CS571 AI LAB 06

11th November 2022

Ishita Singh 1901CS27
Kavya Goyal 1901CS30
Priyanka Sachan 1901CS43

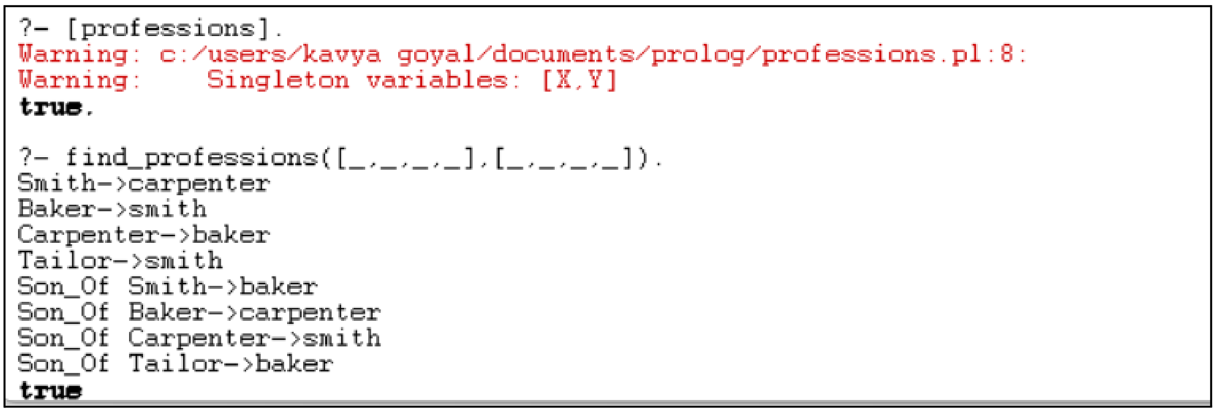
Team Code: 1901cs30_1901cs27_1901cs43
.pl Files attached for questions

1.Given Knowledge:
Smith, Baker, Carpenter, and Tailor each have a profession (smith, baker, carpenter, and tailor) but not shown by their names. Each of them has a son. But the sons also do not have the profession shown by their name. If you know that:
1) No son has the same profession as his father has and
2) Baker has the same profession as Carpenter's son has and
3) Smith's son is a baker.

Question: find the professions of the parents and sons.

```
professions([smith,baker,carpenter,tailor]). % list of professions
/* S = Smith's profession                B = Baker's profession
   C = Carpenter's profession            T = Taylor's profession
   Son_S = Smith son's profession        Son_B = Baker son's profession
   Son_C = Carpenter son's profession    Son_T = Taylor son's profession
*/
different(X,X):-!,fail.
different(X,Y).

find_professions([S,B,C,T],[Son_S,Son_B,Son_C,Son_T]):-
    professions(L),
    member(S,L),different(S,smith),
    member(B,L),different(B,baker),
    member(C,L),different(C,carpenter),
    member(T,L),different(T,tailor),

/* The sons do not have the same profession as their name shows */
    member(Son_S,L),different(Son_S,smith),
    member(Son_B,L),different(Son_B,baker),
    member(Son_C,L),different(Son_C,carpenter),
    member(Son_T,L),different(Son_T,tailor),


    /* The sons do not have the same profession as their fathers either */
    different(S,Son_S),
    different(B,Son_B),
    different(C,Son_C),
    different(T,Son_T),

    B=Son_C,         % Baker has the same profession as Carpenter's son.
    Son_S=baker,     % Smith's son is a baker

    write("Smith" -> S),nl,
    write("Baker" -> B),nl,
    write("Carpenter" -> C),nl,
    write("Tailor" -> T),nl,
    write("Son_Of Smith" -> Son_S),nl,
    write("Son_Of Baker" -> Son_B),nl,
    write("Son_Of Carpenter" -> Son_C),nl,
    write("Son_Of Tailor" -> Son_T).
```

OUTPUT



```
?- [professions].
Warning: c:/users/kavya goyal/documents/prolog/professions.pl:8:
Warning:    Singleton variables: [X,Y]
true.

?- find_professions([_,_,_,_],[_,_,_,_]).
Smith->carpenter
Baker->smith
Carpenter->baker
Tailor->smith
Son_Of Smith->baker
Son_Of Baker->carpenter
Son_Of Carpenter->smith
Son_Of Tailor->baker
true
```

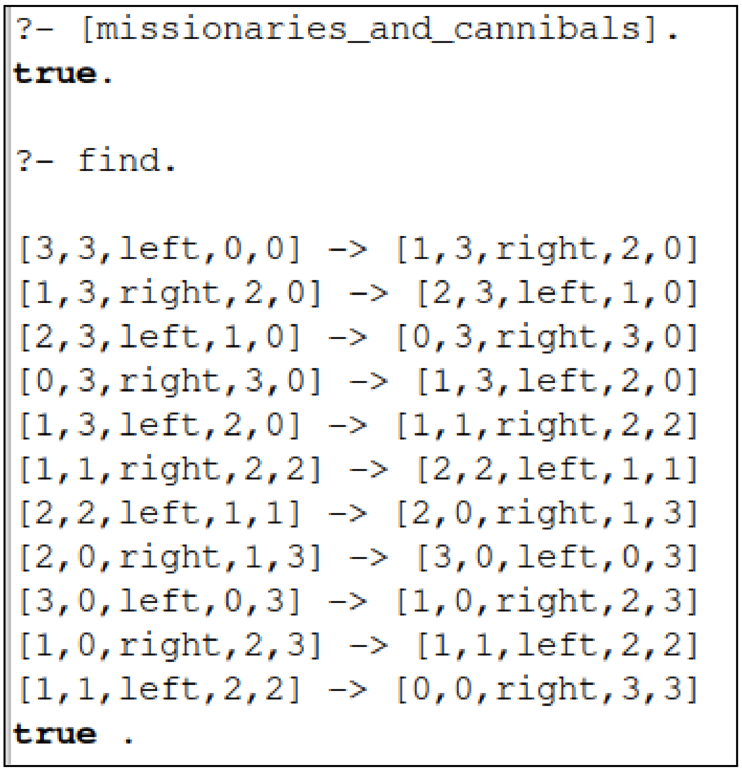
2. Given Knowledge:
On a river bank, there are 3 cannibals and 3 missionaries. Here is a boat with 2 places for crossing the river. If on a bank, there remain more cannibals than missionaries, then it is dangerous since the cannibals may eat them.

Question: How could they all cross the river without being in danger?

```
% Represent a state as
%
[#Cannibals_in_Left,#Missionaries_in_Left,Boat_Position,#Cannibals_in_Right,
#Missionaries_in_Right]
% Start state: 3 cannibals & 3 missionaries & a boat in left side
% start([3,3,left,0,0]).
% Goal state: 3 cannibals & 3 missionaries & a boat in right side
% goal([0,0,right,3,3]).

% Rule to check for valid state
valid(CL,ML,CR,MR) :-
      ML>=0, CL>=0, MR>=0, CR>=0,
      (ML>=CL ; ML=0),
      (MR>=CR ; MR=0).

% Possible moves:

% Two missionaries cross left to right.
move([CL,ML,left,CR,MR],[CL,ML2,right,CR,MR2]):-
      MR2 is MR+2,
      ML2 is ML-2,
      valid(CL,ML2,CR,MR2).

% Two cannibals cross left to right.
move([CL,ML,left,CR,MR],[CL2,ML,right,CR2,MR]):-
      CR2 is CR+2,
      CL2 is CL-2,
      valid(CL2,ML,CR2,MR).

%  One missionary and one cannibal cross left to right.
move([CL,ML,left,CR,MR],[CL2,ML2,right,CR2,MR2]):-
      CR2 is CR+1,
      CL2 is CL-1,
      MR2 is MR+1,
      ML2 is ML-1,
      valid(CL2,ML2,CR2,MR2).

% One missionary crosses left to right.
move([CL,ML,left,CR,MR],[CL,ML2,right,CR,MR2]):-
      MR2 is MR+1,
      ML2 is ML-1,
```

```prolog
	valid(CL,ML2,CR,MR2).

% One cannibal crosses left to right.
move([CL,ML,left,CR,MR],[CL2,ML,right,CR2,MR]):-
	CR2 is CR+1,
	CL2 is CL-1,
	valid(CL2,ML,CR2,MR).

% Two missionaries cross right to left.
move([CL,ML,right,CR,MR],[CL,ML2,left,CR,MR2]):-
	MR2 is MR-2,
	ML2 is ML+2,
	valid(CL,ML2,CR,MR2).

% Two cannibals cross right to left.
move([CL,ML,right,CR,MR],[CL2,ML,left,CR2,MR]):-
	CR2 is CR-2,
	CL2 is CL+2,
	valid(CL2,ML,CR2,MR).

%  One missionary and one cannibal cross right to left.
move([CL,ML,right,CR,MR],[CL2,ML2,left,CR2,MR2]):-
	CR2 is CR-1,
	CL2 is CL+1,
	MR2 is MR-1,
	ML2 is ML+1,
	valid(CL2,ML2,CR2,MR2).

% One missionary crosses right to left.
move([CL,ML,right,CR,MR],[CL,ML2,left,CR,MR2]):-
	MR2 is MR-1,
	ML2 is ML+1,
	valid(CL,ML2,CR,MR2).

% One cannibal crosses right to left.
move([CL,ML,right,CR,MR],[CL2,ML,left,CR2,MR]):-
	CR2 is CR-1,
	CL2 is CL+1,
	valid(CL2,ML,CR2,MR).


% Recursive call to solve the problem - move from state S1 to state S2.
path([CL_S,ML_S,B_S,CR_S,MR_S],[CL_G,ML_G,B_G,CR_G,MR_G],Visited,MovesList)
:-
	% Move from state S_S to state S_G.
	move([CL_S,ML_S,B_S,CR_S,MR_S],[CL_I,ML_I,B_I,CR_I,MR_I]),
	% S_I has not been visited yet.
	not(member([CL_I,ML_I,B_I,CR_I,MR_I],Visited)),
      % And get path from S_I to S_G.

path([CL_I,ML_I,B_I,CR_I,MR_I],[CL_G,ML_G,B_G,CR_G,MR_G],[[CL_I,ML_I,B_I,CR_
I,MR_I]|Visited],[ [[CL_I,ML_I,B_I,CR_I,MR_I],[CL_S,ML_S,B_S,CR_S,MR_S]] |
MovesList ]).

% If both states same, solution found.
path([CL,ML,B,CR,MR],[CL,ML,B,CR,MR],_,MovesList):-
      output(MovesList).

% Recursive call to print path.
output([]) :- nl.
output([[A,B]|MovesList]) :-
      output(MovesList),
      write(B), write(' -> '), write(A), nl.

% Find the path to move from start to goal state.
find :-
  path([3,3,left,0,0],[0,0,right,3,3],[[3,3,left,0,0]],_).
```

OUTPUT



```
?- [missionaries_and_cannibals].
true.

?- find.

[3,3,left,0,0] -> [1,3,right,2,0]
[1,3,right,2,0] -> [2,3,left,1,0]
[2,3,left,1,0] -> [0,3,right,3,0]
[0,3,right,3,0] -> [1,3,left,2,0]
[1,3,left,2,0] -> [1,1,right,2,2]
[1,1,right,2,2] -> [2,2,left,1,1]
[2,2,left,1,1] -> [2,0,right,1,3]
[2,0,right,1,3] -> [3,0,left,0,3]
[3,0,left,0,3] -> [1,0,right,2,3]
[1,0,right,2,3] -> [1,1,left,2,2]
[1,1,left,2,2] -> [0,0,right,3,3]
true .
```