Seminar 3: Heterogeneous lists in Prolog

* A list in which elements are of different types, ex: [1, 2, 3, t, [1,2,3], a, b, [1,2,1,2], 6]
* How do we process such a list?
  + [H|T]
    - is\_list(H) - checks if H is a list
    - number(H) - checks if H is a number
    - atom(H) - checks if H is a symbol
  + How do we build such a list?

|  |  |  |
| --- | --- | --- |
| [H|T] | T = 2 | T = [1,2,3] |
| H = 2 | [2|2] | [2, 1, 2, 3] |
| H = [1,2,3] | [[1,2,3] | 2] | [[1,2,3], 1, 2, 3] |

Obs: [1,2,3,4,5,6,7 | …] w

1. You re given a heterogeneous list, made of numbers and lists of numbers. You will have to remove the odd numbers from the sublists that have a mountain aspect (a list has a mountain aspect if it starts with a sequence of increasing elements, followed by a sequence of decreasing elements).

Ex: [1, 2, [1, 2, 3, 2], 6, [1,2], [1,4,5,6,7,1], 8, 2, [4,3, 1], 11, 5, [6,7,6], 8] =>

[1, 2, [2, 2], 6, [1, 2], [4, 6], 8, 2, [4, 3, 1], 11, 5, [6, 6], 8]

Solution:

* Check if a list is a mountain
* Remove the odd elements from a linear list
* The main function which processes the heterogeneous list

Check if a list is a mountain

* Version 1:
  + Find the maximum
  + Split the list in two (before and after the maximum)
  + Check the first part to be increasing and the second to be decreasing
* **Version 2:**
  + Check if l1 < l2 => keep going
  + Check if l1 > l2 => keep going
  + Add an extra parameter to know on which part we are currently
* Version 3:
  + Check (and remove) the increasing part of the list
  + When you get to l1 > l2 call another function to check if the entire list is decreasing
* Version 4:
  + Find the peak of the mountain: l1<l2 >l3 (it should be only one such triple)
  + [1,2,3,2,1,2,3,2,1], [1,2,3,2,1, 2]
  + Need to check to NOT have a “valley” l1 > l2 < l3

Part – 0 for increasing, 1 for decreasing

Mountain(l1...ln,part)={False, if n<3

true, if l1<l2>l3, n = 3 and part = 0

True,if l1>l2>l3, n = 3 and part = 1

Mountain(l2..ln,part),l1<l2 and part=0

Mountain(l2...ln,1),l1>l2

False,otherwise

}

What happens if part is 1 and l1 < l2?

[1,2,3,4,3,2,1,2,3,4,5,6], 0

[2,3,4,3,2,1,2,3,4,5,6], 0

….

[4, 3, 2, 1, 2, 3, 4, 5, 6], 0

[3, 2, 1, 2, 3, 4, 5, 6] 1

…

[1,2,3,4,5,6] 1 => False (based on the last branch)

Mountain([5,4,3,2,1], 0) => true

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

Mountain([3,2,1], 1) =

True

[1,2,3,4,6, 5,3,4,2]

% Is\_mountain(L: List, P: Integer)

% is\_mountain(L: I, P: I)

Is\_mountain([H1, H2, H3], 0) :-

H1 < H2,

H2 > H3.

Is\_mountain([H1, H2, H3], 1) :-

H1 > H2,

H2 > H3.

Is\_mountain([H1, H2 | T], 0) :-

H1 < H2,

Is\_mountain([H2 | T], 0).

Is\_mountain([H1, H2 | T], \_) :-

H1 > H2,

Is\_mountain([H2 | T], 1).

* Remove all the odd numbers from a linear list.

RemoveOnes(l1,l2,…,ln) {

{} if n =0,

RemoveOnes(l2,l3,...ln) if l1 % 2 == 1,

L1 U removeOnes(l2,l3...,ln), otherwise

}

%RemoveOnes(L-list,RL-resulting list)

%flow model: (I,o)

RemoveOnes([],[]).

RemoveOnes([H|T],[H|RL]):-

H mod 2 =:= 0,

RemoveOnes(T,RL), !.

RemoveOnes([\_|T],RL):-

RemoveOnes(T,RL).

* Remove the odd elements from sublists with mountain aspect from a heterogeneous list.

Main(l1,l2...ln) = { [], if l is empty,

RemoveOnes(l1) U main(l2,..ln), if l1 is a list and l1\_1 < l1\_2 and Mountain(l1, 0) is true,

L1 U main(l2,...ln),otherwise. }

% main (L : input list, R: result list)

% flow model (I,o)

Main ([],[]).

Main([H|T], [H3|R]):- is\_list(H),

H = [H1,H2|\_],

H1 < H2,

Mountain(H,0), !,

RemoveOnes(H, H3),

Main(T, R).

Main([H|T],[H|R]):-

Main(T,R1),

R = R1.

[[1,2,3,4], 3, 5, 2, [1,5,2], 8]

H = [1,2,3,4]

T = [3,5,2,[1,5,2], 8]

H = [H1, H2|T]

1. Consider the following predicates:

%predicate for odd numbers

%odd(I)

Odd(1).

Odd(3).

Odd(5).

Odd(7).

Odd(9).

%even(o)

Even(X) :- odd(N1), odd(N2), X is N1 + N2, X < 9.

Even(X) :- odd(N1), X is N1 \* 2, X > 9.

?-even(X).

* 2;4;6;8;4;6;8;6;8;8;10; 14; 18

Even(X) :- !, odd(N1), odd(N2), X is N1 + N2, X < 9.

Even(X) :- odd(N1), X is N1 \* 2, X > 9.

* 2;4;6;8;4;6;8;6;8;8

Even(X) :- odd(N1), !, odd(N2), X is N1 + N2, X < 9.

Even(X) :- odd(N1), X is N1 \* 2, X > 9.

* 2; 4; 6; 8

Even(X) :- odd(N1), odd(N2), !, X is N1 + N2, X < 9.

Even(X) :- odd(N1), X is N1 \* 2, X > 9.

* 2

1. Let’s consider the following predicate:

P(E, L, [E|L]).

P([E, [H|T], [H|L1]):-

P(E, T, L1).

Depens on the flow model

* I,I, o
* I,I,I
* I, o, I
* O, I, I
* O,o, i