CS 603: Programming Languages

Lecture 25

Spring 2004

Department of Computer Science
University of Alabama
Joel Jones

Prolog

- PROgramming in LOGic
- Algorithm = Logic + Control
- Logic = relation R(I,O) between input I and output O
- Control = method of searching for O that satisfies R(I,O), given input I
- E.g. Find X and Y such that
  - 3*X+2*Y=1
  - X-Y=4
- E.g. find array B such that
  - elements in B are the same as those in A
  - elements of B are in non-descending order

What is Prolog

- Prolog is a 'typeless' language with a very simple syntax.
- Prolog is declarative: you describe the relationship between input and output, not how to construct the output from the input ("specify what you want, not how to compute it")
- Prolog uses a subset of first-order logic

Classical First-Order Logic

- simplest form of logical statements is an atomic formula. e.g.
  - man(tom)
  - woman(mary)
  - married(tom,mary)
- More complex formulas can be built up using logical connectives: □, □, ¬, , □X, □X

Pair Up:
- Define each of these symbols

Examples of First Order Logic

- smart(tom) ∨ dumb(tom)
- smart(tom) ∨ tall(tom)
- ¬ dumb(tom)
- ∃ X married(tom,X)
- ∃ X loves(tom,X)
- ∃ X [married(tom,X) ∧ female(X) ∧ human(X)]
- rich(tom) ∨ ¬ smart(tom)
- ∃ X mother(john,X)
- ∀ X ∀ Y [mother(john,X) ∧ mother(john,Y) → Y=X]

Note: A B B ¬A

Horn Rules

- Logic programming is based on formulas called Horn rules. These have the form

$$\forall x_1,\ldots,x_k\,[A \leftarrow B_1 \wedge B_2 \wedge \ldots \wedge B_j]$$

- Examples:

$\forall$ X,Y[A(X) $\leftarrow$ B(X,Y) $\wedge$ C(Y)]

$\forall$ X[A(X) $\leftarrow$ B(X)]

$\forall$ X[A(X,d) $\leftarrow$ B(X,e)]

A(c,d) $\leftarrow$ B(d,e)

$\forall$ X A(X)

$\forall$ X A(X,d)

A(c,d)

Horn Rules (cont.)

- Note that atomic formulas are also Horn rules, often called facts.
- A set of Horn rules is called a Logic Program.

Logical Inference with Horn Rules

- Logic programming is based on a simple idea: From rules and facts, derive more facts.
- Example 1. Given the facts and rules:

  1. A

  2. B

  3. C

  4. $E \leftarrow A \wedge B$

  5. $F \leftarrow C \wedge E$

  6. $G \leftarrow E \wedge F$

- From 1, derive E; from 2, derive F; from 3, derive G.

Logical Inference

- Example 2: Given these facts:
  - man(plato)
  - man(socrates)
- and this rule:
- $\forall X$ [mortal(X) $\leftarrow$ man(X)]
- derive:
- mortal(plato), mortal(socrates).

Recursive Inference

Example, given

(1) $\forall$ X[mortal(son_of(X)) $\leftarrow$ mortal(X)]

(2) mortal(plato)

derive:

mortal(son_of(plato))

(using X=plato)

mortal(son_of(son_of(plato)))

(using X=son_of(plato))

mortal(son_of(son_of(son_of(plato))))

(using X=son_of(son_of(plato)))

Prolog Notation

A rule:

$\forall X [p(X) \leftarrow (q(X) \wedge r(X))]$

is written as

$p(X) \leftarrow q(X), r(X).$

Prolog conventions:

variables begin with upper case (A, B, X, Y, Big, Small, ACE)

constants begin with lower case (a, b, x, y, plato, aristotle)

Query = list of facts with variables, e.g.

mortal(X)

sorted([5,3,4,9,2], X)

sonOf(martha,S), sonOf(george,S)

Prolog program = facts+rules+query

Prolog Syntax

< fact > → < term > .
< rule > → < term > :- < terms > .
< query > → < terms > .
< term > → < number > | < atom > | <variable>
| < atom > (< terms >)
< terms > → < term > | < term > , < terms >

Syntax

- Integers
- Atoms: user defined, supplied
  - name starts with lower case: john, student2
- Variables
  - begin with upper case: Who, X
  - '_' can be used in place of variable name
- Structures
  - student(ali, freshman, 194).
- Lists
  - [x, y, Z]
  - [Head | Tail]
    - syntactic sugar for . (Head, Tail)
  - [ ]

Constructors like student and "." are called functors in Prolog

Prolog Introduction

- /* list of facts in prolog, stored in an ascii file, 'family.pl'*/
- mother(mary, ann).
- mother(mary, joe).
- mother(sue, mary).
-
- father(mike, ann).
- father(mike, joe).
-
- grandparent(sue, ann).

Prolog Introduction (cont.)

- /* reading the facts from a file */
- ?- consult (family).
- %family compiled, 0.00 sec, 828 bytes
- Comments are either bound by "/*","*/" or any characters following the "%".
- Structures are just relationships. There are no inputs or outputs for the variables of the structures.
- The swipl documentation of the built-in predicates does indicate how the variables should be used.
  - pred(+var1, -var2, +var3).
    - + indicates input variable
    - - indicates output variable

```
/* Prolog the order of the facts and rules is the order it is searched in */
/* Variation from pure logic model  */

2 ?- father( X,Y ).

X = mike   /* italics represents computer output */
Y = ann  ;   /* I type ';' to continue searching the data base */

X = mike
Y = joe  ;

no

3 ?- father( X, joe).

X = mike ;

no
```

Rules

- parent(X ,Y) :– mother(X ,Y). /* If mother(X ,Y) then parent(X ,Y) */
- parent(X ,Y) :– father(X ,Y).

- /* Note: grandparent(sue, ann). redundant */
- /* if parent(X ,Y) and parent(Y,Z) then grandparent(X ,Z). */

Pair Up:
- Define grandparent

grandparent(X , Z) :– parent(X ,Y),parent(Y, Z).