Seminar 3 – Heterogeneous Lists in Prolog

Numarul de aparitii al unui element intr-o lista (rezolvarea standard, fara variabila colectoare)

```
%nrAparitii(L:list, E:element, R:integer)
%model de flux (i,i,o), (i,i,i)
%L - lista in care calculam numarul de aparitii
%E - elementul ale carui aparitii le numaram in lista
%R - rezultatul, numarul de aparitii

nrAparitii([], _, 0).
nrAparitii([H|T], E, R):-
    H =:= E,
    nrAparitii(T, E, R1),
    R is R1 + 1.
nrAparitii([H|T], E, R):-
    H =\= E,
    nrAparitii(T, E, R).
```

Metoda variabilei colectoare

Variabila colectoare (variabila acumulator) este o variabila intermediara pentru rezultate partiale (permite determinarea rezultatului final pe parcurs, nu doar la iesirea din apelul recursiv).

Variabila colectoare se va reflecta printr-un parametru in plus, atat in modelul matematic, cat si in codul Prolog.

Acest parametru va fi un parametru de tip ***input*** (ea este initializata in cadrul primului apel).

```
%nrAparitiiCol(L:list, E:element, Col:integer, R:integer)
%model de flux (i,i,i,o), (i,i,i,i)
%L - lista in care calculam numarul de aparitii
%E - elementul ale carui aparitii le numaram in lista
%Col - variabila colectoare
%R - rezultatul, numarul de aparitii

nrAparitiiCol([], _, C, C).
nrAparitiiCol([H|T], E, C, R):-
    H = E,
    C1 is C + 1,
    nrAparitiiCol(T, E, C1, R).
nrAparitiiCol([H|T], E, C, R):-
    H \=E,
    nrAparitiiCol(T, E, C, R).
```

2. Determinarea inversului unui numar folosind metoda variabilei colectoare

```
%inv_numar(Nr: int, Ncv: int, Rez: int)
%Nr - numarul care trebuie inversat
%Ncv - variabila colectoare in care retinem rezultatul partial
%Rez - rezultatul
%model de flux (i, i, o), (i, i, i)

inv_numar(0, Ncv, Ncv).
inv_numar(N, Ncv, Rez):-
        N > 0,
        Cifra is N mod 10,
        NcvNew is Ncv * 10 + Cifra,
        NN is N div 10,
        inv_numar(NN, NcvNew, Rez).
```

Definirea unei liste eterogene in Prolog

Lista eterogena este o lista formata din atomi, numere si alte liste (subliste).
EG: [[2,4,6],1,4,a,x,[3,6]] este o lista eterogena.

In lucrul cu acetse liste eterogene, vom utiliza din nou mecanismul de a imparti o lista in Head and Tail cu |.
Dupa impartirea listei in H and T, vom verifica tipul Headului.
In contextul acesta, H poate sa fie numar, atom sau lista.

Verificarea se realizeaza utilizand predicatele implicite:

- is_list(H) – returneaza True daca H este o lista
- number(H) – returneaza True daca H este un numar
- atom(H) – returneaza True daca H este un atom (simbol).

3. Se dă o listă de numere și liste de numere. Se cere ca din fiecare sublistă să se șteargă numerele palindrome.

Avand in vedere ca avem deja predicatul care ne determina inversul unui numar, pentru o sublista, parcugem elementele din lista si daca este palindrom, stergem numarul.

```
%delSub(L:list, R: list)
%L - lista liniara din care eliminam numerele care sunt palindrom
%LR - lista rezultat
%model de flux (i, o), (i, i)

delSub([], []).
delSub([H|T], [H|TR]):-
        inv_numar(H, 0, HI),
        H =\= HI,
        delSub(T, TR).
delSub([H|T], TR):-
        inv_numar(H, 0, HI),
        H =:= HI,
        delSub(T, TR).
```

Ne trebuie apoi un Main, pentru a parcurge lista eterogena: daca gasim numar, mergem mai departe si daca gasim lista, intram in sublista si producem modificarea solicitata.

```
%mainHeter(L: lista, LR: list)
%L - lista eterogena initiala
%LR - lista rezultat
%model de flux (i, o), (i, i)

mainHeter([], []).
mainHeter([H|T], [H|LR]):-
        number(H),
        mainHeter(T, LR).
mainHeter([H|T], [H1|LR]):-
        is_list(H),
        delSub(H, H1),
        mainHeter(T, LR).
```

- Când lucrăm in SWI-Prolog cu liste lungi, s-ar putea ca Prolog să ne afișeze ca rezultat o listă de genul: R = [1, 2, [2, 1, 6], 6, [154, 11, 10], 7, 1, 0, [...|...]|...]. Pentru a avea lista întreagă, putem folosi predicatul write:
  - Dacă căutarea în SWI-Prolog nu s-a terminat de tot (adică SWI-Prolog așteaptă; ca să continue căutarea), apăsând tasta w, se va afișa lista completă.
  - Dacă căutarea s-a terminat, predicatul trebuie apelat din nou, adăugând și un apel la predicatul write: numepredicatapelat([1,2,3,4], R), write(R).

4. Se dă o listă eterogenă, formată din numere și liste de numere. Se cere să se determine numărul sublistelor care au aspect de munte. (O listă are aspect de munte dacă este alcătuită dintr-o secvență de numere crescătoare, urmată de o secvență de numere descrescătoare. Orice altă variantă (doar secvență crescătoare, doar secvență descrescătoare, secvență care descrește și după aceea crește, etc.) nu are aspect de munte.)

EG: [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] =>3 (sublistele [1,2,3,2], 1,4,5,6,7,1] și [6,7,6]).

- Metoda de rezolvare presupune utilizarea a 2 funcții: una care verifică dacă o listă are aspect de munte, și una care numără câte subliste sunt cu aspect de munte.
- Începem cu verificarea aspectului de munte.
- O variantă este să luăm un parametru în plus, care va avea o valoare (de exemplu 0) cat timp parcurgem partea crescătoare a listei, iar când ajungem la partea care descrește modificăm valoarea respectivă.

$$munte(l_1 l_2 \dots l_n, f) = \begin{cases} false, n \leq 1, f = 0 \\ true, n \leq 1, f = 1 \\ munte(l_2 \dots l_n, 0), l_1 < l_2 \; f = 0 \\ munte(l_2 \dots l_n, 1), l_1 \geq l_2, f = 0 \\ munte(l_2 \dots l_n, 1), l_1 > l_2, f = 1 \\ false, altfel \end{cases}$$

- Din moment ce am introdus un parametru în plus, va trebui să facem și o funcție în plus, care face primul apel, inițializând valoarea lui f cu 0. Valoarea 0 înseamnă că suntem pe partea crescătoare a listei. Dar ce se întâmplă dacă lista are doar o parte care descrește? Pe baza modelului matematic execuția intră pe a 4-a ramură, iar în final returnează „True". Pentru a evita acest lucru, funcția care face primul apel trebuie să verifice existenta a cel putin a unei perechi de elemente crescătoare.

$$munteMain(l_1 l_2 \dots l_n) = \begin{cases} false, n \leq 2 \\ false, l_1 \geq l_2 \\ munte(l_1 l_2 \dots l_n, 0), altfel \end{cases}$$

```
% munte(L:list, F:integer)
% model de flux: (i,i)
% L - lista pe care o verificam daca are aspect de munte
% F - variabila care arata daca suntem pe partea de crestere sau
% descrestere

        munte([ ],1).
        munte([ _ ], 1).
        munte([H1,H2|T], 0):-
                H1 < H2,
                munte([H2|T], 0).
        munte([H1,H2|T], 0):-
                H1 >= H2,
                munte([H2|T], 1).
        munte([H1,H2|T], 1):-
                H1 > H2,
                munte([H2|T], 1).
```

```
% munteMain(L:list)
% model de flux: (i)
% L - lista pe care o verificam daca are
% aspect de munte

munteMain([H1,H2|T]):-
        H1 < H2,
        munte([H1,H2|T], 0).
```

Acum că avem predicat pentru a verifica dacă o listă are apect de munte, vom scrie predicatul principal care să numere câte subliste cu aspect de munte sunt în lista noastră.

$$nrSubliste(l_1 l_2 \dots l_n) = \begin{cases} 0, n = 0 \\ 1 + nrSubliste(l_2 \dots l_n), & l_1 \text{este listă și munteMain(l1)este adevărat} \\ nrSubliste(l_2 \dots l_n), altfel \end{cases}$$

```
%nrSubliste(L:listE, Rez:integer)
%model de flux: (i,o), (i,i).
%L - lista eterogena in care numaram sublistele cu aspect de munte
%Rez - rezultatul, numarul listelor

nrSubliste([], 0).
nrSubliste([H|T], R):-
        is_list(H),
        munteMain(H),!,
        nrSubliste(T, R1),
        R is R1 + 1.
nrSubliste([ _ |T], R):-
        nrSubliste(T, R).
```