SudokuSAT
Pràctica de Programació Lògica

Paradigmes i Llenguatges de Programació

8/5/18

En aquesta pràctica el que farem serà implementar un predicat per decidir la satisfactibilitat de fórmules Booleanes en CNF mitjançant el procediment de decisió DPLL. Un cop el tinguem, codificarem la resolubilitat de Sudokus en fórmules Booleanes i farem servir el nostre algorisme per trobar-ne la solució (quan en tingui).

La pràctica està estructurada en diferents tasques:

1. Les varibles Booleanes es representaran en el nostre programa Prolog amb nombres positius. Els literals seran nombres enters, les clàusules seran llistes de literals i les CNF llistes de clàusules. Per exemple, la formula CNF $(P \vee Q \vee \neg R) \wedge (\neg Q \vee R)$ la representarem amb prolog amb `[[1,2,-3],[-2,3]]`. Fixeu-vos que la clàusula buida és una llista buida. Una interpretació la representarem amb Prolog com a una llista de literals, per exemple, per a la fórmula anterior, la interpretació $P = 1, Q = 0, R = 1$ la representarem com a `[1,-2,3]`.

   Completeu el predicat `sat(F, I, M)` que donada una fórmula en CNF `F`i una interpretació `I`, es satisfaci quan `M` sigui un model de `F` construit extenent `I` (típicament la primera crida serà de la forma `sat(CNF, [ ], M)`). Aquest predicat implementa el procediment DPLL.

```
sat([],I,I):- write("SAT") ,nl,!.

sat(CNF,I,M):-
    % Ha de triar un literal d'una clausula unitaria, si no n'hi 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).
    % Aquesta simplificacio fara el seguent:
    %  - eliminara les clausules que tinguin el literal
    %  - eliminara el literal negat de totes les clausules, es aqui on
    %     pot sortir la clausula buida i per tant hauria de fallar i fer
    %     backtraking

    simplif(Lit,CNF,CNFS),

    % crida recursiva amb la CNF i la interpretacio actualitzada

    sat( ... , ..., M ).
```

2. Per a codificar el sudoku en CNF el que farem serà codificar que *a certa casella hi ha un número en concret* amb variables Booleanes. Per exemple, la variable Booleana $X_{i,j,3}$ ens servirà per codificar que a la casella de la fila i columna j hi ha un 3. D'aquesta manera per a cada casella (i, j) tindrem k variables per a poder representar que hi ha un nombre entre 1 i k: $X_{i,j,1}, X_{i,j,2}, \ldots, X_{i,j,k}$. Fixem-nos però que per tal que la codificació tingui sentit, només una de les variables $X_{i,j,1}, X_{i,j,2}, \ldots, X_{i,j,k}$ ha de ser certa (no podem tenir dos valors alhora a la mateixa casella i no podem deixar una casella sense valor). D'aquestes llistes de variables per a representar que a certa casella hi ha un numero concret de l'1 al k en direm k-dominis.

   Feu el predicat **exactamenUn(Xs,CNF)** de manera que donada una llista de variables Booleanes `Xs`, `CNF` sigui la fórmula CNF que es satisfaci quan (és a dir, que faci que) exactament una de les variables d'`Xs` sigui certa.

   Aquest predicat el podem fer servir per fer que una casella (i, j) tingui efectivament un valor. Suposem que les variables Booleanes `5,6,7` i `8` representen, respectivament $X_{i,j,1}, X_{i,j,2}, X_{i,j,3}$ i $X_{i,j,k}$, és a dir, que la casella (i, j) prengui el valor 1, 2, 3 o 4, és a dir, que `[5,6,7,8]` és un 4-domini. Llavors, la query:

```
?- exactamentUn([5,6,7,8],CNF).
```

   ens hauria de donar:

```
CNF = [[5,6,7,8],[-5,-6],[-5,-7],[-5,-8],[-6,-7],[-6,-8],[-7,-8]]
```

3. Per a poder codificar que en una fila, columna o quadrat d'un Sudoku no hi ha repetits haurem de fer el predicat **allDiff(XXs, CNF)** que donada una llista de k-dominis (és a dir, dellistes de variables representant valors de caselles), `CNF` sigui una fórmula Boolean que faci que cap casella pugui prendre el mateix valor.

   Per exemple:

```
?- allDiff([[1,2,3,4],[5,6,7,8],[9,10,11,12]],CNF).
```

   on les llistes `[1,2,3,4]`, `[5,6,7,8]` i `[9,10,11,12]` són 4-dominis de tres caselles diferents, ens hauria de donar:

```
CNF = [[-1,-5],[-1,-9],[-5,-9],     % com a molt un 1
       [-2,-6],[-2,-10],[-6,-10],   % com a molt un 2
       [-3,-7],[-3,-11], [-7,-11],  % com a molt un 3
       [-4,-8],[-4,-12], [-8,-12]]  % com a molt un 4
```

4. Podem representar un Sudoku de $n \times n$ com a una llista de llistes on a cada casella hi haurà un valor d'1 a n que es codificarà tal com hem descrit a l'apartat 2 (és a dir, amb n-dominis).

   Feu el predicat **taulerSudoku(N,Inputs,Tauler,CNF)** tal que donat un natural `N` i una llista de caselles plenes `Inputs` (que tindran la forma `[c(1,1,3), c(1,3,2)`

`...]` per indicar que a la casella $(1,1)$ hi ha un 3 i a la $(1,3)$ hi ha un 2), ens retorni un `Tauler` que serà una llista de llistes de N-dominis i ens retorni també la `CNF` codificant els valors que ha pres el tauler d'acord amb els inputs.

Per exemple, la query:

```
?- taulerSudoku(4,[c(1,2,1), c(1,4,3),...],Tauler,CNF).
```

(on fixeu-vos que si la $N = 4$ les caselles poden prendre valors de l'1 al 4), ens hauria de donar:

```
Tauler = [[[1,2,3,4],[5,6,7,8],[9,10,11,12],[13,14,15,16]],
          [[17,18,19,20],  ...                            ],
          ...
          ],
CNF = [[5],[15],...]
```

5. Feu el predicat **codificaSudoku(N,Tauler,CNF1,CNF)** que donat un nombre `N`, el `Tauler` d'$N \times N$ N-dominis i la `CNF1` que ens codifica la inicialització del Sudoku (com hem descrit al predicat `taulerSudoku`), ens retorni la codificació que forci que les caselles de les files, columnes i quadrats pertinents no poden tenir repetits.

6. Feu un predicat **resol(N,Inputs)** que donada una `N` i els `Inputs` d'un Sudoku $N \times N$ ens mostri per pantalla la solució (si en te) o ens digui que no en te.

A part dels predicats demanats, caldrà que en feu d'altres d'auxiliars. Es recomana que feu servir com a base del programa el fitxer `sudokusat.pl` que conté comentaris i suggerències mes pas a pas de com completar la pràctica. Si el paràmetre N us és una complicació podeu pensar la pràctica per sudokus 4×4 i sudokus 9×9. El que no està permès és que feu codi de "copiar enganxar", és a dir, cal que tracteu llistes adequadament.

Us recomano també que verifique individualment els predicats que aneu fent per estar segurs que funcionen, el debugging amb Prolog pot ser difícil amb fórmules grans ...

Lliurament

- S'ha de fer en parelles.
- Caldrà lliurar l'informe imprès amb el codi comentat i jocs de proves.
- Caldrà pujar la pràctica al Moodle, tant informe com codi.
- La data de lliurament serà el **13 de Juny** i les presencials aquella mateixa setmana (ja enviaré el Doodle).