Prolog Assignment

Eli Fereira

Abstract

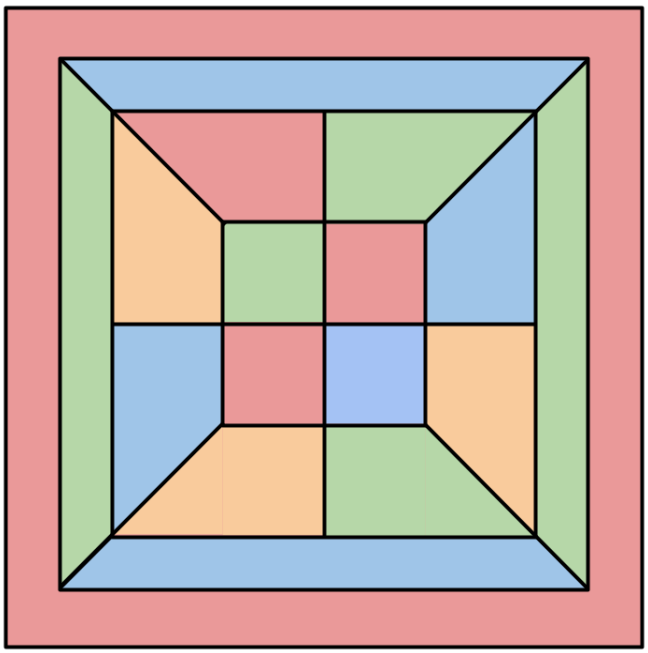
Since this is our only prolog assignment, this lab aims to encapsulate the monstrous topic of prolog, into one cohesive assignment. Starting off with relations, which is prolog's most common data type, getting really comfortable with relations, and then moving into list processing.

Task 1: Map Coloring

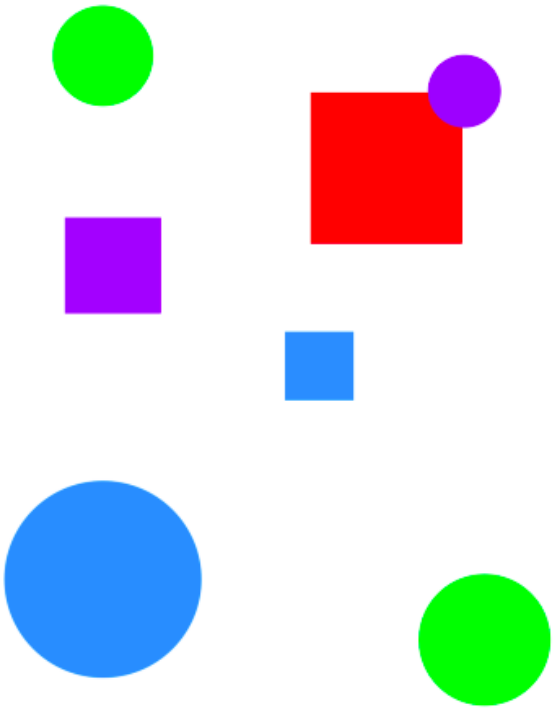
```
% File: map_coloring.pro
% Line Program to find a 4 color map for the given image.


% 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(ORING,LT, LR, LL, LB, MT1, MT2, MR1, MR2, MB1, MB2, ML1, ML2, S1, S2, S3,
S4) :-
    different(ORING, LT),
    different(ORING, LR),
    different(ORING, LL),
    different(ORING, LB),
    different(LT, LR),
    different(LR, LB),
    different(LB, LL),
    different(LL, LT),
    different(MT1, MT2),
    different(MT2, MR1),
    different(MR1, MR2),
    different(MR2, MB1),
    different(MB1, MB2),
    different(MB2, ML1),
    different(ML1, ML2),
    different(ML2, MT1),
    different(LT, MT1),
    different(LT, MT2),
    different(LR, MR1),
    different(LR, MR2),
    different(LL, ML1),
    different(LL, ML2),
    different(LB, MB1),
```

```
    different(LB, MB2),
    different(S1, MT1),
    different(S1, ML2),
    different(S2, MT2),
    different(S2, MR1),
    different(S3, MB1),
    different(S3, MB1),
    different(S4, MB2),
    different(S4, ML1),
    different(S1, S2),
    different(S2, S3),
    different(S3, S4),
    different(S4, S1).
```

```
?- consult("map_coloring.pro").
true.

?- coloring(ORING,LT, LR, LL, LB, MT1, MT2, MR1, MR2, MB1, MB2, ML1, ML2, S1, S2,
S3, S4).
ORING = MT1, MT1 = S2, S2 = S4, S4 = red,
LT = LB, LB = MR1, MR1 = ML1, ML1 = S3, S3 = blue,
LR = LL, LL = MT2, MT2 = MB1, MB1 = S1, S1 = green,
MR2 = MB2, MB2 = ML2, ML2 = orange .

?-
```



Task 2: Shapes World Recreation



```
% File: shapes_world.pro
% Line: Representation of shapes on a canvas


% 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)).

% 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.

% shapes :: 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.

?- consult("shapes_world.pro").
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.

?- 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 .

?- halt.
```

Task 3: Pokemon Knowledge Base Part 1

```
?- consult("pokemon.pro").
true.

?- cen(pikachu).
true.

?- cen(raichu).
False.

?- cen(N).
N = pikachu ;
N = bulbasaur ;
N = caterpie ;
N = charmander ;
N = vulpix ;
N = poliwag ;
N = squirtle ;
N = staryu.

?- cen(N), write(N), nl, fail.
pikachu
bulbasaur
caterpie
charmander
vulpix
poliwag
squirtle
staryu
false.

?- evolves(squirtle, wartortle).
true.

?- evolves(wortortle, squirtle).
false.

?- evolves(squirtle, blastoise).
false.

?- evolves(CEN, EV1), evolves(EV1, EV2).
CEN = bulbasaur,
EV1 = ivysaur,
EV2 = venusaur ;
CEN = caterpie,
```

```
EV1 = metapod,
EV2 = butterfree ;
CEN = charmander,
EV1 = charmeleon,
EV2 = charizard ;
CEN = poliwag,
EV1 = poliwhirl,
EV2 = poliwrath ;
CEN = squirtle,
EV1 = wartortle,
EV2 = blastoise ;
false.

?- pokemon(name(N),_,_,_), write(N), nl, fail.
pikachu
raichu
bulbasaur
ivysaur
venusaur
caterpie
metapod
butterfree
charmander
charmeleon
charizard
vulpix
ninetails
poliwag
poliwhirl
poliwrath
squirtle
wartortle
blastoise
staryu
starmie
false.
?- pokemon(name(N),fire,_,_), write(N), nl, fail.
charmander
charmeleon
charizard
vulpix
ninetails
false.

?- pokemon(N,Type,_,_), write(nks(N,kind(Type))), nl, fail.
nks(name(pikachu),kind(electric))
nks(name(raichu),kind(electric))
```

```
nks(name(bulbasaur),kind(grass))
nks(name(ivysaur),kind(grass))
nks(name(venusaur),kind(grass))
nks(name(caterpie),kind(grass))
nks(name(metapod),kind(grass))
nks(name(butterfree),kind(grass))
nks(name(charmander),kind(fire))
nks(name(charmeleon),kind(fire))
nks(name(charizard),kind(fire))
nks(name(vulpix),kind(fire))
nks(name(ninetails),kind(fire))
nks(name(poliwag),kind(water))
nks(name(poliwhirl),kind(water))
nks(name(poliwrath),kind(water))
nks(name(squirtle),kind(water))
nks(name(wartortle),kind(water))
nks(name(blastoise),kind(water))
nks(name(staryu),kind(water))
nks(name(starmie),kind(water))
false.

?- pokemon(name(N),_,_,attack(waterfall,_)).
N = wartortle .

?- pokemon(name(N),_,_,attack(poison-powder,_)).
N = venusaur .

?- pokemon(_,water,_,attack(A,_)), write(A), nl, fail.
water-gun
amnesia
dashing-punch
bubble
waterfall
hydro-pump
slap
star-freeze
false.

?- pokemon(name(poliwhirl),_,hp(HP),_).
HP = 80.

?- pokemon(name(butterfree),_,hp(HP),_).
HP = 130.

?- pokemon(name(N),_,hp(HP),_), HP > 85, write(N), nl, fail.
raichu
venusaur
```

```
butterfree
charizard
ninetails
poliwrath
blastoise
false.

?- pokemon(name(N),_,_,attack(_,DMG)), DMG > 60, write(N), nl, fail.
raichu
venusaur
butterfree
charizard
ninetails
false.

?- pokemon(name(N),_,hp(HP),_), cen(N), write(N), write(": "), write(HP), nl, fail.
pikachu: 60
bulbasaur: 40
caterpie: 50
charmander: 50
vulpix: 60
poliwag: 60
squirtle: 40
staryu: 40
false.
```

Task 3 Part 2:

```
% ------------------------------------------------------------------------
% --- Part 2

% Display the names of every pokemon
display_names :- pokemon(name(N),_,_,_), write(N), nl, fail.
display_names :- true.

% Display the name of every attack
display_attacks :- pokemon(_,_,_,attack(ATK,_)), write(ATK), nl, fail.
display_attacks :- true.

% Returns true if attack power is larger than 55
powerful(N) :- pokemon(name(N),_,_,attack(_,PWR)), PWR > 55.

% Returns true if HP is larget than 100
tough(N) :- pokemon(name(N),_,hp(HP),_), HP > 100.

% Get the type of a pokemon by name
type(N, T) :- pokemon(name(N),T,_,_).

% Write all pokemon with given type
dump_kind(T) :- pokemon(N,T,HP,ATK), write(pokemon(N,T,HP,ATK)), nl, fail.
dump_kind(_) :- true.

% Write all cen pokemon
display_cen :- pokemon(name(N),_,_,_), cen(N), write(N), nl, fail.
display_cen :- true.

% Display family of pokemon from base pokemon.
family(CEN) :- pokemon(name(CEN),_,_,_), cen(CEN), write(CEN), write(" "), evolves(CEN,
EV1), write(EV1), write(" "), evolves(EV1, EV2), write(EV2).
family(CEN) :- cen(CEN).

% Display all families
families :- cen(C), family(C), nl, fail.
families :- true.

% Print the lineage of each pokemon
lineage(N) :- pokemon(name(N),T,H,A), write(pokemon(name(N),T,H,A)), evolves(N,E1),
pokemon(name(E1),E1T,E1H,E1A), nl, write(pokemon(name(E1),E1T,E1H,E1A)), evolves(E1, E2),
pokemon(name(E2),E2T,E2H,E2A), nl, write(pokemon(name(E2),E2T,E2H,E2A)).
lineage(_) :- true.
```

```
?- consult('pokemon.pro').
true.

?- display_names.
pikachu
raichu
bulbasaur
ivysaur
venusaur
caterpie
metapod
butterfree
charmander
charmeleon
charizard
vulpix
ninetails
poliwag
poliwhirl
poliwrath
squirtle
wartortle
blastoise
staryu
starmie
true.

?- display_attacks.
gnaw
thunder-shock
leech-seed
vine-whip
poison-powder
gnaw
stun-spore
whirlwind
scratch
slash
royal-blaze
confuse-ray
fire-blast
water-gun
amnesia
dashing-punch
bubble
waterfall
hydro-pump
```

```
slap
star-freeze
true.

?- powerful(pikachu).
false.

?- powerful(blastoise).
true.

?- tough(Name), write(Name), nl, fail.
venusaur
butterfree
charizard
poliwrath
blastoise
false.

?- type(caterpie, grass).
true .

?- type(pikachu, water).
false.

?- type(N,electric).
N = pikachu ;
N = raichu.

?- type(N,water), write(N), nl, fail.
poliwag
poliwhirl
poliwrath
squirtle
wartortle
blastoise
staryu
starmie
false.

?- dump_kind(water).
pokemon(name(poliwag),water,hp(60),attack(water-gun,30))
pokemon(name(poliwhirl),water,hp(80),attack(amnesia,30))
pokemon(name(poliwrath),water,hp(140),attack(dashing-punch,50))
pokemon(name(squirtle),water,hp(40),attack(bubble,10))
pokemon(name(wartortle),water,hp(80),attack(waterfall,60))
pokemon(name(blastoise),water,hp(140),attack(hydro-pump,60))
pokemon(name(staryu),water,hp(40),attack(slap,20))
```

```
pokemon(name(starmie),water,hp(60),attack(star-freeze,20))
true.

?- dump_kind(fire).
pokemon(name(charmander),fire,hp(50),attack(scratch,10))
pokemon(name(charmeleon),fire,hp(80),attack(slash,50))
pokemon(name(charizard),fire,hp(170),attack(royal-blaze,100))
pokemon(name(vulpix),fire,hp(60),attack(confuse-ray,20))
pokemon(name(ninetails),fire,hp(100),attack(fire-blast,120))
true.

?- display_cen.
pikachu
bulbasaur
caterpie
charmander
vulpix
poliwag
squirtle
staryu
true.

?- family(pikachu).
pikachu raichu
true.

?- family(squirtle).
squirtle wartortle blastoise
true .

?- families.
pikachu raichu
bulbasaur ivysaur venusaur
caterpie metapod butterfree
charmander charmeleon charizard
vulpix ninetails
poliwag poliwhirl poliwrath
squirtle wartortle blastoise
staryu starmie
true.

?- lineage(caterpie).
pokemon(name(caterpie),grass,hp(50),attack(gnaw,20))
pokemon(name(metapod),grass,hp(70),attack(stun-spore,20))
pokemon(name(butterfree),grass,hp(130),attack(whirlwind,80))
true .
```

```
?- lineage(metapod).
pokemon(name(metapod),grass,hp(70),attack(stun-spore,20))
pokemon(name(butterfree),grass,hp(130),attack(whirlwind,80))
true.

?- lineage(butterfree).
pokemon(name(butterfree),grass,hp(130),attack(whirlwind,80))
true.

?-
```

Task 4: List Processing Part 1

```
?- [H|T] = [red, yellow, blue, green].
H = red,
T = [yellow, blue, green].

?- [H, T] = [red, yellow, blue, green].
false.

?- [F|_] = [red, yellow, blue, green].
F = red.

?- [_|[S|_]] = [red, yellow, blue, green].
S = yellow.

?- [F|[S|R]] = [red, yellow, blue, green].
F = red,
S = yellow,
R = [blue, green].

?- List = [this|[and, that]].
List = [this, and, that].

?- List = [this, and, that].
List = [this, and, that].

?- [a,[b, c]] = [a, b, c].
false.

?- [a|[b, c]] = [a, b, c].
true.

?- [cell(Row, Column)|Rest] = [cell(1,1), cell(3,2), cell(1,3)].
Row = Column, Column = 1,
Rest = [cell(3, 2), cell(1, 3)].

?- [X|Y] = [one(un, uno), two(dos, deux), three(trois, tres)].
X = one(un, uno),
Y = [two(dos, deux), three(trois, tres)].

?-
```

Task 4 Part 2:

```
% -------------------------
% List Processing Excersices
% -------------------------

product([], 1).
product([H|T], Product) :-
       product(T, TailProduct),
       Product is H * TailProduct.

make_list(0,_,[]).
make_list(N,Element,List) :-
       K is N - 1,
       make_list(K,Element,Tail),
       List = [Element|Tail].

but_first([], []).
but_first([_|T], T).

but_last(List, Result) :-
       reverse(List, ReverseList),
       but_first(ReverseList, ButFirstList),
       reverse(ButFirstList, Result).

is_palindrome([]).
is_palindrome([_|[]]).
is_palindrome(List) :-
       first(List,FirstEl),
       last(List,LastEl),
       FirstEl = LastEl,
       but_first(List,ButFirst),
       but_last(ButFirst,TruncList),
       is_palindrome(TruncList).


% -----------------
% Sentence Building
% -----------------

noun_phrase(NP) :-
       Nouns = [king, knight, scientist, wizard, frog, dog, snake, book, sheriff],
       Adjs = [sick, mad, happy, ugly, shy, aggresive],
       pick(Nouns, N),
       pick(Adjs, A),
       NP = [the, A, N].
```

```
sentence(S) :-
       Verbs = [shot, fed, scared, carried, swiped, read, loved],
       pick(Verbs, V),
       noun_phrase(NP1),
       noun_phrase(NP2),
       append(NP1, [V|NP2], S).
```

```
?- consult('list_processors.pro').
true.

?- product([],P).
P = 1.

?- product([1,3,5,7,9],Product).
Product = 945.

?- iota(9,Iota), product(Iota, Product).
Iota = [1, 2, 3, 4, 5, 6, 7, 8, 9],
Product = 362880 .

?- make_list(7,seven,Seven).
Seven = [seven, seven, seven, seven, seven, seven, seven] .

?- make_list(8,2,List).
List = [2, 2, 2, 2, 2, 2, 2, 2] .

?- but_first([a,b,c], C).
C = [b, c].

?- but_last([a,b,c,d,e],X).
X = [a, b, c, d].

?- is_palindrome([x]).
true .

?- is_palindrome([a,b,c]).
false.

?- is_palindrome([a,b,b,a]).
true .

?- is_palindrome([1,2,3,4,5,4,2,3,1]).
false.

?- is_palindrome([c,o,f,f,e,e,e,e,f,f,o,c]).
```

```
true .

?- noun_phrase(NP).
NP = [the, aggressive, sheriff] ;
false.

?- noun_phrase(NP).
NP = [the, aggressive, wizard] .

?- noun_phrase(NP).
NP = [the, aggressive, snake] .

?- noun_phrase(NP).
NP = [the, aggressive, scientist] .

?- noun_phrase(NP).
NP = [the, sick, frog] .

?- sentence(S).
S = [the, aggresive, frog, loved, the, happy, snake] .

?- sentence(S).
S = [the, aggresive, frog, read, the, aggresive, king] .

?- sentence(S).
S = [the, mad, dog, swiped, the, ugly, snake] .

?- sentence(S).
S = [the, shy, dog, shot, the, happy, book] .

?- sentence(S).
S = [the, aggressive, snake, swiped, the, shy, scientist] .

?- sentence(S).
S = [the, happy, wizard, read, the, happy, knight] .

?- sentence(S).
S = [the, mad, dog, loved, the, aggresive, sheriff] .

?- sentence(S).
S = [the, ugly, snake, shot, the, aggresive, knight] .

?- sentence(S).
S = [the, shy, king, carried, the, mad, sheriff] .

?- sentence(S).
S = [the, mad, knight, shot, the, sick, dog] .
```

```
?- sentence(S).
S = [the, shy, frog, fed, the, mad, snake] .

?- sentence(S).
S = [the, aggresive, scientist, scared, the, shy, dog] .

?- sentence(S).
S = [the, mad, book, carried, the, ugly, book] .

?- sentence(S).
S = [the, ugly, snake, read, the, happy, dog] .

?- sentence(S).
S = [the, aggresive, book, scared, the, ugly, knight] .

?- sentence(S).
S = [the, happy, sheriff, loved, the, ugly, dog] .

?-
```