Lecture -1-

Subject: **Introduction**

مدرس مادة الذكاء الاصطناعي – العملي

Lec. Laheeb K. Qurban

Lecture one:

Contents:

1. Introduction to prolog language

Prolog: is a computer programming language that is used for solving problems involves objects and relationships between objects.

Example:

“John owns the book”

Owns (john,book) relationship(object1,object2)

The relationship has a specific order, johns own the book, but the book does not owns john, and this relationship and its representation above called fact.

♦we are using rule to describe relationship between objects.

Example: the rule" two people are sisters if they are both female and have the same parents"

1. Tell us something about what it means to be sisters.

2. Tell us how to find if two people are sisters, simply: check to see if they are both female have the same parents.

2.Component of computer programming in prolog

Computer programming in prolog consist of:

1. Declaring some facts about object and their relationships.

2. Defining some rules about objects and their relationships.

3. Asking questions about objects and their relationships.

if we write our rule about sisters, we could then ask the questions whether Mary and Jane are sisters.

Prolog would search through what we told it about Mary and Jane, and come back with the answer Yes or No, depending on what we told it earlier.

So, we can consider prolog as a store house of facts and rules, and it uses the facts and rules to answer questions.

♦ prolog is a conversational language. Which means you and the computer carry out a kind of conversation, typing a letter from keyboard and displaying it at the screen, prolog work like this manner, prolog will wait for you to type in facts and rules that certain to the problem you want to solve? Then if you ask the right kind of questions prolog will work out the answers and show them.

3. Some of prolog language characteristics:

1. We can solve a particular problem using prolog in less no of line of code.

2. It's an important tool to develop AI application and ES.

3. Prolog program consist of fact and rule to solve the problem and the output is all possible answer to the problem.

4. Prolog language is a descriptive language use the inference depend o n fact and rule we submit to get all possible answer while in other language the 3programmer must tell the computer on how to reach the solution by gives the instruction step by step.

4. Prolog language uses:

1. Construct NLI (Natural Language Interface).

2. Translate language.

3. Constructor symbolic manipulation language packages.

4. Implement powerfully database application.

5. Construct expert system programs.

5. Prolog language component

Facts

Is the mechanism for representing knowledge in the program.

Syntax of fact:

1. The name of all relationship and objects must begin with a lower -case letter, for example likes (john, mary).

2. The relationship is written first, and the objects are written separated by commas, and enclosed by a pair of ro und brackets.

Like (john, mary)

3. The full stop character '.' Must come at the end of fact.

Example:

Gold is valuable valuable (gold).

Jane is female female (jane).

John owns gold owns (johns, gold).

Johns is the father of Mary father (john, marry).

The names of objects that are enclosed within the round brackets are 4 called arguments. And the name of relationship called predicates

Relationship has arbitrary number of argument. If we want to define predicate called play, were we mention two players and a game they play with each other, it can be:

Play (john, Mary, football).

In prolog the collection of facts is called database.

Rules

Rules are used when you want to say that a fact depends on a group of other facts, and we use the following syntax:

1. One fact represents the head (conclusion).

2. The word if used after the head and represented as ":-'.

3. One or more fact represents the requirement (condition).

The syntax of if statement

If (condition) then (conclusion)

[Conclusion: - condition] rule

Example:

I use the umbrella if there is rain

Conclusion condition

Represent both as fact like:

Wheatear (rain).

Use (umbrella)

Use (Iam, umberella):-whether (rain).

Questions

Question used to ask about facts and rules. 5

Question look like the fact and written under the goal program section while fact and rule written under clauses section.

Example: for the following fact owns (mary, book).

We can ask: Does mary own the book in the following manner:

Goal:

Owns (mary,book)

When Q is asked in prolog, it will search through the database you typed before, it look for facts that match the fact in the question.

Two fact matches if their predicates are the same and their corresponding argument are the same, if prolog finds a fact that matches the question, prolog will respond with Yes, otherwise the answer is No.

Variables

If we want to get more interest information about fact or rule, we can use variable to get more than Yes/No answer.

*variables dose not name a particular object but stand for object that we cannot name.

*variable name must begin with capital letter.

*using variable we can get all possible answer about a particular fact or rule.

*variable can be either bound or not bound.

Variable is bound when there is an object that the variable stands for.

The variable is not bound when what the variable stand for is not yet known.6

Example:

Fact

Like (john, mary).

Like (john, flower).

Like (ali, mary).

Questions:

1. Like (john,X)

X= mary

X = flower

2. like(X, mary)

X=john

3. Like(X, Y)

X=john Y=flower

X=john Y=mary

X=ali Y=mary

5. Type of questing in the goal

There are three type of question in the goal summarized as follow:

1. Asking with constant: prolog matching and return Yes/No answer.

2. Asking with constant and variable: prolog matching and produce result for the Variable.

3. Asking with variable: prolog produce result.

Example:

Age(a,10).

Age(b,20).

Age(c,30). 7

Goal:

1.Age(a,X). ans:X=10 Type2

2.age(X,20). Ans:X=b Type2

3.age(X,Y). ans: X=a Y=10, X=b Y=20, X=c Y=30. Type3

4.Age(_,X). ans:X=10 , X=20, X=30. ‘_’ means don’t care Type3

5.Age(_,_). Ans:Yes Type1

Lecture -2-

Subject:

Data type

مدرس

مادة الذكاء الاصطناعي – العملي

Lec. Laheeb K. Qurban

Lecture Two:

1. Data type.

2. Program structure.

3. Read and write functions.

4. Arithmetic and logical operation.

1. data type

Prolog supports the following data type to define program entries.

1. Integer: to define numerical value like 1, 20, 0, -3,-50, ect.

2. Real: to define the decimal value like 2.4, 3.0, 5, -2.67, ect.

3. Char: to define single character, the character can be of type small

letter or capital letter or even of type integer under one condition it

must be surrounded by single quota. For example, 'a','C','123'.

4. string : to define a sequence of character like "good" i.e define word or statement entries the string must be surrounded by double quota for example "computer", "134", "a". The string can be of any length andtype.

5. Symbol: anther type of data type to define single character or

sequence of character but it must begin with small letter and don't

surround with single quota or double quota.

2. program structureProlog program structure consists of five segments, not all of them must appear in each program. The following segment must be included ineach program predicates, clauses, and goal.

1. Domains: define global parameter used in the program.

Domains 9

I= integer

C= char

S = stri ng

R = real

2. Data base: define internal data base generated by the program

Database

Greater (integer)

3. Predicates: define rule and fact used in the program.

Predicates

Mark(symbol,integer).

4. Clauses: define the body of the program.. For the above predicates the

clauses portion may contain Mark (a, 20).

5.Goal: can be internal or external, internal goal written after clauses

portion , external goal supported by the prolog compiler if the program

syntax is correct

This portion contains the rule that drive the program execution.

2. mathematical and logical operation

a .mathematical operation:

operation symbol

addition +

subtraction -multiplication *

Integer part of division div

Remainder of division mod10

B .logical operation

operation symbol

greater >

Less than <

Equal =

Not equal <>

Greater or equal >=

Less than or equal <=

3. Other mathematical function

Function name operation

Cos(X) Return the cosine of its argument

Sine(X) Return the sine of its argument

Tan(X) Return the tranget of its argument

Exp(X) Return e raised to the value to which X is bound

Ln(X) Return the natural logarithm of X (base e)

Log(X) Return the base 10 logarithm of log 10x

Sqrt(X) Return the positive square of X

Round(X) Return the rounded value of X. Rounds X up or down to the nearest integer

Trunc(X) Truncates X to the right of the decimal point

Abs(X) Return the absolute value of X

4. Read and write function

Read function:

readint(Var) : read integer variable.

Readchar(Var) : read character variable.

Readreal(Var) : read read (decimal) variable.

Readln(Var) : read string.

Write function

Write(Var) : write variable of any type.

Example 1: write prolog program to read integer value and print it.

Domains 11

I = integer

Predicates

print.

Clauses

Print: - write ("please read integer number"), readint(X),

write("you read",X).

Goal

Print.

Output:

Please read integer number 4

You read 4

Example2: write prolog program that take two integer input us integer

and print the greater.

Domains

I = integer

Predicates

Greater (i,i)

Clauses

Greater(X,Y):- X>Y,write(“the greater is”,X).

Greater(X,Y):- write (“ the greater is “,Y).

Goal

Greater(4,3).

Output:

The greater is 4

H.W:

1. write prolog program that read any phrase then print it.

2.write prolog program that read an integer number then print it after

multiplying it by any other integer like 5.

Lecture -3-

Subject:

More Examples

مدرس مادة الذكاء الاصطناعي – العملي

Lec. Laheeb K. Qurban

This lecture present several example that intended to display variousway to write prolog program, how to write if –else program ,divide problem into several parts then combine them in a single rule and how to write program describe specific problem.

Example 1:write prolog program to check if the given number is positiveor negative.

Basic rule to check the number

If X>=0 then

X is positive

Else

X is negative

Domains

I= integer

Predicates

Pos_neg(i(

Clauses

Pos_neg(X): -X>=0, write(“positive number”),nl.

Pos_neg(_): -write(“negative number”),nl.

Goal

Pos_neg(4)

Output:

Positive number

Note: nl mean new line.

Example 2: write prolog program to check if a given number is odd oreven.

Basic rule to check number13

If X mod 2=0 then

X is even number

Else

X is odd number

Predicates

Odd_even(integer(

Clauses

Odd_even(X): -X mod 2= 0, write ("even number"), NL.

Odd_even(X):- write ("odd number"), nl.

Goal

Odd_even(5(

Output

Odd number

Example 3:write prolog program to combine both rule in example 1 andexample2.

Domains

I= integer

Predicates

Pos_neg(i)

Odd_even(i)

Oe_pn(i)

Clauses

Oe_pn(X):-pos_neg(X),odd_even(X).

Odd_even(X):-X mod 2= 0, write(“ even number”),nl.

Odd_even(X):- write(“odd number”),nl.

Pos_neg(X): -X>=0, write(“positive number”),nl.

Pos_neg(_): -write(“negative number”),nl.

Goal

Oe_pn(3)

Output:

Odd number

Positive number 14

Note: the rule of same type must be gathering with each other.

Example 4 :write prolog program to describe the behavior of the logical And gate.

Truth table of And gate

X Y Z

0 0 0

0 0 1

0 1 0

1 1 1

Sol 1:

Domains

I= integer

Predicates

And1(I, I , I)

Clauses

And1(0,0,0).

And1(0,1,0).

And1(1,0,0).

And1(1,1,1).

Goal

And1 (0,1,Z)

Output:

Z =0

Sol 2:

From the truth table we can infer the following rule:

If X= Y then

Z= X

Else

Z =0 15

Domains

I= integer

Predicates

And1 (I ,I, I)

Clauses

And1 (X,Y,Z): - X=Y, Z=X.

And1(X,Y,Z): - X<> Y, Z=0.

Goal

And1(0,0,Z)

Output

Z=0

H.W

.1Write prolog program that read character and check if it's a capitalletter, small letter, digit or special character.

.2Modify prolog program in example 3 such that the value of X is readinside the program.

.3Write prolog program that describe the operation of logical Or gate

Lecture -4-

Subject:

1. **Cut function**
2. **fail function**

مدرس مادة الذكاء الاصطناعي – العملي

Lec. Laheeb K. Qurban

1. cut

Represented as "! " is a built in function always True , used to stop backtracking and can be placed anywhere in the rule, we list the cases

where "!" can be inserted in the rule:

1 .R: -f1, f2,!. "f1, f2 will be deterministic to one solution.

2. R: -f1,!,f2. " f1 will be deterministic to one solution while f 2 to all .

3. R: - !,f1,f2. "R will be deterministic to one solution.

Example1 : **program without use cut.**

```
Domains
I= integer
Predicates
No( I )
Clauses
No (5).
No (7).
No (10).
Goal
No (X).
```

Output:

X=5

X=7

X=10

Example 2: **program using cut.**

Domains

I= integer

Predicates

No(I)

Clauses

No (5): -!.

No (7). 17

No (10).

Goal

No (X).

Output:

X=5.

Example3: program without using cut.

Domains

I =integer

S = symbol

Predicates

a (I)

b (s)

c (I, s)

Clauses

a(10).

a(20)

b(a)

b(c)

c (X, Y): - a (X), b (Y).

Goal

c(X,Y).

Output:

X= 10 Y=a

X=10 Y=c

X=20 Y=a

X=20 Y=c

<u>Example 4</u>: using cut in the end of the rule.

Domains

I =integer

S = symbol

Predicates

a(I)

b (s) 18

c (I, s)

Clauses

a(10).

a(20)

b(a)

b(c)

c (X, Y): - a (X), b (Y),!.

Goal

c(X,Y).

Output:

X= 10 Y=a

Example 5: **using cut in the middle of the rule.**

Domains

I =integer

S = symbol

Predicates

a(I)

b (s)

c (I, s)

Clauses

a(10).

a(20)

b(a)

b(c)

c (X, Y): - a (X),!, b (Y).

Goal

c(X,Y).

Output:

X= 10 Y=a

Y=c19

2. Fail

Built in function written as word “fail” used to enforce backtracking, place always in the end of rule, produce false and can be used with internal goal to produce all possible solution.

Example 6:

Predicates

Student (symbol , integer)

Printout.

Clauses

Student (aymen,95).

Student(zainab,44).

Student(ahmed,60).

Printout: -student(N,M),write(N,” “,M),nl,fail.

Goal

Printout.

Output:

aymen 95

zainab 44

ahmed 60

No

Example 7:

Predicates

Student (symbol , integer)

Printout.

Clauses

Student (aymen,95).

Student(zainab,44).

Student(ahmed,60).20

Printout: -student(N,M),write(N,” “,M),nl,fail.

Printout.

Goal

Printout.

Output:

aymen 95

zainab 44

ahmed 60

Yes

Lecture -5-

Subjects:

3. **Repetition**
4. **Recursion**

مدرس مادة الذكاء الاصطناعي – العملي

Lec. Laheeb K. Qurban

1- Repetition

In prolog there is a constant formula to generate repetition; thistechnique can generate repetition for some operation until thestopping condition become true.

Example: prolog program read and write a number of characterscontinue until the input character equal to.'#'

Predicates

Repeat.

Typewriter.

Clauses

Repeat.

Repeat: -repeat.

Typewriter: -repeat,readchar(C),write(C),nl,C.!,'#'=

2- Recursion

In addition to have rules that use other rules as part of theirrequirements, we can have rules that use themselves as part of their

requirements.This kind of rule called "recursive "because the relationship in theconclusion appears again in the body of the rule, where the requirementsare specified. A recursive rule is a way of generating a chain of relationship for arecursive rule to be effective. However, there must be some place in thechain of relationship where the recursio n

stops.This stopping condition must be answerable in the database like anyother rule.

2-1**Tail Recursion**

We place the predicate that cause the recursion in the tail of the rule as

shown below:

Head : - p1,p2,p3, head.

Predicates 1

Predicates 2

Predicates 3

Check point

Setup

variables

Output result 23

Example 1: program to print number from n to 1.

Predicates

A (integer)

Clauses

A(1) : - write (1), nl.!,

A(M): - write (M) , nl, M1 = M -1, A(M1).

Goal

A(4)

Output:

4

3

2

1

Yes

Example 2: program to find factorial.

1*2*3*4*5 = !5

Predicates

Fact (integer, integer, integer)

Clauses

Fact(1, F, F)!- :

Fact(N,F,R): - F1=F*N , N1=N-1, fact(N1,F1,R).

Goal

Fact (5,1,F).

Output:

F = 120.24

Example 3: program to find power.

3

4

3*3*3*3 =

Domains

I= integer

Predicates

Power (I,I,I, I).

Clauses

Power (X,Y,P,R): - P1= P*X, Y1 =Y-1, power(X,Y1,P1,R).

Power (_,0,P,P).!- :

Goal

Power(3,2,1,P)

Output

P= 9

2-2 **Non –Tail Recursion (Stack Recursion)**

This type of recursion us the stack to hold the value of thevariables till the recursion is complete. The statement is self – repeated asmany times as the number of items in the stack.. Below a simplecomparison between tail and non-tail recursion.

Tail recursion

.1Call for rule place in the end

of the rule.

.2It is not fast as much as stack

recursion.

.3Use more variable than stack

recursion.

Non-tail recursion

.1Call for the rule place in the

middle in the rule .

.2Stack recursion is fast to

implement.

.3Few parameters are used.

Example 4: factorial program using non-tail recursion.

Predicates

fact(integer,integer).

Clauses

fact(1,1).

fact(N,F): - N>1,N1=N-1,fact(N1,F1),F=N*F1.

Goal

Fact (4,)

Output:

Y =24.

Example 5: power program using non-tail recursion.

Predicates

Power (integer, integer, integer)

Clauses

Power (_,0,1).!- :

Power (X,Y,P) : - Y> 0, Y1=Y -1, power (X,Y1,P1),P= X*P1.

Goal

Power (3,2,Z)

Output

Z = 9.26

H.W

.1Write prolog program to fin d the sum of 10 integer element using tailand non tail recursion.

.2Write prolog program to find the maximum value between 10 elements.

.3Write prolog program to find the minimum value between 10 elements.

.4Find the sum S = 1+2 + 3 ….+N

Lecture 6

String standard predicates

.1Isname(string) test if the content of the string is name or not

Isname("abc") yes

Isname("123"). No

.2char_int(char,integer) convert the character to its integer value and the opposit

Char_int('A',X(

X=65

Char_int(X,65(

X='A'

.3Str_char(string,char) convert the string (of one char) to char and the opposit

Str_char("A",X(

X='A'

Str_char(X,'A('

X="A"

.4str_real (string,real) convert the string (ofreal) to real and the opposit

Str_real("0.5",X(

X=0.5

Str_real(X,0.5(

X="0.5"

.5Fronttoken(string,string,string.(

Take token of word from the string and return the reminder of the string.

Fronttoken(string,token,rem.(

Fronttoken("ab cd ef",X,Y.(

X="ab" y="cd ef"

Fronttoken("c def",X,Y(

X="cd" Y="ef"

.6Frontstring(integer,string,string,string(

Take a string(str) with length specified by the integer value and

return the reminder

Frontstring(integer,string,str,rem)

Frontstr(3,"ahmed",X,Y)

X="ahm" Y="ed"

Frontstr(2,"abcde",X,Y).

X="ab" Y="cde"

Frontstr(3,S,"ahm","ed").

S="ahmed"

7. Frontchar(string,char,string).

Take one char from a specific string and return the reminder

Frontchar(string,char,rem).

Frontchar("ahmed",X,Y)

X='a' Y="hmed"

Frontchar(X,'a',"hmed")

X="ahmed"

8. Str_len(string,length)

Return the length of specific string

Str_len("ahmed",X)

X=5

Str_len("ab",X)

X=2

Str_len("ab",3) no

Str_len(X,3) X="---"

9. Concat(string,string,string).

Concat two string together to produce one string

Concat("ab","cd",X)

X="abcd"

10. Upper_lower(string,string)

Convert the string in upper case(in capital letter) to the lower case

(small letter) and the opposite.

Upper_lower(capital_letter,small_letter)

Upper_lower("ABC",X)

X="abc" 29

Upper_lower("Abc",X)

X="abc"

Upper_lower(,X,"abc")

X="ABC"

Prolog Programs that deal with string

Ex1:Pogram that read two string and concat them in one string as upper

case.

predicates

start(string)

clauses

start(X): -readln(S),readln(S1),concat(S,S1,S2),upper_lower(X,S2).

Goal

Start(X)

Output:

Ahmed

Ali

X=AHMEDALI yes

Ex2:program that read string of one character then return the integer value of this char.

predicates

start(integer).

clauses

start(X): -readln(S),str_char(S,X).

goal

start(X)

Output:

a

X=97

yes30

Ex3: Program that take a string of words and print each word in a line as

upper case.

predicates

start(string).

clauses

start(S): -fronttoken(S,S3,S2), upper_lower(S1,S3), write(S1),

nl,start(S2).

start("").

Goal

Start("ali is a good boy").

Output:

ALI

IS

A

GOOD

BOY

yes

Ex4: program that take a string and convert each character it contain to its corresponding integer value.

Predicates

start(string).

clauses

start(S): -frontstr(1,S,S1,S2), char_int(S1,I), write(I), nl , start(S2).

start("").

Goal

Start("abc").

Output:

97

98

99

Yes 31

Ex5: program that return the number of names in a specific string.

```
predicates
start(string,INTEGER).
clauses
start(S,X): -fronttoken(S,S1,S2),isname(S1),X1=X+1,start(S2,X1).
start(S,X): -fronttoken(S,_,S2),start(S2,X).
start("",X): -write("the number of names is", X).
goal
start("ali has 2 cars").
```

Output:

The no. of names is 3

Yes

Ex6:program that split a specific string to small string with length 3 char.

```
predicates
start(string).
clauses
start("").
start(S): -str_len(S,I),  I MOD 3=0, frontstr(3,S,S1,S2),   write(S1),
```

nl,start(S2).

start(S): -concat(S," ",S1),start(S1).

Goal

Start("abcdefg").

Output:

abc

def

g

yes

H.W

1- Write a prolog program that do the following: convert the string such as

"abcdef" to 65 66 67 68 69 70.

2-Program tofind the number of tokens and the number of character in a

specific string such as: "ab c def" the output is tokens an d 6 character.

LIST IN PROLOG

Lists in Prolog

Lists are ordered sequences of elements that can have any length. Lists can be represented as a special kind of tree.

A list is either empty, or it is a structure that has two components: the head **H** and tail **T**. List notation consists of the elements of the list separated by commas, and the whole list is enclosed in square brackets.

For example:

[a] and [a,b,c], where a, b and c are symbols type. [1], [2,3,4] these are a lists of integer.

[ ] : is the atom representing the empty list. Lists can contain other lists. Split a list into its head and tail using the operation [X|Y].

Examples about Lists

1. p([1,2,3]).

p([the,cat,sat,[on,the,hat]]).

Goal: p([X|Y]).

Output:

X = 1 Y = [2,3] ;

X = the Y = [cat,sat,[on,the,hat]].

2. p([a]).

Goal: p([H | T]).

Output:

H = a, T = [].

3. p([a, b, c, d]).

Goal: p([X, Y | T]).

Output:

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

2. p([a]).

Goal: p([H | T]).

Output:

H = a, T = [].

3. p([a, b, c, d]).

Goal: p([X, Y | T]).

Output:

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

4. P([[a, b, c], d, e]).

Goal: p(H,T)

Output:

H = [a, b, c], T = [d, e].

List Membership

• Member is possibly the most used user-defined predicate (i.e. you have to define it every time you want to use it!).

• It checks to see if a term is an element of a list.

– it returns yes if it is.

– and fails if it isn't.

member(X,[X|_]).

member(X,[_|Y]) :- member(X,Y).

It 1st checks if the Head of the list unifies with the first argument.

If yes then succeed.

If no then fail first clause.

The 2nd clause ignores the head of the list (which we know doesn't match) and recourses

on the Tail.

Goal: member(a, [b, c, a]).

Output: Yes

Goal: member(a, [c, d]).

Output: No.

Print the contents of the list.

print([]).

print([i(X,Y)|T]):-write(X,Y),print(T).

Goal: print([3,4,5])

Output: 3 4 5

Find the maximum value of the list

list([H],H).

list([H1,H2|T],H1):-H1>H2,list([H1|T],_).

list([_,H2|T],H2):-list([H2|T],H2).

Goal: list([3,9,4,5],M)

Output: M=9

4.6 Append two lists

app([],L,L).

app([X|L1],L2,[X|L3]) :-app(L1,L2,L3).

Goal: app([3,4,5],[6,7,8],L)

Output: L=[3,4,5,6,7,8]

Write the contents of list inside at the given list

list([H|T]):-list([H]),list(T).

Goal: list([[3,4,5],[6,7,8]])

Output: 3 4 5 6 7 8

Reveres the contents of the given list.

```
app([],X,X).

app([H|T1],X,[H|T]):-app(T1,X,T).

rev([X],[X]).

rev([H|T],L):-rev(T,L1),app(L1,[H],L).
```

Goal: rev([a,b,c,d],R)

Output: R=[d,c,b,a].

DATABASE IN PROLOG

Prolog Database

It uses to retrieve information from the database. The database predicate retrieves information from the database by matching its argument with any term stored in the database. Until the

DATABASE keyword is implemented, we have to use this or clause to make a non-destructive

database query.

The database contains related predicates such as: **asserta, assertz, retract**,

retractall, these predicates are standard prolog built-in as shown below:

1.asserta() adds Term to the beginning of the database. Term must should bound to an

atom or a struct, but may contain free variables. The database entry is not removed

upon backtracking. Database entries can only be removed by the use of the.

2.assertz() adds Term to the end of the database. Term must should bound to an atom

or a struct, but may contain free variables. The database entry is not removed upon

backtracking. Database entries can only be removed by the use of the retract()

predicate.

3.retract(): it retrieves information from the database by matching its argument with

any term stored in the database. If it matches, the database entry is removed. Upon

backtracking multiple solutions may occur. You may invoke indefinite loops by using retract(X),...,assertz(X),fail.construct.

Different from standard prolog we may use this with an uninstantiated variable to remove all entries.

4.retractall(): It succeeds once and remove all database entries that match Term on the

way. Different from standard prolog we may use this with an uninstantiated variable

to remove all entries.

To load and save dos file, the predicates consult() and save() are used as shown below:

1- consult (FileName): is used to read and parse the file given by FileName. The file may contain clauses and facts, separated by the dots. The current operator declarations are used.

For example

consult("data.dat"), where “data.dat” is dos file name.

2- save(FileName): is used to save information the file given by FileName.

For example

save("data.dat"), where “data.dat” is dos file name.

Examples: Write prolog program to perform the following:

1- Read person's name S1, gender S2 and age I then save them into file "p.dat".

Solution:

```
repeat.

repeat:-repeat.

a:-repeat,consult("a1.dat"),readln(S1),readln(S2),readint(I),assertz(Per(S1,I,S2)),

I<=18,save("p.dat").
```

2-Read car's name S, owner N and car's model M then save them into file "c.dat".

```
repeat.

repeat:-repeat.

b:-repeat,consult("a2.dat"),readln(S),readln(N),readint(M),assertz(car(S,N,M)),

M<=1980,save("c.dat").
```

3- Count how many cars with "VOLVO" name?

```
c:-consult("c.dat"),findall(X,car(X,_,_),L),count(L,C),write(C).

count([],0).

count([H|T],X):-H="volvo",count(T,X1),X=X1+1.

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

4- Count how many cars with car name "TOYOTA" and owner name "ALI"?

```
d:-consult("c.dat"),findall(X,car("toyota",X,_),L),count(L,C),write(C).

count([],0).

count([H|T],X):-H="ali", count(T,X1),X=X1+1.

count([_|T],X):-count(T,X)
```