ Classical logic formalizes truth and valid conclusions
 - ▶ Examples show that this can be inadequate for common sense reasoning
 - ▶ Nonmonotonic logics formalize rationality and plausible conclusions
- ▶ Beliefs
 - ▶ Agents must be prepared to reject a conclusion
 - ▶ Rules must include a inhibitory mechanism
 - ▶ Conclude unless there is some evidence of the contrary

Nonmonotonic Reasoning

- ▶ In commonsense reasoning it is very difficult to properly characterize all conditions that preclude some conclusions to be drawn
 - ▶ If you turn the ignition key of your car the engine will start; **unless**
 - ▶ the battery is dead
 - ▶ the car is out of petrol
 - ▶ there is a potato blocking the exhaust
- ▶ An insidious problem to be faced in AI is the so-called **frame problem**:
 - ▶ expressing a dynamical domain in logic without explicitly specifying which conditions are **not** affected by an action
 - ▶ takes its name from a common technique for cartoons, in which the moving parts are superimposed on the frame, depicting the background of the scene, which does not change

Nonmonotonic Reasoning

- ▶ The **negation-as-failure** operator in Prolog (early 70s) was the first nonmonotonic constructor in a logic-based system
- ▶ **SLDNF** calculus was introduced extending SLD resolution with negation-as-failure
- ▶ But **SLDNF** is an operational semantics dependent on the underlying search strategy in SLD, without declarative counterpart
- ▶ There is also the problem of **floundering** on resolving non-ground negative literals
- ▶ So negation-as-failure is not an appropriate semantics for nonmonotonic reasoning
- ▶ Other logical systems were needed to formalize this reasoning

Default logic

- ▶ introduced by R. Reiter in 1980
- ▶ a **default rule** is a defeasible inference rule of the form

$$\frac{F : G}{H}$$

where **F**, **G**, **H** are sentences in the language called the **prerequisite**, the **justification**, and, resp., the **consequent** of the default rule

- ▶ the interpretation of the rule is that if **F** is known, and there is no evidence that **G** is false, then **H** can be inferred
- ▶ a **normal default rule** is a default rule of the form

$$\frac{F : G}{G}$$

Default logic

- ▶ so the application of a default rule requires a consistency condition to be satisfied
- ▶ what makes this condition complex is that this consistency depends on the application or not of all default rules in the theory
- ▶ also, rules can interact in complex ways
- ▶ in order to provide a precise semantics for this logic, Reiter introduced the notion of an **extension** in place of models in classical logic
- ▶ a **default theory** is a pair (\mathcal{F}, Γ) , where \mathcal{F} is a set of FOL sentences and Γ is a set of default rules
- ▶ \mathcal{F} represents the strict or background information, and Γ represent the defeasible information

Default logic

The extension \mathbf{E} of a default theory (\mathcal{F}, Γ) is a set such that

- ▶ \mathcal{F} is contained in \mathbf{E}
- ▶ \mathbf{E} is deductively closed
- ▶ \mathbf{E} is closed under the application of default rules

From this, it can be shown that the set of extensions of a default theory is a subset of the set of models for \mathcal{F}

Default logic

- ▶ a pair of sets of sentences $(\mathcal{F}_1, \mathcal{F}_2)$ triggers a default rule $\frac{F:G}{H}$ iff

$$\mathcal{F}_1 \models F \text{ and } \mathcal{F}_2 \not\models \neg G$$

- ▶ let $\text{Cn}(\mathcal{F})$ denote the set of logical consequences of \mathcal{F}
- ▶ an extension for a default theory (\mathcal{F}, Γ) is a set of sentences

$$E = E_0 \cup E_1 \cup E_2 \cup \dots \cup E_n \cup \dots$$

where

$$E_0 = \text{Cn}(\mathcal{F})$$

$$E_{i+1} = \text{Cn}(E_i \cup \{H : \text{there is } \frac{F:G}{H} \in \Gamma \text{ which is triggered by } (E_i, E)\})$$

- ▶ the above definition is not recursive; it is a truly circular characterization of E

Default logic

- ▶ example: let $\mathcal{F} = \{F\}$ and $\Gamma = \left\{ \frac{F:G}{G} \right\}$. This theory has $\text{Cn}(\{F, G\})$ as its only extension.

Theorem

Let (\mathcal{F}, Γ) be a default theory. If Γ has only normal default rules, then the theory has at least an extension.

- ▶
- ▶ example: let $\mathcal{F} = \{F\}$ and $\Gamma = \left\{ \frac{F:G}{\neg G} \right\}$. This theory has no extension.
- ▶ example: let $\mathcal{F} = \{F\}$ and $\Gamma = \left\{ \frac{F:G}{\neg H}, \frac{F:H}{\neg G} \right\}$. This theory has two extensions $\text{Cn}(\{F, \neg H\})$ and $\text{Cn}(\{F, \neg G\})$.

Default logic

- ▶ example: consider the default theory (\mathcal{F}, Γ) :

$$\begin{aligned}\mathcal{F} = & \{ \text{bird}(\text{tweety}), \\ & (\forall X)(\text{bird}(X) \rightarrow \text{fly}(X)), \\ & (\forall X)(\text{penguin}(X) \rightarrow \text{bird}(X)), \\ & (\forall X)(\text{penguin}(X) \rightarrow \neg \text{fly}(X)) \} \\ \Gamma = & \left\{ \frac{\text{bird}(X) : \text{fly}(X)}{\text{fly}(X)} \right\}\end{aligned}$$

Then $\text{fly}(\text{tweety})$ is contained in its only extension, but it is not in the extension of

$$(\mathcal{F} \cup \{ \text{penguin}(\text{tweety}) \}, \Gamma)$$

Default logic

- ▶ these examples are enough to show that there is no “iterative” process to construct an extension
- ▶ one has to **guess** the set of sentences **E**, and then verify that it satisfies the definition
- ▶ from extensions we can define both
 - ▶ **credulous** semantics: the consequences of the default theory are the sentences in one chosen extension of the theory
 - ▶ **skeptical** semantics: the consequences of the default theory are the sentences belonging to all the extensions of the theory

Circumscription

- ▶ introduced by McCarthy in 1980
- ▶ **circumscription** selects from a FOL theory those models that minimally satisfy a given predicate
- ▶ let **F, G** be two fol formulas with the same free variables X_1, \dots, X_n , then we write **$F \leq G$** for

$$(\forall X_1, \dots, X_n)(F \rightarrow G)$$

- ▶ this notation has an intuitive reading within the semantics of FOL
- ▶ let **F** be a FOL sentence containing a predicate $p(X_1, \dots, X_n)$, and Φ a predicate variable with the same arity as **p**
- ▶ then the **circumscription of p in F** is the second order sentence

$$(\forall \Phi)(F[p(X_1, \dots, X_n) \setminus \Phi] \wedge (\Phi \leq p) \rightarrow (p \leq \Phi))$$

We write **CIRC(F; p)**

Circumscription

- ▶ this can be regarded as asserting that the only tuples that satisfy \mathbf{p} are those that have to, as long as \mathbf{F} is true
- ▶ for the semantics of circumscription we need the notion of **minimal model**
- ▶ let \mathbf{M} and \mathbf{N} be two models of sentence \mathbf{F} . We say that \mathbf{M} is a **submodel** of \mathbf{N} in \mathbf{p} , $\mathbf{M} \leq_{\mathbf{p}} \mathbf{N}$ iff \mathbf{M} and \mathbf{N} have the same domain, all other predicate symbols in \mathbf{A} besides \mathbf{p} have the same extensions, but the extension of \mathbf{p} in \mathbf{M} is included in its extension in \mathbf{N}
- ▶ a model \mathbf{M} is **minimal** in \mathbf{p} if $\mathbf{M}' \leq_{\mathbf{p}} \mathbf{M}$ only if $\mathbf{M}' = \mathbf{M}$
- ▶ minimal models don't always exist, so the circumscriptive theory may be inconsistent

Circumscription

Theorem

$\text{CIRC}(\mathbf{F}; \mathbf{p}) \models \mathbf{G}$ iff \mathbf{G} is true in all minimal models of \mathbf{F} in \mathbf{p}

- ▶ example: let $\mathbf{F} = \mathbf{p}(\mathbf{a}) \wedge \mathbf{p}(\mathbf{b}) \wedge \mathbf{p}(\mathbf{c})$, then

$$\text{CIRC}(\mathbf{F}; \mathbf{p}) \models \forall \mathbf{X}(\mathbf{p}(\mathbf{X}) \rightarrow \mathbf{X} = \mathbf{a} \vee \mathbf{X} = \mathbf{b} \vee \mathbf{X} = \mathbf{c})$$

- ▶ example: let $\mathbf{F} = \mathbf{p}(\mathbf{a}) \vee \mathbf{p}(\mathbf{b})$, then

$$\text{CIRC}(\mathbf{F}; \mathbf{p}) \models (\forall \mathbf{X})(\mathbf{p}(\mathbf{X}) \rightarrow \mathbf{X} = \mathbf{a}) \vee (\forall \mathbf{X})(\mathbf{p}(\mathbf{X}) \rightarrow \mathbf{X} = \mathbf{b})$$

Circumscription

► example: let **F** be

$$\begin{aligned} & \text{bird}(\text{tweety}) \wedge (\forall X)(\text{bird}(X) \wedge \neg \text{abnormal}(X) \rightarrow \text{fly}(X)) \\ & \wedge (\forall X)(\text{penguin}(X) \rightarrow \text{bird}(X)) \\ & \wedge (\forall X)(\text{penguin}(X) \rightarrow \neg \text{fly}(X)) \end{aligned}$$

then $\text{CIRC}(\mathbf{F}; \text{abnormal}) \models \text{fly}(\text{tweety})$ but

$\text{CIRC}(\mathbf{F} \cup \{\text{penguin}(\text{tweety})\}; \text{abnormal}) \models \neg \text{fly}(\text{tweety})$

Answer sets

- ▶ **stable models** semantics were introduced for logic programs by Gelfond and Lifschitz in 1988
- ▶ let Π be a normal logic program without negation-as-failure, and X a set of literals in the language. We say that X is **logically closed** if it is consistent, or it is the whole set of literals
- ▶ we say that X is **closed under Π** if for every rule $A \leftarrow B_1, \dots, B_n$ in Π such that $\{B_1, \dots, B_n\} \subseteq X$, then $A \in X$
- ▶ the set of **consequences** of Π , noted $Cn(\Pi)$, is the smallest set of literals in the language which is both logically closed and closed under Π .

Answer sets

- ▶ now we will extend this notion to normal programs with negation-as-failure
- ▶ let Π be a normal logic program, and X a set of literals in the language. We denote with Π^X the **reduct** of Π relative to X , i.e., the propositional logic program without negation as failure obtained by:
 - ▶ deleting from Π all rules of the form

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

with some $C_i \in X$

- ▶ replacing each remaining rule

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

with

$$A \leftarrow B_1, \dots, B_n$$

Answer sets

- ▶ since Π^X does not contain negation-as-failure, $Cn(\Pi^X)$ always exists
- ▶ we say that X is a **stable model** of Π iff $X = Cn(\Pi^X)$
- ▶ example:

$p \leftarrow \text{not } q$

$p \leftarrow \text{not } r$

$q \leftarrow$

- ▶ example:

$p \leftarrow \text{not } p$

Answer sets

▶ example:

$p \leftarrow \text{not } q$

▶ example:

$p \leftarrow \text{not } q$

$q \leftarrow \text{not } p$

Answer sets

- ▶ example:

$p \leftarrow \text{not } q$

$q \leftarrow \text{not } p$

$r \leftarrow \text{not } r$

$r \leftarrow p$

- ▶ stable model also satisfy the **chain property**, if $X \subset Y \subset Z$ and Y is a stable model of Π , then neither X nor Z are stable models of Π
- ▶ **answer sets** are a generalization of stable models for logic programs with negation-as-failure, and strong negation
- ▶ stable models, like default logic, also can be used to define **credulous** and **skeptical** semantics

The well-founded model

- ▶ introduced by van Gelder, Ross and Schlipf in 1990
- ▶ the **well-founded** model is constructed with an extension of the consequence operator T_{Π} for programs without negation-as-failure
- ▶ partial Herbrand interpretation I : a set of ground literals where not both $A \in I$ and $\neg A \in I$
- ▶ Let I be a partial Herbrand interpretation.
- ▶ A subset U of the Herbrand base is called an **unfounded set** of P with respect to I if for each $A \in U$ at least one of the following holds for every $A \leftarrow L_1, \dots, L_n \in \text{ground}(P)$:
 - ▶ Some $L \in L_1, \dots, L_n$ is false in I ;
 - ▶ Some positive literal $A \in L_1, \dots, L_n$ is in U .
- ▶ Greatest unfounded set of P wrt I : set of all ground atomic formulas which are false provided that all literals in I are true
 - ▶ denoted $F_P(I)$

The well-founded model

- ▶ Let P be a general program. The **well-founded** model of P is the least partial Herbrand interpretation I such that:
 - ▶ if $A \in T_P(I)$ then $A \in I$
 - ▶ if $A \in F_P(I)$ then $\neg A \in I$
- ▶ Atoms in $T_P(I)$ are **well-founded**
- ▶ Atoms in $F_P(I)$ are **unfounded**
- ▶ Atoms that are neither in $T_P(I)$ nor in $F_P(I)$ are **undefined**
- ▶ if we consider well-founded atoms as true, and unfounded atoms as false, then we obtain the **well-founded** three-valued model of Π
- ▶ the well-founded model exists, and is unique, for every normal program Π

The well-founded model

- ▶ The well-founded model can be characterized as the least fixed point of the operator $W_P(I) = T_P(I) \cup \neg F_P(I)$ where $\neg F_P(I)$ denotes the set $\{\neg A \mid A \in F_P(I)\}$.
- ▶ example:

$p \leftarrow \neg p$

$q \leftarrow q$

r

has the well-founded model $\{r, \neg q\}$.

The well-founded model

Theorem

Any stable model of Π includes all well-founded literals, and no unfounded literal of Π .

Theorem

The well-founded model of a normal program Π is two-valued iff it is the unique stable model of Π

- ▶ so the well-founded model is a **weaker** semantics for negation-as-failure than the stable models semantics

The well-founded model

- ▶ the advantage of the well-founded model is that it is **always defined** for every normal program
- ▶ also, it has a better **computational complexity** than stable model semantics (answering a query under the well-founded model is **P**)