Answer Set Programming

Quick Summary

Jozef Siska

Department of Applied Informatics
Comenius University in Bratislava

2015/2016

Language

Definition (Language of LP)

- *constants* $a, b, c, \ldots$
- *variables* $X, Y, Z, \ldots$
- *function symbols* $f, g, h, \ldots$ with arity
- *predicate symbols* $p, q, r, \ldots$ with arity
- *logical connectives* $\leftarrow$, ,
- *punctuation symbols* "(", ")", ","

Terms

Definition (Term)

A *term* is inductively defined as follows:

- A variable is a term.
- A constant is a term.
- If f is an n-ary function symbol and $t_1, \ldots, t_n$ are terms then $f(t_1, \ldots, t_n)$ is a term.

A term is said *ground* if no variable occurs in it. A *Herbrand universe* is the set of all ground terms.

Atoms

Definition (Atom)

An *atom* is defined as follows:

- A propositional variable is an atom.
- If p is an n-ary predicate symbol and $t_1, \ldots, t_n$ are terms then $p(t_1, \ldots, t_n)$ is an atom.

An atom is said *ground* if no variable occurs in it. A *Herbrand base* is the set of all ground atoms.

Rules

Definition (Literal)

A *default literal* is an atom preceded by the symbol $\sim$. A *literal* is either an atom or a default literal.

Definition (Rule)

A *rule* is a formula of the form

$$L_1, \ldots, L_m \leftarrow L_{m+1}, \ldots, L_n$$

where $L_i, 1 \leq i \leq n$ are literals.
The formula $L_1, \ldots, L_m$ is called the *head* and the formula $L_{m+1}, \ldots, L_n$ is called the *body* of a rule.
A rule with an empty body ($m = n$) and a single disjunct in the head ($m = 1$) is called a *fact*.
A rule with an empty head ($m = 0$) is called a *constraint*.

Logic Programs

Definition (Logic Program)

A *logic program* is a set of rules.
A *positive* logic program does not contain default negation. A *normal* logic program can contain default negation only in the bodies of rules. A *generalized* logic program can contain default negation also in the heads of rules. If a logic program is *disjunctive*, it can contain rules with disjunction in the head, otherwise it can not.

generalized logic program	normal logic program	positive (definite) logic program
generalized disjunctive logic program	normal disjunctive logic program	positive disjunctive logic program

Example

Example

```
man(dilbert).
single(X) :- man(X), not husband(X).
husband(X) :- man(X), not single(X).
:- single(X), husband(X).
```

$$M_1 = \{\texttt{man(dilbert)}, \texttt{husband(dilbert)}\}$$
$$M_2 = \{\texttt{man(dilbert)}, \texttt{single(dilbert)}\}$$

Positive Logic Programs

Definition (Positive Logic Program)

A *positive (definite) logic program* is a set of rules

$$A_0 \leftarrow A_1 \wedge \cdots \wedge A_n$$

where $n \geq 0$ and $A_i, 0 \leq i \leq n$ are atoms.

Example

```
edge(a, b).
...
path(X, Y) :- edge(X, Y).
path(X, Z) :- edge(X, Y), path(Y, Z).
```

Least Model

Theorem

The intersection of the Herbrand models of a positive logic program is its unique minimal Herbrand model.

Sketch of proof.

Every positive logic program has a model - the Herbrand base is a model. If M_1 and M_2 are models, then $M_1 \cap M_2$ is a model too. $\square$

Immediate Consequence Operator

Definition (Immediate Consequence Operator)

Let Π be a positive logic program. An *immediate consequence operator* is defined as follows:

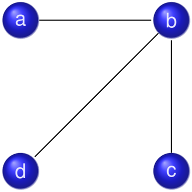
$$T_\Pi(I) = \{A \in \mathcal{B}_\Pi \mid \exists r \in \Pi : head(r) = A, I \models body(r)\}$$

$$T_\Pi \uparrow \alpha = \begin{cases} \emptyset & \text{if } \alpha = 0 \\ T_\Pi(T_\Pi \uparrow \beta) & \text{if } \alpha \text{ is a successor ordinal of } \beta \\ \bigcup_{\beta<\alpha} T_\Pi \uparrow \beta & \text{if } \alpha \text{ is a limit ordinal} \end{cases}$$

Theorem

$T_\Pi \uparrow \omega$ is the least model of a positive logic program Π.

Example



Π_D:
```
edge(a, b).
edge(b, d).
edge(c, b).
```
Π_{PS}:
```
path(X, Y) :- edge(X, Y).
path(X, Z) :- edge(X, Y), path(Y, Z).
```

$$
\begin{array}{lcl}
T_\Pi \uparrow 0 & = & \emptyset \\
T_\Pi \uparrow 1 & += & \{\texttt{edge(a, b), edge(b, d), edge(c, b)}\} \\
T_\Pi \uparrow 2 & += & \{\texttt{path(a, b), path(b, d), path(c, b)}\} \\
T_\Pi \uparrow 3 & += & \{\texttt{path(a, d), path(c, d)}\} \\
T_\Pi \uparrow 4 & = & T_\Pi \uparrow 3 \\
 & \ldots & \\
T_\Pi \uparrow \omega & = & T_\Pi \uparrow 3
\end{array}
$$

Example

Example

```
p(0).
p(f(X)) :- p(X).
```

$$
\begin{array}{lll}
T_\Pi \uparrow 0 & = & \emptyset \\
T_\Pi \uparrow 1 & += & \{\texttt{p(0)}\} \\
T_\Pi \uparrow 2 & += & \{\texttt{p(f(0))}\} \\
T_\Pi \uparrow 3 & += & \{\texttt{p(f(f(0)))}\} \\
T_\Pi \uparrow 3 & += & \{\texttt{p(f(f(f(0))))}\} \\
 & \ldots & \\
T_\Pi \uparrow \omega & = & \{\texttt{p(0), p(f(0)), p(f(f(0))), ...}\}
\end{array}
$$

Normal Logic Programs

Definition (Normal Logic Program)

A *normal logic program* is a set of rules

$$A_0 \leftarrow L_1 \wedge \cdots \wedge L_n$$

where $n \geq 0$, A_0 is an atom, and $L_i, 1 \leq i \leq n$, are literals.

Example

```
man(dilbert).

single(X) :- man(X), not husband(X).
husband(X) :- man(X), not single(X).
```

Immediate Consequence Operator

$$T_\Pi(I) \quad = \quad \{A \in \mathcal{B}_\Pi \mid \exists r \in \Pi : head(r) = A, I \models body(r)\}$$

Example

```
a :- not b.
b :- not a.
```

$$\begin{array}{lcl} T_\Pi \uparrow 0 & = & \emptyset \\ T_\Pi \uparrow 1 & = & \{\texttt{a, b}\} \\ T_\Pi \uparrow 2 & = & \emptyset \\ T_\Pi \uparrow 3 & = & \{\texttt{a, b}\} \\ & \cdots & \end{array}$$

Default Negation

Example

```
fly(X) :- bird(X), not ab(X).
ab(X) :- penguin(X).
bird(X) :- penguin(X).

bird(tweety).
penguin(skippy).
```

Stable Model

Definition (Reduct)

Let I be an interpretation. A *reduct* of a normal logic program Π (denoted by Π^I) is a positive logic program obtained from Π by deleting

- rules containing a default literal $L, I \not\models L$
- default literals $L, I \models L$ from remaining rules

Definition (Stable Model)

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

Example

Example

```
fly(X) :- bird(X), not ab(X).
ab(X) :- penguin(X).
bird(X) :- penguin(X).
bird(tweety).
penguin(skippy).
```

$$M = \{\texttt{bird(tweety)},\ \texttt{penguin(skippy)},\ \texttt{bird(skippy)},\ \texttt{ab(skippy)},\ \texttt{fly(tweety)}\}$$

Example

```
a :- not a.
```

Positive Disjunctive Logic Programs

Definition (Positive Disjunctive Logic Program)

A *positive disjunctive logic program* is a set of rules

$$A_1 \vee \cdots \vee A_m \leftarrow A_{m+1} \wedge \cdots \wedge A_n$$

where $n \geq m \geq 1$ and $A_i, 1 \leq i \leq n$, are atoms.

Example

```
man(dilbert).

single(X) v husband(X) :- man(X).
```

Minimal Models

Theorem

Every positive disjunctive logic program has a Herbrand model.

Sketch of Proof.

The Herbrand base is a model of a positive disjunctive logic program. □

Properties

There exist more minimal models of Π^I.

Example

```
man(dilbert).

single(X) v husband(X) :- man(X).
```

$$M_1 = \{\texttt{man(dilbert), single(dilbert)}\}$$
$$M_2 = \{\texttt{man(dilbert), husband(dilbert)}\}$$

Normal Disjunctive Logic Programs

Definition (Normal Disjunctive Logic Program)

A *normal disjunctive logic program* is a set of rules

$$A_1 \vee \cdots \vee A_m \leftarrow L_{m+1} \wedge \cdots \wedge L_n$$

where $n \geq m \geq 1$ and $A_i, 1 \leq i \leq m$, are atoms, $L_i, m < i \leq n$ are literals.

Example

```
man(dilbert).

single(X) v husband(X) :- man(X).
```