Prolog Programming Assignment #1: Various Computations:

Cameron Francois
April 6th, 2023
CSC 344

Abstract:
Task 1 involves establishing and interacting with the knowledge base detailed in Prolog Lesson 1, a very simple KB pertaining to colors. Task 2 involves establishing and interacting with a very simple KB which is structurally just like the given KB of Task 1, but which you are asked to piece together yourself, one pertaining to food. Task 3, based on Prolog Lesson 3, is all about solving a map coloring problem. Task 4 involves establishing and interacting with a given KB of a bit more complexity than that featured in the first task. This is the KB about floating shapes, inspired by Terry Winograd's blocks world, that was presented in Prolog Lesson 4. Collectively, these tasks afford an opportunity to get acquainted with the basics of Prolog programming.

Task 1 - Colors KB:

Code:

```
% ------------------------------------------------------------------
% File: colors.pro
% Line: Six color facts, structured into primaries and secondaries
% ------------------------------------------------------------------
% primary(P) :: P is a primary color
primary(blue).
primary(red).
primary(yellow).
% ------------------------------------------------------------------
% secondary(S) :: S is a secondary color
secondary(green).
secondary(orange).
secondary(purple).
% ------------------------------------------------------------------
% color(C) :: C is a color
color(C) :- primary(C).
color(C) :- secondary(C).
```

Demo:

```
?- primary(blue).
ERROR: Unknown procedure: primary/1 (DWIM could not correct goal)
?- consult('colors.pl').
true.

?- primary(blue).
true.

?- primary(red).
true.

?- primary(green).
false.

?- secondary(green).
true.

?- secondary(purple).
true.

?- secondary(yellow).
false.

?- color(blue).
true

?- color(purple).
true.

?- primary(P).
P = blue ;
P = red ;
P = yellow.

?- secondary(S).
S = green ;
S = orange ;
S = purple.

?- color(C).
C = blue ;
C = red ;
C = yellow ;
C = green ;
C = orange ;
C = purple.

?- listing(secondary).
secondary(green).
secondary(orange).
secondary(purple).

true.

?- listing(primary).
primary(blue).
primary(red).
primary(yellow).

true.

?- listing(color).
color(C) :-
    primary(C).
color(C) :-
    secondary(C).

true.
```

Task 2 - Food KB:

Code:

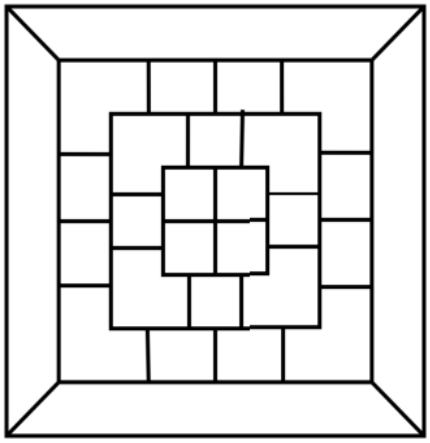
```
% File: Foods.pl
% Line: Six Food facts, structured into fruitd and vegetables
% ------------------------------------------------------------
% fruit(F) :: F is a fruit
fruit(grapefruit).
fruit(avocado).
fruit(date).
% ------------------------------------------------------------
% vegetable(v) :: V is a vegetable
vegetable(asperagus).
vegetable(broccoli).
vegetable(carrot).
% ------------------------------------------------------------
% food(F) :: F is a food
food(F) :- fruit(F).
food(F) :- vegetable(F).
```

Demo:

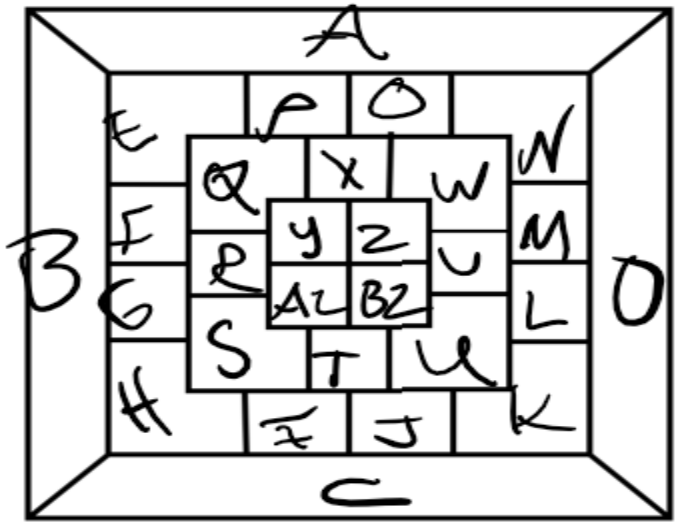
```
?- fruit(grapefruit).
ERROR: Unknown procedure: fruit/1 (DWIM could not correct goal)
?- consult('foods.pl').
true.

?- fruit(grapefruit).
true.

?- fruit(avocado).
true.

?- fruit(asperagus).
false.

?- vegetable(asperagus).
true.

?- vegetable(broccoli).
true.

?- vegetable(date).
false.

?- food(asperagus).
true.

?- food(avocado).
true .

?- food(carrot).
true.


?- fruit(F).
F = grapefruit ;
F = avocado ;
F = date.

?- vegetable(V).
V = asperagus ;
V = broccoli ;
V = carrot.

?- food(F).
F = grapefruit ;
F = avocado ;
F = date ;
F = asperagus ;
F = broccoli ;
F = carrot.

?- listing(fruit).
fruit(grapefruit).
fruit(avocado).
fruit(date).

true.

?- listing(vegetable).
vegetable(asperagus).
vegetable(broccoli).
vegetable(carrot).

true.

?- listing(food).
food(F) :-
    fruit(F).
food(F) :-
    vegetable(F).

true.
```

Task 3 - Map Coloring:

The Given Map:



The Labeled Map:



Code:

```
% ------------------------------------------------------------------------
% File: map_coloring.pro
% Line: Program to find a 4 color map rendering for map.
% More: The colors used will be red, blue, green orange.
% More: The standard abbrieviations are used to stand for the region.
% ------------------------------------------------------------------------
% different(X,Y) :: X is not equal to Y

different(red,blue).
different(red,green).
different(red,orange).
different(green,blue).
different(green,orange).
different(green,red).
different(blue,green).
different(blue,orange).
different(blue,red).
different(orange,blue).
different(orange,green).
different(orange,red).

% ---------------------------------------------
% coloring(A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,Z,A2,B2)

coloring(A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,Z,A2,B2) :-
        different(A,B),
        different(A,D),
        different(A,E),
        different(A,P),
        different(A,O),
        different(A,N),
        different(B,A),
        different(B,C),
        different(B,E),
```

```
different(B,F),
different(B,G),
different(B,H),
different(C,B),
different(C,D),
different(C,H),
different(C,I),
different(C,J),
different(C,K),
different(D,A),
different(D,C),
different(D,K),
different(D,L),
different(D,M),
different(D,N),
different(E,A),
different(E,B),
different(E,F),
different(E,P),
different(E,Q),
different(F,B),
different(F,E),
different(F,G),
different(F,R),
different(F,Q),
different(G,B),
different(G,F),
different(G,H),
different(G,R),
different(G,S),
different(H,B),
different(H,C),
different(H,G),
different(H,S),
```

```
different(H,I),
different(I,C),
different(I,H),
different(I,J),
different(I,S),
different(I,T),
different(J,C),
different(J,I),
different(J,K),
different(J,T),
different(J,U),
different(K,C),
different(K,D),
different(K,J),
different(K,L),
different(K,U),
different(L,D),
different(L,K),
different(L,M),
different(L,U),
different(L,V),
different(M,D),
different(M,L),
different(M,N),
different(M,V),
different(M,W),
different(N,D),
different(N,A),
different(N,M),
different(N,O),
different(N,W),
different(O,A),
different(O,N),
different(O,P),
```

```
different(O,W),
different(O,X),
different(P,A),
different(P,E),
different(P,O),
different(P,X),
different(P,Q),
different(Q,P),
different(Q,E),
different(Q,F),
different(Q,X),
different(Q,Y),
different(Q,R),
different(R,Q),
different(R,Y),
different(R,A2),
different(R,S),
different(R,G),
different(R,F),
different(S,R),
different(S,A2),
different(S,T),
different(S,I),
different(S,H),
different(S,G),
different(T,A2),
different(T,B2),
different(T,S),
different(T,U),
different(T,I),
different(T,J),
different(U,B2),
different(U,V),
different(U,T),
```

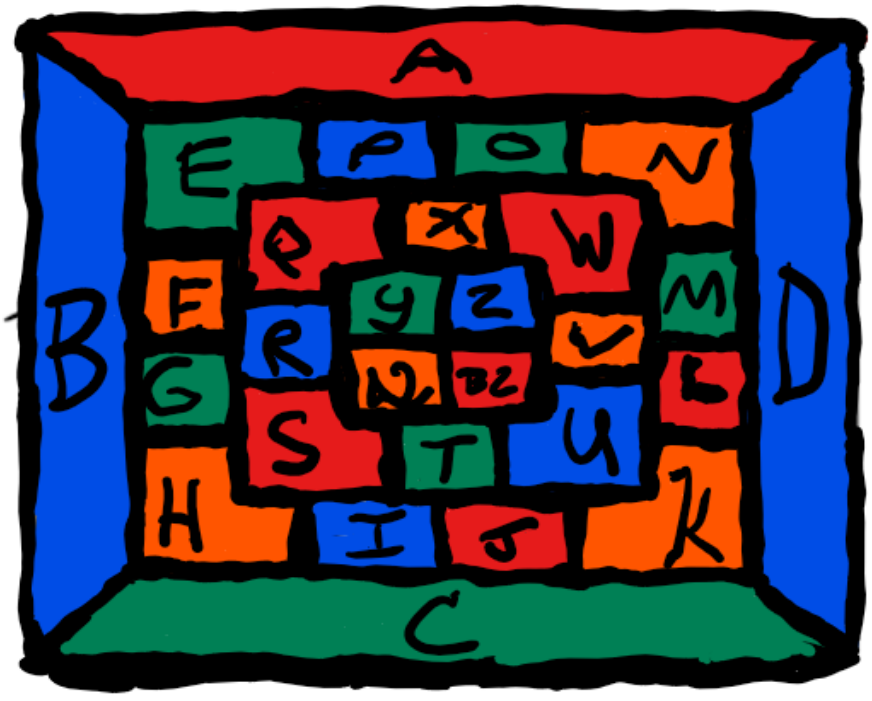
```
different(U,L),
different(U,K),
different(U,J),
different(V,Z),
different(V,B2),
different(V,W),
different(V,U),
different(V,M),
different(V,L),
different(W,X),
different(W,V),
different(W,Z),
different(W,O),
different(W,N),
different(W,M),
different(X,Q),
different(X,P),
different(X,O),
different(X,Y),
different(X,Z),
different(X,W),
different(Y,X),
different(Y,Q),
different(Y,R),
different(Y,A2),
different(Y,B2),
different(Y,Z),
different(Z,X),
different(Z,W),
different(Z,V),
different(Z,Y),
different(Z,A2),
different(Z,B2),
different(A2,Y),
```

```
different(A2,Z),
different(A2,B2),
different(A2,R),
different(A2,S),
different(A2,T),
different(B2,Z),
different(B2,A2),
different(B2,Y),
different(B2,V),
different(B2,U),
different(B2,T).
```

Demo:

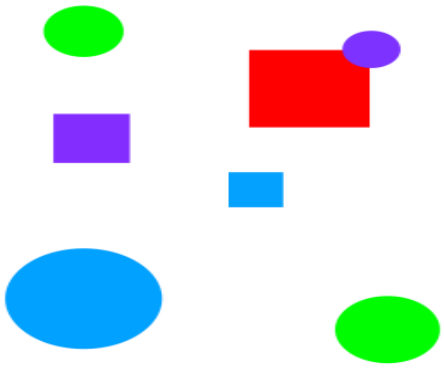
```
?- consult('map.pl').
true.

?- coloring(A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,Z,A2,B2).
A = J, J = L, L = Q, Q = S, S = W, W = B2, B2 = red,
B = D, D = I, I = P, P = R, R = U, U = Z, Z = blue,
C = E, E = G, G = M, M = O, O = T, T = Y, Y = green,
F = H, H = K, K = N, N = V, V = X, X = A2, A2 = orange
```

The Colored Map:



Task 4 - Floating Shapes World KB:

Floating Shapes World Image:



Code:

```
% --- File: shapes_world_1.pro
% --- Line: Loosely represented 2-D shapes world (simple take on SHRDLU)
% ------------------------------------------------------------------------
% ------------------------------------------------------------------------
% --- Facts ...
% ------------------------------------------------------------------------
% ------------------------------------------------------------------------
% --- square(N,side(L),color(C)) :: N is the name of a square with side L
% --- and color C
square(sera,side(7),color(purple)).
square(sara,side(5),color(blue)).
square(sarah,side(11),color(red)).
% ------------------------------------------------------------------------
% --- circle(N,radius(R),color(C)) :: N is the name of a circle with
% --- radius R and color C
circle(carla,radius(4),color(green)).
circle(cora,radius(7),color(blue)).
circle(connie,radius(3),color(purple)).
circle(claire,radius(5),color(green)).
% ------------------------------------------------------------------------
% Rules ...
% ------------------------------------------------------------------------
% ------------------------------------------------------------------------
% --- circles :: list the names of all of the circles
circles :- circle(Name,_,_), write(Name),nl,fail.
circles.
% ------------------------------------------------------------------------
% --- squares :: list the names of all of the squares
squares :- square(Name,_,_), write(Name),nl,fail.
squares.
% ------------------------------------------------------------------------
% --- squares :: list the names of all of the shapes
```

```
shapes :- circles,squares.
% ------------------------------------------------------------------
% --- blue(Name) :: Name is a blue shape
blue(Name) :- square(Name,_,color(blue)).
blue(Name) :- circle(Name,_,color(blue)).
% ------------------------------------------------------------------
% --- large(Name) :: Name is a large shape
large(Name) :- area(Name,A), A >= 100.
% ------------------------------------------------------------------
% --- small(Name) :: Name is a small shape
small(Name) :- area(Name,A), A < 100.
% ------------------------------------------------------------------
% --- area(Name,A) :: A is the area of the shape with name Name
area(Name,A) :- circle(Name,radius(R),_), A is 3.14 * R * R.
area(Name,A) :- square(Name,side(S),_), A is S * S.
```

Demo:

```
?- consult('shape_world_1.pl').
true.

?- listing(squares).
squares :-
    square(Name, _, _),
    write(Name),
    nl,
    fail.
squares.

true.

?- squares.
sera
sara
sarah
true.

?- listing(circles).
circles :-
    circle(Name, _, _),
    write(Name),
    nl,
    fail.
circles.

true.

?- circles.
carla
cora
connie
claire
true.

?- listing(shapes).
shapes :-
    circles,
    squares.
```

```
?- shapes.
carla
cora
connie
claire
sera
sara
sarah
true.

?- blue(Shape).
Shape = sara ;
Shape = cora.

?- large(Name),write(Name),nl,fail.
cora
sarah
false.

?- small(Name),write(Name),nl,fail.
carla
connie
claire
sera
sara
false.

?- area(cora,A).
A = 153.86 .

?- area(carla,A).
A = 50.24 .
```

```
?- consult('shape_world_1.pl').
true.

?- listing(squares).
squares :-
    square(Name, _, _),
    write(Name),
    nl,
    fail.
squares.

true.

?- squares.
sera
sara
sarah
true.

?- listing(circles).
circles :-
    circle(Name, _, _),
    write(Name),
    nl,
    fail.
circles.

true.

?- circles.
carla
cora
connie
claire
true.

?- listing(shapes).
shapes :-
    circles,
    squares.

true.
```

```
?- shapes.
carla
cora
connie
claire
sera
sara
sarah
true.

?- blue(Shape).
Shape = sara ;
Shape = cora.

?-  large(Name),write(Name),nl,fail.
cora
sarah
false.

?- small(Name),write(Name),nl,fail.
carla
connie
claire
sera
sara
false.

?- area(cora,A).
A = 153.86 ,

?-  area(carla,A).
A = 50.24 ,
```