Seminar 2 – Lists in Prolog

1. Write a predicate to remove from a list all the elements that appear only once. For ex: [1,2,1,4,1,3,4] => [1,1,4,1,4].

Check the element in the list and use a collector list in which we add the elements

* Count the number of occurrences of an element in a list.

Count(l1, l2, …, ln, x) = {

0, n = 0,

1 + Count(l2, l3, …, ln, x) if l1 = x,

Count(l2, l3, …, ln, x), otherwise

}

Implementation:

%count(E: elem, Nr: int, L: list)

%flow model (I, o, I) (I, I, I)

count(\_, 0, []).

%count(E, Nr, L):- L = [], Nr = 0.

count(E, Nr1, [E|T]):-

% I, o, i

count(E, Nr, T),

Nr1 is Nr + 1.

count(E, Nr, [H|T]):-

E \= H,

count(E, Nr, T).

?- Count (1, M, [1,4,1,2]).

M = 2;

M=1;

M=1;

M = 0;

False

1 – clause 2 => 1 + …..

* Clause 3

4 – clause 3

1 - clause 2 => 1 + ….

* + Clause 3

2 –clause 3

-> 0

?- Count(1, R, [1,2,3,1,4,1,6]).

R = 3

?- Count(1, 3, [1,2,3,1,4,1,6]).

True

?- Count(1, 2, [1,2,3,1,4,1,6]).

false

A = 3

B is A + 1

B =:= A + 1 // error because B is not bound

Remove\_from\_list(c1,c2,..,cn, a1,a2,…,am)

() if m = 0,

Remove\_from\_list(c1,c2,..cn, a2,a3,…,am), if count(c1,c2...,cn, a1) = 1,

A1 U remove\_from\_list(c1,c2,...cn, a2, a3,…,am) otherwise

%Remove\_from\_list(L: the original list, LC: copy of the list, LRez: the result)

%(I,I,o), (I,I,I)

Remove\_from\_list([], \_, LRez) :- LRez = [].

Remove\_from\_list([L1|T], LC, LRez):-

Count(L1, Nr, LC),

Nr==1,

%Count(L1, 1, LC),

Remove\_from\_list(T, LC, LRez1),

LRez = LRez1.

Remove\_from\_list([L1|T],LC, LRez):-

Count(L1, Nr, LC),

Nr=\=1,

Remove\_from\_list(T, LC, LRez1),

LRez =[L1| LRez1].

Collector variable

Count([1,2,1,4,1,3,4], 1) = 3

1 + count([2,1,4,1,3,4], 1) = 1 + 2 = 3

Count([1,4,1,3,4], 1)= 2

1 + count([4,1,3,4], 1) = 1 + 1 = 2

Count([1,3,4], 1) = 1

1 + count([3,4], 1) = 1 + 0 = 1

Count ([4], 1) = 0

Count([], 1) = 0

= 0

Count([1,2,1,4,1,3,4], 1, 0) =3

Count([2,1,4,1,3,4], 1, 1) = 3

Count([1,4,1,3,4], 1, 1) =3

Count ([4,1,3,4], 1, 2) = 3

Count([1,3,4], 1, 2) =3

Count([3,4], 1, 3) = 3

Count([4], 1, 3) = 3

Count([], 1, 3) =3

3

Count([], 1, 0) => 0

Count(l1l2,…,ln,x,c) {

c, if n=0

Count(l2,l3,…,ln,x,c+1) if x==l1

Count(l2,l3...ln,x,c) if x!=l1

}

Implementation:

% Count( L1 – list, X – Element, C – Collector, Cout – Collector output)

%Flow: (I, I, I, o)

Count( [ ], X, C, C).

%Count([], \_, C, Cout):-Cout = C.

Count( [X|T], X, C, Cout):-

C1 is C + 1,

Count( T, X, C1, Cout).

Count( [H|T], X, C, Cout):-

H \= X,

Count(T, X, C, Cout).

1. Given a list of numbers, remove all the increasing sequences of elements. Ex: [1,2,4,6,5,7,8,2,1] => [2,1]

[2,4,6,5,7,8,2,1], 1

[4,6,5,7,8,2,1], 2

[6,5,7,8,2,1], 4

[5,7,8,2,1], 6

call ([7,8,2,1], 5)

Call([8,2,1], 7)

Call([2,1], 8)

call([1], 2)

2 U call([], 1)

1

[6, 8, 2, 1]

[H1, H2,H3, H4|T]

[]

[H], [H1,H2]