ASP y Logica

Esteban Brzovic ebrzovic@uc.cl
Pablo Flores ptflores1@uc.cl

Si no tienen Clingo descargado o instalado, pueden apoyarse en este link

https://potassco.org/clingo/run/

INTRODUCCIÓN

Un programa lógico P está definido por un set de reglas sobre un set A de Átomos (constantes)

Animal (X) ← Gato (X)

X es una variable, si X es un gato, entonces X es una Animal.

“godo” es un gato

Esto es una regla

CLINGO

```
animal(X) :- gato(X).
```

```
gato(godo).
```

“godo” es un átomo

INTRODUCCIÓN

X es una variable, si X es un gato, entonces X es una Animal.

“godo” es el átomo de constante

```
animal(X) :- gato(X).
gato(godo).
```

Corremos el programa

```
Answer: 1
gato(godo) animal(godo)
SATISFIABLE

Models       : 1
```

Ejemplo

Perro y gato son mamífero

```
mamifero(X) :- gato(X).
mamifero(X) :- perro(X).
```

Los mamíferos son animales

```
animal(X) :- mamifero(X).
```

Los mamíferos duermen

```
sleeps(X) :- mamifero(X).
```

Constraint de restricción

```
:- mamifero(X).
```

Átomos “godo” y “leo”

```
gato(godo).
perro(leo).
```

Ejemplo

Resultado: “godo” es un gato, lo que implica que sea mamífero. Como godo debe ser parte del modelo y descartamos los modelos que tengan el átomo mamifero(X), no existe ninguna solución posible, ya que hay contradicción.

```
mamifero(X) :- gato(X).
mamifero(X) :- perro(X).
animal(X) :- mamifero(X).
sleeps(X) :- mamifero(X).
:- mamifero(X).
gato(godo).
perro(leo).
```

```
Answer: 1
UNSATISFIABLE

Models       : 0
```

Regla de selección

Cada modelo combinan los átomos que se encuentren dentro de {...} de una manera distinta

```
{p(a);q(b)}.
```

```
1{p(1..3)}2.
```

Regla de selección

Cada modelo combinan los átomos que se encuentren dentro de {...} de una manera distinta

```
{p(a);q(b)}.
```

```
1{p(1..3)}2.
```

```
Answer: 1

Answer: 2
q(b)
Answer: 3
p(a)
Answer: 4
p(a) q(b)
SATISFIABLE
Models        : 4
```

Regla de selección

Cada modelo combinan los átomos que se encuentren dentro de {...} de una manera distinta

```
{p(a);q(b)}1        =     {p(a);q(b)}.
                          :- p(a), q(b).
```

```
{p(a);q(b)}.

Answer: 1

Answer: 2
q(b)
Answer: 3
p(a)
Answer: 4
p(a) q(b)
SATISFIABLE
Models       : 4
```

```
1{p(1..3)}2.

Answer: 1
p(2)
Answer: 2
p(3)
Answer: 3
p(2) p(3)
Answer: 4
p(1)
Answer: 5
p(1) p(3)
Answer: 6
p(1) p(2)
SATISFIABLE
Models       : 6
```

Resumen Reglas Simples

q(X) implica p(X)

```
p(X) :- q(X).
```

mini ejemplo

```
par(N) :- N/2 + N/2 == N.
impar(N) :- not par(N).
```

(p(X)y q(X)) no aplican al modelo

```
:- p(X), q(X).
```

r(X) implica (p(X) o q(X))

```
p(X); q(X) :- r(X).
```

(q(X) si r(X))implica p(X)"

```
p(X) :- q(X) : r(X).
```

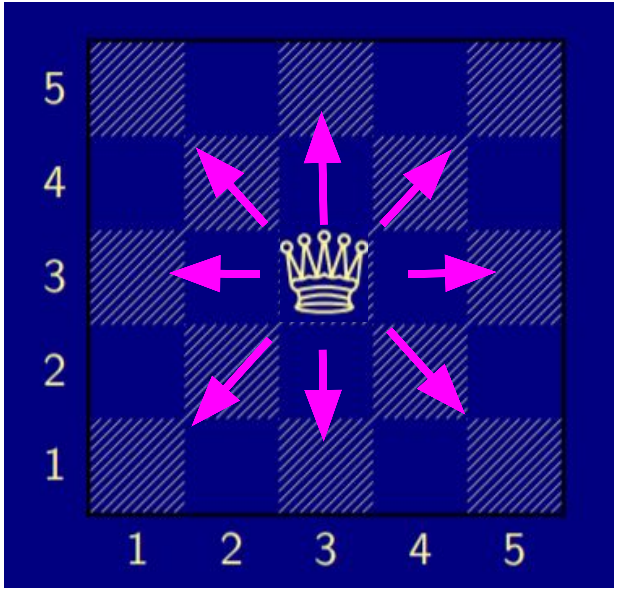
cuando ocurre r(X) entonces escoge entre 2 y 5 p(X)

```
2 { p(X) : q(X) } 5 :- r(X).
```

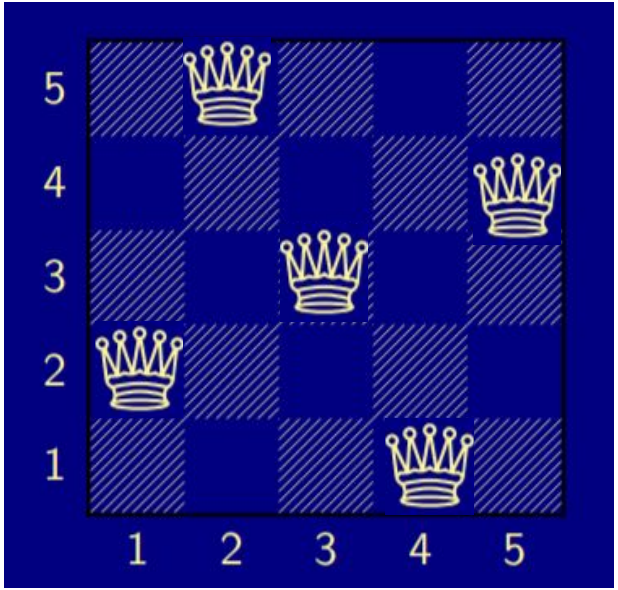
Modelación

Problema de la Dama

El problema consiste en colocar cada “n” damas en un tablero de nxn sin que estos se ataquen entre sí



Una Solución óptima posible



Definimos el tablero

Creamos la condición para que queen(I , J) pueda existir y los coloca queens en el tablero.

Creamos una restricción para que haya exactamente n queens

Podemos simplificar lo anterior

```
row (1.. n ).
col (1.. n ).
```

Definimos el tablero

```
row (1.. n ).
col (1.. n ).
```

Creamos la condición para que queen(I , J) pueda existir y los coloca queens en el tablero.

```
{queen(I,J) : row(I) , col(J)}.
```

Creamos una restricción para que haya exactamente n queens

Podemos simplificar lo anterior

Definimos el tablero

```
row (1.. n ).
col (1.. n ).
```

Creamos la condición para que queen(I , J) pueda existir y los coloca queens en el tablero.

```
{queen(I,J) : row(I) , col(J)}.
```

Creamos una restricción para que haya exactamente n queens

```
:- {queen(I,J)} != n .
```

Podemos simplificar lo anterior

Definimos el tablero

```
row (1.. n ).
col (1.. n ).
```

Creamos la condición para que queen(I , J) pueda existir y los coloca queens en el tablero.

```
{queen(I,J) : row(I) , col(J)}.
```

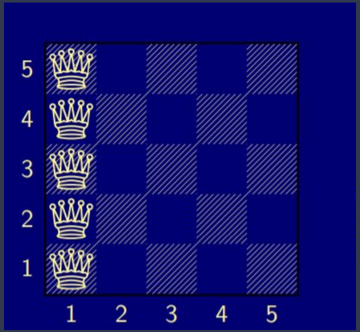
Creamos una restricción para que haya exactamente n queens

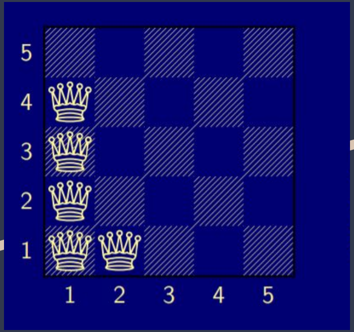
```
:- {queen(I,J)} != n .
```

Podemos simplificar lo anterior

```
{queen(I,J) : row(I), col(J)} = n .
```

Corremos el programa






```
Running . . .
$ clingo queens . lp -- const n =5 2

Answer : 1
row (1) row (2) row (3) row (4) row (5) \
 col (1) col (2) col (3) col (4) col (5) \
queen(5,1) queen(4,1) queen(3,1)
queen(2,1) queen(1,1)

Answer : 2
row (1) row (2) row (3) row (4) row (5) \
col (1) col (2) col (3) col (4) col (5) \
queen(1,2) queen(4,1) queen(3,1)
queen(2,1) queen(1,1)
```

Definimos el tablero

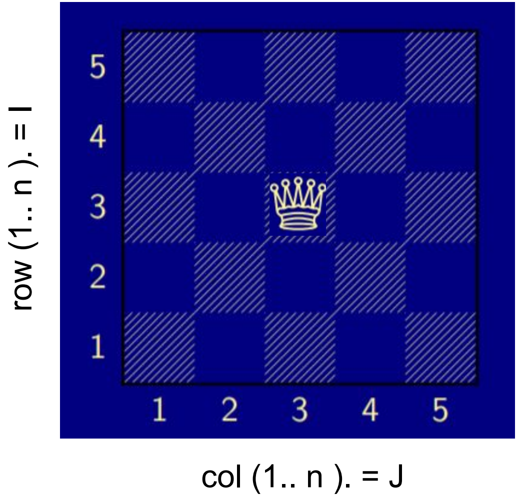
```
row (1.. n ).
col (1.. n ).
```

Posiciona exactamente n queens

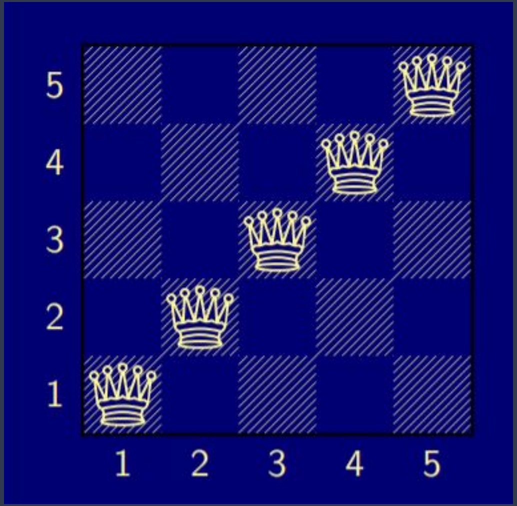
```
{queen (I,J): row(I) , col(J)} = n .
```

Con esto limitamos los movimientos horizontales y verticales de la queen.

```
:- queen(I,J) , queen(I ,J’), J != J’.
:- queen(I,J) , queen(I’,J), I != I’.
```



Corremos el programa



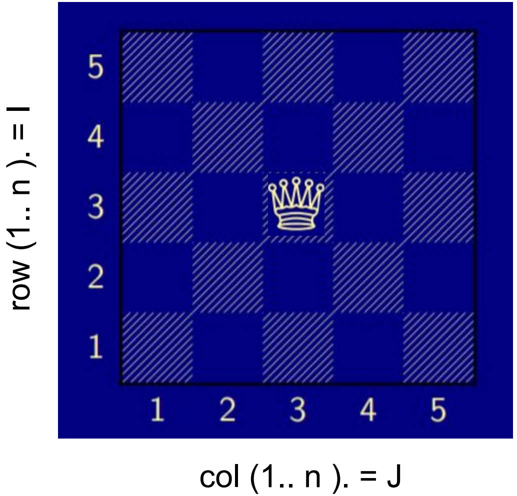

```
$ clingo queens . lp -- const n =5

Answer : 1
row(1) row(2) row(3) row(4) row(5) \
col(1) col(2) col(3) col(4) col(5) \
queen(5,5) queen(4,4) queen(3,3)
queen(2,2) queen(1,1)
```

Reglas anteriores

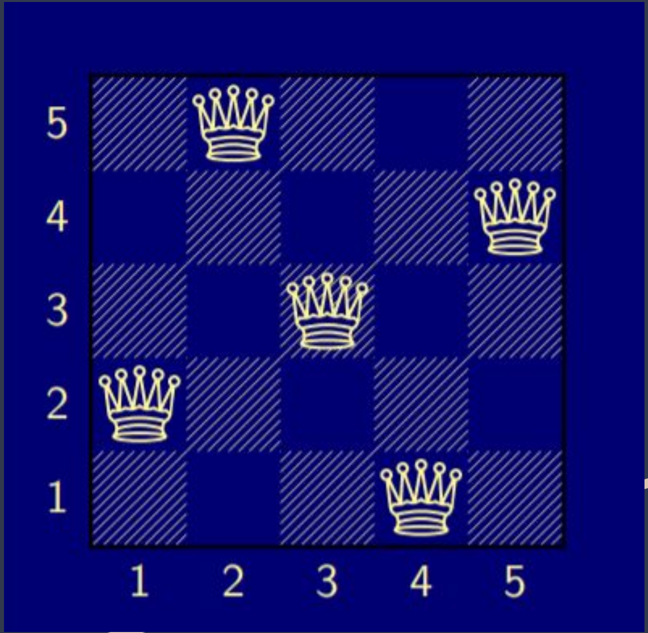
```
row (1.. n ).
col (1.. n ).
{queen (I,J): row(I) , col(J)} = n .
:- queen(I,J) , queen(I ,J’), J != J’.
:- queen(I,J) , queen(I’,J), I != I’.
```

Con esto limitamos los movimientos diagonales de la queen.

```
:- queen(I,J),queen(I’,J’),(I,J)!=(I’,J’)
,I-J==I’-J’.
:- queen(I,J),queen(I’,J’),(I,J)!=(I’,J’)
,I+J==I’+J’.
```



Corremos el programa




```
Answer :1

row (1) row (2) row (3) row (4) row (5) \
col (1) col (2) col (3) col (4) col (5) \
queen(4,5) queen(1,4) queen(3,3)
queen(5,2) queen(2,1)
```

Ejemplos Básicos de Clingo

Si no tienen instalado Clingo en su computador pueden usar https://potassco.org/clingo/run/

Para ejecutar un programa en Clingo:

```
>>>> clingo nombre_del_programa.lp
```

Para especificar la cantidad de modelos usamos `-n`:

```
>>>> clingo programa.lp -n 2
```

y para obtener todos los modelos posibles:

```
>>>> clingo programa.lp -n 0
```

Para especificar el valor de una constante:

```
>>>> clingo programa.lp -c const=9
```

Problema de los Regadores