



Ayudantía 2

Negación y Modelación en Clingo

Por Bernardita Rosas y
Jerónimo Infante

23 de agosto de 2024



Contenidos

1. Repaso
2. Restricciones de cardinalidad
3. Modelación
4. Negación



Repaso



Predicados

- Constantes que representan una propiedad, relación, o característica con sus términos.
- **Siempre** comienzan con **minúscula**.

```
existe(sol).      % Tiene una constante simbólica
existe(1).        % Tiene una constante numérica
existe(X).        % Tiene una variable
```

Ojo: las variables solo existen dentro de los predicados, y siempre comienzan con **mayúscula**



Reglas

Head \leftarrow *Body*

- Si *Body* es verdadero, algo en *Head* también debe serlo.
- Tanto *Head* como *Body* son conjuntos de átomos o proposiciones.
- Se pueden construir hechos a partir de reglas que carezcan de *Body*.

```
llueve.  
mojado(niño) :- llueve.  
enojado(niño) :- mojado(niño). % lamentable :(
```

El modelo es **{llueve, mojado(niño), enojado(niño)}**



Reglas

Body con varios átomos

- Generan una **conjunción** de proposiciones; es decir, se debe cumplir todo en *Body* para que la regla se exija

```
a.          % a se encuentra en el modelo
b.          % b se encuentra en el modelo
c :- a, b.   % c está sólo si a y b lo están
d :- a, m.   % d está sólo si a y m lo están
```

El modelo es **{a, b, c}**



Reglas

Head con varios átomos

- Generan una **disyunción** de proposiciones; es decir, cuando se cumple el *Body*, se cumple sólo uno de los átomos del *Head*
- A excepción de que se fuerce la presencia de más átomos

```
p.  
q, r, k :- p.
```

Los modelos son **$\{p,q\}$, $\{p,r\}$, $\{p,k\}$** .



Reglas

Predicados con variables

- Permiten definir múltiples proposiciones de manera simultánea.

```
pajaro(carpintero).  
pajaro(martin_pescador).  
pajaro(condor).  
vuela(carpintero).  
vuela(martin_pescador).  
vuela(condor).
```

Esto...

```
pajaro(carpintero).  
pajaro(martin_pescador).  
pajaro(condor).  
vuela(Z) :- pajaro(Z).
```

...es equivalente a esto



Reglas

Predicados con variables

```
pajaro(carpintero).  
pajaro(martin_pescador).  
pajaro(condor).  
vuela(Z) :- pajaro(Z).
```

Las variables nos permiten **generalizar** reglas:
“Todo pájaro vuela”

*Las variables siempre se escriben en mayúscula y términos en minúscula



Restricciones de Cardinalidad



Restricciones de Cardinalidad

- En el contexto de la Head de una regla, estas permiten elegir **distintas combinaciones** de átomos o predicados para que aparezcan en los modelos.
- Por ejemplo, para el programa:

```
p.  
{q; r} :- p.      % Si p está en el modelo,  
                  % alguna combinación entre q y r también lo está
```

¿Qué combinaciones de átomos pueden generarse desde la restricción?



Restricciones de Cardinalidad

- En el contexto de la Head de una regla, estas permiten elegir **distintas combinaciones** de átomos o predicados para que aparezcan en los modelos.
- Por ejemplo, para el programa:

```
p.  
{q; r} :- p.      % Si p está en el modelo,  
                  % alguna combinación entre q y r también lo está
```

Las combinaciones pueden ser **{p}**, **{p,q}**, **{p,r}** y **{p,q,r}**.



Restricciones de Cardinalidad

Limitando combinaciones

- Por defecto, Clingo prueba con todas las combinaciones posibles.
- Puede limitarse el número de elementos a incluir rodeando con números el conjunto de la restricción:

```
p.  
1{q; r; s}2 :- p.      % Si p está en el modelo, alguna combinación  
                        % de 1 a 2 elementos entre q, r y s  
                        % también lo está
```

¿Cuántos modelos genera este programa?



Restricciones de Cardinalidad

Limitando combinaciones

- Por defecto, Clingo prueba con todas las combinaciones posibles.
- Puede limitarse el número de elementos a incluir rodeando con números el conjunto de la restricción:

```
p.  
1{q; r; s}2 :- p.      % Si p está en el modelo, alguna combinación  
                        % de 1 a 2 elementos entre q, r y s  
                        % también lo está
```

Ahora, las combinaciones pueden ser **{p;q}**, **{p;r}**, **{p;s}**, **{p,q;r}**, **{p;r;s}** y **{p;q;s}** (6 modelos).



Restricciones de Cardinalidad

Ejercicio

- Supongamos que tenemos un programa con N líneas del tipo:

```
p.  
1 {a_1, b_1} 2 :- p.  
1 {a_2, b_2} 2 :- p.  
(...)  
1 {a_n, b_n} 2 :- p.
```

¿Cuántos modelos genera este programa?



Restricciones de Cardinalidad

Ejercicio

- Supongamos que tenemos un programa con N líneas del tipo:

```
p.  
1 {a_1, b_1} 2 :- p.  
1 {a_2, b_2} 2 :- p.  
(...)  
1 {a_n, b_n} 2 :- p.
```

$\left\{ \begin{array}{l} \{p, a_i\} \\ \{p, b_i\} \\ \{p, a_i, b_i\} \end{array} \right\} \times N \text{ veces } *$

El programa genera **3^N modelos distintos**

* Para cada línea, tres combinaciones posibles



Restricciones de Cardinalidad

Condiciones dentro de las restricciones

- Al poner el carácter ":" es posible crear condiciones dentro de las restricciones de cardinalidad para generar reglas más complejas.

```
num(0..5).  
3{seleccionado(X) : num(X)}3.      % selecciona 3 tal que sean num  
#show seleccionado/1.              % muestra los seleccionados
```

Hay 20 modelos posibles.



Restricciones de Cardinalidad

Condiciones dentro de las restricciones

```
num(0..5).  
3{seleccionado(X) : num(X)}3 :- seleccionado.  
#show seleccionado/1.
```

Modelo vacío.



Restricciones de Cardinalidad

Condiciones dentro de las restricciones

```
seleccionado.  
num(0..5).  
3{seleccionado(X) : num(X)}3 :- seleccionado.  
#show seleccionado/1.
```

Hay 20 modelos posibles.



Restricciones de Cardinalidad

Condiciones dentro de las restricciones

```
#const n = 5.  
  
tiempo(1..n).  
persona(pedro).  
  
1{estudia(P, T) : tiempo(T)}5 :- persona(P).
```

¿Qué simula el programa anterior?



Restricciones de Cardinalidad

Condiciones dentro de las restricciones

```
#const n = 5.  
  
tiempo(1..n).  
persona(pedro).  
  
1{estudia(P, T) : tiempo(T)}5 :- persona(P).
```

**¡Incluye todos los modelos en los que
Pedro estudia entre los tiempos 1 y 5!**



Restricciones de Cardinalidad

```
#const n = 5.
```

```
tiempo(1..n).  
persona(pedro).
```

```
1{estudia(P, T) : tiempo(T)}5 :- persona(P).
```

¡Son equivalentes!

```
persona(pedro).
```

```
1{estudia(P, 1);estudia(P, 2);estudia(P,  
3);estudia(P, 4);estudia(P, 5);}5 :-  
persona(P).
```



Negación



Negación

La palabra not indica la **ausencia de un átomo en un modelo**.
En la lógica formal, not p. equivale a escribir **$\neg p$**

Not p.

$\neg p$.



Negación

Not p .

p no pertenece al modelo, o su equivalente:

$\text{:- } p$.

$p \text{ :- not } q$.

p pertenece al modelo si es que q no pertenece a este



Negación

- Podemos formar nuevos predicados:

```
desmayarse(P):- not desayunar(P), not dormir(P).
```

- Podemos evitar redundancias:

```
catedra(X,Y):- profesor(X), profesor(Y), not catedra(Y,X), X!=Y.
```



Negación

- Podemos realizar restricciones más complejas

```
:- not regar_arbol(A), not abonar_arbol(A), sano_arbol(A).
```

Esto se traduce a que no puede haber un modelo en el que yo no regué ni aboné mi árbol y aún así este está sano.



Negación

```
persona(berni).  
persona(jero).  
vegetarian(berni).  
come_empanada_pino(P):- persona(P), not vegetarian(P).
```

¿Qué modelo resulta?



Modelación



Modelación

Algunos tips

- Dado que Clingo es un lenguaje declarativo, pensar en el problema **resuelto**, no en cómo resolverlo.
- Probar que las reglas funcionan **individualmente** sirve para entender qué funciona y qué no.
- Soltar la mano, especialmente pasando predicados lógicos a Clingo.
- ¡Ejercitar! Hay muchos ejemplos básicos, medios y avanzados.



Modelación

Ejercicio

Se tienen 8 jugadores de tenis que quieren jugar un campeonato de dobles. Modela un programa que indique los posibles equipos.



Ayudantía 2

Negación y Modelación en Clingo

Por Bernardita Rosas y
Jerónimo Infante.

23 de agosto de 2024