AUTOMATED REASONING

Agostino Dovier

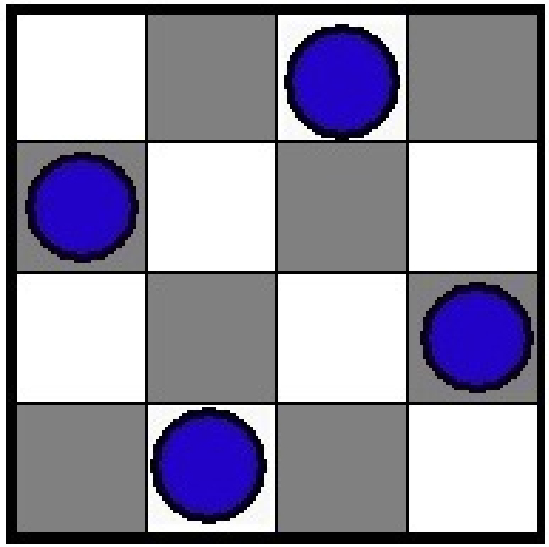
Università di Udine
CLPLAB

Udine, November 2016

Modeling CSP with ASP

N-QUEENS

4-Queens: put 4 queens on a 4×4 chessboard in such a ways as they don't attack each other.



N-Queens

Let us define the predicate `queen`/2, where `queen`(i, j) holds iff in the cell (i, j) there is a queen.

Domain predicate (using a constant n):

```
numero(1..n).
```

For each column j there is exactly one queen:

```
1 {queen(I,J) : numero(I)} 1 :- numero(J).
```

N-QUEENS

HORIZONTAL ATTACK

```
:- numero(I), numero(J1), numero(J2),
   J1 != J2,
   queen(I,J1), queen(I,J2).
```

Read as *It is not possible that there is a row I containing two queens (in column J1 and in column J2)*

In logic:

$$(\exists I)(\exists J_1)(\exists J_2)\,(J_1 \neq J_2 \wedge \texttt{queen}(I, J_1) \wedge \texttt{queen}(I, J_2)) \longrightarrow \texttt{false}$$

Or, equivalently:

$$(\forall I)(\forall J_1)(\forall J_2)\,((J_1 \neq J_2 \wedge \texttt{queen}(I, J_1) \wedge \texttt{queen}(I, J_2)) \longrightarrow \texttt{false})$$

```
:- numero(I), numero(J1), numero(J2),
   J1 != J2,
   queen(I,J1), queen(I,J2).
```

Read as *It is not possible that there is a row I containing two queens (in column J1 and in column J2)*
In logic:

$$(\exists I)(\exists J_1)(\exists J_2)\,(J_1 \neq J_2 \wedge \texttt{queen}(I, J_1) \wedge \texttt{queen}(I, J_2)) \longrightarrow \texttt{false}$$

Or, equivalently:

$$(\forall I)(\forall J_1)(\forall J_2)\,((J_1 \neq J_2 \wedge \texttt{queen}(I, J_1) \wedge \texttt{queen}(I, J_2)) \longrightarrow \texttt{false})$$

N-Queens

Diagonal attack

```
:- numero(I1),numero(I2),numero(J1),numero(J2),
   J1 != J2,
   queen(I1,J1), queen(I2,J2),
   | I1-I2 |  = | J1-J2 |.
```

Read as *It is not possible that there are numbers I1,I2,J1,J2 with J1 $\neq$ J2, such that there the queens in (I1,J1) and (I2,J2) attack each other since | I1-I2 | = | J1-J2 |.*

What happens if we changed I1 $\neq$ I2 with I1 $<$ I2?

N-Queens

Diagonal attack

```
:- numero(I1),numero(I2),numero(J1),numero(J2),
   J1 != J2,
   queen(I1,J1), queen(I2,J2),
   | I1-I2 |  = | J1-J2 |.
```

Read as *It is not possible that there are numbers I1,I2,J1,J2 with J1 $\neq$ J2, such that there the queens in (I1,J1) and (I2,J2) attack each other since | I1-I2 | = | J1-J2 |.*

What happens if we changed I1 $\neq$ I2 with I1 $<$ I2?

Other examples

Coloring
Schur
Hamiltonian

HAMMING CODES

HAMMING CODES

```
word(1..k).
position(1..n).
dmin(d).

%% Values (you can enlarge the alphabet)
val(0).
val(1).
```

HAMMING CODES

We wish to define the predicate
`tupla(indice,posizione,valore)`

```
1{ tupla(I,J,V) : val(V) } 1 :- word(I), position(J).

distanza(I1,I2,S) :-
   word(I1), word(I2), I1 < I2,
   S = #count{ J : tupla(I1,J,V1), tupla(I2,J,V2), V1 != V2,
                   position(J), val(V1), val(V2)}.

badcode :-
   word(I1), word(I2), I1 < I2, dmin(D),
   distanza(I1,I2,S),
   S < D.
:- badcode.
```

HAMMING CODES

We wish to define the predicate
`tupla(indice,posizione,valore)`

```
1{ tupla(I,J,V) : val(V) } 1 :- word(I), position(J).

distanza(I1,I2,S) :-
   word(I1), word(I2), I1 < I2,
   S = #count{ J : tupla(I1,J,V1), tupla(I2,J,V2), V1 != V2,
                                position(J), val(V1), val(V2)}.

badcode :-
   word(I1), word(I2), I1 < I2, dmin(D),
   distanza(I1,I2,S),
   S < D.
:- badcode.
```

Symmetry breaking

```
%%% Zero is in the code
tupla(1,J,0) :- position(J).

%%% Tuples are lexico ordered
ordered(I1,I2) :-
    word(I1), word(I2), I1 < I2,
    ordered(I1,I2,1).
ordered(I1,I2,J) :-
    position(J),
    word(I1), word(I2), I1 < I2,
    val(V1), val(V2),
    tupla(I1,J,V1), tupla(I2,J,V2),
    V1 < V2.
ordered(I1,I2,J) :-
    position(J), position(J+1),
    word(I1), word(I2), I1 < I2,
    val(V),
    tupla(I1,J,V), tupla(I2,J,V),
    ordered(I1,I2,J+1).
:-    word(I1), word(I2), I1 < I2, not ordered(I1,I2).
```

Output

```
% Complete with nonsense values if n < 10
extrapos(n+1..10).
tupla(I,J,x) :- word(I), extrapos(J).

mostra(X1,X2,X3,X4,X5,X6,X7,X8,X9,X10) :-
    word(I),
    tupla(I,1,X1),
    tupla(I,2,X2),
    tupla(I,3,X3),
    tupla(I,4,X4),
    tupla(I,5,X5),
    tupla(I,6,X6),
    tupla(I,7,X7),
    tupla(I,8,X8),
    tupla(I,9,X9),
    tupla(I,10,X10).
#show mostra/10.
```

ESERCISE

- Given a chessboard $n \times n$, and two numbers h and k, put h castles (torri) and k bishops (alfieri) in such a way that they do not attack each other.
- Given a chessboard $m \times n$, find a path for a knight (cavallo) that visits each cell exactly once (starting and arrival places are of course different).
- Consider the Conway's game of life: `https://it.wikipedia.org/wiki/Gioco_della_vita`. Consideri a board of size $n \times n$ and write an ASP program that finds a stable configuration (see the game rules) with exactly h live cells.