```
------------------------------------------------
Teoría de Modelos. Clase del 13 de mayo de 2020
------------------------------------------------
------------------------------------------------
Tema 4: El paradigma de la programación con
        conjuntos de respuestas.
------------------------------------------------
CLINGO: Sintaxis y ejemplos
------------------------------------------------

------------------------------------------------
1. Intervalos (n..m)
------------------------------------------------
```

-) La notación i..j representa los números enteros consecutivos desde i hasta j.

Consideremos un predicado p/1. Si queremos añadir a la base de conocimiento que los números del 1 al 5 satisfacen p, hasta ahora tendríamos que escribir los 5 hechos correspondientes.

```
p(1).
p(2).
p(3).
p(4).
p(5).
```

El uso de intervalos en CLINGO permite abreviar esos cinco hechos como sigue:

```
p(1..5).
```

Podemos usar también números negativos. Por ejemplo:

```
q(-4..4).
```

Añade a la base de conocimiento los 9 hechos que siguen:

```
% Solving...
% Answer: 1
% p(-4) p(-3) p(-2) p(-1) p(0) p(1) p(2) p(3) p(4)
% SATISFIABLE
```

Nota: Si los extremos del intervalo no están en orden, el programa no da ningún mensaje de error. Simplemente, no se añade nada a la base de conocimiento.

Es interesante que también podemos usar intervalos en predicados de dos o más argumentos. En estos casos, se genera el producto cartesiano correspondiente. Veamos algunos ejemplos.

```
p(1..3,0..2).
```

genera los hechos:

```
% Solving...
% Answer: 1
% p(1,0) p(1,1) p(1,2) p(2,0) p(2,1) p(2,2) p(3,0) p(3,1) p(3,2)
% SATISFIABLE
```

```
p(0,1..5).
```

genera los hechos:

```
% Solving...
% Answer: 1
% p(0,1) p(0,2) p(0,3) p(0,4) p(0,5)
% SATISFIABLE
```

-) El intervalo en la cabeza de una regla se expande de forma *conjuntiva*,
   mientras que en el cuerpo de una regla se expande de forma *disyuntiva*.

Podemos usar también intervalos en reglas, su comportamiento es distinto según aparezca en la cabeza o en el cuerpo de la regla. Por ejemplo:

```
p(1..3) :- q(0),r(2..3).
```

equivale al conjunto de reglas:

```
p(1) :- q(0),r(2).
p(2) :- q(0),r(2).
p(3) :- q(0),r(2).
p(1) :- q(0),r(3).
p(2) :- q(0),r(3).
p(3) :- q(0),r(3).
```

O, por ejemplo, la regla:

```
p(X) :- q(X), X=1..3.
```

equivale al conjunto de reglas:

```
p(1) :- q(1).
p(2) :- q(2).
p(3) :- q(3).
```

Un uso típico es el siguiente:

```
size(3).
grid(1..S,1..S) :- size(S).
```

que genera las casillas de un tablero de tamaño 3x3

```
% Solving...
% Answer: 1
% size(3) grid(1,1) grid(2,1) grid(3,1)
%         grid(1,2) grid(2,2) grid(3,2)
%         grid(1,3) grid(2,3) grid(3,3)
% SATISFIABLE
```

Una manera alternativa de generar el tablero es:

```
size(3).
grid(X,Y) :- size(S),X=1..S,Y=1..S.
```

---

2. Agrupaciones (a;b;d)

-) Una agrupación (o1;o2;o3;...;ok) se usa para representar k argumentos (no necesariamente numéricos y no necesariamente consecutivos) de forma compacta. Por ejemplo:

```
p(a;b;d).
q(1;7).
```

generan el conjunto de hechos

```
% Solving...
% Answer: 1
% p(a) p(b) p(d) q(1) q(7)
% SATISFIABLE
```

También pueden usarse con predicados de dos o más argumentos y combinados con intervalos. Por ejemplo:

```
p(1..5,(juan;ana)).
q((a;b),(3;0)).
```

genera el conjunto de hechos

```
% Solving...
% Answer: 1
% p(1,ana) p(2,ana) p(3,ana) p(4,ana) p(5,ana)
% p(1,juan) p(2,juan) p(3,juan) p(4,juan) p(5,juan)
% q(a,3) q(b,3) q(a,0) q(b,0)
% SATISFIABLE
```

Cuidado: Los paréntesis encapsulando (juan;ana) (a;b) y (3;0) son necesarios
en el ejemplo anterior. CLINGO da mayor prioridad al operador coma (,) que al operador (;) a la hora de agrupar si no escribimos los paréntesis.

-) Las agrupaciones en las reglas se expanden de forma análoga a los intervalos, es decir,
   de forma conjuntiva si aparacen en la cabeza de una regla;
   y de forma disyuntiva si aparecen en el cuerpo.

Por ejemplo, la regla:

```
q(X) :- X = (a;7;pepe).
```

genera los hechos

```
% Solving...
% Answer: 1
% q(pepe) q(7) q(a)
% SATISFIABLE
```

Mientras que la regla:

```
q(X,(a;pepe)) :- p(X).
```

daría lugar a las reglas

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

Otros ejemplos típicos de uso:

```
%% 1.1. Descripción de bases de datos familiares:

padre(juan, (ana;julia;pedro)).
```

expresa que Juan es el padre de Ana, Julia y Pedro.

%% 1.2. Descripción de dominios finitos.

```
num(1..10).
```

expresa que consideramos un dominio finito de 10 elementos.

%% 1.3. Descripción de grafos:

```
nodo(1..6).
arco(1,(2;6)).
arco(2,(3;4)).
arco(4,(5;6)).
arco(5,6).
arco(Y,X) :- arco(X,Y).
```

representa un grafo *no dirigido* con seis nodos (1,2,3,4,5,6) y aristas: 1-2, 1-6, 2-3, 2-4, 4-5, 4-6, 5-6.

---
3. Aritmética en CLINGO

CLINGO permite operaciones aritméticas básicas con números enteros.

-) Operadores aritméticos: +,-,*,/,\,**, | |

+ (suma) - (resta) * (producto)
/ (división entera)
\ (resto de la división)
** (exponenciación)
| | (valor absoluto)

-) Operadores de comparación: =,!=,<,>,<=,>=

= (igual)
!= (distinto)
< (menor)
> (mayor)
<= (menor o igual)
>= (mayor o igual)

Por ejemplo, el programa

```
l(2).
r(7).
p(Z) :- l(X),r(Y),Z=X+Y-1.
```

genera el conjunto de respuestas

```
% Solving...
% Answer: 1
% r(7) l(2) p(8)
% SATISFIABLE
```

Nota: La regla anterior también puede escribirse como:

```
p(X+Y-1) :- l(X),r(Y).
```

3.1 Ejemplos de operadores aritméticos:

```
l(2).
r(7).

suma(X+Y) :- l(X),r(Y).
resta(X-Y) :- l(X),r(Y).
prod(X*Y) :- l(X),r(Y).
div(Y/X) :- l(X),r(Y).
mod(Y\X) :- l(X),r(Y).
potencia(X**Y) :- l(X),r(Y).
valor_abs(|X-Y|) :- l(X),r(Y).
compuesto(Z) :- Z=(X+1)**2+((X+3*Y)\7),
                l(X),r(Y).

% Solving...
% Answer: 1
% r(7) l(2) suma(9) resta(-5) prod(14) div(3) mod(1)
% potencia(128) valor_abs(5) compuesto(11)
% SATISFIABLE
```

3.2 Ejemplos de operadores de comparación

```
l(2).
r(7).

eq(X,Y) :- l(X),r(Y),X=Y.
dist(X,Y) :- l(X),r(Y),X!=Y.
menor(X,Y) :- l(X),r(Y),X<Y.
mayor(X,Y) :- l(X),r(Y),X>Y.
menor_igual(X,Y) :- l(X),r(Y),X<=Y.
```

```
mayor_igual(X,Y) :- l(X),r(Y),X>=Y.


% Solving...
% Answer: 1
% r(7) l(2) dist(2,7) menor(2,7) menor_igual(2,7)
% SATISFIABLE
```

3.3 Ejemplos de uso:

```
Descripción de un grafo completo.
---------------------------------

size(4).
nodo(1..S) :- size(S).
arco(X,Y) :- X=1..S-1,Y=X+1..S,size(S).
arco(Y,X) :- arco(X,Y).


% Solving...
% Answer: 1
% size(4) nodo(1) nodo(2) nodo(3) nodo(4)
% arco(1,2) arco(1,3) arco(1,4) arco(2,3)
% arco(2,4) arco(3,4) arco(4,3) arco(4,2)
% arco(3,2) arco(4,1) arco(3,1) arco(2,1)
SATISFIABLE

Nota: CLINGO admite el uso de *constantes*.
Veamos cómo reescribir el ejemplo anterior usando
una constante para guardar el orden del grafo en
lugar del predicado size/1.


#const n=4.
nodo(1..n).
arco(X,Y) :- X=1..n-1,Y=X+1..n.
arco(Y,X) :- arco(X,Y).

Una ventaja del uso de constantes es que también pueden
emplearse como parámetros de ejecución. Aunque siempre pongamos
un valor por defecto en el programa mediante la directiva
  #const n=valor. ,
podemos ejecutar el programa para otros valores de la constante
sin más que añadir a la orden "clingo fichero.lp" la opción
  -c n=nuevoValor.

Por ejemplo, para generar un grafo completo de orden 3 sin modificar
el código del programa anterior, bastará con llamar a CLINGO como sigue:
```

```
> clingo ejemploGrafo.lp -c n=3

% Solving...
% Answer: 1
% nodo(1) nodo(2) nodo(3)
% arco(1,2) arco(1,3) arco(2,3)
% arco(3,2) arco(3,1) arco(2,1)
% SATISFIABLE
```

Cálculo de números primos

Escribamos un programa ASP que genere los números primos del intervalo [1..N].

```
% Fijamos el dominio finito [1..N]
% Un número es primo si es mayor que 1 y
% no es compuesto
% Un número es compuesto si tiene algún
% divisor propio (distinto de 1 y de él mismo).

#const n=20.
num(1..n).
primo(X) :- num(X), X > 1,not compuesto(X).
compuesto(X) :- 0 = X \ Y, num(X),Y=2..X-1.
#show primo/1.

> clingo ejemploPrimos.lp

% Solving...
% Answer: 1
% primo(2) primo(3) primo(5) primo(7) primo(11) primo(13)
% primo(17) primo(19)
% SATISFIABLE

>clingo ejemploPrimos.lp -c n=100

% Solving...
% Answer: 1
% primo(2) primo(3) primo(5) primo(7) primo(11) primo(13)
% primo(17) primo(19) primo(23) primo(29) primo(31)
% primo(37) primo(41) primo(43) primo(47) primo(53)
% primo(59) primo(61) primo(67) primo(71) primo(73)
% primo(79) primo(83) primo(89) primo(97)
% SATISFIABLE
```

---

4. REGLAS DE ELECCIÓN

Veremos ahora las llamadas *reglas de elección* de CLINGO,
un constructor muy potente para modelizar problemas combinatorios.

Introducimos la sintaxis a base de ejemplos.

4.1. Generar todos los subconjuntos de un conjunto finito de átomos dado.

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

```
> clingo ejemplo.lp 0
Solving...
Answer: 1

Answer: 2
p(b)
Answer: 3
p(c)
Answer: 4
p(b) p(c)
Answer: 5
p(a)
Answer: 6
p(a) p(c)
Answer: 7
p(a) p(b)
Answer: 8
p(a) p(b) p(c)
SATISFIABLE
```

Observad la sintaxis: escribimos entre {} una lista finita de 3 átomos separados por ; y CLINGO genera 2^3 conjuntos de respuestas distintos, que corresponden a los subconjuntos del conjunto dado, desde considerar que ninguno de los átomos es el caso (answer 1) hasta considerar que todos los átomos son ciertos (answer 8).

4.2. Generar todos los subconjuntos de tamaño al menos 2.

```
2{p(a);p(b);p(c)}.
```

```
> clingo ejemplo.lp 0
% Solving...
% Answer: 1
```

```
% p(b) p(a)
% Answer: 2
% p(c) p(b) p(a)
% Answer: 3
% p(c) p(a)
% Answer: 4
% p(c) p(b)
% SATISFIABLE
```

4.3. Generar todos los subconjuntos de tamaño a lo más 2.

```
{p(a);p(b);p(c)}2.

> clingo ejemplo.lp 0
% Solving...
% Answer: 1
%
% Answer: 2
% p(b)
% Answer: 3
% p(a)
% Answer: 4
% p(b) p(a)
% Answer: 5
% p(c)
% Answer: 6
% p(c) p(a)
% Answer: 7
% p(c) p(b)
% SATISFIABLE
```

4.4. Generar todos los subconjuntos de tamaño exactamente 2.

```
2{p(a);p(b);p(c)}2.

> clingo ejemplo.lp 0
% Solving...
% Answer: 1
% p(b) p(a)
% Answer: 2
% p(c) p(b)
% Answer: 3
% p(c) p(a)
% SATISFIABLE
```

4.5. Generar todos los subconjuntos no vacíos de un conjunto finito definido por un predicado q/1.

```
q(1..3).
1{p(X) : q(X)}. %R1
```

Idea: La regla R1 dice: si {X1,...Xk} son los elementos que satisfacen el predicado q/1, selecciona todos los subconjuntos no vacíos de {p(X1);...;p(Xk)}.

```
> clingo ejemplo.lp 0
% Solving...
% 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)
% Answer: 7
% p(1) p(2) p(3)
% SATISFIABLE
```

4.6. Asignar a cada número [1..N] un único color de una paleta de colores.

```
#const n=4.
color(rojo;verde).
1{color(X,C) : color(C)}1 :- X=1..n.
#show color/2.
```

```
> clingo ejemplo,lp 0

% Solving...
% Answer: 1
% color(1,verde) color(2,rojo) color(3,rojo) color(4,rojo)
% Answer: 2
% color(1,verde) color(2,verde) color(3,rojo) color(4,rojo)
% Answer: 3
% color(1,verde) color(2,rojo) color(3,verde) color(4,rojo)
% Answer: 4
% color(1,verde) color(2,verde) color(3,verde) color(4,rojo)
% Answer: 5
%color(1,verde) color(2,rojo) color(3,verde) color(4,verde)
% Answer: 6
% color(1,verde) color(2,rojo) color(3,rojo) color(4,verde)
% Answer: 7
```

```
% color(1,verde) color(2,verde) color(3,verde) color(4,verde)
% Answer: 8
% color(1,verde) color(2,verde) color(3,rojo) color(4,verde)
% Answer: 9
% color(1,rojo) color(2,rojo) color(3,verde) color(4,verde)
% Answer: 10
% color(1,rojo) color(2,verde) color(3,verde) color(4,verde)
% Answer: 11
% color(1,rojo) color(2,rojo) color(3,verde) color(4,rojo)
% Answer: 12
% color(1,rojo) color(2,verde) color(3,verde) color(4,rojo)
% Answer: 13
% color(1,rojo) color(2,rojo) color(3,rojo) color(4,verde)
% Answer: 14
% color(1,rojo) color(2,rojo) color(3,rojo) color(4,rojo)
% Answer: 15
% color(1,rojo) color(2,verde) color(3,rojo) color(4,verde)
% Answer: 16
% color(1,rojo) color(2,verde) color(3,rojo) color(4,rojo)
% SATISFIABLE
```

Nota: Si no queréis que CLINGO muestre todos los modelos, sino que solo nos diga cuántos hay, podéis añadir la opción --quiet a la orden de ejecución. Por ejemplo:

```
>clingo ejemplo.lp 0 --quiet

% Solving...
% SATISFIABLE

% Models       : 16
% Calls        : 1
% Time         : 0.003s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s)
% CPU Time     : 0.000s
```

Nota: Si queréis que CLINGO solo muestre los 5 primeros modelos que encuentre, por ejemplo, debéis escribir:

```
> clingo ejemplo.lp 5

% Solving...
% Answer: 1
% color(1,verde) color(2,rojo) color(3,rojo) color(4,rojo)
% Answer: 2
% color(1,verde) color(2,verde) color(3,rojo) color(4,rojo)
% Answer: 3
% color(1,verde) color(2,rojo) color(3,verde) color(4,rojo)
```

```
% Answer: 4
% color(1,verde) color(2,verde) color(3,verde) color(4,rojo)
% Answer: 5
% color(1,verde) color(2,rojo) color(3,verde) color(4,verde)
% SATISFIABLE

% Models       : 5+
% Calls        : 1
% Time         : 0.005s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s)
% CPU Time     : 0.000s
```

Nota: Podéis combinar varias de las opciones que hemos visto. Por ejemplo:

¿De cuántas maneras podemos pintar 7 elementos con 2 colores?

```
> clingo ejemplo.lp 0 --quiet -c n=7

% Solving...
% SATISFIABLE

% Models       : 128
% Calls        : 1
% Time         : 0.004s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s)
% CPU Time     : 0.016s
```

4.7 Ejemplo de uso:

Colorear con K colores un grafo ciclo de orden N
sin adyacentes del mismo color

```
#const n=6. %Orden del grafo
#const k=2. %Núm. de colores

% Generamos el grafo ciclo

nodo(1..n).
arco(X,X+1) :- X=1..n-1.
arco(n,1).
arco(Y,X) :- arco(X,Y).

% Asignamos colores con una regla de elección

1{color(N,C) : C=1..k} :- N=1..n.

% Añadimos la restricción: dos nodos adyacentes
```

```
% no pueden tener el mismo color

:- arco(X,Y),color(X,C),color(Y,C).

#show color/2.

> clingo ejemplo.lp

% Solving...
% Answer: 1
% color(2,1) color(6,1) color(1,2)
% color(3,2) color(4,1) color(5,2)
% SATISFIABLE

> clingo ejemplo.lp -c n=5

% Solving...
% UNSATISFIABLE

En efecto, es imposible colorear un ciclo de orden impar>=3
con solo dos colores. Sin embargo, sí es posible con tres
colores. Veámoslo.

> clingo ejemplo.lp -c n=5 -c k=3

% Solving...
% Answer: 1
% color(5,2) color(1,3) color(2,1) color(3,2) color(4,1)
% SATISFIABLE

-----------------------------------------------------
FIN DE LA CLASE
-----------------------------------------------------
```