CMPE 260 PRINCIPLES OF PROGRAMMING LANGUAGES

PROJECT 1: PROLOG PROGRAMMING PROJECT

Aysu Sayın

23 April 2018

Introduction:

The project is to write a Prolog program which performs some particular tasks for the matches which are played by the football teams in a Football League.

A database is given with a bunch of predicates in the following two form:

1) team(teamName, hometown).
2) match(week, homeTeam, homeTeamScore, awayTeam, awayTeamScore).

First one shows the teams that are existing in the database and their home town.

e.g. team(realmadrid, madrid).

This means realmadrid plays its home matches in madrid.

Second one shows the matches played between two teams, their scores and the week that the match is played.

e.g. match(1, kobenhavn, 1, juventus 2).

A match is played between kobenhavn which is the home team and juventus which is the away team in week 1, and juventus defeated kobenhavn with the score of 2-1.

In the project, the tasks are the following:

1) All Teams:
   List all teams in the database.
2) Match Results:
   The predicates to check the teams that the specified team defeated, is defeated by or tied with.
3) Goals Scored and Conceded:
   Give the number of the goal scored or conceded by the specified team in the specified week.
4) Total Average of a Team:
   Show the average of the given team in the given week. Average is the total number of goals scored minus the total number of goals conceded.
5) Order and Top Three:
   List the league order in the given week. Also list the top three teams in that week.

A few predicates are implemented to achieve these goals which are explained below.

Program Interface:

To run the program please follow these steps:

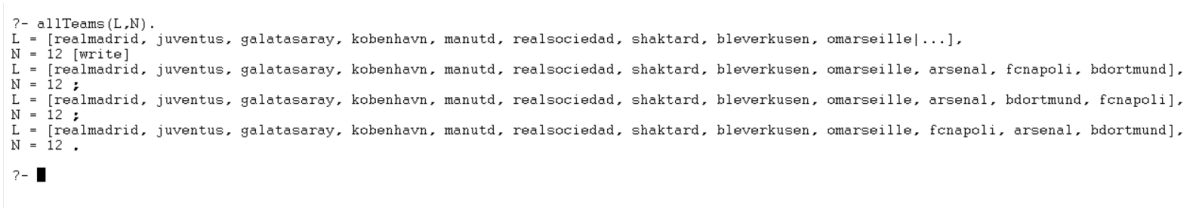
1) Move the program file (predicates.pl) and the database to the Prolog Directory.
2) Start SWI-Prolog.
3) Load the program file:

   Write " [predicates]. " to the console and click enter.

   If you see "true." that means the file is loaded successfully.

   Repeat the same steps for the database file.

4) Write the goal that you wish to check to the console and the result will appear on the screen.



To show all the elements of the list, press "w".

To show the other results, press ";".

To enter a new goal, press enter or ".".

Program Execution

Following part explains the predicates that you can use in the program. If there is no variables in the goal the result will be true or false according to the correctness of the input. Otherwise the program will show the values of the variables if found, false otherwise.

Variables must start with a capital letter.

1) **All Teams**

   Predicate: allTeams(L,N).

   L is the list of all the teams in the database. N is the number of the teams.

   The output will be the list L and integer N if the goal is allTeams(L,N). Also all permutations of teams can be seen by pressing ";". If the L is given in the goal, the program shows N if L is the correct list. If N is given in the goal, the program shows L if the N is correct. If both of L and N is given in the goal program checks if the goal is true. In the other cases the program will give false as an output.

   Some Examples:

```
?- allTeams(L,N).
L = [realmadrid, juventus, galatasaray, kobenhavn, manutd, realsociedad, shaktard, bleverkusen, omarseille|...],
N = 12 [write]
L = [realmadrid, juventus, galatasaray, kobenhavn, manutd, realsociedad, shaktard, bleverkusen, omarseille, arsenal, fcnapoli, bdortmund],
N = 12 ;
L = [realmadrid, juventus, galatasaray, kobenhavn, manutd, realsociedad, shaktard, bleverkusen, omarseille, arsenal, bdortmund, fcnapoli],
N = 12 ;
L = [realmadrid, juventus, galatasaray, kobenhavn, manutd, realsociedad, shaktard, bleverkusen, omarseille, fcnapoli, arsenal, bdortmund],
N = 12 .

?- allTeams([realmadrid, juventus, galatasaray, kobenhavn, manutd, realsociedad, shaktard, bleverkusen, omarseille, arsenal, bdortmund, fcnapoli],N).
N = 12 .

?- allTeams(L,12).
L = [realmadrid, juventus, galatasaray, kobenhavn, manutd, realsociedad, shaktard, bleverkusen, omarseille, arsenal, fcnapoli, bdortmund] .

?- allTeams([realmadrid, juventus, galatasaray, kobenhavn, manutd, realsociedad, shaktard, bleverkusen, omarseille, arsenal, bdortmund, fcnapoli],12).
true .

?- allTeams([galatasaray],4).
false.
```

2) **Wins**

   Predicate: wins(T,W,L,N).

   L involves the teams defeated by team T when we are in week W and N is the number of elements in L.

   T and W must be given.

   Some Examples:

```
?- wins(juventus,5,L,N).
L = [kobenhavn],
N = 1.

?- wins(realmadrid,5,L,N).
L = [kobenhavn, juventus, galatasaray, galatasaray],
N = 4.

?- wins(juventus,5,[kobenhavn],1).
true.
```

3) Losses

Predicate: losses(T,W,L,N).

L involves the teams defeated the team T when we are in week W and N is the number of elements in L.

T and W must be given.

Some Examples:

```
?- losses(galatasaray,6,L,N).
L = [realmadrid, kobenhavn, realmadrid],
N = 3.

?- losses(juventus,6,L,N).
L = [realmadrid, galatasaray],
N = 2.

?- losses(galatasaray,6,L,3).
L = [realmadrid, kobenhavn, realmadrid].
```

4) Draws

Predicate: draws(T,W,L,N).

L involves the teams that is tied with the team T when we are in week W and N is the number of elements in L.

T and W must be given.

Some Examples:

```
?- draws(bleverkusen,5,L,N).
L = [realsociedad, shaktard],
N = 2.

?- draws(bdortmund,1,L,N).
L = [],
N = 0.

?- draws(galatasaray,1,[],0).
true.
```

5) Scored

Predicate: scored(T,W,S).

C is the total number of goals scored by team T up to (and including) week W.

T and W are given as constants.

Some Examples:

```
?- scored(galatasaray,4,S).
S = 6 .

?- scored(realmadrid,4,8).
false.

?- scored(realmadrid,4,S).
S = 14 .

?- scored(realmadrid,4,14).
true .
```

6) Conceded

Predicate: conceded(T,W,S).

C is the total number of goals conceded by team T up to (and including) week W.

T and W are given as constants.

Some Examples:

```
?- conceded(galatasaray,3,C).
C = 9 .

?- conceded(realmadrid,5,5).
true .
```

7) Average

Predicate: average(T,W,A).

A is the of a team T gathered up to (and including) week W.

T and W are given as constants.

Some Examples:

```
?- average(bdortmund,6,A).
A = 5 .

?- average(bdortmund,6,5).
true .

?- average(juventus,6,5).
false.
```

8) Order

Predicate: order(L, W).

League order in week W, retrieved in L.

W is given as constant. The order is decided according to average, the one with the highest average will be at the top. If the two teams have the same average then the order can be in any order.

Some Examples:

```
?- order(L,6).
L = [realmadrid, manutd, bdortmund, arsenal, fcnapoli, shaktard, juventus, realsociedad, bleverkusen, galatasaray, kobenhavn, omarseille] .

?- order(L,3).
L = [realmadrid, manutd, bdortmund, bleverkusen, arsenal, fcnapoli, juventus, shaktard, realsociedad, galatasaray, omarseille, kobenhavn] .
```

9) Top Three

Predicate: topThree(L,W).

L contains the top three teams in week W. The length of L must be 3 and W is given as constant.

Some Examples:

```
?- order(L,3).
L = [realmadrid, manutd, bdortmund, bleverkusen, arsenal, fcnapoli, juventus, shaktard, realsociedad, galatasaray, omarseille, kobenhavn] .

?- topThree(L,6).
L = [realmadrid, manutd, bdortmund] .

?- topThree([realmadrid,manutd,bdortmund],3).
true .

?- topThree([realmadrid,manutd,bdortmund,bleverkusen],3).
false.
```

Input and Output:

Database prolog file must be loaded before asking for goals.

Example database file: database.pl

Loading the file is explained in “Program Interface” section.

The goals’ inputs are explained in detailed in “Program Execution” section.

Outputs can be seen on the screen after asking a goal on the console.

Program Structure:

In this section the program is explained according to its source code, line by line.

```
allTeams(List,N):-
findall(Atom,team(Atom,_),L),length(L,N),permutation(L,List).
```

findAll is a built-in function and finds all predicates matching to the given predicate goal and puts into the list L, length calculates the length of the list and permutation forms all permutation of teams.

```
wins(T,W,L,N):- W1 is W+1,
findall(OT,(match(W2,T,TS,OT,OTS),TS>OTS,W2<W1),L1),findall(OT,(match(W2,OT
,OTS,T,TS),TS>OTS,W2<W1),L2),append(L1,L2,L), length(L,N).
```

If the team's score is more than the other team's score, other team is added to list. This is done by findall function. Two findall functions are used, one is for the matches that the team is home team and the other one is for the away team. These lists are appended to get the final list. length function finds the length of the given list.

```
losses(T,W,L,N):- W1 is
W+1,findall(OT,(match(W2,T,TS,OT,OTS),TS<OTS,W2<W1),L1),findall(OT,(match(W
2,OT,OTS,T,TS),TS<OTS,W2<W1),L2),append(L1,L2,L), length(L,N).
```

If the team's score is less than the other team's score, other team is added to list. This is done by findall function. Two findall functions are used, one is for the matches that the team is home team and the other one is for the away team. These lists are appended to get the final list. length function finds the length of the given list.

```
draws(T,W,L,N):- W1 is W+1,
findall(OT,(match(W2,T,TS,OT,OTS),TS=OTS,W2<W1),L1),findall(OT,(match(W2,OT
,OTS,T,TS),TS=OTS,W2<W1),L2),append(L1,L2,L), length(L,N).
```

If the team's score is equal to the other team's score, other team is added to list This is done by findall function. Two findall functions are used, one is for the matches that the team is home team and the other one is for the away team. These lists are appended to get the final list. length function finds the length of the given list.

```
scored(_,0,S):-S is 0.
```

This is the base case. When week is 0, nobody has scores.

```
scored(T,W,S):-
W>0,findall(TS,match(W,T,TS,_,_),L1),findall(TS,match(W,_,_,T,TS),L2),appen
d(L1,L2,L3),sumOfElements(N,L3), WL is W-1,scored(T,WL,M), S is N+M.
```

Find all matches the team played with findall function and sum up the score they made with the sumOfElements predicate. Call scored recursively to calculate the scores in the previous weeks. And sum them to get the result S.

```
conceded(_,0,C):- C is 0.
```

This is the base case. When week is 0, nobody has conceded any goals.

```
conceded(T,W,C):-
W>0,findall(OTS,match(W,T,_,_,OTS),L1),findall(OTS,match(W,_,OTS,T,_),L2),a
ppend(L1,L2,L3),sumOfElements(N,L3), WL is W-1,conceded(T,WL,M), C is N+M.
```

Find all matches the team played with findall function and sum up the goals it conceded with the sumOfElements predicate. Call conceded recursively to calculate the conceded goals in the previous weeks. And sum them to get the result C.

```
sumOfElements(N,[]):-N is 0.
```

This is the base case. When the list is empty sum of elements is zero.

```
sumOfElements(N,[Head|Tail]):-sumOfElements(M,Tail), N is M+Head.
```

This predicate calls it self recursively with the tail of the given list and sums up the head with the value that is retrieved from the recursive call.

```
average(T,W,A):-scored(T,W,S), conceded(T,W,C), A is S-C.
```

Average is scored-conceded. So, it is calculated by calling scored and conceded predicates and subtracting the results.

```
formTuples(_,[],LT):-append([],[],LT).
```

This is the base case.

```
formTuples(W,[Head|Tail],LT):- formTuples(W,Tail,LT1),average(Head,W,A),
append([(A,Head)],LT1,LT).
```

W week

second parameter is team list

LT is list of tuples

Form tuple (Average, Team) of the head of the list, call formTuples recursively and append the tuple with the list that is retrieved from the recursive call.

```
makeTeamList([],List):-append([],[],List).
```

This is the base case.

```
makeTeamList([(_,T)|Tail],List):- makeTeamList(Tail,L1),
append(L1,[T],List).
```

First parameter is tuple list

Second one is the team list

From the tuple list, take the second element of the head tuple and append it with the list that is retrieved from the recursive call.

```
order(L,W):-
allTeams(L1,_),formTuples(W,L1,LT),sort(LT,LS),makeTeamList(LS,L).
```

Get all the teams in the league and form the tuple list from them. Sort this tuple list, according to averages and call makeTeamList predicate to achieve the result.

```
topThree([A1,A2,A3],W):-order([O1,O2,O3|_],W), A1=O1, A2=O2, A3=O3.
```

Take the top three team of that week from order predicate and add to a list.

Examples:

```
?- allTeams(L,N).
L = [realmadrid, juventus, galatasaray, kobenhavn, manutd, realsociedad,
shaktard, bleverkusen, omarseille|...],
N = 12 [write]
L = [realmadrid, juventus, galatasaray, kobenhavn, manutd, realsociedad,
shaktard, bleverkusen, omarseille, arsenal, fcnapoli, bdortmund],
N = 12 ;
L = [realmadrid, juventus, galatasaray, kobenhavn, manutd, realsociedad,
shaktard, bleverkusen, omarseille, arsenal, bdortmund, fcnapoli],
N = 12 ;
L = [realmadrid, juventus, galatasaray, kobenhavn, manutd, realsociedad,
shaktard, bleverkusen, omarseille, fcnapoli, arsenal, bdortmund],
N = 12 .

?- allTeams([realmadrid, juventus, galatasaray, kobenhavn, manutd,
realsociedad, shaktard, bleverkusen, omarseille, arsenal, bdortmund,
fcnapoli],N).
N = 12 .

?- allTeams(L,12).
L = [realmadrid, juventus, galatasaray, kobenhavn, manutd, realsociedad,
shaktard, bleverkusen, omarseille, arsenal, fcnapoli, bdortmund] .

?- allTeams([realmadrid, juventus, galatasaray, kobenhavn, manutd,
realsociedad, shaktard, bleverkusen, omarseille, arsenal, bdortmund,
fcnapoli],12).
true .

?- allTeams([galatasaray],4).
false.

?- wins(galatasaray,3,L,N).
L = [kobenhavn],
N = 1.

?- wins(galatasaray,6,L,N).
L = [kobenhavn, juventus],
N = 2.
```

```
?- wins(juventus,5,L,N).
L = [kobenhavn],
N = 1.

?- wins(realmadrid,5,L,N).
L = [kobenhavn, juventus, galatasaray, galatasaray],
N = 4.

?- wins(juventus,5,[kobenhavn],1).
true.

?- losses(galatasaray,6,L,N).
L = [realmadrid, kobenhavn, realmadrid],
N = 3.

?- losses(juventus,6,L,N).
L = [realmadrid, galatasaray],
N = 2.

?- losses(galatasaray,6,L,3).
L = [realmadrid, kobenhavn, realmadrid].

?- draws(bleverkusen,5,L,N).
L = [realsociedad, shaktard],
N = 2.

?- draws(bdortmund,1,L,N).
L = [],
N = 0.

?- draws(galatasaray,1,[],0).
true.

?- scored(galatasaray,4,S).
S = 6 .

?- scored(realmadrid,4,8).
false.

?- scored(realmadrid,4,S).
S = 14 .
```

```
?- scored(realmadrid,4,14).
true .

?- conceded(realmadrid,5,C).
C = 5 .

?- conceded(galatasaray,3,C).
C = 9 .

?- conceded(realmadrid,5,5).
true .

?- average(bdortmund,6,A).
A = 5 .

?- average(bdortmund,6,5).
true .

?- average(juventus,6,5).
false.

?- order(L,6).
L = [realmadrid, manutd, bdortmund, arsenal, fcnapoli, shaktard, juventus,
realsociedad, bleverkusen, galatasaray, kobenhavn, omarseille] .

?- order(L,3).
L = [realmadrid, manutd, bdortmund, bleverkusen, arsenal, fcnapoli,
juventus, shaktard, realsociedad, galatasaray, omarseille, kobenhavn] .

?- topThree(L,6).
L = [realmadrid, manutd, bdortmund] .

?- topThree([realmadrid,manutd,bdortmund],3).
true .

?- topThree([realmadrid,manutd,bdortmund,bleverkusen],3).
false.
```

Improvements and Extensions:

This is a very simple program so there can be a few improvements. Order predicate can have implemented in a way that the implementation will be simpler.

As an extantion, some other tasks can be added to the program and other data can be retrieved from the given database.

Difficulties Encountered:

Since Prolog is a logical language and I was not use to it, it was hard to think in that way at the beginning of the project. Understanding the importance of the rescursion, made it easy to implement the predicates. Also, without I didn't managed to find all predicates at first, then I searched and saw findall built-in function. It really helped me to solve the problem.

Conclusion:

The goal of the project is to test Prolog knowledge. In conclusion, this project really taught me about Prolog, how to use it, where to use it, its advantages and disadvantages. I also learned how to think when using a logical programming languages.

Appendices:

Code of the program:

```
%0.1 All Teams

% findAll finds all predicates matching to the given predicate goal and
% puts into the L, length calculates the length of the list and
% permutation forms all permutation of teams.
allTeams(List,N):-
findall(Atom,team(Atom,_),L),length(L,N),permutation(L,List).

%0.2 Match Results

% if the teams score is more than the other team's score, other team is
% added to list.This is done by findall function. Two findall functions are
used, one is for the mathes that the team is home team and the other one is
for the away team. These lists are appended to get the final list.length
function finds the length of the given list.
% T Team
% W Week
% L List of the teams that the team wins.
```

```
% N number of elements in the list
wins(T,W,L,N):- W1 is W+1,
findall(OT,(match(W2,T,TS,OT,OTS),TS>OTS,W2<W1),L1),findall(OT,(match(W2,OT
,OTS,T,TS),TS>OTS,W2<W1),L2),append(L1,L2,L), length(L,N).

% If the teams score is less than the other team's score, other team is
added to list. This is done by findall function. Two findall functions are
used, one is for the matches that the team is home team and the other one
is for the away team. These lists are appended to get the final list.length
function finds the length of the given list.
% T Team
% W Week
% L List of the teams that the team losses.
% N number of elements in the list
losses(T,W,L,N):- W1 is
W+1,findall(OT,(match(W2,T,TS,OT,OTS),TS<OTS,W2<W1),L1),findall(OT,(match(W
2,OT,OTS,T,TS),TS<OTS,W2<W1),L2),append(L1,L2,L), length(L,N).

%If the team’s score is equal to the other team's score, other team is
added to list This is done by findall function. Two findall functions are
used, one is for the matches that the team is home team and the other one
is for the away team. These lists are appended to get the final list.
length function finds the length of the given list.
% T Team
% W Week
% L List of the teams that the team draws.
% N number of elements in the list
draws(T,W,L,N):- W1 is W+1,
findall(OT,(match(W2,T,TS,OT,OTS),TS=OTS,W2<W1),L1),findall(OT,(match(W2,OT
,OTS,T,TS),TS=OTS,W2<W1),L2),append(L1,L2,L), length(L,N).

%0.3 Goals Scored and Conceded

%When week is 0, nobody has scores.(base case)
scored(_,0,S):-S is 0.
% Find all matches the team played with findall function and sum up the
% score they made with the sumOfElements predicate. Call scored again
% recursively to calculate the scores in the previous weeks. T team W
% week S score And sum them to get the result S.
```

```
scored(T,W,S):-
W>0,findall(TS,match(W,T,TS,_,_),L1),findall(TS,match(W,_,_,T,TS),L2),appen
d(L1,L2,L3),sumOfElements(N,L3), WL is W-1,scored(T,WL,M), S is N+M.

%When week is 0, nobody has scores.(base case)
conceded(_,0,C):- C is 0.
% Find all matches the team played with findall function and sum up the
goals it conceded with the sumOfElements predicate. Call conceded
recursively to calculate the conceded goals in the previous weeks. And sum
them to get the result C.
% T team
% W week
% C conceded goals
conceded(T,W,C):-
W>0,findall(OTS,match(W,T,_,_,OTS),L1),findall(OTS,match(W,_,OTS,T,_),L2),a
ppend(L1,L2,L3),sumOfElements(N,L3), WL is W-1,conceded(T,WL,M), C is N+M.


%sums up the values of the elements of the given list.
%This predicate calls it self recursively with the tail of the given list
and sums up the head with the value that is retrieved from the recursive
call.
%N sum
sumOfElements(N,[]):-N is 0.
sumOfElements(N,[Head|Tail]):-sumOfElements(M,Tail), N is M+Head.

%average is the scored - conceded. it is calculated by calling scored and
conceded predicates and subtracting the results.
%T team
%W week
%A average
average(T,W,A):-scored(T,W,S), conceded(T,W,C), A is S-C.

%0.4 Order and Top Three

%form tuples (Average, Team) and add them to the list
%Form tuple (Average, Team) of the head of the list, call formTuples
recursively and append the tuple with the list that is retrieved from the
recursive call.
formTuples(_,[],LT):-append([],[],LT). %base case
%W week
```

```
%second parameter  team list
%LT is list of tuples
formTuples(W,[Head|Tail],LT):- formTuples(W,Tail,LT1),average(Head,W,A),
append([(A,Head)],LT1,LT).

% From the tuple list, take the second element of the head tuple and
% append it with the list that is retrieved from the recursive call.
makeTeamList([],List):-append([],[],List). %base case
%first parameter is tuple list
%second one is the team list
makeTeamList([(_,T)|Tail],List):- makeTeamList(Tail,L1),
append(L1,[T],List).

%give the team order of that week according to the averages. Get all the
teams in the league and form the tuple list from them.  Sort this tuple
list, according to averages and call makeTeamList predicate to achieve
theresult.
%L order list
%W week
order(L,W):-
allTeams(L1,_),formTuples(W,L1,LT),sort(LT,LS),makeTeamList(LS,L).


% take the top three team of that week from order predicate and add to a
% list W week First parameter is the top three list
topThree([A1,A2,A3],W):-order([O1,O2,O3|_],W), A1=O1, A2=O2, A3=O3.
```