Meta-predicados sobre listas

maplist

```
maplist/2, maplist/3, maplist/4, maplist/5

maplist(<Pred/1>, <Lista>)
maplist(<Pred/2>, <Lista1>, <Lista2>)
maplist(<Pred/3>, <Lista1>, <Lista2>, <Lista3>)
maplist(<Pred/4>, <Lista1>, <Lista2>, <Lista3>,  <Lista4>)
```

Suponhamos que estão definidos os seguintes predicados:

```
par(N) - N é par
dobro(N, D) - D é o dobro de N
soma(X, Y, S) - S é a soma de X com Y
```

Meta-predicados sobre listas

maplist

Exemplos de utilização:

```
?- maplist(par,[1,2,3,4,5]).
false.

?- maplist(par,[2,4]).
true.

?- maplist(dobro,[1,2,3,4,5], L).
L = [2, 4, 6, 8, 10].

?- maplist(soma,[1,2,3,4,5], [1,2,3,4,5],L).
L = [2, 4, 6, 8, 10].

?- maplist(soma(10), [0,1,2,3,4],L).
L = [10, 11, 12, 13, 14].
```

Meta-predicados sobre listas

maplist

Escrever os elementos de uma lista, um por linha:

```
?- maplist(writeln,[1,2,3,4,5]).
1
2
3
4
5
true.


?- maplist(length, [[1], [a,b, c, d], [c,d, X]], Comps).
Comps = [1, 4, 3].
```

Meta-predicados sobre listas

include e exclude

```
include(<Pred/1>, <Lista1>, <Lista2>)
exclude(<Pred/1>, <Lista1>, <Lista2>)
```

Suponhamos que está definido o seguinte predicado:

```
menor(X,Y) - X < Y
```

Meta-predicados sobre listas

include e exclude

Exemplos de utilização:

```
?- include(par, [1,2,3,4,5,6,7], L).
L = [2, 4, 6].

?- exclude(par, [1,2,3,4,5,6,7], L).
L = [1, 3, 5, 7].

?- include(menor(3), [1,2,3,4,8,9,3], L).
L = [4, 8, 9].
```

Meta-predicados sobre listas

include e exclude

Predicado conta_variaveis(L, Cont)

```
conta_variaveis(L, Cont) :-
    include(var, L, L_vars),
    length(L_vars, Cont).
```

Predicado conta_maiores(N, L, Cont)

```
conta_maiores(N, L, Cont) :-
    include(menor(N), L, L_maiores),
    length(L_maiores, Cont).
```

Meta-predicados

findall e setof

```
findall(<termo>, <Objectivo>, <Lista>)
setof(<termo>, <Objectivo>, <Lista>)
```

Meta-predicados

findall e setof

Exemplos de utilização:

```
?- findall(Comb, combinacao(2, [a, b ,c], Comb), L_combs).
L_combs = [[a, b], [a, c], [b, c]].

?- findall(X, (member(X,[6,2,4,5,4,7]), X mod 2 =:= 0), L).
L = [6, 2, 4, 4].

?- setof(X, (member(X,[6,2,4,5,4,7]), X mod 2 =:= 0), L).
L = [2, 4, 6].
```

Nova versão de `combina`

```
combina(L1, L2, L3) :-
    findall([E1, E2], (member(E1, L1), member(E2, L2)), L3).
```

Meta-predicados

Útil para o projeto

Definir o predicado `sub_lista_N(SL, L, N)`. Significado: `SL` é uma sublista de `L`, de comprimento `N`.

```
sublista(SL, L, N) :-
     append([_, SL, _], L),
     length(SL, N).

?- L = [1,2,3,4,5,6], sublista(SL, L, 4).
L = [1, 2, 3, 4, 5, 6],
SL = [1, 2, 3, 4]
L = [1, 2, 3, 4, 5, 6],
SL = [2, 3, 4, 5]
L = [1, 2, 3, 4, 5, 6],
SL = [3, 4, 5, 6]
false.
```

Meta-predicados

Útil para o projeto

Se quisermos uma lista com todas as sublistas:

```
sublistas(L, N, SLs) :-
     findall(SL, sublista(SL, L, N), SLs).

?- L = [1,2,3,4,5,6], sublistas(L, 4, SLs).
L = [1, 2, 3, 4, 5, 6],
SLs = [[1, 2, 3, 4], [2, 3, 4, 5], [3, 4, 5, 6]].
                                                MAS...
?- L = [V1,V2,V3,V4,V5,V6], sublistas(L, 4, SLs).
L = [V1, V2, V3, V4, V5, V6],
SLs = [[_1056, _1062, _1068, _1074],
       [_1026, _1032, _1038, _1044],
       [_996, _1002, _1008, _1014]].
```

Meta-predicados

Útil para o projeto

Solução: usar `bagof` em vez de `findall`:

```
sublistas(L, N, SLs) :-
     bagof(SL, sublista(SL, L, N), SLs).

?- L = [V1,V2,V3,V4,V5,V6], sublistas(L, 4, SLs).
L = [V1, V2, V3, V4, V5, V6],
SLs = [[V1, V2, V3, V4], [V2, V3, V4, V5], [V3, V4, V5, V6]].
                                                MAS...
?- L = [V1,V2,V3,V4,V5,V6], sublistas(L, 7, SLs).
false.
```

Meta-predicados

Útil para o projeto

Solução: acrescentar uma cláusula:

```
sublistas(L, N, SLs) :-
     bagof(SL, sublista(SL, L, N), SLs), !.

sublistas(_, _, []).

?- L = [V1,V2,V3,V4,V5,V6], sublistas(L, 7, SLs).
L = [V1, V2, V3, V4, V5, V6],
SLs = [].
```