Генератори (Part II)

**Генериране на Питагорови тройки**

Спомняме си как се генерират естествените числа n(X) , естествените числа в интервала [A, B] between(X, A, B) и накрая се получава това

%p( <променлива>, <променлива>, <променлива> ).

p(A, B, C) :- n(C), between(B,1,C), between(A,1,C), C\*C=:=A\*A+B\*B.

Бихме могли вместо

between(A,1,C)

да напишем

between(A,B,C)

би било малко по-бързо, а резултатът ще се промени само с извеждане на уникални тройки, ако приемем, че (3, 4, 5) \equiv(4, 3, 5).

**Генериране на всички разбивания на списък( Разбиване на число по суми )**

Ако имаме списъкът [1, 2, 3], то всичките му разбивания са [ [1], [2, 3] ], [ [1, 2], [3] ], [ [1, 2, 3] ]

%split(<списък>, <промнелива за резултат>).

split(X, [X]).

split(L, [A|R]) :- append(A,B,L), A \= [], B \= [], split(B, R).

Ако разбиваме число по всичките му суми, можем да изпозлваме тази задача като направим списък с n елемента и за всяко получено негово разбиване да вземем дължините на списъците, които го съставят.

**Задачи с квантори**

Нека p е двуместен предикат.

\exists X\in A, \exists Y \in B p(X, Y)

%p1( <списък>, <списък> ).

p1(A, B) :- member(X, A), member(Y, B), p(X, Y).

member(M, N) генерира всички елементи N в M.

\exists X\in A, \forall Y \in B p(X, Y)

%p2( <списък>, <списък> ).

p2(A, B) :- member(X, A), not ( ( member(Y, B), not( p(X, Y)))).

Допълнителните скоби при not са, защото той е едноместен предикат, без тях **prolog** приема, че подаваме на not два аргумента, което води до компилационна грешка.

\forall X\in A, \exists Y \in B p(X, Y)

%p3( <списък>, <списък> ).

p3(A, B) :- not( (member(X, A), not(member(Y, B), p(X, Y)))).

\forall X\in A, \forall Y \in B p(X, Y)

%p4( <списък>, <списък> ).

p4(A, B) :- not( (member(X, A), member(Y, B), not(p(X, Y)))).

**Декартово произведение на два списъка**

Има шанс да не работи както трябва.

%помощна функция - в <променлива\_за\_резултат> се записва декартовото произведение на <променлива> със <списък>

%one(<променлива>, <списък>, <променлива\_за\_резултат>).

one( \_, [], [] ).

one( X, [H|T], [ [X, H], R ] ) :- one(X, T, R).

%dek( <списък1>, <списък2>, <резултат> ).

dek(A, [], A).

dek([], B, B).

dek([A|A1], B, R) :- one(A, B, ROne), dek(A1, B, RDek), append(ROne, RDek, R).