Ayudantía 1

Introducción a ASP y Clingo

Por Daniel Florea y Kaina Galdames

25 de marzo 2024

Contenidos

1. ¿Qué es ASP y para qué se usa?
2. Predicados
3. Átomos / proposiciones
4. Modelo
5. Reglas
6. Restricciones de cardinalidad
7. Anexo

¿Qué es ASP y para qué se usa?

Answer Set Programming

=

Programación de Conjuntos de Respuestas

- Paradigma de programación lógica
- Busca conjuntos de elementos que cumplan con ciertas reglas o propiedades.

¿Qué es ASP y para qué se usa?

Ejemplo clásico de lógica proposicional:

“Todos los hombres son mortales”

y

“Sócrates es hombre”

↓

“Sócrates es mortal”

¿Qué es ASP y para qué se usa?

Programación declarativa

- Describe hechos como
  - “El sol existe”
  - “El sol emite radiación”
- Lenguajes como clingo

```
existe(sol).
```

Programación imperativa

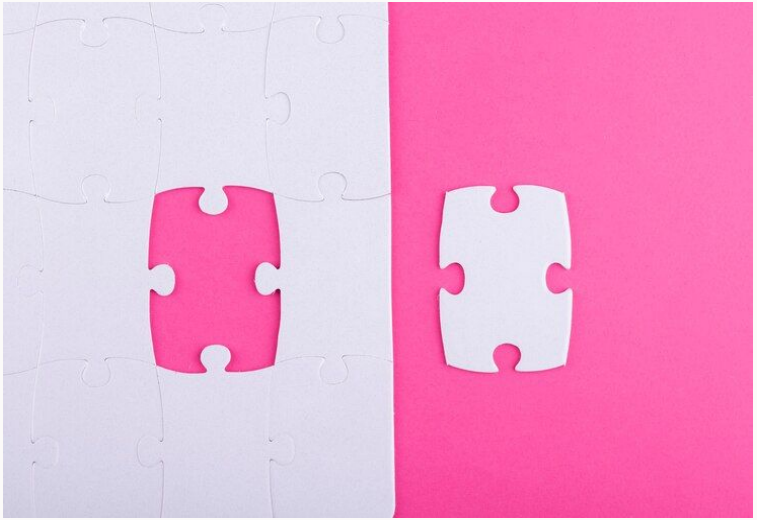
- Da órdenes como
  - “Suma 6+17”
  - “Imprime ‘Hola Mundo’”
- Lenguajes como Python

```
print(‘Hola Mundo’)
```

¿Qué es ASP y para qué se usa?

Se usa para resolver:

- Problemas de búsqueda
- Puzzles lógicos
- Problemas que requieran usar el **sistema 2** ➡ Pensamiento normativo



¿Qué es ASP y para qué se usa?

Clingo

- Es un lenguaje que combina **ASP** con solucionadores de satisfacibilidad **SAT**.
- Es el lenguaje que usaremos para escribir programas lógicos.
- Sus archivos tienen extensión **.lp** y para ejecutarlos se debe escribir en consola:

```
clingo {nombre_archivo}.lp
```

Predicados

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

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

*Las variables sólo existen dentro de los predicados y siempre comienzan con **mayúscula.**

Predicados

Aridad

- Corresponde al **número de términos** que reciben.

```
p.                          % Predicado nulario (aridad 0)
ayudante(dani).             % Predicado unario  (aridad 1)
amigo(shrek, burro).        % Predicado binario (aridad 2)
```

Átomos / proposiciones

- Definen propiedades o reglas que pueden ser ciertas o falsas.
- Un mismo predicado puede definir múltiples proposiciones, si se definen con la misma palabra pero distinta aridad.

```
p.
p(q).

aprende(estudiante).
aprende(estudiante, profesor).
```

Modelo

- Es la **solución** del programa lógico.
- Es un conjunto de **minimal** átomos que satisfacen las condiciones lógicas.
- Pueden existir varios, así como ninguno.

Ejemplo de output de Clingo

```
Solving...
Answer: 1
p r q
SATISFIABLE

Models       : 1
```

El modelo obtenido **{p, r, q}**

Modelo

Minimalidad

- Solo son modelos aquellos conjuntos con la **mínima cantidad** posible de átomos.
- De lo contrario, podrían existir infinitos modelos.
- Para el ejemplo anterior, si **{p, r, q}** es un modelo, **{p, r, q, s}** no puede serlo.

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)  % D:
```

¿Cuál o cuáles son los modelos de este programa?

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)  % D:
```

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

Reglas

Body con varios átomos

- Generan una **conjunción** de proposiciones, es decir, se deben 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 solo si a y b lo están
d :- a, m.      % d está solo si a y m lo están
```

¿Cuál o cuáles son los modelos de este programa?

Reglas

Body con varios átomos

- Generan una **conjunción** de proposiciones, es decir, se deben 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 solo si a y b lo están
d :- a, m.      % d está solo 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.
```

¿Por qué sucede esto? (*psst...* minimalidad)

Reglas

Predicados con variables

- Permiten definir múltiples proposiciones de manera simultánea.

```
arbol(platano_oriental).
arbol(quillay).
arbol(roble).
fotosintesis(platano_oriental).
fotosintesis(quillay).
fotosintesis(roble).
```

Esto...

```
arbol(platano_oriental).
arbol(quillay).
arbol(roble).
fotosintesis(Z) :- arbol(Z)
```

...es equivalente a esto

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.
```

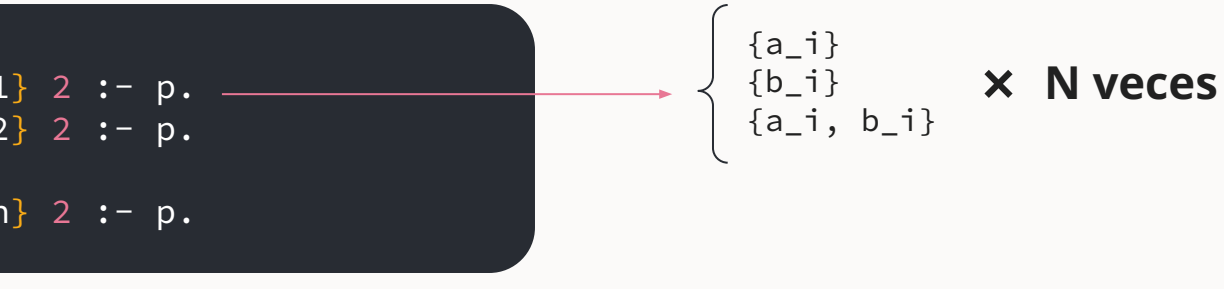
¿Cuántos modelos genera este programa?

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.
```



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

Ejercicios

Ejercicio 2.3.1: Reglas básicas

```
p :- p, q.  % si p y q se encuentran en el modelo, p también
r :- s, t.  % si s y t se encuentran en el modelo, r también
q :- s.     % si s se encuentra en el modelo, q también
p.          % p se encuentra en el modelo
```

¿Qué modelo es la única solución a este programa??

a) {p}
b) {p, q}
c) Vacío
d) El problema es insatisfacible (no existe un modelo que lo solucione)

Ejercicio obtenido de 'Intro a la Inteligencia Artificial' por Daniel Florea

Ejercicios

Ejercicio 2.3.1: Reglas básicas

```
p :- p, q.   % si p y q se encuentran en el modelo, p también
r :- s, t.   % si s y t se encuentran en el modelo, r también
q :- s.      % si s se encuentra en el modelo, q también
p.           % p se encuentra en el modelo
```

¿Qué modelo es la única solución a este programa??

a) {p}
b) {p, q}
c) Vacío
d) El problema es insatisfacible (no existe un modelo que lo solucione)

Ejercicio obtenido de 'Intro a la Inteligencia Artificial' por Daniel Florea

Ejercicios

```
q.            % q se encuentra en el modelo
p :- q, r.    % si q y r se encuentran en el modelo, p también
r :- p.       % si p se encuentra en el modelo, r también
```

¿Qué modelo genera este programa?

Ejercicio inspirado de 'Intro a la Inteligencia Artificial' por Daniel Florea

Ejercicios

```
q.              % q se encuentra en el modelo
p :- q, r.      % si q y r se encuentran en el modelo, p también
r :- p.         % si p se encuentra en el modelo, r también
```

El modelo que genera es **{q}**

Ejercicio inspirado de 'Intro a la Inteligencia Artificial' por Daniel Florea

Ejercicios

```
q.              % q se encuentra en el modelo
p :- q, r.      % si q y r se encuentran en el modelo, p también
r :- p.         % si p se encuentra en el modelo, r también
```

El modelo que genera es **{q}**

Ejercicio inspirado de 'Intro a la Inteligencia Artificial' por Daniel Florea

Anexo

Uso de *statements*

#show

Muestra en las respuestas solo los átomos que nos interesan.

#show predicado/aridad.

```
triste(niño) :- mojado(niño).
mojado(niño) :- llueve.
llueve.
#show triste/1.
```

#const

Permite reemplazar términos de constantes.

Se puede hacer directo por consola también con -c constante=valor

```
grande(c0)
#const c0 = 64.
```

Anexo

- Para que el programa muestre n modelos al ejecutar, se debe escribir en consola el número **N** al final del comando de ejecución.
- Si se quieren ver todos los modelos posibles, se debe escribir el número 0.

```
clingo {nombre_archivo}.lp N
```

Anexo

¿Por qué una regla sin Body es verdadera?

Se puede considerar que el Body vacío es igual a Falso, luego en una implicancia lógica, la tabla de verdad es la siguiente:

p	q	$(p \Rightarrow q)$
V	V	V
V	F	F
F	V	V
F	F	V

La implicancia también se puede escribir como una disyunción:
¬p V q

¬v V v = falso o verdadero = verdadero
¬v V f = falso o falso = falso
¬f V v = verdadero o verdadero = verdadero
¬f V f = verdadero o falso = verdadero

Anexo

Libros/apuntes para más información:

- https://github.com/dfloreaa/Apuntes-IIC2613/blob/main/Intro%20a%20la%20Inteligencia%20Artificial%20-%20Daniel%20Florea.pdf (en proceso de escritura)
- https://www.cs.utexas.edu/users/vl/teaching/378/ASP.pdf
- http://wp.doc.ic.ac.uk/arusso/wp-content/uploads/sites/47/2015/01/clingo_guide.pdf

Ayudantía 1

Introducción a ASP y Clingo

Por Daniel Florea y Kaina Galdames

25 de marzo 2024