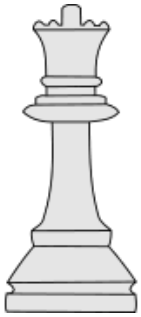
Paradigmes i Llenguatges de Programació
Pràctica de Prolog
[NQueenSAT]

Robert Ripoll
David Suárez

June 2019



Contents

1 Introducció i objectius

En aquesta pràctica resoldrem el problema de col·locar N reines en un tauler d'escacs d'NN caselles de manera que no s'amenacin. El que farem serà implementar un predicat per decidir la satisfactibilitat de fórmules Booleanes en CNF mitjancant 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).

El problema de les N reines consisteix en col·locar N reines en un tauler d'escacs d'NxN de manera que no s'amenacin però nosaltres, addicionalment,deixarem que el tauler tingui ja algunes reines col·locades i que es puguin especificarposicions prohibides [...]

La tasca doncs es divideix en dos blocs:

- **SAT solver**. Aquesta part és en la implementació d'un SAT solver basaten el mètode de DPLL que consisteix en un proc es iteratiu de desició i Backtracking si cal.
- **Codificació N Reines**. Es tracta de expresar en CNF el tauler de forma que no s'amenacin les reines i es compleixin totes les restriccions.

Hem desenvolupat un petit programa en Prolog que ens permet evaluar a partir d'uns valors configurables, com la mida del tauler, retriccions de posicions, així com Reines inicials, les diferents solucions del problema de les N reines.

Tanmateix s'han fet vàries execucions amb mides de tauler creixents, per tal de poder veure l'increment del temps de càlcul de la CPU en poques passes incrementals.

D'altra banda no hem considerat les solucions multiples corresponents a rotacions del tauler com a la mateixa solució. Així doncs, per a N=5 tenim 10 solucions, per a N=6 en tenim 4, etc. A la següent figura podem veure les solucions fins a N=21, amb el temps d'execució sobre una màquina AMD Athlon Dual Core 3030e 2.60GHz.

```
Version 3.2 (2-core)
<------  N-Queens Solutions  -----> <---- time ---->
 N:           Total          Unique days hh:mm:ss.--
 5:              10               2             0.00
 6:               4               1             0.00
 7:              40               6             0.00
 8:              92              12             0.00
 9:             352              46             0.00
10:             724              92             0.00
11:            2680             341             0.00
12:           14200            1787             0.00
13:           73712            9233             0.02
14:          365596           45752             0.05
15:         2279184          285053             0.22
16:        14772512         1846955             1.47
17:        95815104        11977939             9.42
18:       666090624        83263591          1:11.21
19:      4968057848       621012754          8:32.54
20:     39029188884      4878666808       1:10:55.48
21:    314666222712     39333324973       9:24:40.50

AMD Athlon(tm) Dual Core Processor 5050e 2.60 GHz
Microsoft Visual C++ 2008 Express Edition with SP1
Windows SDK for Windows Server 2008 and .NET
```

Font: `http://www.ic-net.or.jp/home/takaken/e/queen/index.html`

2 Exemples d'execució

2.1 Exemple: Tauler de 4x4, sense restriccions.

En aquest exemple, el més simple de tots, mostrem una configuració sense Posicions inicials ni Posicions prohibides. Un cop carregada el nostre univers a Prolog, executem:

```
resol(4,[],[]).
```

```
?- resol(4,[],[]).
SAT!!
---------
| | |Q| |
---------
|Q| | | |
---------
| | | |Q|
---------
| |Q| | |
---------
Reines:
[3,5,12,14]
true ;
SAT!!
---------
| |Q| | |
---------
| | | |Q|
---------
|Q| | | |
---------
| | |Q| |
---------
Reines:
[2,8,9,15]
```

Solucions amb N=4.

2.2 Exemple: Tauler de 7x7, sense restriccions.

En aquest exemple simple, mostrem una configuració sense Posicions inicials ni Posicions prohibides. Un cop carregada el nostre univers a Prolog, executem:

```
resol(7,[],[]).
```

```
?- resol(7,[],[]).
SAT!!
---------------
| | | | | |Q| |
---------------
| | | |Q| | | |
---------------
| |Q| | | | | |
---------------
| | | | | | |Q|
---------------
| | | | |Q| | |
---------------
| | |Q| | | | |
---------------
|Q| | | | | | |
---------------
Reines:
[28,6,16,11,33,38,43]
true ;
SAT!!
---------------
| | | | |Q| | |
---------------
| |Q| | | | | |
---------------
| | | | | |Q| |
---------------
| | |Q| | | | |
---------------
| | | | | | |Q|
---------------
| | | |Q| | | |
---------------
|Q| | | | | | |
---------------
Reines:
[20,5,9,24,35,39,43]
```

Solucions amb N=7.

2.3 Exemple: Tauler de 10x10, sense restriccions.

En aquest exemple simple, mostrem una configuració sense Posicions inicials ni Posicions prohibides. Un cop carregada el nostre univers a Prolog, executem:

```
resol(10,[],[]).
```

```
?- resol(10,[],[]).
SAT!!
---------------------
| | | | | | |Q| | | |
---------------------
| | | |Q| | | | | | |
---------------------
| |Q| | | | | | | | |
---------------------
| | | | | | | | |Q| |
---------------------
| | | | |Q| | | | | |
---------------------
| | | | | | | | | |Q|
---------------------
| | | | | | | |Q| | |
---------------------
| | | | | |Q| | | | |
---------------------
| | |Q| | | | | | | |
---------------------
|Q| | | | | | | | | |
---------------------
Reines:
[39,14,7,22,45,60,68,76,83,91]
true ;
SAT!!
---------------------
| | | | | | | |Q| | |
---------------------
| | | | |Q| | | | | |
---------------------
| |Q| | | | | | | | |
---------------------
| | | |Q| | | | | | |
---------------------
| | | | | | | | | |Q|
---------------------
| | | | | | |Q| | | |
---------------------
| | | | | | | | |Q| |
---------------------
| | | | | |Q| | | | |
---------------------
| | |Q| | | | | | | |
---------------------
|Q| | | | | | | | | |
---------------------
Reines:
[8,15,50,34,22,57,69,76,83,91]
```

Solucions amb N=10.

2.4 Exemple: Tauler de 13x13, sense restriccions.

En aquest exemple simple, mostrem una configuració sense Posicions inicials ni Posicions prohibides. Un cop carregada el nostre univers a Prolog, executem:

```
resol(13,[],[]).
```

```
?- resol(13,[],[]).
SAT!!
-------------------------
| | | | | | |Q| | | | | |
-------------------------
| | | | | | | | | | |Q| | |
-------------------------
| | | | | | | |Q| | | | | |
-------------------------
| | | | | |Q| | | | | | | |
-------------------------
| | | |Q| | | | | | | | | |
-------------------------
| | | | | | | | | | | | |Q|
-------------------------
| | | | | | | | | |Q| | | |
-------------------------
| | | | | | | | | | | |Q| |
-------------------------
| | | | | | | | |Q| | | | |
-------------------------
| |Q| | | | | | | | | | | |
-------------------------
| | | | |Q| | | | | | | | |
-------------------------
| | |Q| | | | | | | | | | |
-------------------------
|Q| | | | | | | | | | | | |
-------------------------
Reines:
[78,34,24,7,56,45,88,103,113,119,135,146,157]
true ;
SAT!!
-------------------------
| | | | | | | | | | | |Q| |
-------------------------
| | | | | | | | | |Q| | | |
-------------------------
| | | | | | | |Q| | | | | |
-------------------------
| | | | | |Q| | | | | | | |
-------------------------
| | | |Q| | | | | | | | | |
-------------------------
| |Q| | | | | | | | | | | |
-------------------------
| | | | | | | | | | | |Q| |
-------------------------
| | | | | | | | | |Q| | | |
-------------------------
| | | | | | | | |Q| | | | |
-------------------------
| | | | |Q| | | | | | | | |
-------------------------
| | |Q| | | | | | | | | | |
-------------------------
|Q| | | | | | | | | | | | |
-------------------------
Reines:
[45,12,34,23,56,67,91,102,113,124,135,146,157]
```

Solucions amb N=13.

2.5 Exemple: Tauler de 7x7, amb restriccions.

En aquest exemple, mostrem una configuració amb Posicions inicials de dues reines, sense Posicions prohibides. Un cop carregada el nostre univers a Prolog, executem:

```
resol(7,[(1,1),(3,2)],[]).
```

```
?- resol(7,[(1,1),(3,2)],[]).
SAT!!
---------------
|Q| | | | | | |
---------------
| | | | |Q| | |
---------------
| |Q| | | | | |
---------------
| | | | | |Q| |
---------------
| | |Q| | | | |
---------------
| | | | | | |Q|
---------------
| | | |Q| | | |
---------------
Reines:
[31,12,27,42,46,16,1]
```

Solucions amb N=7 i 2 reines inicials.

2.6 Exemple: Tauler de 10x10, amb més restriccions.

En aquest exemple, mostrem una configuració amb Posicions inicials de dues reines, sense Posicions prohibides. Un cop carregada el nostre univers a Prolog, executem:

```
resol(10,[(1,1),(3,2)],[]).
```

```
?- resol(10,[(1,1),(3,2)],[]).
SAT!!
---------------------
|Q| | | | | | | | | |
---------------------
| | | | | | |Q| | | |
---------------------
| |Q| | | | | | | | |
---------------------
| | | | | |Q| | | | |
---------------------
| | | | | | | |Q| | |
---------------------
| | | | | | | | | |Q|
---------------------
| | | |Q| | | | | | |
---------------------
| | | | | | | | |Q| |
---------------------
| | |Q| | | | | | | |
---------------------
| | | | |Q| | | | | |
---------------------
Reines:
[36,17,60,64,48,79,83,95,22,1]
```

Solucions amb N=10 i 2 reines inicials.

A continuació volem acurar més les restriccions, i veiem que a la primera solució que troba el SATSolver, apareix una reina a la casella (2,7).

Així doncs, tornem a executar, restringint aquesta posició.

```
resol(10,[(1,1),(3,2)],[(2,7)]).
```

```
?- resol(10,[(1,1),(3,2)],[(2,7)]).
SAT!!
---------------------
|Q| | | | | | | | | |
---------------------
| | | | | | | |Q| | |
---------------------
| |Q| | | | | | | | |
---------------------
| | | | | | | | |Q| |
---------------------
| | | | | |Q| | | | |
---------------------
| | |Q| | | | | | | |
---------------------
| | | | | | | | | |Q|
---------------------
| | | |Q| | | | | | |
---------------------
| | | | | | |Q| | | |
---------------------
| | | | |Q| | | | | |
---------------------
Reines:
[39,18,46,53,70,74,87,95,22,1]
```

Solucions amb N=10, 2 reines inicials, i (2,7) Prohibida.

Com veiem, la solucio que troba a aquesta nova configuració passa compleix amb les restriccions indicades.

3 Codi font i funcions

3.1 SAT Solver

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.

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.

simplif(Lit, F, FS)

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.

3.2 Codificació *N*-Reines

treurePrimer(+L1, L2)

Donada una llista d'enters L1, "retorna" L2 tota la llista L1 a excepció del primer element.

parelles(+L1, +L2, P)

Donada una llista d'enters L1, construeix una llista P amb les parelles resultants de combinar L1 amb L2, on els enters que formen les parelles són negatius.

parelles(+L, P)

Donada una llista d'enters L, construeix una llista de parelles amb tots els elements (en negatiu)de la llista combinats sense repeticions. Per repeticions s'entén: [4,3] == [3,4] (per tant només hi haurà [3,4])

comaminimUn(L,CNF)

Donat una llista de variables booleanes, -> el segon parametre sera la CNF que codifica que com a minim una sigui certa.

comamoltUn(L,CNF)

Donat una llista de variables booleanes, -> el segon parametre sera la CNF que codifica que com a molt una sigui certa.

exactamentUn(L,CNF)

Donat una llista de variables booleanes, -> el segon parametre sera la CNF que codifica que exactament una sigui certa.

fesTauler(+N,+PI,+PP,V,I)

Donat una dimensio de tauler N, unes posicions inicials PI i unes prohibides PP -> V sera el la llista de llistes variables necessaries per codificar el tauler -> I sera la CNF codificant posicions inicials i prohibides.

toCNF(N,Signe,Posicions,CNF)

Donada la mida N del costat del tauler NxN i donat un Signe (valor positiu o valor negatiu) i donada una llista de parells d'enters (posicions (X,Y) del tauler) -> CNF serà la llista d'enters a partir de l'index amb la formula
Signe*((X-1)*N+Y) (posicions dins un vector).

llista(I,F,L)

Donat un inici i un fi -> el tercer parametre sera una llista de numeros d'inici a fi.

extreu(+Inici, +Fi, +Actual, +L1, L2)

Donada una llista d'enters L1, "retorna" a L2 tots els enters d'L1 que es trobin dins l'interval ["Inici","Fi"] començant per "Actual". "Actual" és la variable que permet controlar l'índex de l'element iterat.

trosseja(+L1, +Inici, +Fi, +MidaTros, L2)

Donada una llista d'enters L1, "retorna" a L2 tots els "Fi" trossos de mida "MidaTros" començant per "Inici".

noAmenacesFiles(+V,F)

donada la matriu de variables, -> F sera la CNF que codifiqui que no s'amenecen les reines de les mateixes files

noAmenacesColumnes(+V,C)

donada la matriu de variables, -> C sera la CNF que codifiqui que no s'amenecen les reines de les mateixes columnes

noAmenacesDiagonals(+N,C)

Donada la mida del tauler,-> D sera la CNF que codifiqui que no s'amenecen les reines de les mateixes diagonals

expandeix$_c omamoltUn(V, L)$

Donada una llista V que conté llistes de enters, s'aplica per a cada llista el mètode ComamoltUn, L és la concatenació de totes aquestes noves clausules.

diagonalsIn(D,N,L)

Generem les diagonals dalt-dreta a baix-esquerra, D es el numero de diagonals, N la mida del tauler i L seran les diagonals generades.

diagonals2In(D,N,L)

Generem les diagonals baix-dreta a dalt-esquerra

coordenadesAVars(L,N,Ls)

Passa una llista de coordenades (Tuples d'enters) de tauler NxN a variables corresponents.

llistesDiagonalsAVars(Lparells,N,Lvars)

Passa una llista de diagonals a llistes de llistes de variables.

minimNReines(+V,FN)

donada la matriu de variables (inicialment d'un tauler NxN), -> FN sera la CNF que codifiqui que hi ha d'haver com a minim N reines al tauler.

llegeixNombre(X)

Llegeix un enter per teclat i el "retorna" a X.

filtrarPositius(+L1, L2)

Donada una llista d'enters L1, "retorna" a L2 tots els enters d'L1 positius.

resol()

Ens demana els parametres del tauler i l'estat inicial, codifica les restriccions del problema i en fa una formula que la enviem a resoldre amb el SAT solver i si te solucio en mostrem el tauler.

resol(N, +I, +P)

N és la mida del tauler NxN, I és la llista de posicions inicials, P és la llista de posicions prohibides. Codifica les restriccions del problema i en fa una formula que la enviem a resoldre amb el SAT solver i si te solucio en mostrem el tauler.

4 Resultat i valoracions

El resultat és un petit programa que permet trobar solucions en un temps raonable per al problema de les N-Reines. No obstant no s'ha fet l'estudi ni l'implementació d'optimitzacions per translació de matrius, ni altres mètodes per a fer més eficient el programa.

Com a valoracions finals, hem d'admetre que a l'inici del desenvolupament de la pràctica ens ha sigut difícil "interioritzar" i assolir el funcionament intern i comportament de prolog, amb conceptes com: la unificació, iteració de llistes i creació de llistes recursivament.

En la nostra opinió, prolog té una corba de nivell d'aprenentatge bastant elevada al principi i moderada posteriorment, ja que un cop ja ens trobàvem "enmig" del desenvolupament ja ens ha sigut més fàcil seguir el fil i continuar el desenvolupament més lleugera i ràpidament. Pensem que a classe se'ns han explicat aquests conceptes "sobre paper" i se'ns ha proporcionat predicats ja fets perquè els comprenguéssim, però que no és suficient per entendre a efectes pràctics el funcionament de prolog i com programar en paradigma de programació lògica.

En la nostra opinió, aquesta pràctica ha sigut una bona ocasió per "interioritzar" i posar en pràctica tots els conceptes explicats a classe, a més de tenir la possibilitat de crear codi nou de zero pel nostre propi peu per tal d'acabar assolint tots aquests conceptes. Hem de reconèixer que al principi érem molt escèptics de que es poguessin fer programes complexes (com per exemple el que hem desenvolupat en aquesta pràctica) en prolog, ja que no érem conscients de les possibilitats que ofereix aquest paradigma.

Durant la programació de la pràctica hem sigut capaços d'entendre un concepte clau com és la unificació, que permet: realitzar operacions aritmètiques, iterar i construir llistes, etc., i d'aquesta forma hem arribat a comprendre com programar amb aquest paradigma.

Per altra banda, també hem vist de primera mà el funcionament del predicat de tall de prolog, per tal d'evitar visitar possibles branques alternatives, i així millorar l'eficiència del nostre programa.

References

[1] **Pàgina de consulta**
`http://www.ic-net.or.jp/home/takaken/e/queen/index.html`

[2] **Pàgina de consulta**
`https://www.swi-prolog.org/pldoc/index.html`

[3] **Pàgina de consulta**
`https://sicstus.sics.se/sicstus/docs/latest4/html/sicstus.html/`