2017/2018-2 CMPE 260.01

**PRINCIPLES OF PROGRAMMING LANGUAGES**

## PROLOG ASSIGNMENT

Şadi Uysal

2015400162

Submission Date: 21.04.2018

Problem Description

The file cl base.pl posted on the class contains a database of facts about 4 teams and the played matches. The predicates that will be used are as follows:

**team(teamName, hometown).**

**match(week, homeTeam, homeTeamScore, awayTeam, awayTeamScore).**

The following is a portion of a sample database illustrating the two relations that are defined: team(realmadrid, madrid).

**team(juventus, torinp).**

**match(2, juventus, 2, galatasaray, 2).**

To illustrate, match(2, realmadrid, 4, kobenhavn, 0). implies that “In the 2nd week, realmadrid defeated kobenhavn with the score of 4-0.

Our Tasks

**-All Teams**

Implement the predicate allTeams(L,N). L is the list containing all the teams in the database where N is the number of elements in the list. In this predicate, you should be able to specify queries with any combination of variables and constants.

**-Match Results**

**•** **wins(T,W,L,N)** implies that L involves the teams defeated by team T when we are in week W and N is the number of elements in L.

**• losses(T,W,L,N)** implies that L involves the teams that defeated team T when we are in week W and N is the number of elements in L.

**• draws(T,W,L,N)** is very similar but now L involves the teams that team T could not defeat also did not lose to.

**-Goals Socred and Conceded**

Implement **scored(T,W,S)** where S is the total number of goals scored by team T up to (and including) week W.

Implement **conceded(T,W,C)** where C is the total number of goals conceded by team T up to week W. Assume T and W are given as constants.

**-Total Average of a Team**

Implement **average(T,W,A)** where A is the average (goals scored – goals conceded) of a team T gathered up to (and including) week W. Assume T and W are given as constants.

**-Order and Top Three**

Implement **order(L,W)** where W (week) is given as constant and league order in that week will be retrieved in L.

Additionally, implement **topThree([T1,T2, T3],W)** where T1 T2 and T3 are the top teams when we are in the given week W. Assume that the order is decided according to average i.e 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.

The user will start running the program firstly by opening file predicates.pl with prolog and after the command “[database\_name].” , the user will be ready to use program with any command.

In Prolog you can continue querying by pressing ; and the next result is retrieved. If there is no other result left, then it returns False. If you don’t make a new query and press enter it returns True.

Below, some input- output explanations and examples.

In the first query, I pressed ; two times and then I pressed enter. In the second and the third query, I checked if the given data is correct according to knowledge base. In the fourth query I asked to show list of teams and in the fifth query I asked to show number of teams.

**?- allTeams(L,N).**

**L = [galatasaray, realmadrid, juventus, kobenhavn, manutd, realsociedad, shaktard, bleverkusen, omarseille, arsenal, fcnapoli, bdortmund]**

**N = 12;**

**L = [galatasaray, realmadrid, juventus, kobenhavn, manutd, realsociedad, shaktard, bleverkusen, omarseille, arsenal, bdortmund, fcnapoli]**

**N = 12;**

**L = [galatasaray, realmadrid, juventus, kobenhavn, manutd, realsociedad, shaktard, bleverkusen, omarseille, bdortmund, arsenal, fcnapoli]**

**N = 12 True**

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

**?- allteams([], 12). False**

**?- allteams(L, 12).**

**L = [galatasaray, realmadrid, juventus, kobenhavn, manutd, realsociedad, shaktard, bleverkusen, omarseille, bdortmund, arsenal, fcnapoli] True**

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

**N = 12 False**

**?- wins(galatasaray,4,L,N).**

**L = [kobenhavn]**

**N = 1 ; False**

**?- losses(galatasaray,4,L,N).**

**L = [realmadrid, kobenhavn]**

**N = 2 ; False**

**?- draws(galatasaray,4,L,N).**

**L = [juventus]**

**N = 1 ; False**

**?- draws(galatasaray,4,[juventus],N).**

**N = 1 ; False**

**?- draws(galatasaray,4,L,1).**

**L = [juventus] ; False**

**?- draws(galatasaray,4,[juventus],1).**

**True**

**?- scored(juventus,5,S).**

**S = 9**

**?- conceded(juventus,5,C).**

**C = 8**

**?- order(L, 6).**

**L = [realmadrid, manutd, bdortmund, arsenal, fcnapoli, shaktard, juventus, bleverkusen, galatasaray, realsociedad, kobenhavn, omarseille]**

**?-topThree(L, 6).**

**L = [realmadrid, manutd, bdortmund]**

* Program Structure

**allTeams(L,N):-**

**findall(X,team(X,\_),L0), /\* for team list\*/**

**permutation(L0,L), /\* for permutation of lists\*/**

**length(L0,N).**

**wins(T,W,L,N):-**

**winsW(T,W,L), /\* recursive predicate to find wins\*/**

**length(L,N). /\* finds length of wins \*/**

**winsW(T,W,L):-**

**sth(W),**

**findall(X,someLogic(X,T,W),L1), /\*Team is home? \*/**

**findall(X,someLogic2(X,T,W),L2), /\* Team is away\*/**

**append(L1,L2,L3), /\*finds win for that week if it exist \*/**

**decr(W,W1), /\* decrement week\*/**

**winsW(T,W1,L4), /\* look for previous week\*/**

**append(L3,L4,L). /\* append recursively week's wins \*/**

**winsW(T,0,[]). /\* to end recursive loop\*/**

**losses(T,W,L,N):- /\* same structure for losses\*/**

**lossesW(T,W,L),**

**length(L,N).**

**lossesW(T,W,L):-**

**sth(W),**

**findall(X,someLogic(T,X,W),L1),**

**findall(X,someLogic2(T,X,W),L2),**

**append(L1,L2,L3),**

**decr(W,W1),**

**lossesW(T,W1,L4),**

**append(L3,L4,L).**

**lossesW(T,0,[]).**

**draws(T,W,L,N):- /\* same structure for draws\*/**

**drawsW(T,W,L),**

**length(L,N).**

**drawsW(T,W,L):-**

**sth(W),**

**findall(X,someLogic3(X,T,W),L1),**

**findall(X,someLogic3(T,X,W),L2),**

**append(L1,L2,L3),**

**decr(W,W1),**

**drawsW(T,W1,L4),**

**append(L3,L4,L).**

**drawsW(T,0,[]).**

**someLogic(X,T,W):- /\* finds home team's wins or losses\*/**

**match(W,T,S1,X,S2),**

**S1>S2.**

**someLogic2(X,T,W):- /\* finds away team's wins or losses \*/**

**match(W,X,S1,T,S2),**

**S1<S2.**

**someLogic3(X,T,W):- /\* finds away or home team's draws \*/**

**match(W,X,S1,T,S2),**

**S1=S2.**

**scored(T,W,S):-**

**scoredW(T,W,S0), /\*recursively finds scored goals \*/**

**sum\_list(S0,S). /\* sums scores \*/**

**conceded(T,W,C):-**

**concededW(T,W,C0), /\*same structure to find conceded goals \*/**

**sum\_list(C0,C).**

**scoredW(T,W0,S0):-**

**sth(W0), /\* condition for week>0\*/**

**findall(X,someLogic4(X,T,W0),S1), /\* team is home?\*/**

**findall(X,someLogic5(X,T,W0),S2), /\* team is away?\*/**

**append(S1,S2,S3), /\* append scores whether team is away or home\*/**

**decr(W0,W1), /\* decrement week\*/**

**scoredW(T,W1,S4), /\*recursively find scores of previous weeks \*/**

**append(S3,S4,S0). /\* append scores list\*/**

**scoredW(T,0,[]). /\* to end recursive loop\*/**

**concededW(T,W0,C0):- /\*same structure to find conceded goals \*/**

**sth(W0),**

**findall(X,someLogic6(X,T,W0),C1),**

**findall(X,someLogic7(X,T,W0),C2),**

**append(C1,C2,C3),**

**decr(W0,W1),**

**concededW(T,W1,C4),**

**append(C3,C4,C0).**

**concededW(T,0,[]).**

**someLogic4(X,T,W):-**

**match(W,\_,\_,T,X).**

**someLogic5(X,T,W):-**

**match(W,T,X,\_,\_).**

**someLogic6(X,T,W):-**

**match(W,\_,X,T,\_).**

**someLogic7(X,T,W):-**

**match(W,T,\_,\_,X).**

**decr(X,NX) :-**

**NX is X-1 .**

**sth(X):-**

**X>0.**

**average(T,W,A):- /\* scored goals-conceded goals\*/**

**scored(T,W,S),**

**conceded(T,W,C),**

**A is (S-C).**

**order(L,W):-**

**setof([X,Y],someLogic8(X,W,Y),Z), /\*finds some [Team,Average] \*/**

**reverse(Z,Z1), /\* to order list \*/**

**findall(X,member([\_,X], Z1),L). /\*get ordered list of teams\*/**

**topThree(L,W):- /\* first three element of ordered list\*/**

**order(L0,W),**

**take(L0,3,L).**

**take(Src,N,L) :- findall(E, (nth1(I,Src,E), I =< N), L). /\*to take first N element of list\*/**

**someLogic8(X,W,Y):- /\* to gather [Team,Average] lists\*/**

**team(Y,\_),**

**average(Y,W,A),**

**X is A.**

End of program.

**The weak points of the program**

* Since checking a list to be whether a permutation of another list or not is costly , that feature has high cost for efficiency of the program.
* Rules for input forms very strict, since we did not code for checking possible wrong inputs.

**The strong points of the program**

* All predicates we used can be explained by similar algorithms since their structures are alike.
* Most of the time we used simple recursions for predicates so that program’s readibility is high.

Since prolog is a new language for me , it was quite hard to understand the basics of language and writing code in that language but as i tried and researched on the internet it was getting better. So the Project provided us a chance to learn basics of logic programming ,prolog and how to deal with problems that we encounter while using that language.