Computational Logic

Answer Set Programming

Martin Baláž

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

2011

Answer Set Program

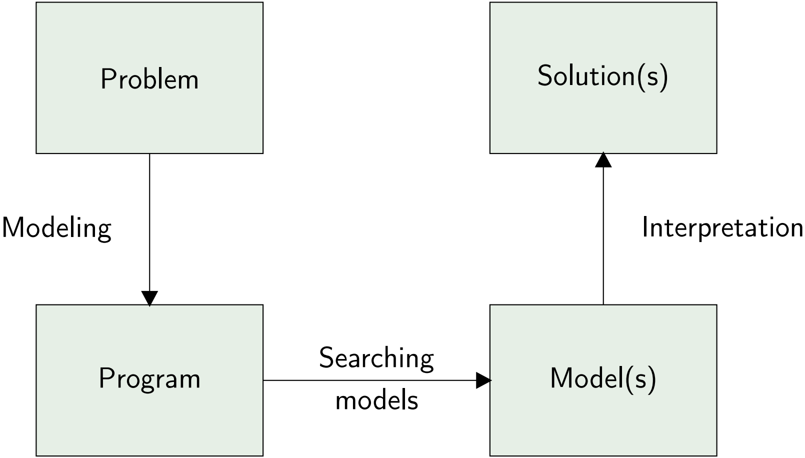
Logic Program:

$$
\begin{aligned}
father(abraham, isaac) &\leftarrow \\
mother(sarah, isaac) &\leftarrow \\
father(isaac, jacob) &\leftarrow \\
\\
parent(X, Y) &\leftarrow father(X, Y) \\
parent(X, Y) &\leftarrow mother(X, Y) \\
grandparent(X, Z) &\leftarrow parent(X, Y), parent(Y, Z) \\
ancestor(X, Y) &\leftarrow parent(X, Y) \\
ancestor(X, Z) &\leftarrow parent(X, Y), ancestor(Y, Z)
\end{aligned}
$$

Models:

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

Answer Set Programming



Semantics for Definite Logic Programs

The *stable model* of a definite logic program P is its least model M_P.

There always exists exactly one stable model of a definite logic program.

It can be computed by iterating the immediate consequence operator.

Default Negation

Closed World Assumption

Example 1:

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

Example 2:

$$a \leftarrow \sim a$$

Default Negation:
We can assume that $\sim p$ is true (p is false) by default, if we can not prove p.

Semantics for Normal Logic Programs

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.

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.

Note that

- Rules with false assumption are trivially satisfied, they are not applicable.
- Rules containing only true assumptions are applicable if their conclusion is not true (it is false or unknown).
- If a rule contains a positive assumption with unknown value, it is not applicable.
- If a rule contains some default assumptions with unknown value, but all other assumptions are true, we can assume they are false by default.

How we can compute stable models?

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

1. Start with the empty interpretation.
2. Apply all applicable rules containing only true assumptions. If an inconsistency is to be derived, backtrack.
3. If there exists a rule containing some default assumptions with unknown value, but all other assumptions are true, assume they are false and go to 2. Otherwise go to 4.
4. Assume all atoms with unknown value are false. The resulting interpretation is a stable model of P.

Example

Example 1:

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

Example 2:

$$\begin{aligned} p &\leftarrow \sim q \\ q &\leftarrow \sim p \\ q &\leftarrow p \end{aligned}$$

Properties

There are normal logic programs with no stable model.

There are normal logic programs with more stable models.

Stable models of a nomal logic program P are minimal models of P.

Stable models of a normal logic program P are models of $Comp(P)$.

Disjunctive Logic Programs

A *disjunctive rule* is a rule of the form

$$A_0 \vee \cdots \vee A_m \leftarrow L_{m+1} \wedge \cdots \wedge L_n$$

where $0 \leq m \leq n$, each A_i, $0 \leq i \leq m$ is an atom, and each L_i, $m < i \leq n$, is a literal.

A *disjunctive logic program* is a finite set of disjunctive rules. A *positive* disjunctive logic program does not contain default negation.

A *stable model* of a positive disjunctive logic prorgam P is a minimal model of P.

Positive disjunctive logic programs may have more minimal models.

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

Constraints

A *constraint* is a rule of the form

$$\leftarrow L_1 \land \cdots \land L_n$$

where $0 \leq n$ and each L_i, $1 \leq i \leq n$, is a literal.

An interpretation is a *stable model of* a logic program P with a set of constraints C iff it is a stable model of P and satisfies C.

Example

$$
\begin{gathered}
s(a,b,x,y,1) \vee s(a,b,x,y,2) \vee \cdots \vee s(a,b,x,y,9) \leftarrow \\
\leftarrow s(a,b,x,y,n_1) \wedge s(a,b,x,y,n_2) \wedge n_1 \neq n_2 \\
\leftarrow s(a,b,x_1,y_1,n) \wedge s(a,b,x_2,y_2,n) \wedge (x_1,y_1) \neq (x_2,y_2) \\
\leftarrow s(a_1,b,x_1,y,n) \wedge s(a_2,b,x_2,y,n) \wedge (a_1,x_1) \neq (a_2,x_2) \\
\leftarrow s(a,b_1,x,y_1,n) \wedge s(a,b_2,x,y_2,n) \wedge (b_1,y_1) \neq (b_2,y_2)
\end{gathered}
$$

Explicit Negation

$$\underbrace{train \qquad \rlap{\overbrace{}^{\sim train}}\emptyset}_{\sim \neg\, train} \qquad \neg\, train$$

$$cross \leftarrow \sim train$$

versus

$$cross \leftarrow \neg\, train$$

Extended Logic Programs

A *classical literal* is an atom or an atom preceded by explicit negation. A *default literal* is a classical literal preceded by default negation. A *literal* is either classical or default literal.

An *extended logic program* is a finite set of rules

$$L_1 \vee \cdots \vee L_m \leftarrow L_{m+1} \wedge \cdots \wedge L_n$$

where $0 \leq m \leq n$ and each L_i, $0 \leq i \leq n$, is a literal.

An *extended Herbrand base* is a set of ground classical literals. A set of classical literals is *coherent* if it does not contain an atom A and its explicit negation $\neg A$. A *Herbrand interpretation* is a *coherent* subset of the extended Herbrand base.

Example

$$
\begin{array}{rcl}
fly(X) & \leftarrow & bird(X) \wedge \sim \neg fly(X) \\
\neg fly(X) & \leftarrow & penguin(X) \\
bird(X) & \leftarrow & penguin(X) \\
bird(skippy) & \leftarrow & \\
penguin(tweety) & \leftarrow &
\end{array}
$$