Principles of Programming Languages(CS F301)

BITS Pilani
Hyderabad Campus

Prof.R.Gururaj
CS&IS Dept.

Introduction to Logic Programming(Ch.16 of T1)

BITS Pilani
Hyderabad Campus

Prof R Gururaj

Introduction to Logic Programming (Ch.16)

In logic programming methodology, the approach is to express programs in a form of *symbolic logic* and use a *logical inferencing* process to produce results.

Logic programs are declarative rather than procedural.

Which means that only the specifications of the desired results are stated rather than detailed procedures for producing them.

Programs in logic programming are collections of *facts* and *rules*.

Such a program is used by asking it questions, which it attempts to answer by consulting the facts and rules.

The programming that uses a form of symbolic logic as programming language is called as *Logic Programming*, and the languages based on symbolic logic are called *Logic programming languages* OR *declarative languages*.

We discuss *Prolog* which is a widely used logic programing language.

The syntax of Logic programming is remarkably different from that of imperative and functional languages.

Semantics is also different.

A *proposition* can be thought as a logical statement that may or may not be true.

Formal language was developed to provide a method for describing propositions, with goal of allowing those formally stated propositions to be checked for validity.

Symbolic logic can be used for three different needs:

1. To express propositions.
2. To express relationships between propositions.
3. To describe how propositions can be inferred from other propositions that are assumed o be true.

The particular form of symbolic logic that is used for logic programming is called *first-order-logic* (usually referred to as predicate logic).

Propositions

The simple proposition – called a atomic proposition consists of compound terms.

A compound term is one element of a mathematical relation, written in a form that has the appearance of mathematical function notation .

Ex: man(James)

A mathematical function is a mapping which can be represented as an expression, a table or list of tuples.

A compound term is composed of two parts- a functor, which is a function symbol which names the relation, and an ordered list of parameters, which together represents an element of the relation.

A compound with a single parameter is called a 1-tuple,

Man(James)

Man(Gill)

with two parameters called a 2-tuple

Likes(Bill, Coffee)

etc.

Now the relation *Man* has two elements- {James, Gill}

Constants- Man, Likes, James, Gill, Bill, Coffee

Compound propositions:

$a \cap b \supset c$

Meaning: *a* and *b* implies *c*.

Variables can occur in propositions only when introduced by special symbols called quantifiers.

Predicate calculus
Logical connectors

Name	*Symbol*	*Example*	*Meaning*
negation	$\neg$	$\neg a$	not *a*
conjunction	$\cap$	$a \cap b$	*a* and *b*
disjunction	$\cup$	$a \cup b$	*a* or *b*
equivalence	$\equiv$	$a \equiv b$	*a* is equivalent to *b*
implication	$\supset$	$a \supset b$	*a* implies *b*
	$\subset$	$a \subset b$	*b* implies *a*

$a \cap b \supset c$
$a \cap \neg b \supset d$

The $\neg$ operator has the highest precedence. The operators $\cap$, $\cup$, and $\equiv$ all have higher precedence than $\supset$ and $\subset$ So, the second example is equivalent to

$(a \cap (\neg b)) \supset d$

Universal quantifier:

Existential quantifier:

Name	*Example*	*Meaning*
universal	$\forall X.P$	For all X, P is true.
existential	$\exists X.P$	There exists a value of X such that P is true.

Clausal form propositions.

Antecedent (RHS)

Consequent (LHS)

Because it is the consequence of the truth of the antecedent.

Likes(Bob, Trout) $\subset$ Likes(Bob, Fish) $\cap$ Fish(Trout)

Prolog

Alain Colmerauer and *Phillippe Roussel* developed the fundamental design for prolog in mid 1970s.

Around 1980 it became popular in USA, Europe, Japan for developing AI and Logic programs.

In 1990 it lost its charm.

Due to its deficiencies it could not comeback as a popular language. (Sec.7 of Ch. 16)

Prolog consists of collection of statements.

A prolog *term* is- constant, a variable or a structure.

A variable is a string of letter/digit/under-score.

Binding a value to variable and thus a type is called *instantiation*.

A variable not assigned any value is said to be *uninstatiated*.

Structure: represents atomic proposition.

Functor(parameter list)

Functor-an atom used to identify the structure;

Parameter List- can be list of atoms.

Fact Statements

Female(Shelley)

Male(Jake)

Father(Bill, Jake)

These simple structures state certain facts.

Prolog propositions have no intrinsic meaning.

They mean whatever the programmer wants them to mean.

Rule statements

Conjunction specified by comma ‘,’. The Disjunction by semicolon ‘;’.

Headed horn clause statement (Rules):

Consequent :- antecedent

Ex:

Ancestor(Mary, Shelly) :- Mother(Mary, Shelly)

Variables in prolog:

Parent(X, Y) :- Mother(X,Y)

Is read as if there are instantiations of X and Y such that Mother(X,Y) is true then for those same instantiations of X and Y, Parent(X,Y) is also true.

Use of variables in Prolog:

```
parent(X, Y) :- mother(X, Y).
parent(X, Y) :- father(X, Y).
grandparent(X, Z) :- parent(X, Y) , parent(Y, Z).
```

These statements give rules of implication among some variables, or universal objects. In this case, the universal objects are `X`, `Y`, and `Z`. The first rule states that if there are instantiations of `X` and `Y` such that `mother(X, Y)` is true, then for those same instantiations of `X` and `Y`, `parent(X, Y)` is true.

Goal statement

Queries are called Goals.

When a goal is a compound proposition, each of the facts (structures) is called a subgoal.

Headless horn clause

Man(fred)

To which the system will respond saying YES or NO.

If yes, the system had proved the goal was true under the given database or facts and relationships.

Father(Bob). Man(X):-Father(X).

Man(Bob)

Instantiate temporary X with Bob.

To use numeric computation in Prolog.

We have average speed of automobiles participated in a race, and the time they were in race.

This basic information can be coded as facts, and the relationship between speed, time, and distance can be written as a rule, as in the following:

```
speed(ford, 100).
speed(chevy, 105).
speed(dodge, 95).
speed(volvo, 80).
time(ford, 20).
time(chevy, 21).
time(dodge, 24).
time(volvo, 24).
distance(X, Y) :- speed(X, Speed),
                  time(X, Time),
                  Y is Speed * Time.
```

Now, queries can request the distance traveled by a particular car. For example, the query

```
distance(chevy, Chevy_Distance).
```

instantiates `Chevy_Distance` with the value 2205. The first two clauses in the right side of the distance computation statement instantiate the variables `Speed` and `Time` with the corresponding values of the given automobile functor. After satisfying the goal, Prolog also displays the name `Chevy_Distance` and its value.

Applications of Prolog

1. RDBS.

   Tables can be seen as prolog structures.

   Relationships between tables can be seen as rules.

   Goal statements are queries.

   Hence it is natural match for RDBMS.

Advantages:

(i)One single language for programming and querying.

(ii) The deductive capability is a built-in feature.

2. Expert Systems: These are computer systems designed to emulate human expertise in some particular domain.

It consists of database facts, inferencing process, some heuristics about the domain.

And friendly human computer interface.

Expert systems can also learn.

3. Natural language Processing:

Providing Natural language interfaces to computer SW, such as intelligent systems.

Language syntax can also be represented using Logic programming.

Proof procedures are like parsing.

Summary of Ch.16 (Logic Programming.)

innovate achieve lead

1. Introduction
2. Propositions
3. Introduction to Prolog
4. Facts, Rules and Goals
5. Applications