CMSC 433 – Programming Language Technologies and Paradigms Spring 2007

Logic Programming

May 1, 2007

A Prolog Example

- Facts
  - 'It is sunny'.
  - 'It is summer'.
- Query
  - ?- 'It is sunny'.
    - Yes
- Rules
  - 'It is hot' :- 'It is summer', 'It is sunny'.
  - 'It is cold' :- 'It is winter', 'It is snowing'.
- Queries
  - ?- 'It is cold'.
    - No
  - ?- 'It is hot'.
    - Yes

Another Example

- female(amy).
- female(johnette).

- male(anthony).
- male(bruce).
- male(ogden).

- parentof(amy, johnette).
- parentof(amy, anthony).
- parentof(amy, bruce).
- parentof(ogden, johnette).
- parentof(ogden, anthony).
- parentof(ogden, bruce).

- siblingof(X,Y) :-
  parentof(Z,X),
  parentof(Z,Y), X != Y.
- brotherof(X,Y) :-
  parentof(Z,X),
  male(X),
  parentof(Z,Y),
  X != Y.
- ? – parentof(amy, Y).
- ? – parentof(X, anthony).
- ? – siblingof(X, Y).
- ? – siblingof(johnette, Y).
- ? – siblingof(johnette, bruce).

SWI Prolog

- A Prolog program consists of a database of facts and rules, and queries (questions).
  - Fact: ... .
    - 'It is sunny'.
    - male(anthony).
  - Rule: ... :- ... .
    - 'It is hot' :- 'It is summer', 'It is sunny'.
    - siblingof(X,Y) :- parentof(Z,X), parentof(Z,Y).
  - Query: ?- ... .
    - ?- 'It is hot'.
  - Variables: must begin with an upper case letter.
  - Constants: numbers, begin with lowercase letter, or enclosed in single quotes.

Important Concepts

- Unification
  - Pattern matching
- Backtracking

Lists

- Find the total cost of a list of items
  - cost(cornflakes, 230).
  - cost(cocacola, 210).
  - cost(chocolate, 250).
  - cost(crisps, 190).

  - total_cost([], 0).
  - total_cost([Item|Rest], Cost) :-
    cost(Item, ItemCost),
    total_cost(Rest, CostOfRest),
    Cost is ItemCost + CostOfRest.
- Sample query:
  - ?- *total_cost([cornflakes, crisps], X).*
- Output
  - X = 420

Complex Structures

- tree(
  - tree(empty, jack, empty),
  - fred,
  - tree(empty, jill, empty)
-).

Computing the Size of a Tree

- Size of tree = number of nodes
- The size of an empty tree is zero.
  - tree_size(empty, 0).
- The size of a non-empty tree is the size of the left sub-tree plus the size of the right sub-tree plus one for the current tree node.
  - tree_size(tree(L, _, R), Total_Size) :-
    tree_size(L, Left_Size),
    tree_size(R, Right_Size),
    Total_Size is Left_Size + Right_Size + 1.

Learning

- assert
- siblingof(X,Y) :-
  parentof(Z,X),
  parentof(Z,Y), X != Y,
  asserta(siblingof(X,Y)).
- brotherof(X,Y) :-
  parentof(Z,X),
  male(X),
  parentof(Z,Y),
  X != Y,
  asserta(brotherof(X,Y)).
- retract