The Syntax and Semantics of A-Prolog

In this section we give a mathematical introduction to a knowledge representation language A-Prolog.

The language will be used to illustrate many of the theoretical ideas of AI and at the same time serve as a basic for the development of various applications.

Despite the simplicity of the language understanding of this material will require certain level of mathematical sophistication.

Declarative Languages - basic idea

• A declarative program (DP) is a collection of statements describing objects of a domain and their properties.

• Semantics defines a notion of a model of a DP (i.e. a possible state of the world compatible with the DP statements) and characterizes the collection of valid consequences of a program.

• Various tasks are reduced to finding models or computing consequences of a DP.

• Models are found and/or consequences are computed by general purpose reasoning algorithms often called inference engines.

Declarative Languages - basic terminology

SIGNATURE is a four-tuple

$$\sigma = \langle \mathcal{O}, \mathcal{F}, \mathcal{P}, \mathcal{V} \rangle$$

of (disjoint) sets.

Elements of $\mathcal{O}, \mathcal{F}, \mathcal{P}$ are called Object, Function, and Predicate symbols (or constants) respectively. Predicate constants are used to name relations between the domain's objects, Each function and predicate symbol is associated with its arity - an integer indicating the number of symbol's parameters. Normally, arity will be determined from the context.

Elements of $\mathcal{V}$ are called (object) Variables.

Declarative Languages - basic terminology

TERMS (over σ) are defined as follows:

1. Variables and object constants are terms.

2. If $t_1, \ldots, t_n$ are terms and $f \in \mathcal{F}$ then $f(t_1, \ldots, t_n)$ is a term. Terms not containing variables are called Ground. They are used to name the domain's objects.

ATOM is an expression of the form $p(t_1, \ldots, t_n)$ where $p \in \mathcal{P}$ and $t_1, \ldots, t_n$ are terms. If t's are ground then $p(t_1, \ldots, t_n)$ says that objects denoted by $t_1, \ldots, t_n$ satisfy property p.

LITERAL is an atom, $p(t_1, \ldots, t_n)$, or its negation, $\neg p(t_1, \ldots, t_n)$.

The syntax of A-Prolog

• A program Π of A-Prolog (sometimes called a knowledge base) consists of a signature σ and a collection of rules of the form (1):

$$l_0 \; or \; \ldots \; or \; l_i \leftarrow l_{i+1}, \ldots, l_m, \; not \; l_{m+1}, \ldots, \; not \; l_n.$$

where l's are literals of σ.

The left-hand side of a rule is called the Head and the right-hand side the Body. Both, the head and the body can be empty.

A rule with the empty head is often referred to as a Constraint. A rule with the empty body is often referred to as a Fact and written as

$$l_0 \; or \; \ldots \; or \; l_i.$$

The syntax of A-Prolog

Object, function and predicate symbols of σ are denoted by identifiers starting with small letters. Variables are identifiers starting with the capital ones.

Variables of Π range over ground terms of σ. A rule r with variables is viewed as a set of its ground instantiations - rules obtained from r by replacing r's variables by ground terms of σ. This means that it is enough to define the semantics of ground (i.e. not containing variables) programs.

Symbol not is called Default Negation, or Negation as Failure. The disjunction or is sometimes called Epistemic Disjunction.

More notation and terminology

The following notation will be used throughout the course: Given a rule r,

$$head(r) = \{l_0, \ldots, l_i\}$$

$$pos(r) = \{l_{i+1}, \ldots, l_m\}$$

$$neg(r) = \{l_{m+1}, \ldots, l_n\}$$

Rule r can be written as $H \leftarrow B^+, B^-$ where $H = head(r)$, $B^+ = pos(r)$, and $B^- = neg(r)$.

A ground set S of literals Satisfies a rule r (Closed under r) if one of the following conditions hold:

$$pos(r) \not\subseteq S$$

$$neg(r) \cap S \neq \emptyset$$

$$head(r) \cap S \neq \emptyset$$

For example, let r be the rule

$$p(a) \leftarrow q(b), \neg t(c).$$

and let

$$S = \{\neg p(a), q(b), \neg t(c)\}$$

Let us check if S satisfies r. By definition,

$$pos(r) = \{q(b), \neg t(c)\} \quad neg(r) = \{\ \} \quad head(r) = \{p(a)\}$$

Since,

$$\{q(b), \neg t(c)\} \subset \{\neg p(a), q(b), \neg t(c)\},$$

$$\emptyset \cap \{\neg p(a), q(b), \neg t(c)\} = \emptyset, \textbf{ and}$$

$$\{p(a)\} \cap \{\neg p(a), q(b), \neg t(c)\} = \emptyset,$$

every satisfiability condition fails. Therefore, S does *not* satisfy r. There are many sets that do satisfy r including $\{\ \}, \{p(a)\}, \{p(b)\}, \{t(c)\},$ and $\{p(a), q(b), \neg t(c)\}$. For practice, check to see that this is so.

Informal semantics of A-Prolog

• Ground program Π can be viewed as a specification for the sets of beliefs to be held by a rational reasoner associated with Π. Such sets will be represented by collection of ground literals. In forming such sets the reasoner must:

1. Satisfy the rules of P.

2. Satisfy the "the rationality principle" which says: "Believe nothing you are not forced to believe".

Informal semantics of A-Prolog

Beliefs of Π are represented by sets of ground literals called Answer Sets (Stable Models) of Π, e.g., a program

$$\Pi_0 \begin{cases} p(a) \leftarrow not\ q(a). \\ p(b) \leftarrow not\ q(b). \\ q(a). \end{cases}$$

has one answer set $S_0 = \{q(a), p(b)\}$. Notice, that $q(a)$ is true in S_0 while, say, $q(b)$ is unknown.

The answer set of

$$\Pi_1 = \Pi_0 \cup \{\neg q(X) \leftarrow not\ q(X).\}$$

is $S_1 = \{q(a), \neg q(b), p(b)\}$. This time $q(b)$ is false in S_1 (while $p(a)$ is still unknown).

Defining answer sets - Part 1

Let program Π consist of rules of the form:

$$l_0 \; or \; \ldots \; or \; l_i \leftarrow l_{i+1}, \ldots, l_m. \qquad (1)$$

Answer Set of Π is a consistent set S of ground literals such that:

- **S is closed under the rules of Π;**
- **S is minimal i.e. no proper subset of S satisfies the rules of Π.**

Examples

- $p(a) \leftarrow \neg p(b).$ $\neg p(a).$

  $A = \{\neg p(a)\}$

- $p(b) \leftarrow \neg p(a).$ $\neg p(a).$

  $A = \{\neg p(a), p(b)\}$

- $p(b) \leftarrow \neg p(a).$ $p(b) \leftarrow p(a).$

  $A = \{\quad\}$

- $p(a)\ or\ p(b).$

  $A1 = \{p(a)\}$ $A2 = \{p(b)\}$

- $p(a)\ or\ p(b).$ $\leftarrow p(a)$

  $A = \{p(b)\}$

Defining answer sets - Part 2

Let Π be an arbitrary program. By Π^S we denote the program obtained from Π by

(i) removing all rules containing $not\ l$ such that $l \in S$;

(ii) removing all other premises containing not .

Definition: S is an Answer Set of Π iff S is an answer set of Π^S.

Example:

Π	Π^S	$S = \{q(a), p(b)\}$
$p(a) \leftarrow\ not\ q(a).$		
$p(b) \leftarrow\ not\ q(b).$	$p(b).$	
$q(a).$	$q(a).$	

$\{q(a), p(b)\}$ is an answer set of Π

Examples

- $\Pi_0 = \{p(a) \leftarrow \ not\ p(a).\}$

  NO ANSWER SET

- $\Pi_1 = \{p(a) \leftarrow \ not\ p(b). \qquad p(b) \leftarrow \ not\ p(a).\}$

  $A1 = \{p(a)\} \qquad A2 = \{p(b)\}$

- $\Pi_2 = \Pi_1 \cup \{\leftarrow p(b).\}$

  $A = \{p(a)\}$

- $\Pi_3 = \Pi_2 \cup \{\neg p(a).\}$

  NO ANSWER SET