NQueensSAT
Pràctica de Programació Lògica

Paradigmes i Llenguatges de Programació

En aquesta pràctica resoldrem el problema de col·locar N reines en un tauler d'escacs d'$N \times N$ caselles de manera que no s'amenacin. 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 del problema de les N reines en fórmules Booleanes i farem servir el nostre predicat de satisfactibilitat Booleana per trobar-ne la solució (quan en tingui).

Com hem dit, el problema de les N reines consisteix en col·locar N reines en un tauler d'escacs d'$N \times N$ de manera que no s'amenacin però nosaltres, addicionalment, deixarem que el tauler tingui ja algunes reines col·locades i que es puguin especificar posicions prohibides.

La pràctica està estructurada en dues tasques principals:

1. **SAT solver.** Aquesta part consisteix en la implementació d'un SAT solver basat en el mètode de DPLL que consisteix en un procés iteratiu de decisió (amb propagació unitaria) i Backtracking si cal.

   Les variables 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 fórmula 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]`. Aquesta interpretació seria un model de la fórmula anterior de fet.

   Completeu el predicat `sat(F, I, M)` que donada una fórmula `F` en CNF i una interpretació `I`, es satisfaci quan `M` sigui un model de `F` construit extenent `I` (assumim que la primera crida serà de la forma `sat(CNF, [ ], M)`). Aquest predicat ha d'implementar 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 la resta de 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 ).
```

Per exemple:

```
| ?- sat([[1,2],[-1,2,3],[-2,3],[4,-1],[-4,-2,-3]],[],A).
SAT
A = [3,-2,4,1] ? ;
SAT
A = [-4,3,2,-1] ? ;
no

| ?- sat([[1,2],[-1,2],[1,-2],[-1,-2]],[],A).
no
```

2. Per a codificar el problema de col·locar N reines en un tauler d'escacs de manera que no s'amenacin en CNF el que farem serà codificar que *a certa casella hi ha una reina* amb variables Booleanes. Per exemple, la variable Booleana $X_{i,j}$ ens servirà per codificar que a la casella de la fila i columna j hi ha una reina. D'aquesta manera per a cada casella (i, j) tindrem una variable per a poder representar que hi ha o que no hi ha una reina. Així si tinguéssim un tauler de 2×2 ens caldrien només les variables `1`, `2`, `3` i `4`, i la variable `1` es referiria a si hi ha una reina a $(1, 1)$, la `2` a $(1, 2)$, la `3` a $(2, 1)$ i la `4` a $(2, 2)$.

   Feu el predicat **comaminimUn(+Xs,F)** de manera que donada una llista de variables Booleanes `Xs`, `F` sigui la fórmula CNF que es satisfaci quan (és a dir, que faci que) com a mínim una de les variables d'`Xs` sigui certa. Aquest predicat és trivial.

   Feu el predicat **comamoltUn(+Xs,F)** de manera que donada una llista de variables Booleanes `Xs`, `F` sigui la fórmula CNF que es satisfaci quan (és a dir, que faci que) com a molt una de les variables d'`Xs` sigui certa. Aquest predicat el podeu fer amb clàusules que evitin, per cada parell de variables, que totes dues siguin certes.

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

   Feu el predicat **fesTauler(+N,+Posinicial,+Posprohibides,Vars,Inicial)** de manera que donat un nombre `N` de files (i columnes) que ha de tenir el tauler, una llista de parells d'enters de'1 a `N`: `Posinicial` (possiblement buida) indicant a quines posicions hi ha una reina, una llista de parells d'enters de 1 a `N`:

`Posprohibides` (possiblement buida) indicant a quines posicions no hi pot haver cap reina, faci que `Vars` sigui una llista de llistes de variables Booleanes tal que la variable a la posició j de la i-èssima llista representi si hi ha o no una reina a aquella posició del tauler, i faci també que `Inicial` sigui la CNF representant l'ocupació del tauler d'acord amb `Posinicial`, `Posprohibides` i `Vars`.

Feu el predicat **noAmenacesFiles(+Vars,F)** de manera que `F` sigui la fórmula CNF que es satisfaci quan al tauler representat per les variables `Vars` no hi hagi dues reines o més a la mateixa fila. Semblantment feu els predicats **noAmenacesColumnes(+Vars,F)** (pista: transposada d'una matriu).

Feu també el predicat **noAmenacesDiagonals(+N,F)**. Pel de les diagonals, penseu bé quantes i quines diagonals hi ha. La recomanació és que feu una llista de diagonals i aquestes diagonals serien una llista de coordenades. Haurieu d'usar l'auxiliar que tradueix coordenades a les corresponents variables. Imposar que no s'amenacin per diagonals llavors és molt fàcil i podreu reusar predicats fets anteriorment.

Feu el predicat **minimNReines(+Vars,F)** de manera que `F` sigui la fórmula CNF que es satisfaci quan al tauler `N x N` representat per les variables `Vars` hi hagi com a mínim `N` reines. Aquest predicat és trivial.

Feu el predicat **mostraTauler(N,M)** que donada una mida de tauler `N` i un model Booleà `M` (on hi haurà `N` literals positius de les variables corresponents a les posicions on s'ha de posar una reina), mostri un tauler com:

```
-----------
|Q| | | | |
-----------
| | | |Q| |
-----------
| |Q| | | |
-----------
| | | | |Q|
-----------
| | |Q| | |
-----------
```

Feu el predicat **resol()** que demani a l'usuari una dimensió de tauler, una llista de coordenades on hi ha reines, una llista de coordenades on no pot haver-hi reines, codifiqui a CNF el problema, pregunti al SAT solver per una solució i si la troba la representi a pantalla adequadament. Si voleu/podeu, feu que mostri diferenciadament les posicions de reines fixades i de les trobades i també les posicions prohibides, us pot anar bé per debugar.

A part dels predicats demanats, possiblement caldrà que en feu altres d'auxiliars. Es recomana que feu servir com a base del programa el fitxer `nqueensat.pl` que conté comentaris i suggerències pas a pas de com completar la pràctica.

Si el paràmetre N us és una complicació podeu començar pensant la pràctica per taulers de 8×8 i després la generalitzeu. El que no està permès és que feu codi de "copiar enganxar", és a dir, cal que tracteu llistes adequadament.

Us recomano que verifiqueu individualment els predicats que aneu fent per estar segurs que funcionen, el debugging amb Prolog pot ser dur 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 **16 de Juny** i les presencials aquella mateixa setmana (ja enviaré el Doodle).