EECS 3401

ASSIGNMENT 2 SOLUTIONS

Deadline March. 7, 2016

Skyler Layne (cse23170, 212166906)

PROBLEM 1.

(1) X is a grandfather of Z, if X is the father of Y and Y is a parent of Z.

(2) The father of Y is a parent of Y.

(3) The mother of Y is a parent of Y.

Given the following facts:
(4) adam is the father of beth and bill.
(5) beth is the mother of chris.
(6) bill is the father of ann.

A. Convert (1)-(6) into clauses.

Use the following predicates:

gf(X,Y) - to denote the fact that X is a grandfather of Y;
f(X,Y) - to denote the fact that X is the father of Y;
m(X,Y) - to denote the fact that X is the mother of Y;
p(X,Y) - to denote the fact that X is a parent of Y;

A.sol

(1) C1. gf(X,Y) -> (f(X,Y) & p(Y,Z))
gf(X, Z) :- f(X, Y), p(Y, Z).

(2) C2. f(X, Y) :- p(X, Y).

(3) C3. m(X, Y) :- p(X, Y).

(4) C4.1. f(adam, beth).
C4.2. f(adam, bill).

(5) C5. m(beth, chris).

(6) C6. f(bill, ann).

B. Formulate an appropriate query to solve the problem

B.sol: find all A's such that adam is the grandparent

Q. gf(adam, X).

C. Construct a complete SLD search tree

C.sol:

```
              :- gf(adam, X).
                    *
                    | C1.
                    *
          gf(X, Z) :- f(X, Y), p(Y, Z).
          :- f(adam, Y'), p(Y', X').
                    *
                    |
        **************************
       / C2, C4.1.                 \ C2, C4.2.
      *                             *
f(X, Y) :- p(X, Y).           f(X, Y) :- p(X, Y).
f(adam, beth).                f(adam, bill).
p(beth, X).                   f(bill, X).
      *                             *
      |  C5.                        | C6.
      *                             *
m(X, Y) :- p(X, Y).           f(X, Y) :- p(X, Y).
m(beth, chris).               f(bill, ann).
X = chris.                    X = ann.
```

PROBLEM 2.

Specification of merge(L1,L2,L)

```
%%%%%%%%%%%%%%%%%%%%%%%%%
%% Merge a sorted list %%
%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%
 CASE 1: If the empty lists.
```

```
%%%%
merge([],[],[]).

%%%%
 CASE 2: The lists contain 1 element
%%%%
merge([],[Y],[Y|L]) :- merge([],[],L).
merge([X],[],[X|L]) :- merge([],[],L).

%%%%
  CASE 3: The first element, X, in L1 is less than the first element, Y, in L2, add X to L a
%%%%
merge([X|L1],[Y|L2],[X|L]) :- lt(X, Y), merge(L1,[Y|L2],L).

%%%%
 CASE 4: The first element, X, in L1 is greater than or equal to the first element, Y, in L2
%%%%
merge([X|L1],[Y|L2],[Y|L]) :- merge([X|L1],L2,L).

%%%%%%%%%%%%%%%%%%%%%%%
%% Helper Predicates %%
%%%%%%%%%%%%%%%%%%%%%%%
% True if X < Y is true.
lt(X, Y) :- X<Y.
```

Specification of delete(X,L,L1)

PROBLEM 3

PROBLEM 4