# Prolog

## introducción al lenguaje

David Gelpi Fleta

[ correo ]

Laboratorio de Introducción a los Sistemas Informáticos Inteligentes

Escuela Superior de Ingeniería Informática - Universidad Vigo

*Nota:*

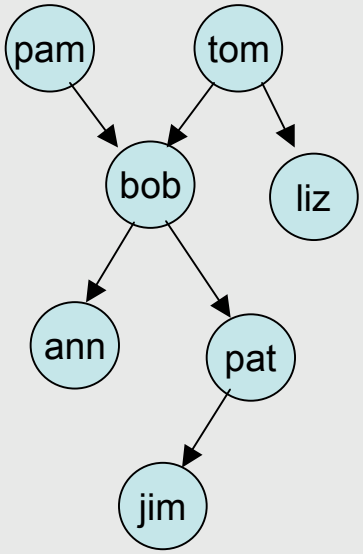
*Exposición basada en el capítulo 1 del libro Prolog: Programming for Artificial Intelligence de Ivan Bratko - Ed. Pearson - contenido en la bibliografía recomendada de la asignatura.*

# Introducción a Prolog

- Prolog
- Definir relaciones mediante hechos.
- Definir relaciones mediante reglas.
- Reglas recursivas.
- Cómo Prolog responde cuestiones.
- Sentido declarativo y procedimental de los programas.

# Definiendo relaciones mediante hechos.

- Prolog es un lenguaje de programación para computación simbólica, no numérica.
- Especialmente adecuado para resolver problemas que involucran objetos y relaciones entre ellos.



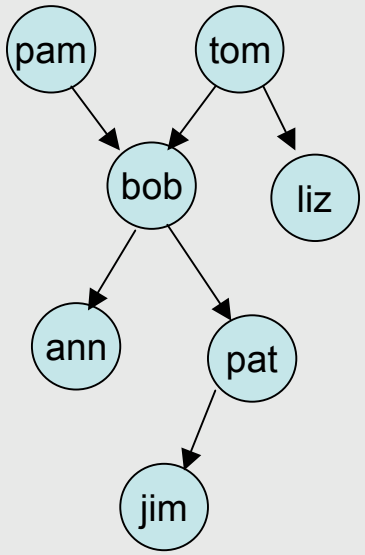

```
padre(pam,bob).
padre(tom,bob).
padre(tom,liz).
padre(bob,ann).
padre(bob,pat).
padre(pat,jim).
```

- Padre relación
- Tom, bob argumentos
- *Tom es padre de bob*
- Este programa consta de seis cláusulas
- Cada cláusula declara un hecho sobre la relación padre
- padre(tom,bob) es una instancia de la relación padre

# Definiendo relaciones mediante hechos.

- Prolog puede contestar preguntas sobre la relación padre.

```
?-padre(bob,pat).
yes
?-padre(liz,pat).
no
?-padre(X,liz).
X=tom
?-padre(bob,X).
X=ann
X=pat
?padre(X,Y).
X=pam
Y=bob;
X=tom
Y=bob;
X=tom
Y=liz;
```



# Conclusiones

- Una relación se define estableciendo las n-tuplas de objetos que satisfacen la relación.
- Un programa en Prolog consiste de cláusulas (terminan con un punto).
- Los argumentos de las relaciones pueden ser:
  - Objetos concretos: tom, pat (átomos)
  - Constantes (átomos)
  - Objetos genéricos: X, Y (variables)
- Las cuestiones consisten en uno o más objetivos.

  padre(X,ann),padre(X,pat) es la conjunción de los objetivos:

  X es padre de ann y X es padre de pat
- Prolog acepta cuestiones como objetivos que han de ser satisfechos.

# Definiendo relaciones mediante reglas.

¿Cómo extender nuestro programa “padre”?

- p.e. la relación “descendencia”:
  - añadir hecho: `descendencia(liz,tom).` ó
  - emplear hechos existentes + razonamiento: regla

```
Para todo X e Y,
  Y es un descendiente de X si
  X es padre de Y.
```

```
descendiente(Y,X):- parent(X,Y).
```

- regla ≠ hecho
  - Un hecho es siempre cierto.
  - Una regla especifica algo cierto si alguna condición se satisface.

```
descendiente(Y,X):- parent(X,Y).
```

cabecera *conclusión* — cuerpo *condición*

# ¿Cómo emplea Prolog las reglas?

- Ejemplo de computación:

1. Formular pregunta: `?-descendiente(liz,tom)`
2. ¿Existe hecho que satisfaga el objetivo?: no ⇒aplicar reglas:

   `descendiente(Y,X):-padre(X,Y)`
3. Para aplicar la regla a liz y tom, instanciamos las variables:

   `X = tom, Y = liz`
3. Obtenemos un caso particular de la regla:

   `descendiente(liz,tom):-padre(tom,liz)`
4. El objetivo inicial `descendiente(liz,tom)` ha sido sustituido por el objetivo `padre(tom,liz)`
5. ¿Existe hecho que satisfaga el objetivo?: si ⇒ conclusión de la regla es cierta.
6. Respuesta: si

# Extensiones a un programa

Continuemos extendiendo nuestro programa ejemplo.

- Mediante hechos:

```
mujer(pam).
hombre(tom).
hombre(bob).
mujer(liz).
mujer(pat).
mujer(ann).
hombre(jim).
```

  `hombre(tom)` es una relación unaria: declaran una propiedad si/no simple de un objeto.

- Mediante reglas:

```
madre(X,Y):-padre(X,Y),mujer(X)
```

conjunción

```
Para todo X e Y
    X es madre de Y si
    X es padre de Y y
    X es mujer
```

# Conclusiones

- Un programa Prolog puede ser extendido añadiendo nuevas cláusulas.
- Tres tipos de cláusulas: hechos, reglas y cuestiones.
- Los hechos declaran cosas que son siempre incondicionalmente ciertas.
- Las reglas declaran cosas que son ciertas dependiendo de una condición dada.
- Mediante cuestiones un usuario puede preguntar al programa Prolog qué cosas son ciertas.
- Una cláusula Prolog consta de encabezamiento y cuerpo.
  - El cuerpo es una lista de objetivos separadas por comas.
  - Las comas se entienden como conjunciones.
- Los hechos son cláusulas que poseen un encabezamiento (conclusión) y el cuerpo (condición) vacío. Las cuestiones sólo poseen cuerpo. Las reglas tienen encabezado y cuerpo no vacío.
- En el transcurso de la computación, una variable puede ser sustituida por otro objeto. Decimos así que la variable es instanciada.
- Las variables son universalmente cuantificadas y se leen “para todo”. Si la variable aparece sólo en el cuerpo, existen lecturas alternativas:

```
tienehijo(X):- padre(X,Y).
```

```
Para todo X e Y
 si X es padre de Y entonces
 X tiene hijo
```

```
Para todo X
 X tiene hijo si
 hay algún Y tal que X es padre de Y
```

# Reglas recursivas

- Intentemos añadir la relación “predecesor” a nuestro programa ejemplo.
- La relación consta de dos reglas:
  - la primera define el predecesor directo:

```
Para todo X y Z
  X es un predecesor de Z si
  X es padre de Z.
```

```
predecesor(X,Z):- padre(X,Z).
```

  - y la segunda los predecesores indirectos:

```
predecesor(X,Z):-
 padre(X,Z).
```

```
predecesor(X,Z):
 padre(X,Y),
 padre(Y,Z).
```

```
predecesor(X,Z):
 padre(X,Y1),
 padre(Y1,Y2),
 padre(Y2,Z).
```

```
predecesor(X,Z):
 padre(X,Y1),
 padre(Y1,Y2),
 padre(Y2,Y3),
 padre(Y3,Z).
...
```

# Reglas recursivas

- La segunda regla solo trabaja hasta la profundidad definida por el cuerpo de la regla… y el programa además de limitado es extenso.
- Necesitamos definir una regla que defina el predecesor a cualquier nivel de profundidad.
- Solución: definir la relación predecesor en términos de si misma.

```
Para todo X y Z,
 X es predecesor de Z si
 existe Y tal que
 (1) X es padre de Y y
 (2) Y es predecesor de Z.
```

```
predecesor(X,Z):-
 padre(X,Y),
 predecesor(Y,Z).
```

- La relación “predecesor” consiste así de dos reglas:

```
predecesor(X,Z):-
 padre(X,Z).

predecesor(X,Z):-
 padre(X,Y),
 predecesor(Y,Z).
```

# ¿Cómo es un programa Prolog?

```
parent( pam, bob).                % Pam is a parent of Bob
parent( tom, bob).
parent( tom, liz).
parent( bob, ann).
parent( bob, pat).
parent( pat, jim).
female( pam).                     % Pam is female
male( tom).                       % Tom is male
male( bob).
female( liz).
female( ann).
female( pat).
male( jim).

offspring( Y, X):-                % Y is an offspring of X if
     parent( X, Y).               % X is a parent of Y

mother( X, Y) :-                  % X is the mother of Y if
     parent( X, Y),               % X is a parent of Y and
     female( X).                  % X is female

grandparent( X, Z) :-             % X is a grandparent of Z if
     parent( X, Y),               % X is a parent of Y and
     parent( Y, Z).               % Y is a parent of Z

sister( X, Y) :-                  % X is a sister of Y if
     parent( Z, X),
     parent( Z, Y),               % X and Y have the same parent and
     female( X),                  % X is female and
     different( X, Y).            % X and Y are different

predecessor( X, Z) :-             % Rule prl: X is a predecessor of Z
     parent( X, Z).

predecessor( X, Z) :-             % Rule pr2: X is a predecessor of Z
     parent( X, Y),
     predecessor( Y, Z).
```

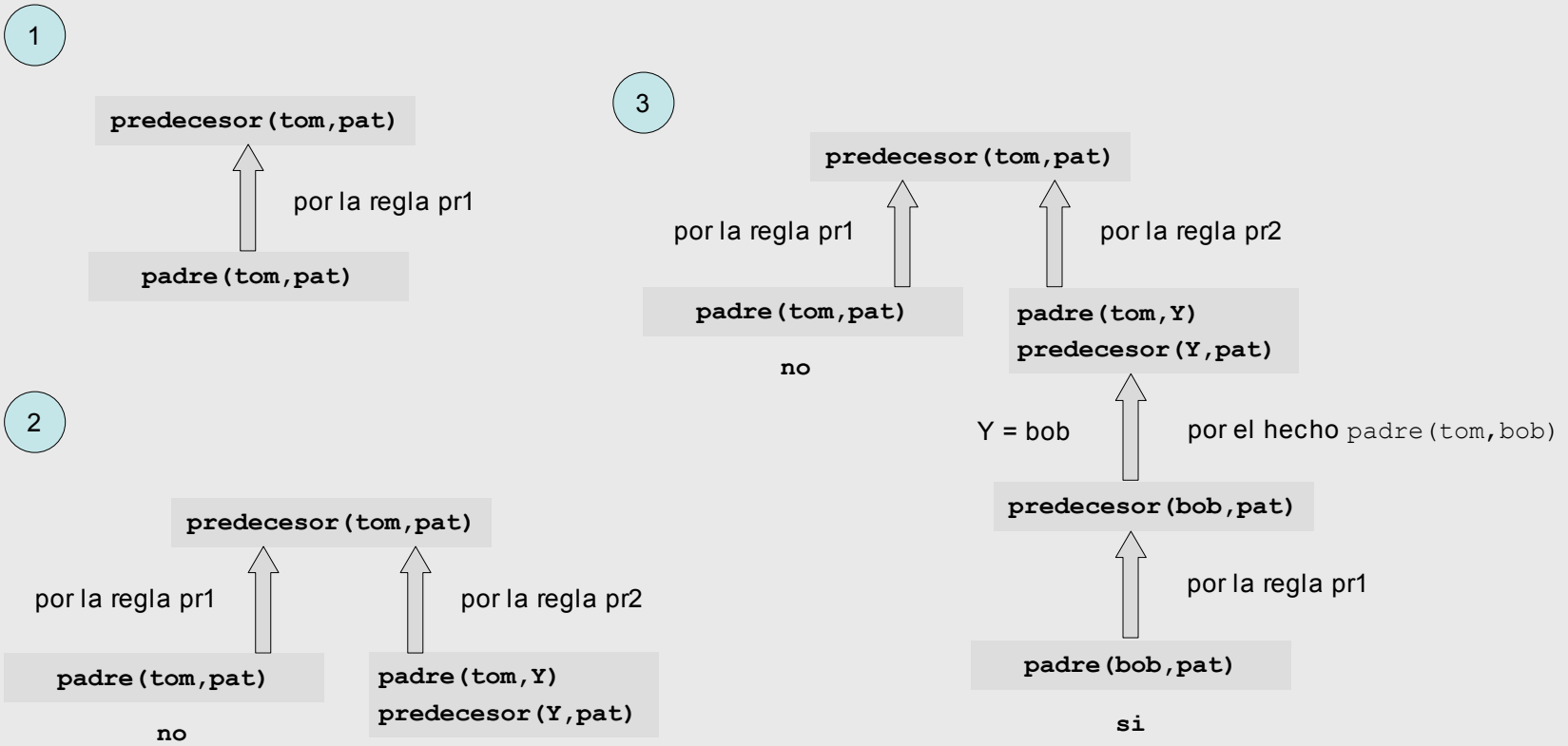
comentario

procedimiento

# ¿Cómo contesta Prolog cuestiones?

- Una cuestión para Prolog es siempre una secuencia de uno o más objetivos.
- Para contestar una cuestión, Prolog intenta satisfacer todos los objetivos.
- Para satisfacer un objetivo es preciso demostrar que es cierto, asumiendo que las relaciones dadas en el programa son ciertas.
- Satisfacer un objetivo significa demostrar que se deduce lógicamente de los hechos y reglas del programa.
- Si la cuestión contiene variables, Prolog tiene que encontrar los objetos particulares para los cuales los objetivos se satisfacen.
- La instanciación de las variables se muestra al usuario. Prolog contesta “no” si no puede demostrar que el objetivo se deduce del programa para alguna instanciación de las variables.
- Interpretación de este proceso en términos matemáticos:
  - Prolog acepta hechos y reglas como axiomas y cuestiones como un teorema
  - Intenta probar demostrar el teorema, i,e., que se deriva lógicamente de los axiomas.

# ¿Cómo contesta Prolog cuestiones?



# Significado declarativo y procedimental de los programas

- Hemos visto cómo es posible entender el resultado del programa sin comprender exactamente cómo el sistema lo encuentra.
- Un programa Prolog posee:
  - Sentido declarativo
  - Sentido procedimental
- El sentido declarativo concierne a las relaciones definidas por el programa; determina *cuál* será la salida del programa.
- El sentido procedimental determina *cómo* se obtiene la salida, cómo las relaciones son evaluadas por el sistema Prolog.
- Ventaja que presenta Prolog: es posible concentrarse en las relaciones que definen el problema más que en los detalles de la ejecución.
  - Esto es impensable en programas escritos en C o Pascal.
  - El aspecto procedimental no puede ser ignorado por razones de eficiencia del código.

# Ejercicios propuestos

1. Con la definición de la relación padre de la transparencia 5, cuál es la respuesta de Prolog a las cuestiones:

   a) padre(jim, X).

   b) padre(pam, X), padre(X, pat).

   c) padre(pam, X), padre(X, Y), padre (Y, jim).

2. Expresa en Prolog:

   a) Todo el que tiene hijos es feliz

   b) La relación nieto usando la relación padre.

   c) La relación tía usando las relaciones parent y sister de la transparencia 13.

3. Explica cómo encontraría Prolog solución a las siguientes cuestiones, usando el programa de la transparencia 13.

   a) parent(pam, bob).

   b) mother(pam, bob).

   c) grandparent (pam, ann).

   d) grandparent (bob, jim).