

Programación en Lógica

Jorge A. Baier

Departamento de Ciencia de la Computación
Pontificia Universidad Católica de Chile

Santiago, Chile



- Programación Orientada a Objetos (Smalltalk)
- Programación Imperativa (Algol, Pascal, C)
- Combinación de las dos anteriores.
- Programación **declarativa**.
 - Programación Funcional.
 - Programación en Lógica.



- Los programas son fórmulas lógicas.
- Computación = Deducción automática = Demostración Automática de Teoremas



La Base: Lógica Matemática

$h(x)$: “ x es un humano”

$m(x)$: “ x es mortal”

¿Qué significa $h(Denis) \rightarrow m(Denis)$?



La Base: Lógica Matemática

$h(x)$: “ x es un humano”

$m(x)$: “ x es mortal”

¿Qué significa $h(Denis) \rightarrow m(Denis)$?

¿Qué significa $\forall x (h(x) \rightarrow m(x))$?



$h(x)$: “ x es un humano”

$m(x)$: “ x es mortal”

¿Qué significa $h(Denis) \rightarrow m(Denis)$?

¿Qué significa $\forall x (h(x) \rightarrow m(x))$?

¿Y ésta: $\neg \exists x m(x) \rightarrow \forall x \neg h(x)$?

Si estamos dispuestos a aceptar $\forall x (h(x) \rightarrow m(x))$ como cierto, ¿qué podemos concluir si $\neg m(Denis)$?



$h(x)$: “ x es un humano”

$m(x)$: “ x es mortal”

¿Qué significa $h(Denis) \rightarrow m(Denis)$?

¿Qué significa $\forall x (h(x) \rightarrow m(x))$?

¿Y ésta: $\neg \exists x m(x) \rightarrow \forall x \neg h(x)$?

Si estamos dispuestos a aceptar $\forall x (h(x) \rightarrow m(x))$ como cierto,
¿qué podemos concluir si $\neg m(Denis)$?

$$\{\forall x (h(x) \rightarrow m(x)), \neg m(Denis)\} \models \neg h(Denis)$$



La Consecuencia Lógica es Computable!

En 1965, John A. Robinson inventó el principio de **resolución**.

La regla de resolución toma dos *cláusulas* C_1 y C_2 y genera una tercera cláusula que es **consecuencia lógica** de la anterior.

[Ejemplo en pizarra]

El razonamiento lógico proposicional **se puede automatizar**.



- No todos los lenguajes de programación en lógica usan resolución.
- Prolog, desarrollado en los '70, es un ejemplo.
- Usaremos el intérprete de acá:
<http://www.swi-prolog.org/>



Un Ejemplo

```
padre(juan, amanda). % juan es padre de amanda
madre(ximena, amanda).
madre(laura, juan).
padre(andres, juan).
padre(patricio, bonifacio).
padre(juan, patricio).
padre(juan, ana).
madre(ximena, ana).
```

```
% X es progenitor de Y si X es madre de Y
progenitor(X,Y) :- madre(X,Y).
% X es progenitor de Y si X es padre de Y
progenitor(X,Y) :- padre(X,Y).
```

```
ancestro(X,Y) :- progenitor(X,Y).
ancestro(X,Y) :- progenitor(X,Z),ancestro(Z,Y).
```



- Las siguientes tres reglas podrían ser entendidas como que los *predicados* padre y madre están definidos por extensión.

padre(juan, amanda).

madre(ximena, amanda).

madre(laura, juan).

- También podemos hacer 'definiciones por comprensión' usando la noción de variable, así:

progenitor(X,Y) :- madre(X,Y).

dice que X es progenitor de Y si es que X es madre de Y.



La regla para ancestro

- Las definiciones que usan variables pueden apelar a la recursión.
- Las variables se escriben con letra mayúscula en Prolog.
- En la definición de ancestro:

```
ancestro(X,Y) :- progenitor(X,Y).  
ancestro(X,Y) :- progenitor(X,Z),ancestro(Z,Y).
```

vemos que la segunda regla es recursiva, lo que permite encontrar todos los ancestros.
- Cuando una variable solo aparece al lado derecho de una regla, se lee como una variable existencial. Así, la segunda regla para ancestro se lee “X es ancestro de Y si existe un Z tal que X es progenitor de Z y Z es ancestro de Y”.



- En el intérprete de Prolog podemos hacer consultas, que Prolog determinará si son verdaderas o falsas.
- Supongamos que tenemos el siguiente programa en el archivo `familia.pl`

```
padre(juan, amanda).  
padre(alberto, juan).  
padre(alberto, lucia).  
padre(juan, vicente).  
madre(sara, juan).  
madre(ximena, amanda).  
madre(sara, lucia).  
madre(lucia, felipe).  
madre(lucia, camila).  
madre(amanda, vicente).  
  
progenitor(X,Y) :- padre(X,Y).  
progenitor(X,Y) :- madre(X,Y).  
  
abuelo(X,Y) :- padre(X,Z),progenitor(Z,Y).
```



Haciendo Consultas en Prolog

- Desde el intérprete de Prolog, usamos:

```
?- padre(juan, amanda).
```

```
true.
```

```
?- padre(juan, juan).
```

```
false.
```

- Al usar variables en las consultas, por ejemplo:

```
?- abuelo(alberto, X).
```

```
    X = amanda ;
```

```
    X = vicente ;
```

```
    X = felipe ;
```

```
    X = camila.
```

preguntamos a Prolog “existe un X tal que alberto es abuelo de X”. En otras palabras, preguntamos si alberto tiene nietos.

- Prolog entrega todas las respuestas si se presiona ';'.



Cómo lo hace Prolog para entregar las Respuestas

- El algoritmo de demostración de Prolog se puede entender como construir un árbol de demostración que se recorre usando backtracking.
- En clases vemos esto en la pizarra, pero para más detalles, puede ver:
`http://www.learnprolognow.org/lpnpage.php?pagetype=html&pageid=lpn-htmlse6`
- En esta misma página web se puede encontrar mucha más información sobre Prolog.

