Lecture 10: Answer Set Programming

2-AIN-108 Computational Logic

Martin Baláž, Martin Homola

Department of Applied Informatics
Faculty of Mathematics, Physics and Informatics
Comenius University in Bratislava

2 Dec 2014

Example

Logic Program:

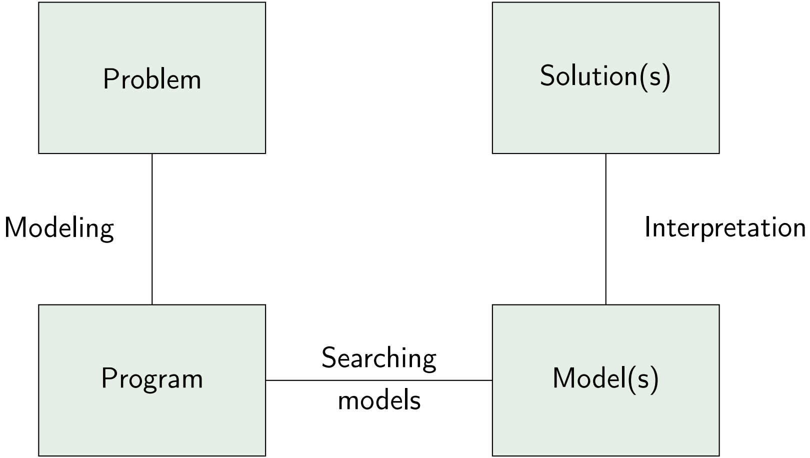
$$\begin{aligned}
father(abraham, isaac) &\leftarrow \\
mother(sarah, isaac) &\leftarrow \\
father(isaac, jacob) &\leftarrow
\end{aligned}$$

$$\begin{aligned}
parent(X, Y) &\leftarrow father(X, Y) \\
parent(X, Y) &\leftarrow mother(X, Y) \\
ancestor(X, Y) &\leftarrow parent(X, Y) \\
ancestor(X, Z) &\leftarrow parent(X, Y), ancestor(Y, Z)
\end{aligned}$$

Models:

$$M = \{parent(abraham, isaac), parent(sarah, isaac), \ldots\}$$

Answer Set Programming



Definite Logic Programs

Definition (Stable Model)

An interpretation I is a stable model of a definite logic program P iff I is the least model of P.

Fact (Existence of Stable Model)

Each definite logic program has exactly one stable model.

Default Negation

$\sim p$ is true (p is false) by default unless we prove p.

Example (One Stable Model)

$$
\begin{aligned}
student(peter) &\leftarrow \\
student(alice) &\leftarrow \\
inf(peter) &\leftarrow \\
\\
ain(X) &\leftarrow student(X), \sim inf(X)
\end{aligned}
$$

Default Negation

$\sim p$ is true (p is false) by default unless we prove p.

Example (Two Stable Models)

$$
\begin{aligned}
student(peter) &\leftarrow \\
student(alice) &\leftarrow \\
inf(peter) &\leftarrow \\
\\
ain(X) &\leftarrow student(X), \sim inf(X) \\
inf(X) &\leftarrow student(X), \sim ain(X)
\end{aligned}
$$

Example (No Stable Model)

$$p \leftarrow \sim p$$

Normal Logic Programs

Definition (Program Reduct)

Let I be an interpretation. A program reduct of a normal logic program P is a definite logic program P^I obtained from P by

- deleting all rules with a default literal L in the body not satisfied by I
- deleting all default literals from remaining rules.

Definition (Stable Model)

An interpretation I is a stable model of a normal logic program P iff I is the least model of the program reduct P^I.

Fact (Existence of Stable Model)

A normal logic program may have zero, one, or multiple stable models.

Properties

Theorem

Stable model of a normal logic program P is a model of P.

Theorem

Stable model of a normal logic program P is a minimal model of P.

Definition (Support)

A normal rule $A \leftarrow A_1, \ldots, A_m, \sim A_{m+1}, \ldots, \sim A_n$ supports an atom A (w.r.t. an interpretation I) iff $\{A_1, \ldots, A_m\} \subseteq I$ and $\{A_{m+1}, \ldots, A_n\} \cap I = \emptyset$.
An interpretation I is supported by a normal logic program P iff for each atom A in I there exists a rule r in P supporting A.

Theorem

Stable model of a normal logic program P is supported by P.

Answer Set Programming and Completion

Theorem

Each stable model of a normal logic program P is a model of $Comp(P)$.

Example

Logic Program P:

$$p \leftarrow q$$
$$q \leftarrow p$$

Completion $Comp(P)$:

$$p \leftrightarrow q$$

$\{p, q\}$ is a model of $Comp(P)$ but it is not a stable model of P.

Answer Set Programming and Completion

Definition (Tight Logic Program)

A normal logic program P is tight if there exists a mapping ℓ from the Herbrand base $\mathcal{B}$ to the set of natural numbers $\mathbb{N}$ such that for each rule $A \leftarrow A_1, \ldots, A_m, \sim A_{m+1}, \ldots, \sim A_n$ in P and each $1 \leq i \leq m$ holds $\ell(A) > \ell(A_i)$.

Theorem

Let P be a tight normal logic program. A model of $Comp(P)$ is a stable model of P.

How we can compute stable models?

Example 1:

$$
\begin{array}{lll}
p & \leftarrow & \\
r & \leftarrow & p, q \\
s & \leftarrow & p, \sim q
\end{array}
$$

Example 2:

$$
\begin{array}{lll}
p & \leftarrow & \sim q \\
q & \leftarrow & \sim p \\
q & \leftarrow & p
\end{array}
$$

How we can compute stable models?

Definition (Consistency, Totality)

A set S of literals is

- consistent iff for each atom A holds $\{A, \sim A\} \not\subseteq S$
- total iff for each atom A holds either $A \in S$ or $\sim A \in S$

Definition (Applicability)

Let S be a set of literals. A normal rule $A \leftarrow L_1, \dots, L_n$ is

- applicable iff $\{L_1, \dots, L_n\} \subseteq S$ but $A \notin S$
- conditionally applicable iff $\emptyset \subset \{L_1, \dots, L_n\} \setminus S \subseteq \sim B_P$ and $A \notin S$

How we can compute stable models?

Input: Grounded normal logic program P.
Output: Stable model of P.

1. Start with the empty set of literals.
2. Apply all applicable rules. If an inconsistency is derived, backtrack (go to the last step 3 and choose another conditionally applicable rule if exists).
3. If there exists a conditionally applicable rule, add all its preconditions to S and go to 2. Otherwise go to 4.
4. Assume all atoms with unknown value are false.
   The resulting set S is a stable model of P.