CSPC31: Principles of Programming Languages

Dr. R. Bala Krishnan
Asst. Prof.
Dept. of CSE
NIT, Trichy – 620 015
Ph: 999 470 4853
E-Mail: balakrishnan@nitt.edu

Books

- **Text Books**
  - ✓ **Robert W. Sebesta, *"Concepts of Programming Languages"*, Tenth Edition, Addison Wesley, 2012.**
  - ✓ Michael L. Scott, *"Programming Language Pragmatics"*, Third Edition, Morgan Kaufmann, 2009.
- **Reference Books**
  - ✓ Allen B Tucker, and Robert E Noonan, *"Programming Languages – Principles and Paradigms"*, Second Edition, Tata McGraw Hill, 2007.
  - ✓ R. Kent Dybvig, *"The Scheme Programming Language"*, Fourth Edition, MIT Press, 2009.
  - ✓ Jeffrey D. Ullman, *"Elements of ML Programming"*, Second Edition, Prentice Hall, 1998.
  - ✓ Richard A. O'Keefe, *"The Craft of Prolog"*, MIT Press, 2009.
  - ✓ W. F. Clocksin, C. S. Mellish, *"Programming in Prolog: Using the ISO Standard"*, Fifth Edition, Springer, 2003.

Chapters

Chapter No.	Title
1.	Preliminaries
~~2.~~	~~Evolution of the Major Programming Languages~~
3.	Describing Syntax and Semantics
4.	Lexical and Syntax Analysis
~~5.~~	~~Names, Binding, Type Checking and Scopes~~
6.	Data Types
7.	Expressions and Assignment Statements
8.	Statement-Level Control Structures
9.	Subprograms
10.	Implementing Subprograms
~~11.~~	~~Abstract Data Types and Encapsulation Constructs~~
12.	Support for Object-Oriented Programming
13.	Concurrency
14.	Exception Handling and Event Handling
15.	Functional Programming Languages
16.	Logic Programming Languages

Chapter 16 – Logic Programming Languages

Introduction

- Expresses 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 languages 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
- Programming that uses a form of symbolic logic as a programming language is often called logic programming, and languages based on symbolic logic are called logic programming languages, or declarative languages
- We have chosen to describe the logic programming language Prolog, because it is the only widely used logic language
- The syntax of logic programming languages is remarkably different from that of the imperative and functional languages
- The semantics of logic programs also bears little resemblance to that of imperative-language programs

Predicate Calculus

Universal Quantifier → $\forall X.(woman(X) \supset human(X))$

Existential Quantifier → $\exists X.(mother(mary, X) \cap male(X))$

- The first of these propositions means that for any value of X, if X is a woman, then X is a human
- The second means that there exists a value of X such that mary is the mother of X and X is a male; in other words, mary has a son

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	$=$	$a = b$	*a* is equivalent to *b*
implication	$\supset$	$a \supset b$	*a* implies *b*
	$\subset$	$a \subset b$	*b* implies *a*

The $\neg$ operator has the highest precedence. The operators $\cap$, $\cup$, and $=$ all have higher precedence than $\supset$ and $\subset$.

Clausal Form

- One problem with predicate calculus as we have described it thus far is that there are too many different ways of stating propositions that have the same meaning -> Great deal of redundancy
- This is not such a problem for logicians, but if predicate calculus is to be used in an automated (computerized) system, it is a serious problem
- To simplify matters, a standard form for propositions is desirable
- Clausal form, which is a relatively simple form of propositions, is one such standard form
- All propositions can be expressed in clausal form
- A proposition in clausal form has the following general syntax:

$$B_1 \cup B_2 \cup \ldots \cup B_n \subset A_1 \cap A_2 \cap \ldots \cap A_m$$

in which the *A's and B's are terms*

- If all of the A's are true, then at least one B is true

Primary Characteristics of Clausal Form

- Existential quantifiers are not required
- Universal quantifiers are implicit in the use of variables in the atomic propositions
- No operators other than conjunction and disjunction are required
- Also, conjunction and disjunction need appear only in the order shown in the general clausal form:
  - Disjunction on the left side and conjunction on the right side
- All predicate calculus propositions can be algorithmically converted to clausal form
- The right side of a clausal form proposition is called the antecedent
- The left side is called the consequent because it is the consequence of the truth of the antecedent

Primary Characteristics of Clausal Form

$$likes(bob, trout) \subset likes(bob, fish) \cap fish(trout)$$

- States that "if bob likes fish and a trout is a fish, then bob likes trout"

$$father(louis, al) \cup father(louis, violet) \subset$$
$$father(al, bob) \cap mother(violet, bob) \cap grandfather(louis, bob)$$

- States that "if al is bob's father and violet is bob's mother and louis is bob's grandfather, then louis is either al's father or violet's father

Overview of Logic Programming

- Languages used for logic programming are called declarative languages, because programs written in them consist of declarations rather than assignments and control flow statements

- These declarations are actually statements, or propositions, in symbolic logic

- One of the essential characteristics of logic programming languages is their semantics, which is called declarative semantics
- Prolog has two basic statement forms; these correspond to the **headless** and **headed Horn** clauses of predicate calculus

Basic Elements of Prolog

- **Fact Statements**
  - Our discussion of Prolog statements begins with those statements used to construct the hypotheses, or database of assumed information—the statements from which new information can be inferred
  - Simplest form of **headless Horn clause** in Prolog is a single structure, which is interpreted as an unconditional assertion, or fact
  - Logically, facts are simply propositions that are assumed to be true

```
female(shelley).
male(bill).
female(mary).
male(jake).
father(bill, jake).
father(bill, shelley).
mother(mary, jake).
mother(mary, shelley).
```

Basic Elements of Prolog

- **Fact Statements**
  - Our discussion of Prolog statements begins with those statements used to construct the hypotheses, or database of assumed information—the statements from which new information can be inferred

- These simple structures state certain facts about jake, shelley, bill, and mary
- For example, the first states that shelley is a female
- The last four connect their two parameters with a relationship that is named in the functor atom; for example, the fifth proposition might be interpreted to mean that bill is the father of jake
- Note that these Prolog propositions, like those
- of predicate calculus, have no intrinsic semantics
- They mean whatever the programmer wants them to mean
- For example, the proposition father(bill, jake). could mean bill and jake have the same father or that jake is the father of bill
- The most common and straightforward meaning, however, might be that bill is the father of jake

```
female(shelley).
male(bill).
female(mary).
male(jake).
father(bill, jake).
father(bill, shelley).
mother(mary, jake).
mother(mary, shelley).
```

Basic Elements of Prolog

- **Rule Statements**
- **Headed Horn** clause

consequence :- antecedent_expression.

- It is read as follows: "consequence can be concluded if the antecedent expression is true or can be made to be true by some instantiation of its variables."

ancestor(mary, shelley) :- mother(mary, shelley).

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

Goal Statements

- So far, we have described the Prolog statements for logical propositions, which are used to describe both known facts and rules that describe logical relationships among facts
- These statements are the basis for the theorem-proving model
- The theorem is in the form of a proposition that we want the system to either prove or disprove
- In Prolog, these propositions are called goals, or queries
- The syntactic form of Prolog goal statements is identical to that of headless Horn clauses
- For example, we could have man(fred)., to which the system will respond either yes or no.
  - The answer yes means that the system has proved the goal was true under the given database of facts and relationships
  - The answer no means that either the goal was determined to be false or the system was simply unable to prove it

Sample Program

- Prolog always performs depth-first-search, Matches facts & rules (i.e. knowledge base) in top-down manner and resolves the goals or subgoals in left-to-right manner
- Most important thing to keep in mind while writing prolog program - "order of writing facts & rules always matters“

```
Facts                  English meanings
food(burger).          // burger is a food
food(sandwich).        // sandwich is a food
food(pizza).           // pizza is a food
lunch(sandwich).       // sandwich is a lunch
dinner(pizza).         // pizza is a dinner

Rules
meal(X) :- food(X).    // Every food is a meal OR Anything is a meal if it is a food
```

```
Queries / Goals
?- food(pizza).            // Is pizza a food?
?- meal(X), lunch(X).      // Which food is meal and lunch?
?- dinner(sandwich).       // Is sandwich a dinner?
```

https://www.jdoodle.com/execute-prolog-online/

Sample Program

- Prolog always performs depth-first-search, Matches facts & rules (i.e. knowledge base) in top-down manner and resolves the goals or subgoals in left-to-right manner
- Most important thing to keep in mind while writing prolog program - "order of writing facts & rules always matters“

Facts **English meanings**

```
studies(charlie, csc135).      // charlie studies csc135
studies(olivia, csc135).       // olivia studies csc135
studies(jack, csc131).         // jack studies csc131
studies(arthur, csc134).       // arthur studies csc134

teaches(kirke, csc135).        // kirke teaches csc135
teaches(collins, csc131).      // collins teaches csc131
teaches(collins, csc171).      // collins teaches csc171
teaches(juniper, csc134).      // juniper teaches csc134
```

Rules

```
professor(X, Y) :-  teaches(X, C), studies(Y, C).   // X is a professor of Y if X teaches C and Y studies C.
```

Queries / Goals

```
?- food(pizza).             // Is pizza a food?
?- meal(X), lunch(X).       // Which food is meal and lunch?
?- dinner(sandwich).        // Is sandwich a dinner?
```

https://www.jdoodle.com/execute-prolog-online/

Thank You