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
- 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 \( \Phi \models_B \psi \) (\( \psi \) \( B \)-minimally follows from \( \Phi \)) if \( I \models \psi \) for all interpretations \( I \) such that \( I \models \Phi \) and there is no \( I' \) such that \( I' \models \Phi \) and \( \{ b \in B | I' \models b \} \subset \{ b \in B | I \models b \} \).
Minimal models: example

Φ = \{ student \land \neg ABstudent \rightarrow \neg earnsmoney, student, \
\quad adult \land \neg ABadult \rightarrow earnsmoney, student \rightarrow adult \ \}\}

Φ has the following models.

\begin{align*}
I_1 &\models student \land adult \land earnsmoney \land ABstudent \land ABadult \\
I_2 &\models student \land adult \land \neg earnsmoney \land ABstudent \land ABadult \\
I_3 &\models student \land adult \land earnsmoney \land ABstudent \land \neg ABadult \\
I_4 &\models student \land adult \land \neg earnsmoney \land \neg ABstudent \land ABadult
\end{align*}

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 \( \Phi \vdash_B \psi \) if and only if \( \psi \) follows from \( \langle D, W \rangle \) skeptically, where

\[
D = \{ \neg b \mid b \in B \} \quad \text{and} \quad W = \Phi.
\]

Relation to Default Logic: Proof

Proof sketch.

\( \Rightarrow \): Assume there is extension \( E \) of \( \langle D, W \rangle \) such that \( \psi \not\in E \). 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 \( E \subset F \), \( I \) 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 an \( B \)-minimal model \( I \) of \( \Phi \) such that \( I \not\models \psi \).
Define

\[
E = Th(\Phi \cup \{ \neg b \mid b \in B, I \models \neg b \}).
\]

Now \( I \models E \) and because \( I \not\models \psi \), \( \psi \not\in E \).
We can show that \( E \) is an extension of \( \langle D, W \rangle \).
Because there is an extension \( E \) such that \( \psi \not\in E \), \( \psi \) does not skeptically follow from \( \langle D, W \rangle \). \( \square \)

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 the propositional logic in some applications.
Nonmonotonic Logic Programs

- Rules $c \leftarrow b_1, \ldots, b_m, \text{not } d_1, \ldots, \text{not } d_k$
  where $\{c, b_1, \ldots, b_m, d_1, \ldots, d_k\} \subseteq A$ for a set $A = \{a_1, \ldots, a_n\}$ of propositions.
- Meaning similar to default logic: If
  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: $c \leftarrow$
- Rules without left-hand side: $\leftarrow b_1, \ldots, b_m, \text{not } d_1, \ldots, \text{not } d_k$

Answer Sets – Formal Definition

- Reduct of a program $P$ with respect to a set of atoms $\Delta \subseteq A$:
  $$P^\Delta := \{c \leftarrow b_1, \ldots, b_m | (c \leftarrow b_1, \ldots, b_m, \text{not } d_1, \ldots, \text{not } d_k) \in P, \{d_1, \ldots, d_k\} \cap \Delta = \emptyset\}$$
- The closure $\text{dcl}(P) \subseteq A$ of a set $P$ of rules without not is defined by iterative application of the rules in the obvious way.
- A set of propositions $\Delta \subseteq A$ is an answer set of $P$ iff $\Delta = \text{dcl}(P^\Delta)$.

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:
   $$p \leftarrow \text{not } \hat{p}$$
   $$\hat{p} \leftarrow \text{not } p$$
   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 $l_i = \lnot p$. 
Programs for Reasoning with Answer Sets

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

\[
\begin{align*}
p(X) & :- \neg q(X). \\
q(X) & :- \neg p(X). \\
r(a). \\
r(b). \\
r(c). \\
\text{anc}(X,Y) & :- \text{par}(X,Y). \\
\text{anc}(X,Y) & :- \text{par}(X,Z), \text{anc}(Z,Y). \\
\text{par}(a,b). \text{par}(a,c). \text{par}(b,d). \\
\text{female}(a). \\
\text{male}(X) & :- \neg \text{female}(X). \\
\text{forefather}(X,Y) & :- \text{anc}(X,Y), \text{male}(X). \\
\end{align*}
\]

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.

\[
\begin{align*}
\text{par}(X,Y) & \rightarrow \text{anc}(X,Y) \\
\text{par}(X,Z) \land \text{anc}(Z,Y) & \rightarrow \text{anc}(X,Y)
\end{align*}
\]

The above formulae only guarantee that \( \text{anc} \) is a superset of the transitive closure of \( \text{par} \).
- 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
\[
P = P_1 \cup \cdots \cup P_n
\]
so that for all \( i \in \{1, \ldots, n\} \) and
\[
(c \leftarrow b_1, \ldots, b_m, \neg d'_1, \ldots, \neg d'_k) \in P_i,
\]
1. there is no \( \neg c \) in \( P_i \) and
2. there are no occurrences of \( c \) anywhere in \( P_1 \cup \cdots \cup P_{i-1} \).

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:
\[
P_3 = \{ p \leftarrow \neg \neg p \} \\
P_4 = \{ p \leftarrow \neg q, \quad q \leftarrow \neg \neg p \}
\]
Applications of Logic Programs

1. Simple forms of default reasoning (inheritance networks)
2. A solution to the frame problem: instead of using frame axioms, use defaults
   \[ a_{t+1} \leftarrow a_t, \neg \neg a_{t+1} \]
   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