Programación en Lógica: *Answer Set Programming*

Jorge A. Baier

Departamento de Ciencia de la Computación
Pontificia Universidad Católica de Chile

Santiago, Chile

Programación Basada en Lógica

- Programación basada en lógica es un paradigma de programación *declarativo* basado en lógica.
- En el paradigma declarativo, el usuario *declara los objetivos* del programa o describe la solución a un problema.
- El usuario **no define** cómo es que el programa debe ejecutar.
- Veremos un tipo especial de programación en lógica llamado *Answer Set Programming* (ASP).

ASP

Características de ASP

- Permite plantear problemas de razonamiento deductivo usando un dialecto de lógica de primer orden.
- Algunas aplicaciones incluyen diagnóstico y planning.
- Posee implementaciones **muy eficientes**.

Programa en Lógica

- Un programa en lógica es un conjunto de fórmulas lógicas.
- Hay al menos dos tipos de fórmulas que se utilizan en ASP: (1) *reglas*, y (2) *restricciones*.
- Inicialmente nos concentraremos las primeras.

Hechos

- Un *hecho* es de la forma:

$$L.$$

  donde L es una atómo de primer orden sin variables libres o una variable proposicional.

- El "." es un terminador; es parte de la sintaxis usada en ASP).

- Si un hecho L pertenece a un programa, diremos que del programa de *deduce* L.

Reglas

- Una *regla* es de la forma:

$$L_0 \leftarrow L_1, \ldots, L_m, not\ L_{m+1}, \ldots, not\ L_n,$$

donde L_i son atómos de primer orden o variables proposicionales.
Un átomo de primer orden es de la forma $r(t_1, t_2, \ldots, t_n)$, donde r es un símbolo de relación y $t_1, t_2, \ldots, t_n$ son términos.
Un término, a su vez, es un **constante** o una función $f(t_1, \ldots, t_n)$ donde $t_1, \ldots, t_n$ son términos.

Intuición Básica de la Semántica

Si la regla

$$L_0 \leftarrow L_1, \ldots, L_m, \mathit{not}\ L_{m+1}, \ldots, \mathit{not}\ L_n,$$

pertenece al programa Π. Entonces:

1. Si es posible *deducir* $L_1, \ldots, L_m$, y
2. *no* es posible deducir $L_{m+1}, L_{m+1}, \ldots, L_n$,

Entonces es posible deducir L_0.

Un Ejemplo Sencillo

$$p,$$
$$q \leftarrow p,$$
$$t \leftarrow r, not\ p,$$

¿Qué sería razonable deducir de este programa?

Otro ejemplo

En este ejemplo, representamos un grafo y la noción de camino.

$$nodo(a).$$
$$nodo(b).$$
$$nodo(c).$$
$$nodo(d).$$
$$arco(a, b).$$
$$arco(c, b).$$
$$arco(c, d).$$
$$alcanzable(X, Y) \leftarrow arco(X, Y).$$
$$alcanzable(X, Y) \leftarrow alcanzable(X, Z), alcanzable(Z, Y).$$

Sintaxis: En ASP, las variables se representan siempre con mayúsculas y las constantes con minúsculas.

Otro ejemplo

En este ejemplo, representamos un grafo y la noción de camino.

$$nodo(a).$$
$$nodo(b).$$
$$nodo(c).$$
$$nodo(d).$$
$$arco(a, b).$$
$$arco(c, b).$$
$$arco(c, d).$$
$$alcanzable(X, Y) \leftarrow arco(X, Y).$$
$$alcanzable(X, Y) \leftarrow alcanzable(X, Z), alcanzable(Z, Y).$$

Sintaxis: En ASP, las variables se representan siempre con mayúsculas y las constantes con minúsculas.

Reglas Disyuntivas

Una regla disyuntiva se ve de la siguiente forma:

$$H_1; H_2; \ldots; H_k \leftarrow L_1, \ldots, L_m, not\ L_{m+1}, \ldots, not\ L_n,$$

Notación: Usaremos $Head \leftarrow Pos \cup not(Neg)$ para denotar una regla de esta forma.
Ejemplo: Si $H = \{p, q\}$, $P = \{r, s\}$ y $N = \{t, u\}$, entonces $H \leftarrow P \cup not(N)$ representa a

$$p; q \leftarrow r, s, not\ t, not\ u$$

Semántica Para Programas Sin Negación: *Ground Terms*

Definición (Término Instanciado (ground term) de Π)

Para un programa Π con símbolos de función $\mathcal{F}$ y constantes en $\mathcal{C}$, un término instanciado (*ground term*) es de la forma:

- C, donde $C \in \mathcal{C}$ es una constante.
- $f(t_1, \ldots, t_n)$, donde f es una función de aridad n, y t_i $(1 \leq i \leq n)$ son términos instanciados.

Definición

Una sustitución θ es una función parcial de variables a términos instanciados. Si a es un átomo, entonces $a\theta$ denota el término que resulta de sustituir en a toda ocurrencia de la variable x por $\theta(x)$, para cada variable x que es asignada por θ.

Instanciación de un Programa con Variables

Definición

La instanciación de un programa Π es

$$\Pi_G = \{\pi\theta \mid \pi \in \Pi \text{ y } \theta \text{ asigna todas las variables en } \pi \text{ a algún término instanciado de } \Pi\}$$

Definición (Modelo de un Programa Sin Negación)

Un **modelo** de un programa Π instanciado y sin negación, es un conjunto minimal (respecto de la relación subconjunto) de átomos instanciados M, tales que si $Head \leftarrow Body \in \Pi$ y $Body \subseteq M$, entonces $Head \cap M \neq \emptyset$.

Reducción y Conjunto Respuesta

Definición (Reducción)

La *reducción* un programa Π relativa a un conjunto X, denotada por Π^X es la que resulta de hacer:

1. $\Pi^X := \Pi$
2. **Borrar** toda regla $Head \leftarrow Pos \cup not(Neg)$ de Π^X cuando $Neg \cap X \neq \emptyset$.
3. **Reemplazar** cada regla $Head \leftarrow Pos \cup not(Neg)$ en Π^X por $Head \leftarrow Pos$ cuando $Neg \cap X = \emptyset$.

Reducción y Conjunto Respuesta

Definición (Reducción)

La *reducción* un programa Π relativa a un conjunto X, denotada por Π^X es la que resulta de hacer:

1. $\Pi^X := \Pi$
2. **Borrar** toda regla $Head \leftarrow Pos \cup not(Neg)$ de Π^X cuando $Neg \cap X \neq \emptyset$.
3. **Reemplazar** cada regla $Head \leftarrow Pos \cup not(Neg)$ en Π^X por $Head \leftarrow Pos$ cuando $Neg \cap X = \emptyset$.

Definición (Modelo de un programa con negación)

X es un modelo de un programa con negación Π ssi X es un modelo para Π^X.

Restricciones

Una **restricción** es una regla con cabeza vacía. Ej:

$$\leftarrow p, q, r.$$

¿Qué propiedad cumple un programa con esta regla?

Restricciones de Cardinalidad

- Restricciones de cardinalidad son de la forma:

$$n\,\{p_1; p_2; \ldots; p_n\}\,m \leftarrow Body$$

- Para definir la semántica anotamos este tipo de reglas como $n \leq |Head| \leq m \leftarrow Body$.

Definición (Modelo con restricciones de cardinalidad)

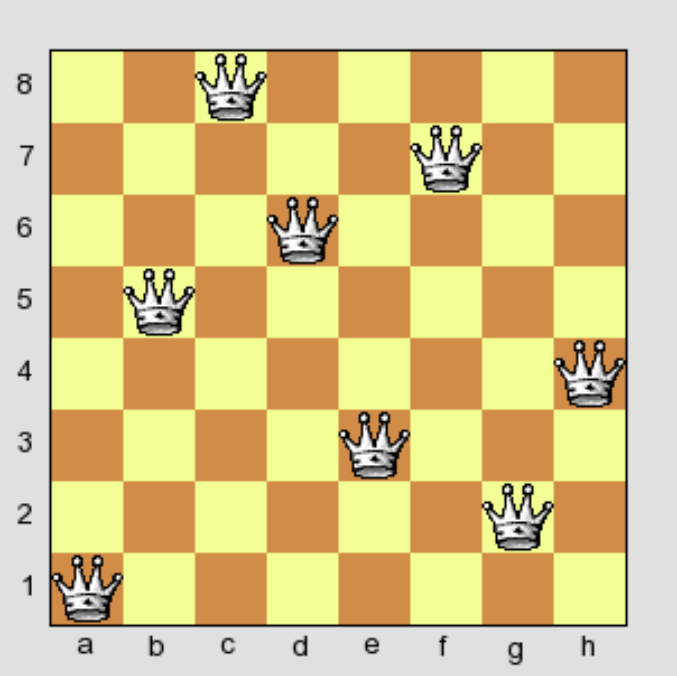
Un **modelo** de un programa Π instanciado y sin negación, es un conjunto minimal (respecto de la relación subconjunto) de átomos instanciados M, tales que

- si $Head \leftarrow Body \in \Pi$ y $Body \subseteq M$, entonces, algún átomo de $Head$ está en Π.
- si $n \leq |Head| \leq m \leftarrow Body \in \Pi$ y $Body \subseteq M$, entonces, $n \leq |Head \cap M| \leq m$.

El Problema de las 8 Reinas

Problema Ubicar 8 reinas en un tablero de Ajedrez, de tal forma que ningún par se ataque.

Una solución:[a]



[a] Imagen tomada de `http://paulbutler.org`

Un Programa con Restricciones

Usamos $q(X, Y)$ para representar el hecho que hay una regla en la posición (X, Y).
Queremos que haya una reina por columna:

$$\bigvee_{i \in \{1,\ldots,8\}} q(i,j) \leftarrow, \qquad \text{para cada } j \in \{1,\ldots,8\}$$

No queremos dos reinas en la misma columna:

$$\leftarrow q(i,j), q(i',j), i' \neq i, \qquad \text{para cada } i, i', j \in \{1,\ldots,8\}$$

No queremos dos reinas en la misma fila:

$$\leftarrow q(i,j), q(i,j'), j' \neq j, \qquad \text{para cada } i, j, j' \in \{1,\ldots,8\}$$

No queremos dos reinas en la misma diagonal:

$$\leftarrow q(i,j), q(i',j'), i \neq i', |i-i'| = |j-j'|, \quad \text{para cada } i, i', j, j' \in \{1,\ldots,8\}$$

Reinas en formato clingo

```
num(1).
num(2).
num(3).
num(4).
num(5).
num(6).
num(7).
num(8).

1 {q(1,X);q(2,X);q(3,X);q(4,X);q(5,X);q(6,X);q(7,X);q(8,X)} :- num(X).

:- q(I,J), q(Ip,J), I!=Ip.
:- q(I,J), q(I,Jp), J!=Jp.
:- q(I,J), q(Ip,Jp), Ip<I,|I-Ip|==|J-Jp|.
```

En forma más compacta: 20 reinas

```
num(1..20).

1 {q(Y,X) : num(Y)} :- num(X).

:- q(I,J), q(Ip,J), I!=Ip.

:- q(I,J), q(I,Jp), J!=Jp.

:- q(I,J), q(Ip,Jp), Ip<I, |I-Ip|== |J-Jp|.
```

Elementos Sintácticos Útiles en Clingo

1. Restricción

$$m\{L_1, L_2, \ldots, L_n\}m'$$

al menos m literales y a lo más m' son verdaderos de entre $\{L_1, \ldots, L_n\}$.

2. Notación

$$q(X) : r(X)$$

representa a la lista de los $q(X)$ tales que $r(X)$ se deduce del programa.

3. Expresiones aritméticas se evalúan usando el operador ==. Ejemplos: `X==Y*2`.

Coloreo de Grafos

Suponemos un grafo no dirigido donde el predicado arista especifica si un par de nodos está conectado.

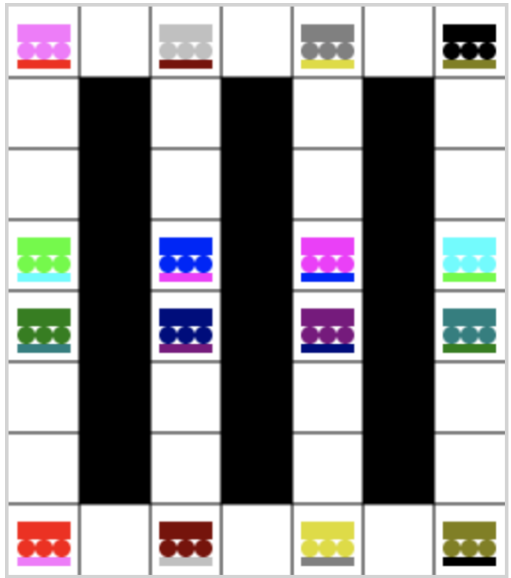
Coloreo de Grafos

Suponemos un grafo no dirigido donde el predicado arista especifica si un par de nodos está conectado.

```
color(1).
color(2).

nodo(a).
nodo(b).
nodo(c).
nodo(d).
nodo(e).

arista(a,e).
arista(a,b).
arista(a,d).
arista(c,b).
arista(c,e).
arista(a,c).

unidos(X,Y) :- arista(X,Y).
unidos(X,Y) :- arista(Y,X).

1 {pintado(X,Y):color(Y)} 1 :- nodo(X).

:- pintado(X,C), pintado(Y,C), unidos(X,Y).
```

Otra Aplicación: Multi-Agent Pathfinding

El problema consiste en mover un número de agentes desde una posición inicial hasta una final. Las acciones permitidas para cada agentes son *up*, *down*, *left*, *right*, *wait*.



[ver modelación en Syllabus]