Review on Prolog

Example 1

```
contact_list([],List,List).
contact_list([Elem|List1], List2,[Elem|List3]):-
contact_list(List1,List2,List3).
```

```
? contact_list(X,Y,[a,b,c,d]).
```

Example 1 (Solution)

X = [],
Y = [a, b, c, d]

X = [a],
Y = [b, c, d]

X = [a, b],
Y = [c, d]

X = [a, b, c],
Y = [d]

X = [a, b, c, d],
Y = []

(Try to solve it yourself and write all the steps of unification/Resolution)

Example 2

```
solve([_|T],X):- solve(T,X).
solve([H|_],H).

solve([a,a,a,b,b],Y).
```

X = b
X = b
X = a
X = a
X = a

Pay Attention to the order of found unifications and resolutions

Example 2 Solution elaboration

solve([a,a,a,b,b],Y)=> unification solve([_|a,a,b,b],Y)=>resolution:-solve([a,a,b,b,Y]) =>unification solve([_|a,a,b,b],Y)=>resolution:-

solve([a,a,b,b,X1],Y)=> unification solve([_|a,b,b],Y)=>resolution:-

solve([a,b,b],Y) =>unification solve([_|b,b],Y)=>resolution:- solve([b,b],Y) =>unification solve([_|b],Y)=> resolution:- solve(b,Y)

=>unification with fact solve([H|_],H) => Y=b **(1st found answer)**

Backtrack to unification of solve([b,b],Y) with the fact (second line)=> solve(b|_,b)=> Y=b **(2nd found answer)**

Backtrack: unification of solve([a,b,b],Y) with the fact(second line)=>solve(a|_,a) => Y=a **(3rd found answer)**

Backtracking: unification of solve([a,a,b,b],Y) with the fact (second line) => solve(a|_, a)=> Y=a **(4th found answer)**

Backtracking: unification of solve([a,a,a,b,b],Y) with the fact (second line) => solve(a|_, a)=> Y=a **(5th found answer)**

Example 3

```
f(0,0).
f(1,1).
f(N,X):- N1 is N-1, N2 is N-2, f(N1,X1), f(N2,X2), X is X1+X2.
```

? f(4,X)

Example 3 (Solution)

f(4,X) => unification with the rule (line 3):
:- N1 is 3, N2 is 2, f(3,X1), f(2,X2), X is X1+X2.

=> f(3,X1)=>unification:- N3 is 2, N4 is 1, f(2,X3), f(1,X4), X1 is X3+X4.
: X4=1
=> f(2,X2)=> N5 is 1, N6 is 0,f(1,X5),f(0,X6), X2 is X5+X6
: X5=1, X6=0 => X2=1

f(2,X3):- N7 is 1, N8 is 0, f(1,X7), f(0,X8), X3 is X7+X8
: X7=1, X8=0, X3 is 1
X1= 1+1=2
X= 2(X1) + 1(X2)=**3 (Final answer)**

Example 4

```
function(1,_,[]).
function(0,_,[]).
function(_,[],[]).
function(M,[A|A1],[A|B1]) :- M > 1, C is M-1,number(A), function(C,A1,B1).
function(M,[A|A1],[A|B1]) :- M > 1, C is M-2,\+ number(A), function(C,A1,B1).
```

(rule 1)

(rule 2)

?function(6, [8,a,11,4], M).

Solution

function(6, [8,a,11,4], M).
⇒Unification with head of rule 1: function(6,[8,a,11,4], [8|B1])
⇒Resolution:-function(5,[a,11,4], B1)
⇒Unification with head of rule 2: function (5,[a,11,4], [a|B2])
⇒Resolution: :-function(3,[11,4],B2)
⇒Unification with head of rule 1: function(3,[11,4], [11|B3])
⇒Resolution: :-function (2,[4],B3)
⇒Unification with head of rule 1: function (2,[4], 4|B4)
⇒Resolution :- function(1,[],B4).
⇒Unification with fact: B4=[] => B3(4|B4)= [4] => B2=[11,4] => B1= [a,11,4] =>
M=[8,a,11,4]

Recommendation

- Review all the Examples solved in the class and textbook.
- Review **Cut !** In prolog
- In multiple-choice question, first remove irrelevant answers to save your time. Maybe you do not need to solve the whole problem to find the right answer.
- If a ground query is false means there is no fact or rule in data base to be matched with
- findall and forall are different! (what is the difference?)
- Constants always start with a lower case letter and variables start with an upper case letter:

Example:

func (func,X):-.....

func: name of rule

fun inside the parenthesis is constant