Theory of Programming Languages

Logic Programming Languages

Sajid Anwer

Department of Computer Science,
FAST-NUCES, CFD Campus

Chapter Outline

- Introduction
- A Brief Introduction to Predicate Calculus
- Predicate Calculus and Proving Theorems
- An Overview of Logic Programming
- Applications of Logic Programming

Introduction

- Programs in logic languages are expressed in a form of *symbolic logic*

- Use a logical *inferencing* process to produce results

- *Declarative* rather that *procedural*:
  - » Only specification of *results* are stated (not detailed *procedures* for producing them)

- Proposition
  - » A logical statement that may or may not be true
    - Consists of objects and relationships of objects to each other

Symbolic Logic

- Logic which can be used for the basic needs of formal logic:
  - Express propositions
  - Express relationships between propositions
  - Describe how new propositions can be inferred from other propositions
- Particular form of symbolic logic used for logic programming called *predicate calculus*

Object Representation

- *Objects* in propositions are represented by simple terms: either constants or variables

- *Constant*: a symbol that represents an object

- *Variable*: a symbol that can represent different objects at different times

  - » Different from variables in imperative languages

Compound Terms

- *Atomic propositions* consist of compound terms

- *Compound term*: one element of a mathematical relation, written like a mathematical function
  - » Mathematical function is a mapping
  - » Can be written as a table

- Compound term composed of two parts
  - » Functor: function symbol that names the relationship
  - » Ordered list of parameters (tuple)

  - » Examples:

```
student(jon)
like(seth, OSX)
like(nick, windows)
like(jim, linux)
```

Forms of a Proposition

- Propositions can be stated in two forms:
  - *Fact*: proposition is assumed to be true
  - *Query*: truth of proposition is to be determined
- Compound proposition:
  - Have two or more atomic propositions
  - Propositions are connected by operators

Logical Operators

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	$\supset$ $\subset$	a $\supset$ b a $\subset$ b	a implies b b implies a

Quantifiers

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

$$\forall X.(woman(X) \supset human(X))$$

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

Clausal Form

- Too many ways to state the same thing
- Use a standard form for propositions
- *Clausal form*:
  - » $B_1 \cup B_2 \cup \ldots \cup B_n \subset A_1 \cap A_2 \cap \ldots \cap A_m$
  - » means if all the As are true, then at least one B is true
- *Antecedent*: right side
- *Consequent*: left side

Predicate Calculus and Proving Theorems

- A use of propositions is to discover new theorems that can be inferred from known axioms and theorems

- *Resolution*: an inference principle that allows inferred propositions to be computed from given propositions

Resolution

- *Unification*: finding values for variables in propositions that allows matching process to *succeed*
- *Instantiation*: assigning temporary values to variables to allow unification to succeed
- After instantiating a variable with a value, if matching fails, may need to *backtrack* and instantiate with a different value

Theorem Proving

- Basis for logic programming
- When propositions used for resolution, only restricted form can be used
- *Horn clause* - can have only two forms
  - » *Headed*: single atomic proposition on left side

    $$\text{likes(bob, trout)} \subset \text{likes(bob, fish)} \cap \text{fish(trout)}$$

  - » *Headless*: empty left side (used to state facts)

    $$\text{father(bob, jake)}$$

- Most propositions can be stated as Horn clauses

Overview of Logic Programming

- Declarative semantics
  - » There is a simple way to determine the meaning of each statement
  - » Simpler than the semantics of imperative languages
- Programming is nonprocedural
  - » Programs do not state how a result is to be computed, but rather the form of the result

Example: Sorting a List

- Describe the characteristics of a sorted list, not the process of rearranging a list

sort(old_list, new_list) $\subset$ permute (old_list, new_list) $\cap$ sorted (new_list)

sorted (list) $\subset$ $\forall_j$ such that $1 \leq j < n$, list(j) $\leq$ list (j+1)

Applications of Logic Programming

- Relational database management systems
- Expert systems
- Natural language processing