TP2-AI

Zhenxi Li(1883294) Jinling Xing(1915481)

November 6, 2017

1 Introduction

Prolog is functional declarative language. In TP2, we use Prolog to do some exercises. The first question is about the genealogical tree. We have four mathematical problems, such as sum and derivative function. Then, we solved the genealogical search tree and the crossword problem. Use the recursive method and list, we solved six questions. Last, we studied CLPFD and constraint programming, we have two questions about that, the second one is more complex than the first one, but they are related with each other.

2 Genealogical tree(3.7)

2.1 Prolog knowledge base

```
man("Hugo").
woman("Catherine").
man("Loic").
woman("Justine").
woman("Lea").
man("Gabriel").
woman("Alice").
man("Maxime").
man("Mathieu").
man("Alexis").
woman("Rose").
woman("Emma").

parent("Hugo","Catherine","Lea").
parent("Hugo","Catherine","Gabriel").
parent("Loic","Justine","Alice").
parent("Loic","Justine","Maxime").
parent("Loic","Justine","Mathieu").
parent("Gabriel","Alice","Alexis").
parent("Gabriel","Alice","Rose").
parent("Gabriel","Alice","Emma").

child(X,Y) :- parent(Y,Z,X);parent(Z,Y,X).
daughter(X,Y) :- child(X,Y),woman(X).
son(X,Y) :- child(X,Y),man(X).
mother(X,Y) :- child(Y,X),woman(X).
father(X,Y) :- child(Y,X),man(X).
uncle(X,Y) :- child(Y,Z),child(Z,P),child(X,P),man(X).
aunt(X,Y) :- child(Y,Z),child(Z,P),child(X,P),woman(X).
grand_parent(X,Y) :- child(Y,Z),child(Z,X).
grand_father(X,Y) :- grand_parent(X,Y),man(X).
grand_mother(X,Y) :- grand_parent(X,Y),woman(X).
grand_child(X,Y) :- grand_parent(Y,X).
grand_daughter(X,Y) :- grand_child(X,Y),woman(X).
```

```
grand_son(X,Y) :- grand_child(X,Y),man(X).
nephew(X,Y) :- (uncle(Y,X);aunt(Y,X)),man(X).
niece(X,Y) :- (uncle(Y,X);aunt(Y,X)),woman(X).
brother(X,Y) :- child(X,Z),child(Y,Z),man(X).
sister(X,Y) :- child(X,Z),child(Y,Z),woman(X).
```

2.2 Test and Result

?- child("Lea","Hugo").
true .
?- child("lea","Alice").
false.
?- nephew("Alexis","Maxime").
true .
?- nephew("Emma","Maxime").
false.
?- grandmother("Catherine","Rose").
true .
?- grandmother("Hugo","Rose").
false.

2.3 Explanation

We wrote a series of facts, predicates and define some rules of genealogical relationship according to some conventions.

3 Mathematic Methods(4.3)

3.1 Prolog knowledge base

```
sum(X,Y,R) :-  R is X+Y.
max2(X,Y,M) :-  M is X,X>Y;M is Y.
max2(X,Y,Z,M) :- M is X,X>Y,X>Z;M is Y,Y>X,Y>Z;M is Z.
d(X,X,1).
d(C,X,0) :- atomic(C).
d(U+V,X,A+B) :- d(U,X,A),d(V,X,B).
d(U*V,X,U*B+V*A) :- d(U,X,A),d(V,X,B).
d(U^N,X,R) :- atomic(N),N\=X,d(U,X,A),R=N*A*U^(N-1).
d(sin(W),X,Z*cos(W)) :- d(W,X,Z).
d(exp(W),X,Z*exp(W)) :- d(W,X,Z).
d(log(W),X,Z/W) :- d(W,X,Z).
d(cos(W),X,-(Z*sin(W))) :- d(W,X,Z).
d(tan(W),X,(Z*sec(W)^2)) :- d(W,X,Z).
```

3.2 Test and Result

1.Compute the sum of two numbers:
?- sum(1,2,R).
R = 3.
2.Compute the maximum of two numbers
?- max2(2,5,M).
M = 5.
3.Compute the maximum of three numbers:
?- max2(1,44,22,M).
M = 44

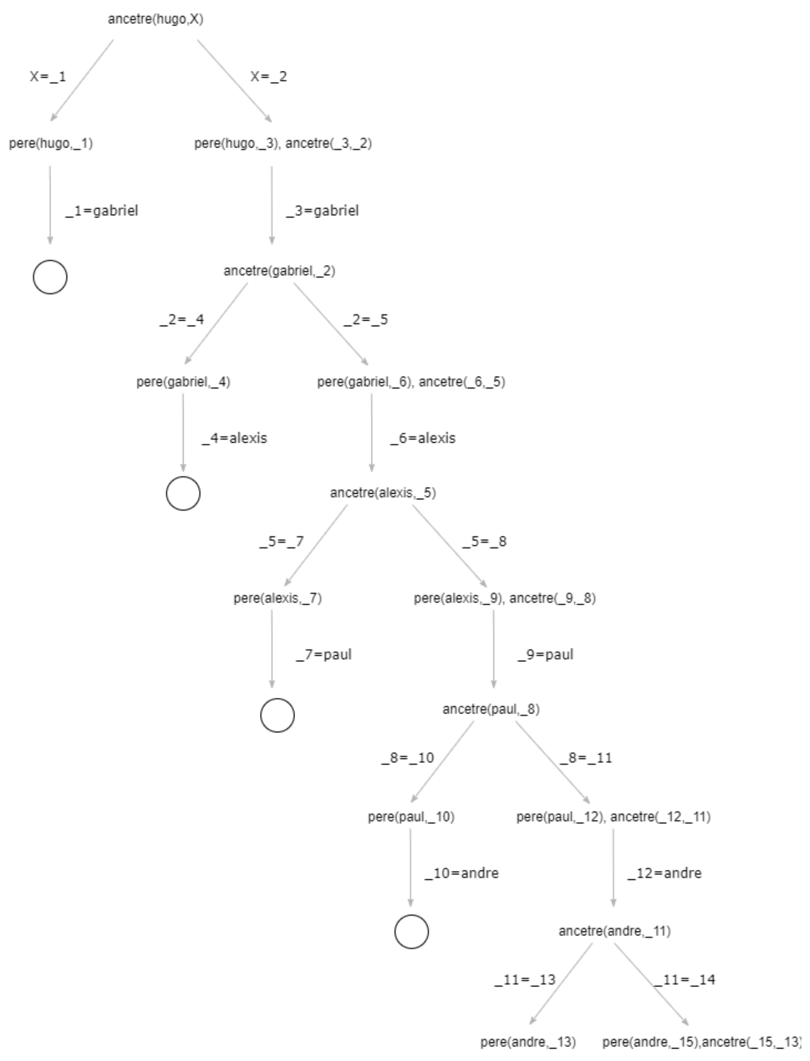
4.Compute the derivative of Function:
$?-d(5*x^3+3,x,R).$
$R = 5*(3*1*x^(3-1)) + x^3*0+0$

3.3 Explanation

We use unification operations to definite above functions.
1)use X is Y, to unify X to the arithmetic value of Y(such as the sum function, unify the sum R to the arithmetic value of the sum of X,Y(X+Y))
2)use X > Y, to compare variables and get the result
3)solve the derivative in a recursive way, list all base situations to solve derivative and solve the general problem in a recursive way

4 Search Tree(4.5.1)

We use abbreviation to represent the variable name, which we shown besides the directed line. We construct the search tree as follows:



5 Crossword problem(4.5.2)

5.1 Prolog knowledge base

```
word(abalone, a, b, a, l, o, n, e).
word(abandon, a, b, a, n, d, o, n).
word(enhance, e, n, h, a, n, c, e).
word(anagram, a, n, a, g, r, a, m).
word(connect, c, o, n, n, e, c, t).
word(elegant, e, l, e, g, a, n, t).

crossword(V1, V2, V3, H1, H2, H3) :-
word(V1, V11, V12, V13, V14, V15, V16, V17),
word(V2, V21, V22, V23, V24, V25, V26, V27),
word(V3, V31, V32, V33, V34, V35, V36, V37),
word(H1, H11, H12, H13, H14, H15, H16, H17),
word(H2, H21, H22, H23, H24, H25, H26, H27),
word(H3, H31, H32, H33, H34, H35, H36, H37),
H12==V12, H14==V22, H16==V32, H22==V14,
H24==V24, H26==V34, H32==V16, H34==V26, H36==V36.
```

5.2 Test and Result

?- crossword(V1, V2, V3, H1, H2, H3).
V1 = abalone,
V2 = anagram,
V3 = connect,
H1 = abandon,
H2 = elegant,
H3 = enhance

5.3 Explanation

We defined a predicate crossword(V1,V2,V3,H1,H2,H3) that gives the problem solution. Because of the cross of rows and columns are same words, we defined the equation, such as $H12 == V12$.

6 Recursive problem(4.8)

6.1 Prolog knowledge base

```
accMax([H|T], A, Max) :- H > A, accMax(T,H,Max).
accMax([H|T], A, Max) :- H =< A, accMax(T,A,Max).
accMax([], A, A).
max(L, M) :-  L=[H|_], accMax(L, H, M).
%compute the sum of all elements in the list(two methods)
somme1([], 0).
somme1([L|T], S) :- somme(T, Rest), S is L + Rest.
somme(L, S) :- L=[H|T], somme(T, S1), S is S1+H.
somme([], S) :- S is 0.
%find the n-th element of a list(two methods)

nth1(0,[H|_],H).
nth1(N,[_|T],H):- N>0,N1 is N-1, nth(N1,T,H).

nth(N, L, R) :- L=[H|T], N1 is N-1, nth(N1, T, R).
nth(1, L, R) :- L=[H|_], R is H.
```

```
%write a method that zips two lists zip(L1,L2,R). example : zip([1,2,3],[a,b,c],R)gives R = [[1,a]
zip(L1, L2, R) :- L1=[H1|T1], L2=[H2|T2], zip(T1,T2,R1), R=[[H1,H2]|R1].
zip(L1, L2, R) :- L1=[H1], L2=[H2], R=[[H1,H2]].
zip1([],[],[]).
zip1([H1|L1],[H2|L2],[[H1,H2]|R]):- zip(L1,L2,R).

enumerate(0,L) :- L=[0].
enumerate(N,L) :- N1 is N-1, enumerate(N1, L1), L=[N|L1].

give_change(Money,Price) :- Change is Money-Price, Ipart is floor(Change),
    N2 is div(Ipart, 2), N1 is mod(Ipart, 2), Fpart is floor((Change-Ipart)*100),
    N25 is div(Fpart, 25),
    N10 is div(Fpart-25*N25, 10), N05 is div(Fpart-25*N25-10*N10, 5),
    write('A rendre:'), nl, write(N05), write(' coin of 0.05'), nl,
    write(N10), write(' coin of 0.10'), nl,
    write(N25), write(' coin of 0.25'), nl,
    write(N1), write(' coin of 1'), nl,
    write(N2), write(' coin of 2').
```

6.2 Test and Result

1.Compute the maximum over a list. The testing result is as follows:
?- max([1,2,3],X).
X = 3
2.Computes the sum of all elements in a list. The testing result is as follows(two methods):
?- somme1([1,2,3,4],S).
S = 10.
?- somme([1,2,3,4,5,6],S).
S = 21.
3.Return the n-th element of a list.
?- nth1(4,[1,2,3,4,5,6,7,8],H).
H = 4.
4.A method of zips
?- zip([1,2,3,4,5,6],[a,b,c,d,e,f],R).
R = [[1, a], [2, b], [3, c], [4, d], [5, e], [6, f]]
5.enumerates all number from 0 to N in the list
?- enumerate(9,L).
L = [9, 8, 7, 6, 5, 4, 3, 2, 1|...]
6.prints the change to do if we pay Price with Money.
?- givechange(45,30).
A rendre:
0 coin of 0.05
0 coin of 0.10
0 coin of 0.25
1 coin of 1
7 coin of 2
true.

6.3 Explanation

The key point to solve this problem is that we need to construct the recursive formula by list.

7 CLPFD and constraint programming(4.9)

7.1 Prolog knowledge base(problem 1)

```
:- use_module(library(clpfd)).
```

```
people(1).
people(2).
people(3).
getPeople(P, ID):- (ID#=1, P=british);(ID#=2, P=spanish); (ID#=3, P=japanese).
animal(1).
animal(2).
animal(3).
getAnimal(A, ID):- (ID#=1, A=jaguar);(ID#=2, A=snail);(ID#=3, A=snake).
pos(1).
pos(2).
pos(3).
color(1).
color(2).
color(3).
getColor(C, ID):- (ID#=1, C=red); (ID#=2, C=blue); (ID#=3, C=unknowncolor).

house(C, Pos) :- color(C), pos(Pos).
owns(P, A, C, Pos) :- people(P), animal(A), house(C, Pos).

snakeowner(X) :-
        owns(P1, A1, C1, Pos1), P1 in 1, C1 in 1, % member(P1, [british]), member(C1, [red]),
        owns(P2, A2, C2, Pos2), P2 in 2, A2 in 1, % member(P2, [spanish]), member(A2, [jaguar]),
        owns(P3, A3, C3, Pos3), P3 in 3, % member(P3, [japanese]),

        owns(P4, A4, C4, Pos4), A4 in 2, Pos3 #= Pos4+1, % member(A4, [snail]), Pos3 #= Pos4+1,
        owns(P5, A5, C5, Pos5), P4 #= Pos5-1, C5 in 2, % member(C5, [blue]),
        all_different([P1, P2, P3]),
        all_different([A1, A2, A3]),
        all_different([C1, C2, C3]),
        all_different([Pos1, Pos2, Pos3]),

        L = [[P1, A1, C1, Pos1], [P2, A2, C2, Pos2],
             [P3, A3, C3, Pos3]],
        tuples_in([[P4, A4, C4, Pos4]], L),
        tuples_in([[P5, A5, C5, Pos5]], L),

        owns(P6, A6, C6, Pos6), A6 in 3, % member(A6, [snake]),
        tuples_in([[P6, A6, C6, Pos6]], L),
        getPeople(X, P6).
```

7.2 Test and Result

?- snakeowner(X).
X = japanese

7.3 Explanation

The problem gives us three persons, three animals and two color. We assume the color number is also three, one of these we labeled unknown. First, we did this question by labeling all the people, color and animals. Then we defined snakeowner(X) to add the constraints given by questions and we use the $all_different()$ to describe the variables are totally different.

7.4 Prolog knowledge base(problem 2)

```
:- use_module(library(clpfd)).


people(1). % British
people(2). % Spanish
people(3). % Ukrainian
people(4). % Japanese
people(5). % Noregian
getPeople(X, ID) :- (ID#=1, X=british);(ID#=2, X=spanish);(ID#=3, X=ukrainian);(ID#=4, X=japanese)

animal(1). % dog
animal(2). % snail
animal(3). % fox
animal(4). % Zebra
animal(5). % horse

drink(1). % coffee
drink(2). % tea
drink(3). % milk
drink(4). % orange juice
drink(5). % water

job(1). % sculptor
job(2). % diplomat
job(3). % doctor
job(4). % violinist
job(5). % Acrobat

color(1). % red
color(2). % green
color(3). % white
color(4). % yellow
color(5). % blue

pos(1).
pos(2).
pos(3).
pos(4).
pos(5).

problem([[P1,P2,P3,P4,P5],
[A1,A2,A3,A4,A5],
[D1,D2,D3,D4,D5],
[J1,J2,J3,J4,J5],
[C1,C2,C3,C4,C5],
[T1,T2,T3,T4,T5]]).

answer(ANS1, ANS2) :-
problem(X),
length(X, 6),
append(X, Vs), Vs ins 1..5,
maplist(all_distinct, X),
X = [Ps, As, Ds, Js, Cs, Ts],
Ps = [P1, P2, P3, P4, P5],
As = [A1, A2, A3, A4, A5],
Ds = [D1, D2, D3, D4, D5],
Js = [J1, J2, J3, J4, J5],
Cs = [C1, C2, C3, C4, C5],
Ts = [T1, T2, T3, T4, T5],
P1 = 1, C1 = 1,
P2 = 2, A2 = 1,
P3 = 3, D3 = 2,
P4 = 4, J4 = 5,
P5 = 5, T5 = 1,

nth0(I1, [C1,C2,C3,C4,C5], 3),
nth0(I1, [T1,T2,T3,T4,T5], E1),

nth0(I2, [D1,D2,D3,D4,D5], 1),
nth0(I2, [C1,C2,C3,C4,C5], 2),
nth0(I2, [T1,T2,T3,T4,T5], E2),
E2 #= E1+1,

nth0(I3, [A1,A2,A3,A4,A5], 2),
nth0(I3, [J1,J2,J3,J4,J5], 1),

nth0(I4, [J1,J2,J3,J4,J5], 2),
nth0(I4, [C1,C2,C3,C4,C5], 4),
nth0(I4, [T1,T2,T3,T4,T5], E5),

nth0(I5, [D1,D2,D3,D4,D5], 3),
nth0(I5, [T1,T2,T3,T4,T5], 3),

nth0(I6, [A1,A2,A3,A4,A5], 3),
nth0(I6, [T1,T2,T3,T4,T5], E3),

nth0(I7, [J1,J2,J3,J4,J5], 3),
nth0(I7, [T1,T2,T3,T4,T5], E4),
(E4 #= E3+1; E4 #= E3-1),

nth0(I8, [A1,A2,A3,A4,A5], 5),
nth0(I8, [T1,T2,T3,T4,T5], E6),
(E6 #= E5+1; E6 #= E5-1),

nth0(I9, [J1,J2,J3,J4,J5], 4),
nth0(I9, [D1,D2,D3,D4,D5], 4),

nth0(I10, [C1,C2,C3,C4,C5], 5),
nth0(I10, [T1,T2,T3,T4,T5], 2),

nth0(I11, [D1,D2,D3,D4,D5], 5),
nth0(I11, [P1,P2,P3,P4,P5], ANS1_P),

nth0(I12, [A1,A2,A3,A4,A5], 4),
nth0(I12, [P1,P2,P3,P4,P5], ANS2_P),

getPeople(ANS1, ANS1_P),% who drinks water
getPeople(ANS2, ANS2_P). % who owns a Zebra
```

7.5 Test and Result

?- answer(ANS1, ANS2).
ANS1 = noregian,

ANS2 = japanese
Noregian drinks water and Japanese owns a Zebra.

7.6 Explanation

The problem gives us five persons, five animals and five colors, five jobs and five drinks. First, we did this question by labeling all the people, colors, animals, drinks and jobs. Then we defined answer(ANS1, ANS2) to add the constraints given by questions and we use problem() to construct the five people, drinks, animals, colors and jobs.