**CONCATENAR**

concat([],L,L).

concat([Car|Cdr],L,[Car|R]) :- concat(Cdr,L,R).

**INVERTIR**

invertir([ ], [ ]).

invertir([Car|Cdr], Invertir) :- invertir(Cdr, Invertir\_Cdr),

concatenar(Invertir\_Cdr,[Car],Invertir).

**IF THEN ELSE**

if\_then\_else(P,Q,R) :- P, !, Q.

if\_then\_else(P,Q,R) :- R.

**SUMA**

suma(0,X,X).

Suma(s(X),Y,s(Z)):- suma(X,Y,Z).

**PRODUCTO**

Producto(0,X,0).

Producto(s(0),X,X).

Producto(s(X),Y,Res):- producto(X,Y,Z),suma(Z,Y,Res).

**UNION**

union(Conjunto,[],Conjunto).

union(Conjunto,[Car|Cdr],Res) :- miembro(Car,Res),

union(Conjunto,Cdr,Res),!.

union(Conjunto,[Car|Cdr],[Car|Res]) :-union(Conjunto,Cdr,Res).

**INVERTIR\_DL**

invertir(Lista,Invertir) :- invertir\_dl(Lista,Invertir\[]).

invertir\_dl([],Invertir\Invertir).invertir\_dl([Car|Cdr],Invertir\Cola) :- invertir\_dl(Cdr,Invertir\[Car|Cola]).

**DUPLICAR**

Duplicar([],[]).

Duplicar([Car|Cdr],[Car,Car|Res]):- duplicar(Cdr,Res).

**APLANAR**

aplanar([],[]).

aplanar(Atomo,[Atomo]) :- atomic(Atomo), Atomo \== [].

aplanar([Car|Cdr], Resultado) :- aplanar(Car,Car\_aplanado),

aplanar(Cdr,Cdr\_aplanado),

concatenar(Car\_aplanado,Cdr\_aplanado,Resultado).

**APLANAR\_DL**

aplanar(Lista,Resultado) :- aplanar\_dl(Lista,Resultado\[]).

aplanar\_dl([],Lista\Lista).

aplanar\_dl(Atomo,[Atomo|Cdr]\Cdr) :- atomic(Atomo), Atomo \== [].

aplanar\_dl([Car|Cdr],Cabeza\Dif) :- aplanar\_dl(Car,Cabeza\Dif\_car),

aplanar\_dl(Cdr,Dif\_car\Dif).

**GET\_ASOC**

get\_asoc(Llave,[Llave|Valor],Valor).

get\_asoc(Llave,[Car|\_],Res):-get\_asoc(Llave,Car,Res),!.

get\_asoc(Llave,[\_|Cdr],Res):-get\_asoc(Llave,Cdr,Res).

**ESCALAR**

escalar([],\_,0).

escalar([Car1|Cdr1],[Car2|Cdr2],Resultado):-escalar(Cdr1,Cdr2,Aux),                                                                                                                                       Resultado is Aux+Car1\*Car2.

**ELIMINAR**

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

eliminar([\_|Y],1,Y):- !.

eliminar([Car|Cdr],X,[Car|R]) :- Y is X-1, eliminar(Cdr,Y,R).

**AÑADIR**

anadir(X,[],[X]).

anadir(X,Lista,Lista):-miembro(X,Lista),!.

anadir(X,Lista, Res):-insertar(X,Lista,Res),!.

**MIEMBRO**

miembro(Car,[Car|\_]):- !.

miembro(Car,[\_|Cdr]):- miembro(Car,Cdr).

**INSERTAR**

insertar(X,[],[X]).

insertar(X,[Car|Cdr],[X,Car|Cdr]):- X < Car.

insertar(X,[Car|Cdr],[X|Cdr]) :- X=Car.

insertar(X,[Car|Cdr],[Car|CdrX]):- X>Car,insertar(X,Cdr,CdrX).

**INSERTAR CEROS**

insert\_ceros([],[]).

insert\_ceros([Car|Cdr],[Car,0|R]) :- insert\_ceros(Cdr,R).

**BORRAR UNICOS**

Borrar\_unicos([],[]).

Borrar\_unicos([Car|Cdr],Res):- miembro(Car,Cdr),!,borrar\_unicos(Cdr,Res).

Borrar\_unicos([Car|Cdr],Res):- borrar\_unicos(Cdr,Res).

**COMPLEMENTARIO COMP([3,1],[1,3,2,6],X) X=[2,6]**

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

comp(conj,[Car|Cdr],Res):- miembro(Car,Conj),!,comp(conj,Cdr,Res).

comp(conj,[Car|Cdr],Res):- comp(conj,Cdr,Res).

**INTERSECCIÓN**

inter([],C,[]).

Inter([Car|Cdr],C,[Car|Res]):- miembro(Car,C),!,inter(Cdr,C,Res).

inter([Car|Cdr],C,Res):- inter(Cdr,C,Res).

**INCLUSION**

inclusion([],Conjunto).

inclusion([Car|Cdr],Conjunto) :- miembro(Car,Conjunto),

inclusion(Cdr,Conjunto).

**PREORDEN**

:-op(600,xfy,[\]).

preorden(X,R) :-preorden\_dl(X,R\[]).

preorden\_dl([],X\X).

preorden\_dl([Valor | [HijoIzq,HijoDer] ], [Valor|Izq]\Der ) :- preorden\_dl(HijoIzq,Izq\Z), preorden\_dl(HijoDer,Z\Der).

**ENORDEN**

:-op(600,xfy,[\]).

enorden(X,R) :-enorden\_dl(X,R\[]).

enorden\_dl([],X\X).

enorden\_dl([Car,Izq,Der],X\Z) :- enorden\_dl(Izq,X\[Car|Y]),

          enorden\_dl(Der,Y\Z).

**QUICKSORT**

quicksort([],[]).

quicksort([Car|Cdr], Res) :- partir(Car,Cdr,Izq,Der),

quicksort (Izq, OrdenI),

quicksort(Der, OrdenDer),

  concat(OrdenI,[Car|OrdenDer],Res).

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

partir(Pivote,[Car|Cdr],[Car|Izq],Der) :- Car =< Pivote,partir(Pivote,Cdr,Izq,Der).

partir(Pivote,[Car|Cdr],Izq,[Car|Der]) :- Car  >  Pivote,partir(Pivote,Cdr,Izq,Der).

concat([],L,L).

concat([Car|Cdr],L,[Car|R]) :- concat(Cdr,L,R).

**QUICKSORT\_DL**

:- op(600,xfy,[\]).

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

partir(Pivote,[Car|Cdr],[Car|Izq],Der) :- Car =< Pivote,partir(Pivote,Cdr,Izq,Der).

partir(Pivote,[Car|Cdr],Izq,[Car|Der]) :- Car  >  Pivote,partir(Pivote,Cdr,Izq,Der).

quicksort(Lista,Orden) :- quicksort\_dl(Lista,Orden\[]).

quicksort\_dl([],X\X).

quicksort\_dl([Car|Cdr], OrdenI\X) :- partir(Car,Cdr,Izq,Der),

  quicksort\_dl(Izq, OrdenI\[Car|OrdenD]),

    quicksort\_dl(Der,OrdenD\X).

**SUBARBOL(SUBARBOL,ARBOL\_BINARIO) ARBOL(IZQ,DER)**

Subárbol(arbol(L,R), arbol(L,R)).

Subárbol(S,arbol(S,\_)) :- !.

Subárbol(S,arbol(\_,S)) :- !.

Subárbol(S,árbol(L,R)):-arbol(S,L), arbol(S,R)

**CARTESIANO**

Cart([],Conj,[]).

Cart([Car|Cdr],Conj,Res):- línea(Car,Conj,Linea),

Cartesiano(Cdr,Conj,Resto)

Concat(Linea,Resto,Res).

Línea(Elem,[],[]).

Línea(Elem,[Car|Cdr],[Elem,Car|Res]):- línea(elem,Cdr,Res).

concat([],L,L).

concat([Car|Cdr],L,[Car|R]) :- concat(Cdr,L,R).

**HANOI**

:- op(600,yfx,a).

concat([],L,L) :- !.

concat([Car|Cdr],L,[Car|R]) :- concat(Cdr,L,R).

hanoi(1,A,B,\_,[A a B]).

hanoi(N,A,B,C,Movs) :- N > 1, N1 is N - 1,

hanoi(N1,A,C,B,Movs\_1),

hanoi(N1,C,B,A,Movs\_2),

concat(Movs\_1,[A a B|Movs\_2],Movs).

**HANOI ASSERTA**

:- op(600,yfx,a).

concat([],L,L) :- !.

concat([Car|Cdr],L,[Car|R]) :- concat(Cdr,L,R).

hanoi(1,A,B,\_,[A a B]).

hanoi(N,A,B,C,Movs):- N>1, N1 is N-1,

hanoi(N1,A,C,B,Movs\_1),

asserta((hanoi(N1,A,C,B,Movs\_1):-!)),

hanoi(N1,C,B,A,Movs\_2),

retract((hanoi(N1,A,C,B,Movs\_1):-!)),

concat(Movs\_1,[A a B|Movs\_2],Movs).

**VENTAJAS DESVENTAJAS DE LA RESOLUCION SLD RELACION CONSTR TRANSVER DEL ARBOL DE RESOLCION**

Nos ahorramos la recursividad de izquierdas, es decir, no caeremos en un bucle infinito.

Fail rompe la lectura transversal, backtraking o fails darían problema con ese tipo de recorrido.