Minimal Model Reasoning

- Conflicts between defaults in default logic lead to multiple extensions
- Each extension corresponds to a maximal set of non-violated defaults
- Reasoning with defaults can also be achieved by a simpler mechanism: predicate or propositional logic + minimize the number of cases where a default (expressed as a conventional formula) is violated \( \implies \) minimal models
- Notion of minimality: cardinality vs. set-inclusion

Entailment with respect to Minimal Models

**Definition**

Let \( A \) be a set of atomic propositions. Let \( \Phi \) be a set of propositional formulae on \( A \), and \( B \subseteq A \) a set (called abnormalities).

Then \( \psi \) \( B \)-minimally follows from \( \Phi \) \( (\Phi \models_B \psi) \) if \( I \models \psi \) for all interpretations \( I \) such that

- \( I \models \Phi \) and
- there is no \( I' \) such that \( I' \models \Phi \) and \( \{ b \in B \mid I' \models b \} \subset \{ b \in B \mid I \models b \} \).
Minimal Model Reasoning Example

$\Phi = \{ \text{student} \land \neg \text{ABstudent} \rightarrow \neg \text{earnsmoney}, \text{student}, \text{adult} \land \neg \text{ABadult} \rightarrow \text{earnsmoney}, \text{student} \rightarrow \text{adult} \}$

$\Phi$ has the following models:

$I_1 = \{ \text{student} \land \text{adult} \land \text{earnsmoney} \land \text{ABstudent} \land \text{ABadult} \}$

$I_2 = \{ \text{student} \land \text{adult} \land \neg \text{earnsmoney} \land \text{ABstudent} \land \text{ABadult} \}$

$I_3 = \{ \text{student} \land \text{adult} \land \text{earnsmoney} \land \text{ABstudent} \land \neg \text{ABadult} \}$

$I_4 = \{ \text{student} \land \text{adult} \land \neg \text{earnsmoney} \land \neg \text{ABstudent} \land \text{ABadult} \}$

Relation to Default Logic

We can embed propositional minimal model reasoning in the propositional default logic.

Theorem

Let $A$ be a set of atomic propositions. Let $\Phi$ be a set of propositional formulae on $A$, and $B \subseteq A$.

Then $\Phi \models_B \psi$ if and only if $\psi$ follows from $(D, W)$ skeptically, where

$$D = \{ \neg b \mid b \in B \}$$

and $W = \Phi$.

Relation to Default Logic: Proof

Proof sketch.

$\Rightarrow$: Assume there is an extension $E$ of $(D, W)$ such that $\psi \notin E$. Hence there is an interpretation $I$ such that $I \models E$ and $I \models \neg \psi$.

By the fact that there is no extension $F$ such that $E \subseteq F$, $I$ is a $B$-minimal model of $\Phi$. Hence $\psi$ does not $B$-minimally follow from $\Phi$.

$\Leftarrow$: Assume $\psi$ does not $B$-minimally follow from $\Phi$. Hence there is a $B$-minimal model $I$ of $\Phi$ such that $I \nmodels \psi$.

Define

$$E = \text{Th}(\Phi \cup \{ \neg b \mid b \in B, I \models \neg b \})$$

Now $I \models E$ and because $I \nmodels \psi$, $\psi \notin E$.

We can show that $E$ is an extension of $(D, W)$.

Because there is an extension $E$ such that $\psi \notin E$, $\psi$ does not skeptically follow from $(D, W)$.

Nonmonotonic Logic Programs: Background

- Answer set semantics: a formalization of negation-as-failure in logic programming (Prolog)
- Other formalizations: well-founded semantics, perfect-model semantics, inflationary semantics, ...
- Can be viewed as a simpler variant of default logic
- A better alternative to propositional logic in some applications
Nonmonotonic Logic Programs

Let \( A = \{a_1, \ldots, a_n\} \) be a set of propositions.

Rules:

\[
\begin{align*}
   c & \leftarrow b_1, \ldots, b_m, \text{not } d_1, \ldots, \text{not } d_k
\end{align*}
\]

where \( \{c, b_1, \ldots, b_m, d_1, \ldots, d_k\} \subseteq A \)

- Meaning similar to default logic:
  1. we have derived \( b_1, \ldots, b_m \) and
  2. cannot derive any of \( d_1, \ldots, d_k \),

  then derive \( c \).

- Rules without right-hand side (facts): \( c \leftarrow \)

- Rules without left-hand side (constraints):
  \( \leftarrow b_1, \ldots, b_m, \text{not } d_1, \ldots, \text{not } d_k \)

Answer Sets – Formal Definition

Definition

Let \( P \) be a set of rules without \( \text{not} \), \( \Delta \subseteq A \).

The closure \( \text{dcl}(P) \subseteq A \) of \( P \) is defined by iterative application of the rules in the obvious way. \( \Delta \) is an answer set of \( P \) if \( \Delta = \text{dcl}(P) \) and there is no constraint in \( P \) violated by \( \Delta \).

Definition (Reduct)

The reduct of a program \( P \) with respect to a set of atoms \( \Delta \subseteq A \) is defined as:

\[
P^\Delta := \{c \leftarrow b_1, \ldots, b_m | (c \leftarrow b_1, \ldots, b_m, \text{not } d_1, \ldots, \text{not } d_k) \in P, \{d_1, \ldots, d_k\} \cap \Delta = \emptyset\}
\]

Definition (Answer set)

\( \Delta \subseteq A \) is an answer set of \( P \) if \( \Delta \) is an answer set of \( P^\Delta \).

Examples

- \( P_1 = \{a \leftarrow, b \leftarrow a, c \leftarrow b\} \)
- \( P_2 = \{a \leftarrow b, b \leftarrow a\} \)
- \( P_3 = \{p \leftarrow \text{not } p\} \)
- \( P_4 = \{p \leftarrow \text{not } q, q \leftarrow \text{not } p\} \)
- \( P_5 = \{p \leftarrow \text{not } q, q \leftarrow \text{not } p, \leftarrow p\} \)

Complexity: existence of answer sets is NP-complete

1. Membership in NP: Guess \( \Delta \subseteq A \) (nondet. polytime), compute \( P^\Delta \), compute its closure, compare to \( \Delta \) (everything det. polytime).

2. NP-hardness: Reduction from 3SAT: an answer set exists iff clauses are satisfiable:

\[
\begin{align*}
p & \leftarrow \text{not } \hat{p} \\
\hat{p} & \leftarrow \text{not } p
\end{align*}
\]

for every proposition \( p \) occurring in the clauses, and

\[
\leftarrow \text{not } l_1', \text{not } l_2', \text{not } l_3'
\]

for every clause \( l_1 \lor l_2 \lor l_3 \), where \( l_i' = p \) if \( l_i = p \) and \( l_i' = \hat{p} \) if \( l_i = \neg p \).
Programs for Reasoning with Answer Sets

- smodels (Niemelä & Simons), dlv (Eiter et al.), ...
- Schematic input:

```
p(X) :- not q(X).
qu(X) :- not p(X).
par(a). par(b). par(a,c). par(b,d).
female(a).
forefather(X,Y) :- anc(X,Y), male(X).
```


difference to the propositional logic

- The ancestor relation is the transitive closure of the parent relation.
- Transitive closure cannot be (concisely) represented in propositional/predicate logic.

```
par(X,Y) -> anc(X,Y)
par(X,Z) ∨ anc(Z,Y) -> anc(X,Y)
```

The above formulae only guarantee that `anc` is a superset of the transitive closure of `par`.
- For transitive closure one needs the minimality condition in some form: nonmonotonic logics, fixpoint logics, ...

**Stratification**

The reason for multiple answer sets is the fact that `a` may depend on `b` and simultaneously `b` may depend on `a`.

The lack of this kind of circular dependencies makes reasoning easier.

**Definition**

A logic program `P` is stratified if `P` can be partitioned to `P = P_1 ∪ ... ∪ P_n` so that for all `i ∈ {1, ..., n}` and `(c ← b_1, ..., b_m, not d_1, ..., not d_k) ∈ P_i`,

1. there is no `not c` in `P_i` and
2. there are no occurrences of `c` anywhere in `P_1 ∪ ... ∪ P_{i-1}`.

**Theorem**

A stratified program `P` has exactly one answer set. The unique answer set can be computed in polynomial time.

**Example**

Our earlier examples with more than one or no answer sets:

```
P_3 = {p ← not p}
P_4 = {p ← not q, q ← not p}
```
Applications of Logic Programs

1. Simple forms of default reasoning (e.g., inheritance networks, see later)
2. A solution to the frame problem: instead of using frame axioms, use defaults
   \[ a_{t+1} \leftarrow a_t, \neg a_{t+1} \]
   By default, truth-values of facts stay the same.
3. deductive databases (Datalog\(^-\))
4. et cetera: Everything that can be done with propositional logic can also be done with propositional nonmonotonic logic programs.

Literature