P1: Liste in Prolog(1)

# 7.

# a. Sa se scrie un predicat care intoarce reuniunea a doua multimi.

Fie multimile, A = (A1…An), B = (B1…Bm)

## Model Recursiv:

,n=0 si m=0



reuniune( (A1, A2,.. An), (B1, B2…Bn) ) = (B1..Bm) , n=0

(A1..An) ,m=0



A1 + reuniune((A2…An),(B1…Bm)), A1 nu apartine (B1..Bm)



reuniune((A2..An), (B1..Bm)) , A1 apartine (B1…Bm)

A white paper with black text

Description automatically generated

## Functie ajutatoare:

A math equations on a white background

Description automatically generated

Exemple:

A = (1,4,3) si B = (2, 5, 1, 6, 3) n = 3, m = 5

reuniune((1,4,3), (2,5,1,6,3)) = reuniune((4,3), (2,5,1,6,3)) = 4 + reuniune((3), (2,5,1,6,3)) =4 + reuniune((), (2,5,1,6,3)) = (4,2,5,1,6,3)

A=(), B=(1,2)

reuniune((),(1,2)) = (1,2)

A=(1,2,3), B= ()

reuniune((1,2,3), ()) = (1,2,3)

A=(), B=()

()

## Model de flux:

(i,i,i) – daca M = A U B

(i,i,o) – daca M := A U B

## Cod:

apartine(E,[E|\_]):-!.

apartine(\_,[]):-false.

apartine(E,[\_|Tail]):-

apartine(E,Tail).

reuniune([],[],[]):-!.

reuniune([],B,B):-!.

reuniune(A,[],A):-!.

reuniune([Head\_A|Tail\_A],B,[Head\_A|Multime]):-

\+ apartine(Head\_A,B),

!,

reuniune(Tail\_A,B,Multime).

reuniune([Head\_A|Tail\_A],B,Multime):-

apartine(Head\_A,B),

!,

reuniune(Tail\_A,B,Multime).

test1:-reuniune([1,4,3],[2,5,1,6,3],[4,2,5,1,6,3]).

test2:-reuniune([],[1,2],[1,2]).

test3:-reuniune([1,2,3],[],[1,2,3]).

test4:-reuniune([],[],[]).

# b. sa se scrie un predicat care, primind o lista, intoarce multimea tuturor perechilor din lista. De ex, cu [a,b,c,d] va produce [[a,b] , [a,c], [a,d], [b,c], [b,d], [c,d]]

## Model Recursiv:

A white board with black text

Description automatically generated

A math equations on a white background

Description automatically generated

## Functie ajutatoare:

0, m = 0 si n = 0

adauga((A2…An), (B1…Bm), A1 (+) P), n!=0



Adauga((A1…An), (B…Bm), P) = adauga((A1…An)(B2…Bm), B1 (+) P), n=0 si m!=0

## Exemple:

Multime((a,b,c,d),()) = multime((b,c,d), perechi(a,(b,c,d),()))

=> multime((b,c,d), ( (a,b), (a,c), (a,d) )) = multime((c,d), perechi(b,(c,d), ( (a,b), (a,c), (a,d) )))

=> multime((c,d), ((a,b),(a,c),(a,d),(b,c),(b,d)) = multime((d), perechi(c, (d),P))

=> multime((d),((a,b),(a,c),(a,d),(b,c),(b,d), (c,d)))=multime( () , perechi(d, (), M)

= ((a,b),(a,c),(a,d),(b,c),(b,d), (c,d))

## Model de flux:

(i,i) - daca M = multimea de perechi care se formeaza din lista (l1…ln)

(i,o) – daca M:= multimea de perechi care se formeaza din lista (l1…ln)

## Cod:

adauga([],[],[]):-!.

adauga([Head\_A|Tail\_A],B,[Head\_A|P]):-

adauga(Tail\_A,B,P).

adauga([],[Head\_B|Tail\_B],[Head\_B|P]):-

adauga([],Tail\_B,P).

perechi(\_,[],[]):-!.

perechi(E,[Head|Tail],[[E,Head]|P]):-

!,

perechi(E,Tail,P).

multime([],[]):-!.

multime([Head|Tail],P):-

perechi(Head,Tail,P1),

multime(Tail,P2),

adauga(P1,P2,P).

test:-multime([1,2,3],[[1,2],[1,3],[2,3]]).