Ayudantía 3

Fin Clingo y presentación tarea 2

Por Alejandro Held

31 de marzo de 2025

Contenidos

1. Restricciones de cardinalidad
2. Negación y filtros
3. Modelación
4. Presentación tarea 2

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

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 y filtros

Negación

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

```
Not p.
```

```
:- p.
```

Negación

```
Not p.
```

p no pertenece al modelo, o su equivalente:

```
:- p.
```

```
p :- 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.

Filtros

- Mediante la negación podemos generar filtros:

```
:- estudiar(R1, H), estudiar(R2, H), R1!=R2
```

Esto implica que no podemos estudiar a los ramos R1 y R2 en el mismo horario H

Filtros

- Se pueden tener distintas formas lógicas admitidas por clingo

```
tiempo_total(H1, H2, H3, H4, Total) :- Total = H1 + H2 + H3 + H4.
:- estudiar(H1), comer(H2), dormir(H3), jugar(H4), tiempo_total(H1, H2,
H3, H4, Total), Total > 24.
```

Estas reglas determinan que no podemos ocupar más de 24 horas en las acciones de estudiar, comer, dormir y jugar

Modelación

Modelación

Algunos tips

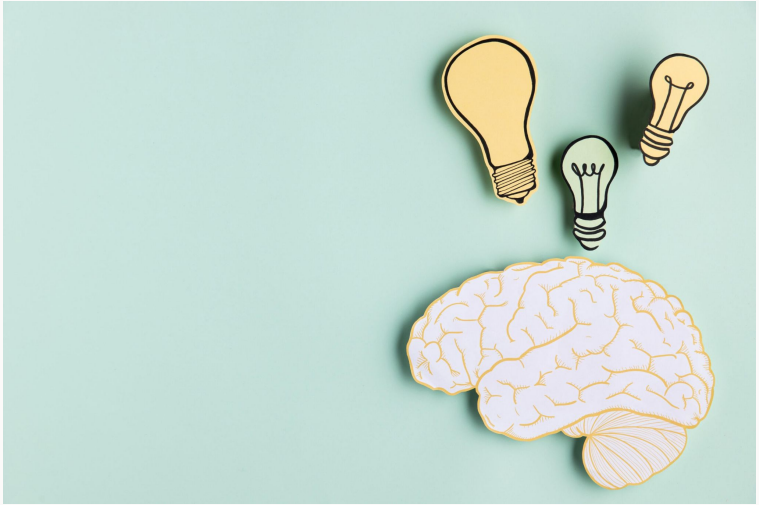
- Dado que Clingo es un lenguaje declarativo, pensar en el problema **resuelto**, no en cómo resolverlo.
- Probar que las reglas funcionan **individualmente** sirve para entender qué funciona y qué no.
- Soltar la mano, especialmente pasando predicados lógicos a Clingo.
- ¡Ejercitar! Hay muchos ejemplos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

Apoyo T2

Parte 1: Reflexión y Teoría

Algunos tips

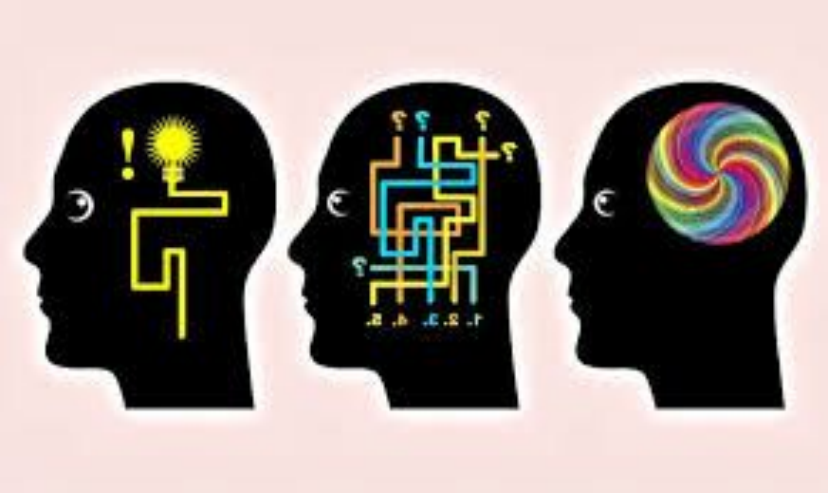
- Sean claros y concretos.
- Vean los videos con atención.



Parte 2: DCCesafíos de la Ingeniería

Algunos tips

- Piensa en el contexto de la pregunta (Desafíos de la Ingeniería en la UC)
- Documentar no es repetir lo mismo que dice el código
- Comprende el código bien antes de realizar las tareas



Parte 3: DCC Bodega

Algunos tips

- Aprovechen el código de clingo visto en clases
- No es necesario realizar las tareas del enunciado en orden
- Aprovechen el bonus que no está tan difícil