Sudoku Sat - Pràctica Prolog

Tutor de la pràctica : Mateu Villaret

Francesc Xavier Bullich Parra i Marc Sànchez Pifarré,
GEINF (UDG-EPS)

2018-06-13

Contents

Informe sobre la pràctica Prolog.

En aquest informe s'hi allotja tot el codi degudament documentat sobre la pràctica prolog anomenada com sudokusat. Aquest informe es construeix sobre un exemple en concret de un sudoku de n=4. Amb aquest exemple s'anirà escenificant què ocorre a cada un dels predicats que considerem més important.

També consta d'un apartat d'exemples d'execució amb sudokus N=9.

Propietats Sudoku

Com a propietats importants d'un sudoku tenim :

- Només podem fer sudokus de tamany N on N és un nombre que té una arrel entera. Per tant es poden fer sudokus de N=4, N=9, N=16…

Predicats de la pràctica.

En aquest apartat es mostren tots els predicats que no son el mostra, el mostraM i el mostraLinia. S'expliquen i es posa un exemple d'execució de cada un d'ells. Així es pot comprovar el seu funcionament i evaluar si el tall està ben posat.

Resol

Aquest predicat seria com el main del programa, és qui delega feina als predicats de manera ordenada i qui estructura les execucions per que el sat rebi els paràmetres que requereix i per que els sudokus es pintin de manera correcte.

Predicat

S'hi ha afegit en tot el codi una série de writes per modificar la visualització del sudoku i fer-la més visual.

```
%%%%%%%%%%%%%%%%%%
% resol(N,Inputs)
% Donada la N del sudoku (NxN), i una llista d'inputs,
% -> es mostra la solucio per pantalla si en te o es diu que no en te.
resol(N,Inputs):- nl, write('NEM A RESOLDRE EL SUDOKU : '),nl,
                  write('.................................'),nl,
                  mostraSudoku(N,Inputs), taulerSudoku(N, Inputs, T, C0
                     ), codificaSudoku(N,T,C0,CNF),
                  sat(CNF,[],M), mostra(M,N).
```

Execució

Veiem l'execució següent que és la que utilitzarem a tot l'informe per explicar de manera adequada el funcionament de cada un dels predicats.

```
| ?- resol(4,[c(1,1,3),c(2,2,1),c(4,4,3)]).

NEM A RESOLDRE EL SUDOKU :
.................................
---------
|3| | | |
---------
| |1| | |
---------
| | | | |
---------
```

```
| | | |3|
---------

SAT!!, SUDOKU PISCINAS!!!
.....................................
---------
|3|2|1|4|
---------
|4|1|3|2|
---------
|2|3|4|1|
---------
|1|4|2|3|
---------
```

Crides internes

- Tauler Sudoku
- Codifica Sudoku
- Sat
- Mostra

Tauler Sudoku

És un predicat que s'encarrega de generar un tauler a partir d'un nombre i una sèrie de inputs. Aquest tauler està representat en forma de llista de llistes de K-domins. On cada K-Domini es representa com a 4 variables que poden prendre el valor cert o fals. Llavors per representar que un valor és cert deixarem la variable com a enter positiva i per representar que és falça com un enter negatiu.

Predicat :

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% DONE MACARRONE!!
% taulerSudoku(N,C,Tauler,CNF)
% Donat el domini de les caselles (N), una inicialitzacio del Sudoku (
    de forma [c[1,2,5],...]),
% -> Tauler serà Llista de llistes de N-dominis del tauler,
% -> i CNF codifica els valors assignats amb clausules unitaries que
    forcen valor a les caselles inicialitzades
taulerSudoku(N,C,Tauler,CNF):-
  % Genera totes les possibles variables per cada domini dins de cada
     casella i cada llista.
  festauler(N,Tauler),
```

```
9    % Agafa les caselles passades com a inputs C i genera la CNF que
         codifica aquestes caselles com a fixes.
10   inicialitzar(N,C,CNF).
```

Execució

```
1  ?-  taulerSudoku(4, [c(1,1,3),c(2,2,1),c(4,4,3)], T, CNF).
2  T = [[[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12], [13, 14, 15, 16]],
       [[17, 18, 19, 20], [21, 22, 23, 24], [25, 26, 27, 28], [29, 30,
       31|...]], [[33, 34, 35, 36], [37, 38, 39, 40], [41, 42, 43|...],
       [45, 46|...]], [[49, 50, 51, 52], [53, 54, 55|...], [57, 58|...],
       [61|...]]],
3  CNF = [[3], [21], [63]].
```

En el resultat de l'execució podem veure que T té una llista de files on cada fila té una llista de caselles i on finalment per cada casella hi ha codificat la possibilitat de que prengui el valor 1, 2, 3 o 4 en forma de variables. Per la casella Fila 1 columna 1 el fet que hi hagi un 1 es representa amb un 1. A la casella Fila 1 columna 2 el fet que hi hagi un 1 es representa amb la variable 5.

D'aquesta manera CNF serà el llistat de clàusules que codifiquen que hem forçat un 3 a la fila 1, columna 1, un 1 a la fila 2 columna 2 i un 3 a la fila 4 columna 4.

Crides internes

Es fa la crida al festauler i al inicialitzar.

- Fes Tauler
- Inicialitzar

Fes Tauler

La idea és simple, es genera una llista de K on K és N^3 nombres. Un cop generada la llista la dividim en grups de N, dos vegades. És a dir sobre una llista de K varibales generem una llista LLN de K/N Llistes de N variables cada llista. I sobre la llista de K/N llistes dividim altre cop per N llistes de llistes quedant un llista de N llistes que té N llistes de N dominis on tot plegat son K dominis.

Predicat

```
1  %%%%%%%%%%%%%%%% DONE MACARRONE!!
2  % festauler(N,T)
3  % Donat el domini de les caselles (N),
4  % -> T sera una llista de N llistes de N llistes de N nombres
5  %    es a dir, una llista de files de N-dominis.
```

```
%    El primer N-domini tindra la forma [1,2,...N] i aixi
    successivament de fila en fila fins
%    a l'ultim N-domini corresponent a la casella (N,N) que sera [N*N*(
    N-1)+1,N*N*(N-1)+2,...N*N*N]
% festauler(0, []). %F is N*N, K is F*N
festauler(N, T):- K is N*N*N, generaLLista(1, K, LL), separaPerN(N, LL,
     LLN), separaPerN(N, LLN, T), !.
```

Execució

```
?- festauler(4,T).
T = [[[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12], [13, 14, 15, 16]],
    [[17, 18, 19, 20], [21, 22, 23, 24], [25, 26, 27, 28], [29, 30,
    31|...]], [[33, 34, 35, 36], [37, 38, 39, 40], [41, 42, 43|...],
    [45, 46|...]], [[49, 50, 51, 52], [53, 54, 55|...], [57, 58|...],
    [61|...]]].
```

Crides internes

- Genera Llista
- Separa Per N

Genera llista

Generar una llista de K variables on K és N^3.

```
%%%%%%%%%%%%%%%%%%%%%%%%%% DONE MACARRONE!!
% generaLLista(P, U, LL)
% Genera una llista de números de P fins a U i ho posa a LL
% -> P < U.
% -> P és l'inici
% -> U és el nombre final
% -> LL és la llista d'enters de P fins a U. [P..U]
generaLLista(P, P, [P]).
generaLLista(P, U, LL):- P < U,  K is P+1, generaLLista(K, U, R),
    append([P], R, LL),!.
```

Execució

```
?- generaLLista(1, 64, LL).
LL = [1, 2, 3, 4, 5, 6, 7, 8, 9|...].
```

Separa Per N

Donats un N i una llista de K elements, divideix la llista de K elements en X subllistes de N elements.

Predicat

```
%%%%%%%%%%%%%%%%%%%% DONE MACARRONE!!
% separaPerN(N, L, R)
% Donada una llista L d'elements, R serà L dividida en K parts de N
    elements cada part.
% -> N és el nombre d'elements que han de tenir les subllistes
% -> L és la llista a tractar
% -> R és una llista de llistes de N elements cada subllista.
separaPerN(_, [], []).
separaPerN(N, L, R):- treuN(N, L, EXTRETS, CUA), separaPerN(N, CUA,
    NOUEXTRETS), append([EXTRETS], NOUEXTRETS, R), !.
```

Execució

```
?- generaLLista(1, 64, LL), separaPerN(4, LL, RESULTAT).
LL = [1, 2, 3, 4, 5, 6, 7, 8, 9|...],
RESULTAT = [[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12], [13, 14, 15,
    16], [17, 18, 19, 20], [21, 22, 23|...], [25, 26|...], [29|...],
    [...|...]|...].
```

Un cop feta aquesta crida està tot dividit en caselles, s'ha de tornar a fer la mateixa crida per tenir dividida la llista dos vegades i així obtenir les files a partir de les caselles.

```
?- generaLLista(1, 64, LL), separaPerN(4, LL, LL2), separaPerN(4, LL2,
    RESULTAT).
LL = [1, 2, 3, 4, 5, 6, 7, 8, 9|...],
LL2 = [[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12], [13, 14, 15, 16],
    [17, 18, 19, 20], [21, 22, 23|...], [25, 26|...], [29|...],
    [...|...]|...],
RESULTAT = [[[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12], [13, 14, 15,
    16]], [[17, 18, 19, 20], [21, 22, 23, 24], [25, 26, 27, 28], [29,
    30, 31|...]], [[33, 34, 35, 36], [37, 38, 39, 40], [41, 42, 43|...],
     [45, 46|...]], [[49, 50, 51, 52], [53, 54, 55|...], [57, 58|...],
    [61|...]]].
```

Crides internes

- Treu N

Treu N

Divideix una llista en EXTRETS i CUA on EXTRETS és una llista de N elements de L i CUA és la resta expressada com a llista.

Predicat

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%% DONE MACARRONE!!
% treuN(N, L, EXTRETS, CUA)
% Divideix L en EXTRETS i CUA, on EXTRETS conté N elements de L i CUA
    la resta.
% N -> nombre d'elements a extreure
% L -> Llista d'elements a tractar
% EXTRETS -> N elements de L en forma de LLISTA
% CUA -> L - els N primers elements de L en forma de LLISTA
treuN(0, L, [], L).
treuN(N, L, EXTRETS, CUA):- K is N-1, K >= 0, append([A], B, L), treuN(
    K, B, NOUEX, CUA), append([A], NOUEX, EXTRETS), !.
```

Al final tenim una matriu cúbica representada en llistes aniuades on cada casella té una llista de variables que representen la possibilitat que hi hagi un valor en concret en aquella casella.

Execució

Fem un exemple concret i petit que no forma part de la resolució del sudoku de l'exemple.

```
?- treuN(4, [1,2,3,4,5,6,7,8], EXTRETS, CUA).
EXTRETS = [1, 2, 3, 4],
CUA = [5, 6, 7, 8].
```

Inicialitzar

Aquest predicat tradueix els inputs representats en llista de c(F,C,V) en una CNF amb clàusules untàries on cada clàusula representa el fet que es fixi un valor en concret en el tauler generat.

Predicat

```
%%%%%%%%%%%%%%%%%%%%%%%%%
% inicialitzar(N,LI,F)
% Donat el domini de les caselles (N), i la llista d'inicialitzacions (
    de forma [c[1,2,5],...]),
% -> el tercer parametre sera la CNF formada de clausules unitaries que
     forcen els valors corresponents
```

```
%    a les caselles corresponents (als N-dominis corresponents)
inicialitzar(_,[],[]):-!.
inicialitzar(N,LI,CNF):- append([c(F,C,D)],Resta,LI),
                         POS is N*N*(F-1) + (C-1)*N+D,
                         inicialitzar(N,Resta,W),
                         append([[POS]],W,CNF).
```

Execució

```
?- inicialitzar(4, [c(1,1,3),c(2,2,1),c(4,4,3)], CNF).
CNF = [[3], [21], [63]].
```

Codifica sudoku

El que es proposa amb aquest predicat és generar les cnf que codifiquen que només hi ha un nombre possible per cada casella (kdominis), les cnf que codifiquen que tots els nombres de les caselles son diferents a cada una de les files, les cnf que codifiquen que tots els nombres de les caselles son diferents a cada una de les columnes i les cnf que codifiquen que tots els nombres de les caselles son diferents per cada un dels subquadrats del sudoku.

Els appends següents empalmen les diferents cnf per produïr la CNF del paràmetre per tot el sudoku.

Predicat

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% codificaSudoku(N,Tauler,C0,CNF)
% Donat el domini de les caselles (N) d'un Sudoku, el seu tauler i la
    CNF que l'inicialitza (C0),
% -> el quart parametre sera la CNF que codifica el Sudoku.
codificaSudoku(N,Tauler,C0,CNF):-  kdominis(Tauler,C1),
                                   allDiffFiles(Tauler,C2),
                                   allDiffColumnes(Tauler,C3),
                                   allDiffQuadrats(Tauler,N,C4),
                                   append(C1,C2,C12), append(C3,C4,C34)
                                       ,
                                   append(C0,C12,C012), append(C012,C34
                                       ,CNF).
```

Execució

```
?- festauler(4, T), codificaSudoku(4, T, [[3],[21],[63]], CNF).
```

```
T = [[[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12], [13, 14, 15, 16]],
    [[17, 18, 19, 20], [21, 22, 23, 24], [25, 26, 27, 28], [29, 30,
    31|...]], [[33, 34, 35, 36], [37, 38, 39, 40], [41, 42, 43|...],
    [45, 46|...]], [[49, 50, 51, 52], [53, 54, 55|...], [57, 58|...],
    [61|...]]],
CNF = [[3], [21], [63], [1, 2, 3, 4], [-1, -2], [-1, -3], [-1, -4],
    [-2|...], [...|...]|...].
```

Crides internes

- K dominis
- All Diff Files
- All Diff Columnes
- All Diff Quadrats

K dominis

Reb el tauler i codifica el "exactament un" per a cada casella monta la codificació conforme només accepta un valor tal com es mostra en l'exemple d'execució següent.

Predicat

```
%%%%%%%%%%%%%%%
% kdominis(T,F)
% Donat un Tauler,
% -> el segon parametre es la CNF que codifica els exactamentUn per
    cada casella (K-domini)
kdominis([],[]).
kdominis([[]|T],F):-kdominis(T,F).
kdominis(T,F):- append([PF],CUA,T), append([PE],CUAFILA,PF),
    exactamentUn(PE,CNF1),
                append([CUAFILA],CUA,RESTA),kdominis(RESTA,CNFFINAL),
                   append(CNF1,CNFFINAL,F),!.
```

Execució

Podem observar que la resposta CNF és part de la resposta CNF del predicat codifica sudoku.

```
?- festauler(4,T), kdominis(T,CNF).
T = [[[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12], [13, 14, 15, 16]],
    [[17, 18, 19, 20], [21, 22, 23, 24], [25, 26, 27, 28], [29, 30,
    31|...]], [[33, 34, 35, 36], [37, 38, 39, 40], [41, 42, 43|...],
    [45, 46|...]], [[49, 50, 51, 52], [53, 54, 55|...], [57, 58|...],
    [61|...]]],
```

```
3 CNF = [[1, 2, 3, 4], [-1, -2], [-1, -3], [-1, -4], [-2, -3], [-2, -4],
      [-3, -4], [5|...], [...|...]|...].
```

Crides internes

- Exactament Un

Exactament Un

Per cada una de les caselles és cridat aquest predicat que colabora en la generació de la CNF per codificar el sudoku sencer. Valida que una casella només prengui un dels seus valors possibles.

Una CNF "exactament un" que codifica un únic nombre per la primera casella d'un sudoku de N=4 pren la forma que es veu en l'exemple d'execució d'aquest mateix apartat. Aquest exemple d'execució es realitza sobre el cas explicat al llarg de l'informe però només per una de les caselles del tauler.

Predicat

```
1 %%%%%%%%%%%%%%%%%%%%%
2 % exactamentUn(L,CNF)
3 % Donat una llista de variables booleanes,
4 % -> el segon parametre sera la CNF que codifica que exactament una
      sigui certa.
5 exactamentUn([],[]).
6 exactamentUn([A],[[A]]).
7 exactamentUn(L,CNF):- negat(L, LNEG), montaParelles(LNEG,PARELLES),
      append([L],PARELLES, CNF).
```

Execució

Veiem l'execució només per la primera casella d'un sudoku n=4.

```
1 ?- exactamentUn([1,2,3,4],CNF).
2 CNF = [[1, 2, 3, 4], [-1, -2], [-1, -3], [-1, -4], [-2, -3], [-2, -4],
      [-3, -4]].
```

Crides internes

- Negat
- Monta Parelles

Negat

Les CNF's estan formades de llistes de llistes de k dominis on els k dominis son nombres enters. Cada nombre enter simbolitza un nombre donat una fila i una columna. Per tant codifica una variable, el fet de que la variable sigui certa o falça es representa amb un enter positiu o negatiu. La feina d'aquest predicat és la de capgirar el valor de les variables i passar-les de certes a falces o vicevesa. I per fer-ho s'utiltiza el 0 per restar-li el valor enter negatiu o positiu en funció de l'entrada.

Predicat

```
%%%%%%%%%%%%%
% negat(A,ANEG)
% Donat una llista de literals enters torna una llista amb els literals
    enters negats.
negat([],[]).
negat([A],[ANEG]):- ANEG is 0-A,!.
negat([A|CUA], [ANEG|CUANEG]):- negat([A],[ANEG]), negat(CUA,CUANEG).
```

Execució

```
?- negat([1,-2,-3,4],NEGAT).
NEGAT = [-1, 2, 3, -4].
```

Monta Parelles

La idea és que donada una llista amb n variables es cridi al predicat combina que genera una llista amb la combinació d'una de les variables amb la resta, fent així una llista de parelles de variables. Veure l'especificació del Combina.

La idea és que a mesura que es vagin combinant les variables no es tornint a combinar repetits i per tant s'exclou un cop combinada la variable i es torna a cridar el montaParelles amb la resta de variables que no poden haver-se combinat.

Predicat

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% montaParelles(L, PARELLES)
% PARELLES és el resultat de combinar tots els elements de L entre sí.
montaParelles([],[]).
montaParelles([A,B],[[A,B]]):-!.
montaParelles(L, PARELLES):- append([_],CUA,L),
              % [_] ja no ens importa per que ja s'ha combinat amb l'
                  element extret.
```

```
8                               combina(L,PIVOTA), % [[1,2],[1,3],[1,4]]
9                               montaParelles(CUA,ITSEG),% Montarà amb
                                    [2,3,4]
10                              append(PIVOTA,ITSEG,PARELLES),!.
```

Execució

```
1  ?- montaParelles([1,2,3,4], PARELLES).
2  PARELLES = [[1, 2], [1, 3], [1, 4], [2, 3], [2, 4], [3, 4]].
```

Crides internes

- Combina

Combina

Donades una llista de variables, agafarà la primera i la combinarà amb la resta (només la primera). Ja es tornarà a cridar més endavant sense el primer element que ja s'ha combinat.

Predicat

```
1  %%%%%%%%%%%%%%%%%%%%
2  % combina(L,COMBINAT)
3  % COMBINAT és la llista resultant de combinar el primer element de L
       amb la resta
4  % L -> és una llista d'elements a tractar
5  combina([],[]).
6  combina([A],[[A]]).
7  combina([A,B],[[A,B]]).
8  combina(L, COMBINAT):- append([A,B], CUA, L),
9                         append([A],CUA,PIVOTA),
10                        combina(PIVOTA,ITSEG), append([[A,B]],ITSEG,
                             COMBINAT),!.
```

Execució

```
1  ?- combina([1,2,3,4],COMBINAT).
2  COMBINAT = [[1, 2], [1, 3], [1, 4]].
```

All Diff Files

Controla que donades totes les files de la matriu, no hi hagi cap element repetit en cap de les files, complint així el primer dels tres propòsits d'un sudoku (No hi ha elements repetits a les files, no hi ha elements repetits a les columnes, no hi ha elements repetits als quadrats). CNF la llista de clàusules resultat que verifica que no hi ha cap repetit a les files.

Predicat

```
%%%%%%%%%%%%%%%%%%%%
% allDiffFiles(T,F)
% Donat un Tauler,
% -> el segon parametre es la CNF que codifica que no hi hagi repetits
    (allDiff) als K-dominis de cada fila
allDiffFiles([],[]).
allDiffFiles(T,F):- append([PF],CUA,T),
                    allDiff(PF,CNF),
                    allDiffFiles(CUA,CNFCUA),
                    append(CNF,CNFCUA,F),!.
```

Execució

```
?- festauler(4,T), allDiffFiles(T, CNF).
T = [[[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12], [13, 14, 15, 16]],
    [[17, 18, 19, 20], [21, 22, 23, 24], [25, 26, 27, 28], [29, 30,
    31|...]], [[33, 34, 35, 36], [37, 38, 39, 40], [41, 42, 43|...],
    [45, 46|...]], [[49, 50, 51, 52], [53, 54, 55|...], [57, 58|...],
    [61|...]]],
CNF = [[-1, -5], [-1, -9], [-1, -13], [-5, -9], [-5, -13], [-9, -13],
    [-2, -6], [-2|...], [...|...]|...].
```

Crides internes

- All Diff

All Diff

All diff reb una llista de llistes de variables que representen un subconjunt de caselles. Aquest subconjunt de caselles pot ser un a fila, una columna o bé un quadrat. El que codifica el alldiff és que totes les caselles que entren siguin diferents. Com que el format d'entrada és una llista i nosaltres volem combinar les diferents opcions que hi ha dins del llistat de varibles el que fem és transposar aquest llistat per que el seu format passi d'aquí :

```
[[1,2,3,4],[5,6,7,8],[9,10,11,12],[13,14,15,16]]
```

A aquí :

```
[[1,5,9,13],[2,6,10,14],[3,7,11,15],[4,8,12,16]]
```

Realitzant aquesta transposició ja podem accedir directament a vincular els diferents elements cridant al montaparelles per cada una de les possibilitats forçant que per exemple només hi hagi un 1, o només hi hagi un 2 o només hi hagi un 3 … fin a N.

Predicat

```
%%%%%%%%%%%%%%
% allDiff(L,F)
% Donat una llista (L) de Kdominis,
% -> CNF  codifica que no poden prendre el mateix valor.
% - Extraurem el primer literal de cada k-domini
% - muntem una llista amb tots els literals extrets
% - cridem a alldif amb totes les cues.
allDiff(L,CNF):-matTransposa(L,T), iAllDiff(T,CNF).


%%%%%%%%%%%%%%%
% INMERSIÓ (AHUUUUUHA)
% iAllDiff(L,CNF)
% Donada una llista de caselles L
% -> CNF serà la clausla que valida que totes les caselles son
    diferents
iAllDiff([],[]).
iAllDiff(L,CNF):-append([PCOL],MAT,L), negat(PCOL, PCOLNEG),
    montaParelles(PCOLNEG, PCOLNEGCOMB),
        iAllDiff(MAT, CNFPARCIAL), append(PCOLNEGCOMB, CNFPARCIAL, CNF)
           ,!.
```

Execució

Codifiquem la primera fila del sudoku N=4.

```
?- allDiff([[1,2,3,4],[5,6,7,8],[9,10,11,12],[13,14,15,16]], CNF).
CNF = [[-1, -5], [-1, -9], [-1, -13], [-5, -9], [-5, -13], [-9, -13],
    [-2, -6], [-2|...], [...|...]|...].
```

Crides internes

- Mat Transposa
- Negat
- Monta Parelles

Mat Transposa

Donada qualsevol matriu, fa la seva transposada i la posa a RES. Per fer-ho es realitza fila per fila, fent que la fila es divideix en una fila per cada element i unint el resultat de cridar recursivament a la transposada de la cua per després unir-la per files.

Predicat

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% matTransposa(MAT,RES)
% Donat una matriu MAT.
% -> RES es la matriu transposada de MAT
matTransposa([],[]):-!.
matTransposa(MAT,RES):- append([PF],CUA,MAT),
                        transposarFila(PF,PFT),
                        matTransposa(CUA,REST),
                        matUnio(PFT,REST,RES),!.
```

Execució

Fem un exemple de transposar la primera fila de una matriu de N=4.

```
?- matTransposa([[1,2,3,4],[5,6,7,8],[9,10,11,12],[13,14,15,16]],
    TRANSPOSADA).
TRANSPOSADA = [[1, 5, 9, 13], [2, 6, 10, 14], [3, 7, 11, 15], [4, 8,
    12, 16]].
```

Un altre exemple seria transposar tot el tauler a nivell de Files i columnes, que tracta els dominins de cada casella no com una llista sinó com un element.

```
?- festauler(4,T), matTransposa(T, TRANSPOSADA).
T = [[[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12], [13, 14, 15, 16]],
    [[17, 18, 19, 20], [21, 22, 23, 24], [25, 26, 27, 28], [29, 30,
    31|...]], [[33, 34, 35, 36], [37, 38, 39, 40], [41, 42, 43|...],
    [45, 46|...]], [[49, 50, 51, 52], [53, 54, 55|...], [57, 58|...],
    [61|...]]],
TRANSPOSADA = [[[1, 2, 3, 4], [17, 18, 19, 20], [33, 34, 35, 36], [49,
    50, 51, 52]], [[5, 6, 7, 8], [21, 22, 23, 24], [37, 38, 39, 40],
    [53, 54, 55|...]], [[9, 10, 11, 12], [25, 26, 27, 28], [41, 42,
```

```
      43|...], [57, 58|...]], [[13, 14, 15, 16], [29, 30, 31|...], [45,
      46|...], [61|...]]].
```

Crides internes

- Transposar Fila
- Mat Unió

Transposar Fila

Donada una fila es retorna una llista de files on cada element de la llista L és una sola llista dins de RES.

Predicat

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%
% transposarFila(L,RES)
% Donada una llista (fila) L
% -> RES es el resultat de dividir la fila en columnes i retorna com
    una matriu
transposarFila([],[]).
transposarFila(L,RES):- append([P],CUA,L),
                        transposarFila(CUA,CUARES),
                        append([[P]],CUARES,RES), !.
```

Execució

```
?- transposarFila([[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12], [13,
    14, 15, 16]], FILA_TRANSPOSADA).
FILA_TRANSPOSADA = [[[1, 2, 3, 4]], [[5, 6, 7, 8]], [[9, 10, 11, 12]],
    [[13, 14, 15, 16]]].
```

Mat Unió

Donades dues matrius ESQ i DRE, el matunió genera la unió de les dues matriu a partir de unir-les fila a fila. És a dir la matriu [[A],[C]] U [[B],[D]] = [[A,B],[C,D]].

Aquesta unió és genèrica per qualsevol matriu ESQ i qualsevol Matriu DRE.

Predicat

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% matUnio(ESQ,DRE,MAT)
% Donades 2 matrius ESQ i PDRE
% -> MAT es la unió per files de les 2 matrius
% -> EX: matUnio([[1]],[[2],[3]],[[1,2],[3]])
matUnio([],T,T).
matUnio(T,[],T).
matUnio(ESQ,DRE,MAT):-append([PESQ],CUAESQ,ESQ), append([PDRE],CUADRE,
    DRE), append(PESQ,PDRE,PF),
                      matUnio(CUAESQ,CUADRE,MATRES),
                      append([PF],MATRES,MAT),!.
```

Execució

Els següents exemplifiquen el comportament del mat unió. (És la unió de dues matrius de tota la vida unides per files).

```
?- matUnio([[1],[2]],[[3]], RES).
RES = [[1, 3], [2]].

?- matUnio([[1],[2]],[[[3]]], RES).
RES = [[1, [3]], [2]].

?- matUnio([[1]],[[3],[4]], RES).
RES = [[1, 3], [4]].

?- matUnio([[1],[3]],[[],[4]], RES).
RES = [[1], [3, 4]].
```

All Diff Columnes

Simplement transposar la matriu que tenim, i cridar al all diff files. És un clar exemple d'abstracció.

Predicat

```
%%%%%%%%%%%%%%%%%%%%%%%%%
% allDiffColumnes(T,F)
% Donat un Tauler,
% -> el segon parametre es la CNF que codifica que no hi hagi repetits
    (allDiff) als K-dominis de cada columna
allDiffColumnes(MAT,F):- matTransposa(MAT,T),
                         allDiffFiles(T,F),!.
```

Execució

Veiem que ha transposat la matriu i que ha combinat els diferents parells de variables per que no hi hagi cap element repetit a les columnes.

```
?- festauler(4,T), allDiffColumnes(T, CNF).
T = [[[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12], [13, 14, 15, 16]],
    [[17, 18, 19, 20], [21, 22, 23, 24], [25, 26, 27, 28], [29, 30,
    31|...]], [[33, 34, 35, 36], [37, 38, 39, 40], [41, 42, 43|...],
    [45, 46|...]], [[49, 50, 51, 52], [53, 54, 55|...], [57, 58|...],
    [61|...]]],
CNF = [[-1, -17], [-1, -33], [-1, -49], [-17, -33], [-17, -49], [-33,
    -49], [-2, -18], [-2|...], [...|...]|...].
```

Crides internes

- Mat Transposa
- All diff Files

All Diff Quadrats

Aquest predicta extreu un llistat de caselles per a cada possible quadrat del sudoku. Plantegem un sistema de quadrats basat en una fòrmula matemàtica utiltizant les arrels de N, on N és el nombre del sudoku.

Aquests quadrats segueixen la lògica expressada en el següent esquema :

```
|---|---|---|---|
| 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 |
| 2 | 2 | 3 | 3 |
| 2 | 2 | 3 | 3 |
|---|---|---|---|
```

D'aquesta manera el primer quadrat estarà format per les caselles c(1,1), c(1,2), c(2,1), c(2,2).

El procediment que es segueix és el d'agafar fila per fila i verificar que les caselles estan dins del subquadrat que es demana. Per aquest motiu s'utilitza inmersió. Per poder generar una llista de valors enters de 0 fins a N-1 que permeti demanar per cada valor el subquadrat X del sudoku on 0 <= X < N.

Utilitzem el subset per generar el llistat de caselles que s'han de combinar per generar la CNF corresponent a cada quadrat.

Predicat

```
%%%%%%%%%%%%%%%%%%%%%%%%%%
% allDiffQuadrats(T,N,F)
% Donat un Tauler, i la mida del K-domini (que es tambe el nombre de
    quadrats que hi ha),
% -> el tercer parametre es la CNF que codifica que no hi hagi repetits
     (allDiff) als K-dominis de cada quadrat
allDiffQuadrats([],_,[]).
allDiffQuadrats(T,N,F):- iAllDiffQuadrats(T,N,0,F).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% iAllDiffQuadrats(T,N,M,F)
% Inmersió per allDiffQuadrats.
iAllDiffQuadrats(_,N,N,[]).
iAllDiffQuadrats(T,N,M,F):- X is M+1,
                            subset(T,N,M,S),
                            allDiff(S,CNF),
                            iAllDiffQuadrats(T,N,X,CNFX),
                            append(CNF,CNFX,F),!.
```

Execució

```
?- N=4, festauler(N,T), allDiffQuadrats(T, N, CNF).
N = 4,
T = [[[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12], [13, 14, 15, 16]],
    [[17, 18, 19, 20], [21, 22, 23, 24], [25, 26, 27, 28], [29, 30,
    31|...]], [[33, 34, 35, 36], [37, 38, 39, 40], [41, 42, 43|...],
    [45, 46|...]], [[49, 50, 51, 52], [53, 54, 55|...], [57, 58|...],
    [61|...]]],
CNF = [[-1, -5], [-1, -17], [-1, -21], [-5, -17], [-5, -21], [-17,
    -21], [-2, -6], [-2|...], [...|...]|...].
```

Crides internes

- Subset
- All Diff

Subset

A partir de una matriu, extraurem el llistat de caselles S corresponent a el subquadrat Q demanat per paràmetre. Per fer-ho es sitúa un inici en funció del nombre del quadrat demanat seguint la fòrmula:

fila = (Q mod SQRT(N)) * SQRT(N) columna = (Q / SQRT(N)) * SQRT(N)

UN cop calculats els inicis passem tant el tauler com els inicis a un predicat extreu que acaba extreient tots els elements que estan compresos entre els inicis i els inicis + SQRT(N).

Predicat

```
%%%%%%%%%%%%%%%%%%
% subset(T,N,Q,S)
% Serveix per extreure Un subquadrat amb el valor SUBQUADRAT.
% T és tauler
% N és el valor del sudoku.
% S Serà el subset referent al subquadrat de numero subquadrat.
% -----------------
% | 0 | 0 | 1 | 1 |
% | 0 | 0 | 1 | 1 |
% | 2 | 2 | 3 | 3 |
% | 2 | 2 | 3 | 3 |
% -----------------
% SUBQUADRAT pot prendre valor de 1 - N on en la matriu expressada
    anteriorment cada valor
% refereix al subquadrat marcat amb els nombres de 1 a 4.
subset(T,N,SUBQUADRAT,S):- P is truncate(sqrt(N)),
                           X is (SUBQUADRAT mod P)*P,
                           Y is (SUBQUADRAT div P)*P,
                           extreu(T,X,Y,N,S).
```

Execució

Veiem la crida de subset per cada un dels quadrats en un sudoku N=4.

```
?- festauler(4,T), subset(T,4,0,SUBSET).
T = [[[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12], [13, 14, 15, 16]],
    [[17, 18, 19, 20], [21, 22, 23, 24], [25, 26, 27, 28], [29, 30,
    31|...]], [[33, 34, 35, 36], [37, 38, 39, 40], [41, 42, 43|...],
    [45, 46|...]], [[49, 50, 51, 52], [53, 54, 55|...], [57, 58|...],
    [61|...]]],
SUBSET = [[1, 2, 3, 4], [5, 6, 7, 8], [17, 18, 19, 20], [21, 22, 23,
    24]].

?- festauler(4,T), subset(T,4,1,SUBSET).
T = [[[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12], [13, 14, 15, 16]],
    [[17, 18, 19, 20], [21, 22, 23, 24], [25, 26, 27, 28], [29, 30,
    31|...]], [[33, 34, 35, 36], [37, 38, 39, 40], [41, 42, 43|...],
    [45, 46|...]], [[49, 50, 51, 52], [53, 54, 55|...], [57, 58|...],
    [61|...]]],
```

```
SUBSET = [[9, 10, 11, 12], [13, 14, 15, 16], [25, 26, 27, 28], [29, 30,
    31, 32]].

?- festauler(4,T), subset(T,4,2,SUBSET).
T = [[[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12], [13, 14, 15, 16]],
   [[17, 18, 19, 20], [21, 22, 23, 24], [25, 26, 27, 28], [29, 30,
   31|...]], [[33, 34, 35, 36], [37, 38, 39, 40], [41, 42, 43|...],
   [45, 46|...]], [[49, 50, 51, 52], [53, 54, 55|...], [57, 58|...],
   [61|...]]],
SUBSET = [[33, 34, 35, 36], [37, 38, 39, 40], [49, 50, 51, 52], [53,
   54, 55, 56]].

?- festauler(4,T), subset(T,4,3,SUBSET).
T = [[[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12], [13, 14, 15, 16]],
   [[17, 18, 19, 20], [21, 22, 23, 24], [25, 26, 27, 28], [29, 30,
   31|...]], [[33, 34, 35, 36], [37, 38, 39, 40], [41, 42, 43|...],
   [45, 46|...]], [[49, 50, 51, 52], [53, 54, 55|...], [57, 58|...],
   [61|...]]],
SUBSET = [[41, 42, 43, 44], [45, 46, 47, 48], [57, 58, 59, 60], [61,
   62, 63, 64]].
```

Crides internes

- Extreu

Extreu

L'objectiu de l'extreu és que donats dos inicis referents a la fila i la columna extregui els elements compressos entre els inicis i els inicis + SQRT(N) que son els que delimiten el quadrat.

Es realitza una verificació de tots els elements del tauler (tallant quan s'ha s'ha extret tot el quadrat) per cada quadrat que es vol extreure. Comprovant si la fila i la columna avaluats en aquest moment estan compresos dins del rang estipulat entre inici i final per files i columnes.

Un cop hem trobat tots els elements d'una fila, la fila es suprimeix i es segueix amb la cua que son la resta de files de la matriu. Si No s'ha trobat l'element encara es van passant elements i files fins arrivar a dins del rang. En cas que es trobi tots els elements de dins del quadrat i encara quedin files per mirar es talla.

Predicat

```
%%%%%%%%%%%%%%%%%%%%%%
% extreu(T,XI,YI,N,S)
```

```
3  % Extreu el subquadrat que estigui entre XI - XI + SQRT(N) i  YI - YI +
       SQRT(N) on X refereix a files i Y a columnes.
4  % T és Tauler
5  % XI és Fila inicial
6  % YI és columna inicial
7  % N és el valor del SUDOKU
8  % S serà el subquadrat de tamany SQRT(N) que està entre  XI - XI + SQRT
       (N) i  YI - YI + SQRT(N).
9  extreu(T,XI,YI,N,S):- ARR is truncate(sqrt(N)),
10                       XF is XI + ARR, YF is YI+ARR,
11                       iExtreu(T,XI,YI,XF,YF,0,0,S), !.
12
13 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
14 % iExtreu(T,XI,YI,XF,YF,X,Y,S)
15 % Inmersió d'extreu. Retorna el subquadrat entre XI i XF, YI i YF.
16 % T és Tauler
17 % XI és Fila inici, XF és fila Final, X és Fila actual
18 % YI és columna inici, YF és columna Final, Y és columna actual
19 % S és el subquadrat format per tots els elements entre XI i XY, YI i
       YF.
20
21 % ja ha trobat tot el quadrat
22 iExtreu(_,_,_,_,YF,_,YF,[]):-!.
23 % Encara no he arribat a la fila que cerco.
24 iExtreu(T,XI,YI,XF,YF,_,Y,S):-    Y < YI, append([_],CUAF,T), YSEG is Y
       +1,
25                                  iExtreu(CUAF,XI,YI,XF,YF,0,YSEG,S).
26 % me passat de columnes (X) puc saltar a la linia seguent
27 iExtreu(T,XI,YI,XF,YF,XF,Y,S):-   append([_],CUAF,T), YSEG is Y+1,
28                                  iExtreu(CUAF,XI,YI,XF,YF,0,YSEG,S).
29 % No me passat ni de files ni de columnes i miro si estic dins del rang
       (extrec)
30 iExtreu(T,XI,YI,XF,YF,X,Y,S):-    append([PF],CUAF,T), append([PC],
       CUAC, PF),
31                                  dinsRang(XI,XF,YI,YF,X,Y),
32                                  append([CUAC],CUAF,MAT), XSEG is X+1,
33                                  iExtreu(MAT,XI,YI,XF,YF,XSEG,Y,Q),
34                                  append([PC],Q,S),!.
35 % Encara no he trobat l'inici del rang i vaig passant elements
36 iExtreu(T,XI,YI,XF,YF,X,Y,S):-    append([PF],CUAF,T),
37                                  append([_], CUAC, PF),
38                                  append([CUAC],CUAF,MAT),  X<XF, XS is
                                        X+1,
```

```
                                iExtreu(MAT,XI,YI,XF,YF,XS,Y,S).
```

Execució

```
?- festauler(4, T), extreu(T,2,2,4,QUADRAT).
T = [[[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12], [13, 14, 15, 16]],
    [[17, 18, 19, 20], [21, 22, 23, 24], [25, 26, 27, 28], [29, 30,
    31|...]], [[33, 34, 35, 36], [37, 38, 39, 40], [41, 42, 43|...],
    [45, 46|...]], [[49, 50, 51, 52], [53, 54, 55|...], [57, 58|...],
    [61|...]]],
QUADRAT = [[41, 42, 43, 44], [45, 46, 47, 48], [57, 58, 59, 60], [61,
    62, 63, 64]].
```

Crides internes

- Dins Rang

Dins Rang

Únicament comprova que x i y estan ditre de Xi i XF i Yi i Yf.

Predicat

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% dinsRang(XI,XF,YI,YF,X,Y)
% Comprova que X i Y estan entre XI i XF, YI i YF els finals no
    inclosos.
% XI és Fila inici, XF és fila Final, X és Fila actual
% YI és columna inici, YF és columna Final, Y és columna actual
dinsRang(XI,XF,YI,YF,X,Y):-  X>=XI, X<XF, Y>=YI, Y<YF.
```

Execució

```
?-  dinsRang(0,2,0,2,1,1).
true.

?- dinsRang(0,2,0,2,3,3).
false.
```

Crides internes

Sat

Mira la satisfactibilitat de una CNF. utiltiza l'algoritme de simplificació de variables eliminant clàusules de dins de la CNF a mesura que va decidint literals. Cada literal que fixa el valor per aquell literal com a part d'un model. Llavors elimina totes les clàusules on apareix aquest literal, i quan es troba ell mateix negat, el treu de la clàusula. Va aplicant aquest algoritme fins que es queda sense clàusules i per tant completa el model.

Si entre el les clàusules restants es troba la clàusula buida és que s'ha trobat un contraexemple i que per tant no té solució per a la interpretació donada fins el moment. Llavors el prolog genera el backtracking per veure si en alguna altre branca de l'arbre de cerca la CNF és satisfactible.

Predicat

```
%%%%%%%%%%
% sat(F,I,M)
% si F es satisfactible, M sera el model de F afegit a la interpretació
    I (a la primera crida I sera buida).
% Assumim invariant que no hi ha literals repetits a les clausules ni
   la clausula buida inicialment.
sat([],I,I):-     write('SAT!!, SUDOKU PISCINAS!!! '),nl,!.
sat(CNF,I,M):-
    % Ha de triar un literal 'duna clausula unitaria, si no 'nhi ha cap
        , llavors un literal pendent qualsevol.
    decideix(CNF,Lit),

    % Simplifica la CNF amb el Lit triat (compte pq pot fallar, es a
       dir si troba la clausula buida fallara i fara backtraking).
    simplif(Lit,CNF,CNFS),

    % crida recursiva amb la CNF i la interpretacio actualitzada
      append(I,[Lit],ISEG),
    sat(CNFS,ISEG ,M).
```

Execució

En el següent exemple d'execució li demanarem al Sat tots els possibles models entrant ";". No considerem que sigui una bona idea tallar aquest sat per que així podem aconseguir diferents models donada una CNF que representa el sudoku.

```
?- taulerSudoku(4, [c(1,1,3),c(2,2,1),c(4,4,3)],T, C), codificaSudoku
   (4, T, C, CNF), sat(CNF, [], MODEL).
SAT!!, SUDOKU PISCINAS!!!
```

```
3  T = [[[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12], [13, 14, 15, 16]],
       [[17, 18, 19, 20], [21, 22, 23, 24], [25, 26, 27, 28], [29, 30,
       31|...]], [[33, 34, 35, 36], [37, 38, 39, 40], [41, 42, 43|...],
       [45, 46|...]], [[49, 50, 51, 52], [53, 54, 55|...], [57, 58|...],
       [61|...]]],
4  C = [[3], [21], [63]],
5  CNF = [[3], [21], [63], [1, 2, 3, 4], [-1, -2], [-1, -3], [-1, -4],
       [-2|...], [...|...]|...],
6  MODEL = [3, 21, 63, -1, -2, -4, -22, -23, -24|...] ;
7  SAT!!, SUDOKU PISCINAS!!!
8  T = [[[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12], [13, 14, 15, 16]],
       [[17, 18, 19, 20], [21, 22, 23, 24], [25, 26, 27, 28], [29, 30,
       31|...]], [[33, 34, 35, 36], [37, 38, 39, 40], [41, 42, 43|...],
       [45, 46|...]], [[49, 50, 51, 52], [53, 54, 55|...], [57, 58|...],
       [61|...]]],
9  C = [[3], [21], [63]],
10 CNF = [[3], [21], [63], [1, 2, 3, 4], [-1, -2], [-1, -3], [-1, -4],
       [-2|...], [...|...]|...],
11 MODEL = [3, 21, 63, -1, -2, -4, -22, -23, -24|...] ;
12 SAT!!, SUDOKU PISCINAS!!!
13 T = [[[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12], [13, 14, 15, 16]],
       [[17, 18, 19, 20], [21, 22, 23, 24], [25, 26, 27, 28], [29, 30,
       31|...]], [[33, 34, 35, 36], [37, 38, 39, 40], [41, 42, 43|...],
       [45, 46|...]], [[49, 50, 51, 52], [53, 54, 55|...], [57, 58|...],
       [61|...]]],
14 C = [[3], [21], [63]],
15 CNF = [[3], [21], [63], [1, 2, 3, 4], [-1, -2], [-1, -3], [-1, -4],
       [-2|...], [...|...]|...],
16 MODEL = [3, 21, 63, -1, -2, -4, -22, -23, -24|...] ;
17 SAT!!, SUDOKU PISCINAS!!!
18 T = [[[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12], [13, 14, 15, 16]],
       [[17, 18, 19, 20], [21, 22, 23, 24], [25, 26, 27, 28], [29, 30,
       31|...]], [[33, 34, 35, 36], [37, 38, 39, 40], [41, 42, 43|...],
       [45, 46|...]], [[49, 50, 51, 52], [53, 54, 55|...], [57, 58|...],
       [61|...]]],
19 C = [[3], [21], [63]],
20 CNF = [[3], [21], [63], [1, 2, 3, 4], [-1, -2], [-1, -3], [-1, -4],
       [-2|...], [...|...]|...],
21 MODEL = [3, 21, 63, -1, -2, -4, -22, -23, -24|...] ;
22 SAT!!, SUDOKU PISCINAS!!!
23 T = [[[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12], [13, 14, 15, 16]],
       [[17, 18, 19, 20], [21, 22, 23, 24], [25, 26, 27, 28], [29, 30,
       31|...]], [[33, 34, 35, 36], [37, 38, 39, 40], [41, 42, 43|...],
```

```
    [45, 46|...]], [[49, 50, 51, 52], [53, 54, 55|...], [57, 58|...],
    [61|...]]],
C = [[3], [21], [63]],
CNF = [[3], [21], [63], [1, 2, 3, 4], [-1, -2], [-1, -3], [-1, -4],
    [-2|...], [...|...]|...],
MODEL = [3, 21, 63, -1, -2, -4, -22, -23, -24|...] ;
SAT!!, SUDOKU PISCINAS!!!
T = [[[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12], [13, 14, 15, 16]],
    [[17, 18, 19, 20], [21, 22, 23, 24], [25, 26, 27, 28], [29, 30,
    31|...]], [[33, 34, 35, 36], [37, 38, 39, 40], [41, 42, 43|...],
    [45, 46|...]], [[49, 50, 51, 52], [53, 54, 55|...], [57, 58|...],
    [61|...]]],
C = [[3], [21], [63]],
CNF = [[3], [21], [63], [1, 2, 3, 4], [-1, -2], [-1, -3], [-1, -4],
    [-2|...], [...|...]|...],
MODEL = [3, 21, 63, -1, -2, -4, -22, -23, -24|...].
```

Crides internes

- Decideix
- Simplif

Decideix

Si trobem una clàusula amb un sol literal, tirem pel dret amb aquest literal, altrament tirem pel dret amb el primer literal de la primera clàusula o el primer literal de la primera clàusula negat.

Predicat

```
%%%%%%%%%%%%%%%%%%%%
% decideix(F, Lit)
% Donat una CNF,
% -> el segon parametre sera un literal de CNF
%  - si hi ha una clausula unitaria sera aquest literal, sino
%  - un qualsevol o el seu negat.
                % detectar clàusula unitària
decideix(L,A):- append(_,[X|_], L),
                append([A],[],X),!.
                        % Retorna el primer element
decideix([PL|_], A) :- append([A],_,PL).
                       % Retorna el negat del primer element
decideix([PL|_], B) :- append([A],_,PL), B is 0-A.
```

Execució

```
| ?- decideix([[3],[21],[63],[1,2,3,4]], LITERAL).

LITERAL = 3

yes
| ?- decideix([[1,2,3,4]], LITERAL).

LITERAL = 1 ? ;
LITERAL = -1
```

Simplif

Va treien totes les clàusules on hi aparegui el literl que li passem i va treient els negats d'aquest literal de dins de les clàusules.

Predicat

```
%%%%%%%%%%%%%%%%%%%%%%
% simplif(Lit, CNF, CNFS)
% Donat un literal Lit i una CNF,
% -> el tercer parametre sera la CNF que ens han donat simplificada:
%  - sense les clausules que tenen lit
%  - treient -Lit de les clausules on hi es, si apareix la clausula buida fallara.
simplif(_, [], []). % QUAN TINGUEM LA LLISTA BUIDA RETORNEM LA LLISTA BUIDA.
                                        % Comprovem si conté lit a primera
simplif(Lit, [PRIMERA|CUA], CNFS) :- member(Lit, PRIMERA),
                                        % seguim amb la cua.
                                        simplif(Lit, CUA, CNFS), !.
                                        % neguem el literal i el fotem a x
simplif(Lit, [PRIMERA|CUA], CNFS) :- X is 0-Lit,
                                        append(A, [X|XS], PRIMERA),
                                        % Treure el literal negat a primera serà res
                                        append(A, XS, RES),
                                        % crida recursiva
                                        simplif(Lit, CUA, CUACNFS),
                                        % Muntem la cadena final
                                        append([RES], CUACNFS, CNFS), !.
```

```
21                                      % Crida recursiva quan no conté
                                            Lit
22 simplif(Lit, [PRIMERA|CUA], CNFS) :- simplif(Lit, CUA, CUACNFS),
23                                      % Concatenar resultats
24                                      append([PRIMERA], CUACNFS, CNFS),
                                            !.
```

Execució

```
1 ?- taulerSudoku(4, [c(1,1,3),c(2,2,1),c(4,4,3)],T, C), codificaSudoku
      (4, T, C, CNF), simplif(3, CNF, RESTA_CNF).
2 T = [[[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12], [13, 14, 15, 16]],
      [[17, 18, 19, 20], [21, 22, 23, 24], [25, 26, 27, 28], [29, 30,
      31|...]], [[33, 34, 35, 36], [37, 38, 39, 40], [41, 42, 43|...],
      [45, 46|...]], [[49, 50, 51, 52], [53, 54, 55|...], [57, 58|...],
      [61|...]]],
3 C = [[3], [21], [63]],
4 CNF = [[3], [21], [63], [1, 2, 3, 4], [-1, -2], [-1, -3], [-1, -4],
      [-2|...], [...|...]|...],
5 RESTA_CNF = [[21], [63], [-1, -2], [-1], [-1, -4], [-2], [-2, -4],
      [-4], [...|...]|...].
```

Mostra Sudoku

Mostra un sudoku ja sigui complert o incomplert a partir d'una entrada per teclat amb el format [c(F1,C1,V1)..c(Fn,Cn,Vn)].

Predicat

```
1 %%%%%%%%%%%%%%%%%%%%%%%
2 % mostraSudoku(N, IN)
3 % Mostra un sudoku expressat en format llistat de c(F,C,V) per pantalla
      .
4 % N és el tamany del sudoku
5 % IN és el sudoku representat en llistat de c(F,C,V).
6 mostraSudoku(N, IN) :- pintaSeparador(N), nl, iMostraSudoku(N,1,1,IN).
7
8 iMostraSudoku(N, F, _, _):-  F>N, !.
9 iMostraSudoku(N, F, C, IN):- C > N, write('|'), nl, pintaSeparador(N),
      nl,
10                             FSEG is F+1, iMostraSudoku(N, FSEG, 1, IN)
                                   .
```

```
iMostraSudoku(N, F, C, IN):- pintaCasella(F,C,IN), CSEG is C+1,
                             iMostraSudoku(N, F, CSEG, IN), !.
iMostraSudoku(N, F, C, IN):- CSEG is C+1, write('| '),
                             iMostraSudoku(N, F, CSEG, IN).
```

Execució

```
?- mostraSudoku(4, [c(1,1,3),c(2,2,1),c(4,4,3)]).
---------
|3| | | |
---------
| |1| | |
---------
| | | | |
---------
| | | |3|
---------
true.
```

Crides internes

- Pinta Separador
- Pinta casella

Pinta Separador

Pinta (N+2)+(N-1) guions com una sola linia.

Predicat

```
%%%%%%%%%%%%%%%%%%%%
% pintaSeparador(N)
% Pinta un separador generat amb '-' amb tamany N+2+N-1 o N*2+1 carà
    cters
pintaSeparador(N):- NF is (N+2)+(N-1), iPintaSeparador(1, NF).

%%%%%%%%%%%%%%%%%%%%%%%%%
% iPintaSeparador(N, NF)
% Crida inmersiva del pintaSeparador.
iPintaSeparador(N, NF):- N>NF, !.
iPintaSeparador(N, NF):- write('-'), NSEG is N+1, iPintaSeparador(NSEG,
    NF).
```

Execució

```
?- pintaSeparador(4).
---------
true.
```

Pinta Casella

Cerca dins de la llista in si hi ha una casella que coincideixi amb files i columnes a partir del l'append i si hi és el pinta. Si no hi és retorna no.

Predicat

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% pintaCasella(F,C,IN, FCUA)
% Pinta el valor de una casella amb el separador al davant si
    coincideixen F = IN[F] i C = IN[C].
% F és el valor de la fila, C és el valor de la Columna
% IN son les caselles possibles a pintar per la casella [F,C]
% FCUA seran les caselles restants si es compleix que es pot pintar.
pintaCasella(F,C,IN) :- append(_,[c(F,C,V)|_], IN), write('|'), write(V
    ), !.
```

Execució

```
?- pintaCasella(1,1,[c(1,1,3),c(2,2,1),c(4,4,3)]).
|3
true.


?- pintaCasella(1,2,[c(1,1,3),c(2,2,1),c(4,4,3)]).
false.
```

Exemples i jocs de proves.

Hi ha una série de fitxers que comencen per la paraula sudoku i que son de tipus txt on hi ha diferents jocs de proves.

SUDOKUS AMB SOLUCIÓ

- **sudoku1.txt** -> Té solució
- **sudoku2.txt** -> Té solució

sudoku1

```
| ?- resol(9,[c(1,2,5),c(1,5,8),c(1,8,3),c(2,2,2),c(2,3,3),c(2,4,9),c
   (2,6,5),c(2,7,7),c(3,1,9),c(3,3,6),c(3,4,7),c(4,2,4),c(4,3,2),c
   (4,5,7),c(4,7,8),c(4,8,9),c(5,1,6),c(5,9,5),c(6,2,9),c(6,3,1),c
   (6,5,5),c(6,7,4),c(6,8,6),c(7,6,1),c(7,7,9),c(7,9,3),c(8,3,4),c
   (8,4,3),c(8,6,6),c(8,7,5),c(8,8,7),c(9,2,3),c(9,5,4),c(9,8,1)]).

NEM A RESOLDRE EL SUDOKU :
.................................
------------------
| |5| | |8| | |3| |
------------------
| |2|3|9| |5|7| | |
------------------
|9| |6|7| | | | | |
------------------
| |4|2| |7| |8|9| |
------------------
|6| | | | | | | |5|
------------------
| |9|1| |5| |4|6| |
------------------
| | | | | |1|9| |3|
------------------
| | |4|3| |6|5|7| |
------------------
| |3| | |4| | |1| |
------------------
SAT!!, SUDOKU PISCINAS!!!
.................................
------------------
|4|5|7|1|8|2|6|3|9|
------------------
|1|2|3|9|6|5|7|8|4|
------------------
|9|8|6|7|3|4|1|5|2|
```

```
------------------
|5|4|2|6|7|3|8|9|1|
------------------
|6|7|8|4|1|9|3|2|5|
------------------
|3|9|1|2|5|8|4|6|7|
------------------
|7|6|5|8|2|1|9|4|3|
------------------
|2|1|4|3|9|6|5|7|8|
------------------
|8|3|9|5|4|7|2|1|6|
------------------
```

sudoku2

```
| ?- resol(9, [c(1,4,8), c(1,8,4), c(1,9,7), c(2,4,2), c(2,5,3), c
   (2,9,5), c(3,4,7), c(3,8,8), c(4,2,8), c(4,3,7), c(4,6,3),c(5,2,5),
   c(5,5,4), c(5,6,8), c(5,9,2), c(6,3,3), c(6,4,5), c(6,5,2), c(6,7,8)
   ,c(6,8,9), c(7,1,7), c(7,4,3), c(8,1,6), c(8,2,2), c(8,6,5), c
   (8,7,9) ,c(8,8,7), c(8,9,3), c(9,2,4), c(9,8,2), c(9,9,6)]).

NEM A RESOLDRE EL SUDOKU :
.................................
------------------
| | | |8| | | |4|7|
------------------
| | | |2|3| | | |5|
------------------
| | | |7| | | |8| |
------------------
| |8|7| | |3| | | |
------------------
| |5| | |4|8| | |2|
------------------
| | |3|5|2| |8|9| |
------------------
|7| | |3| | | | | |
------------------
|6|2| | | |5|9|7|3|
------------------
```

```
| |4| | | | | |2|6|
-------------------
SAT!!, SUDOKU PISCINAS!!!
.....................................
-------------------
|9|1|2|8|5|6|3|4|7|
-------------------
|8|7|4|2|3|9|1|6|5|
-------------------
|5|3|6|7|1|4|2|8|9|
-------------------
|2|8|7|1|9|3|6|5|4|
-------------------
|1|5|9|6|4|8|7|3|2|
-------------------
|4|6|3|5|2|7|8|9|1|
-------------------
|7|9|5|3|6|2|4|1|8|
-------------------
|6|2|1|4|8|5|9|7|3|
-------------------
|3|4|8|9|7|1|5|2|6|
-------------------

true ? ;

(3984 ms) no
```

SUDOKUS SENSE SOLUCIÓ

- **sudokuNoQ.txt** -> No té solució per que té elements repetits en un quadrat.
- **sudokuNoF.txt** -> No té solució per que té elements repetits en una fila.
- **sudokuNoC.txt** -> No té solució per que té elements repetits en una columna.

sudokuNoQ

```
| ?- resol(9,[c(1,2,5),c(1,5,8),c(1,8,3),c(2,2,2),c(2,3,3),c(2,4,9),c
   (2,6,5),c(2,7,7),c(3,1,9),c(3,3,6),c(3,4,7),c(4,2,4),c(4,3,2),c
   (4,5,7),c(4,7,8),c(4,8,9),c(5,1,6),c(5,9,5),c(6,2,9),c(6,3,1),c
   (6,5,5),c(6,7,4),c(6,8,6),c(7,6,1),c(7,7,9),c(7,9,3),c(8,3,4),c
```

```
    (8,4,3),c(8,6,6),c(8,7,5),c(8,8,7),c(9,2,3),c(9,5,4),c(9,8,1), c
    (9,9,9)]).

NEM A RESOLDRE EL SUDOKU :
.....................................
-------------------
| |5| | |8| | |3| |
-------------------
| |2|3|9| |5|7| | |
-------------------
|9| |6|7| | | | | |
-------------------
| |4|2| |7| |8|9| |
-------------------
|6| | | | | | | |5|
-------------------
| |9|1| |5| |4|6| |
-------------------
| | | | | |1|9| |3|
-------------------
| | |4|3| |6|5|7| |
-------------------
| |3| | |4| | |1|9|
-------------------

(2094 ms) no
```

sudokuNoF

```
| ?- resol(9,[c(1,2,5),c(1,5,8),c(1,8,3),c(2,2,2),c(2,3,3),c(2,4,9),c
    (2,6,5),c(2,7,7),c(3,1,9),c(3,3,6),c(3,4,7),c(4,2,4),c(4,3,2),c
    (4,5,7),c(4,7,8),c(4,8,9),c(5,1,6),c(5,9,5),c(6,2,9),c(6,3,1),c
    (6,5,5),c(6,7,4),c(6,8,6),c(7,6,1),c(7,7,9),c(7,9,3),c(8,3,4),c
    (8,4,3),c(8,6,6),c(8,7,5),c(8,8,7),c(9,2,3),c(9,5,4),c(9,8,1), c
    (9,9,4)]).

NEM A RESOLDRE EL SUDOKU :
.....................................
-------------------
| |5| | |8| | |3| |
-------------------
```

```
| |2|3|9| |5|7| | |
------------------
|9| |6|7| | | | | |
------------------
| |4|2| |7| |8|9| |
------------------
|6| | | | | | | |5|
------------------
| |9|1| |5| |4|6| |
------------------
| | | | | |1|9| |3|
------------------
| | |4|3| |6|5|7| |
------------------
| |3| | |4| | |1|4|
------------------

(2109 ms) no
```

sudokuNoC

```
| ?- resol(9,[c(1,2,5),c(1,5,8),c(1,8,3),c(2,2,2),c(2,3,3),c(2,4,9),c
   (2,6,5),c(2,7,7),c(3,1,9),c(3,3,6),c(3,4,7),c(4,2,4),c(4,3,2),c
   (4,5,7),c(4,7,8),c(4,8,9),c(5,1,6),c(5,9,5),c(6,2,9),c(6,3,1),c
   (6,5,5),c(6,7,4),c(6,8,6),c(7,6,1),c(7,7,9),c(7,9,3),c(8,3,4),c
   (8,4,3),c(8,6,6),c(8,7,5),c(8,8,7),c(9,2,3),c(9,5,5),c(9,8,1)]).

NEM A RESOLDRE EL SUDOKU :
.................................
------------------
| |5| | |8| | |3| |
------------------
| |2|3|9| |5|7| | |
------------------
|9| |6|7| | | | | |
------------------
| |4|2| |7| |8|9| |
------------------
|6| | | | | | | |5|
------------------
| |9|1| |5| |4|6| |
```

```
-------------------
| | | | | |1|9| |3|
-------------------
| | |4|3| |6|5|7| |
-------------------
| |3| | |5| | |1| |
-------------------

(2141 ms) no
```

Comentaris sobre la solució i aspectes positius.

Abstracció

Com és d'esperar, en les nostres pràctiques mirem d'abstreure les funcionalitats per reutilitzar-les en diferents parts del programa. Per exemple podem observar les anbstraccions dels predicats montaParelles o del matTransposa i fins i tot del allDiff entre altres.

Talls.

S'ha procurat que el prolog faci la feina que ha de fer i cap altre. Per demostrar-ho s'ha realitzat un exemple d'execució de cada predicat per que e pugui veure la solució que es proposa i juntament amb la definició del predicat es pugui interpretar que no hi ha més opcions possibles.

S'ha de descartar el tall en el sat, ja que el sat interessa que torni models si el cridem independentment de la resta de predicats.

L'us de l'append

L'append és el més utilitzat per el tractament de llistes en aquesta pràctica.

Cerques

L'ús de l'append per fer una cerca en un llistat per exemple de caselles (Cerca si hi ha la casella que coincideix en files i columnes a dins de IN i la pinta):

```
pintaCasella(F,C,IN) :- append(_,[c(F,C,V)|_], IN), write('|'), write(V
    ), !.
```

Detectar clàusula unitària

Podem usar-la per detectar si hi ha una llista amb un sol element dins d'una llista.

```
decideix(L,A):- append(_,[X|_], L), append([A],[],X),!.
```

Construir i desconstruir Llistes

És molt usat en general per poder desengranar les llistes a plaer. També per construir-les.

```
combina(L, COMBINAT):- append([A,B], CUA, L),
                               append([A],CUA,PIVOTA),
                               combina(PIVOTA,ITSEG),
                          append([[A,B]],ITSEG,COMBINAT),!.
```

Conclusió

La pràctica de prolog ha sigut molt clara i fàcil d'interpretar un cop estudiat l'esquelet de la pràctica. Reconeixem que sense l'esquelet ens hi hauríem fet bastant de mal. Ens ha agradat la manera que té el prolog de treballar, com evalúa i genera l'arbre de cerca i com realitza el backtracking ell solet. És quelcom curiós i amb aplicació directe a problemes que son molt interessants.