Prolog

Sintaxis y significado de un programa Prolog

David Gelpi Fleta

[correo]

Laboratorio de Introducción a los Sistemas Informáticos Inteligentes

Escuela Superior de Ingeniería Informática - Universidad Vigo

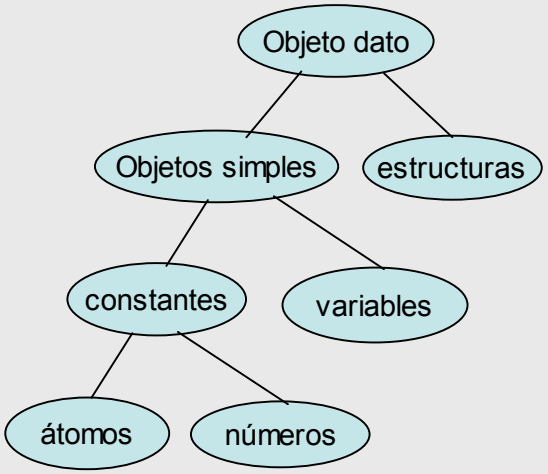
Nota:

Exposición basada en el capítulo 2 del libro Prolog: Programming for Artificial Intelligence de Ivan Bratko - Ed. Pearson - contenido en la bibliografía recomendada de la asignatura.

Contenido

- Datos simples y estructurados.
- Emparejamiento (*matching*)
- Sentido declarativo de un programa.
- Sentido procedimental de un programa.
- Efectos producidos por la reordenación de cláusulas.

Tipos de datos: átomos y números



- El sistema Prolog reconoce el tipo de un objeto en el programa por su forma sintáctica.
- La sintaxis de Prolog especifica distintas formas para cada tipo de dato. Ejemplo: átomos `tony` y variables `X`
  - Átomos:
    - Formados por cadenas de los siguientes caracteres:
      - Letras mayúsculas: A, B , ..., Z
      - Letras minúsculas: a, b, ..., z
    - Tres posibles maneras de construir átomos:
      - Cadenas de caracteres, dígitos y el carácter subrayado '_', **comenzando con una letra minúscula.**

        ```
        ana   x25   x_   tony_clifton
        ```

      - Cadenas de caracteres especiales:

        ```
        <--->   ====>   ::=
        ojo con cadenas predefinidas
        ```

      - Cadenas encerradas entre comillas simples: no confundir con variables

        ```
        'Ana'   'Tony_Clifton'
        ```

  - Números: enteros y reales:
    - Números reales no son necesarios en programación simbólica

      ```
      1   1313   -97   3.14   -0.0035
      ```

Tipos de datos: variables

- Las variables son cadenas de letras, dígitos y guión bajo.
- Comienzan con un letra mayúscula o un guión bajo.

```
X   Resultado   ListaParticipantes   _x   _23
```

- Variables anónimas:

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

  - variables que sólo aparecen una vez en la cláusula
  - su valor no es devuelto cuando Prolog responde a una cuestión de la forma: `?-padre(X,_).`

Tipos de datos: estructuras

- Una estructura es un objeto que tiene varios componentes.
  - Ejemplo: estructura *fecha* con componentes *día*, *mes* y *año*.
- Los componentes pueden ser, a su vez, estructuras.
- Las estructuras son tratadas como objetos simples.
- Los componentes se combinan en un objeto simple mediante un conector funcional (*functor*).

Prolog distingue dos conectores funcionales:
- por su nombre
- por su aritmética: n° de componentes

```
fecha(1,mayo,2001)
```

```
functor(argumento_1,argumento_2,…, argumento_n)
```

- Los componentes pueden ser constantes (átomos y números), variables y estructuras.

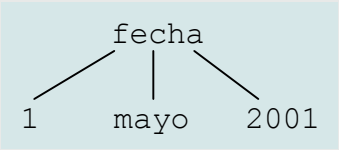
```
fecha(Dia,mayo,2001)
```

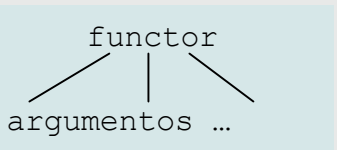
- Sintacticamente, todos los objetos dato en Prolog son términos:

```
mayo
fecha(1,mayo,2001)
```

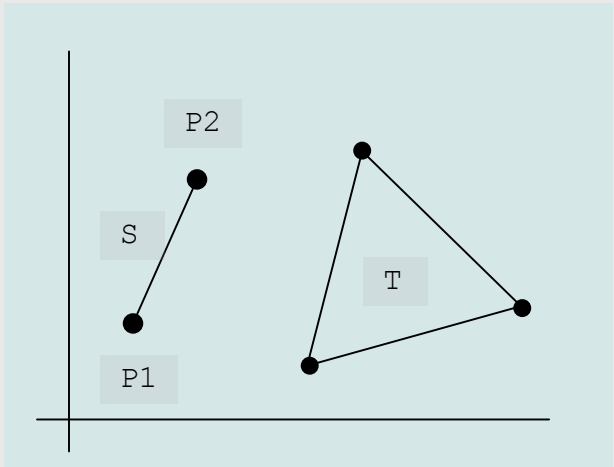
- Todos los objetos estructurados pueden ser representados como árboles.

```
fecha(1,mayo,2001)
```

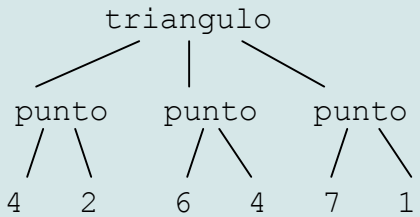




Estructuras: ejemplos




```
Objetos:
    P1 = punto(1,1)
    P2 = punto(2,3)
    S = seg(P1,P2) = seg(punto(1,1),punto(2,3))
    T = triangulo(punto(4,2),punto(6,4),punto(7,1))
```



Algunos objetos geométricos y su representación en Prolog.

Emparejamiento (*matching*)

- Dados dos términos, decimos que emparejan si:

  1) son idénticos

  2) las variables en ambos términos pueden ser instanciadas a objetos de tal modo que tras la sustitución de las variables por estos objetos los términos resultan idénticos.

```
?-fecha(D,M,2001)=fecha(D1,mayo,Y1)
  D=D1   D es instanciada a D1
  M=mayo
  Y1=2001
fecha(D,M,2001)    fecha(D1,M1,1444)   no!
Fecha(D,M,2001)    punto(X,Y,Z)
```

- Dos términos, S y T, emparejan:
  1. Si S y T son constantes entonces S y T emparejan sólo si son el mismo objeto.
  2. Si S es una variable y T es cualquier objeto, entonces emparejan y S es instanciada a T. Si T es una variable, entonces T es instanciada a S.
  3. Si S y T son estructuras, emparejan sólo si:
     a) S y T tienen el mismo functor principal y
     b) todos sus componentes correspondientes emparejan.

  La instanciación resultante se determina por el emparejamiento de los componentes.

Sentido declarativo y procedimental de un programa Prolog

Consideremos la cláusula:

```
P :- Q, R
```

donde P, Q y R poseen sintaxis de términos.

- lecturas declarativas de la cláusula:
  - P es cierto si Q y R son ciertos.
  - De Q y R se sigue P.
- lecturas procedimentales de la cláusula:
  - Para resolver P, resuelve primero el subproblema Q y entonces el subproblema R.
  - Para satisfacer P, satisface primero Q y entonces R.

[El sentido procedimental define no sólo las *relaciones lógicas* entre la cabecera y los objetivos, también el *orden* en el cual los objetivos se procesan]

Sentido declarativo de un programa Prolog

[El sentido declarativo determina cuándo un objetivo es cierto y para qué valores de las variables]

cláusula:

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

Una instancia de una cláusula C es la cláusula C con cada una de sus variables sustituidas por algún término.

instancias:

```
tienehijo(pedro):- padre(pedro,Z).
tienehijo(roberto):- padre(roberto,pequeña(eva)).
```

Una variante de una cláusula C es así una instancia de la cláusula C donde cada variable ha sido sustituida por otra variable.

variantes:

```
tienehijo(A):- padre(A,B).
tienehijo(X1):- padre(X1,X2).
```

Lista de objetivos

Una cuestión se interpreta como una *lista de objetivos* separados por comas.

Una lista de objetivos es cierta si todos los objetivos en la lista con ciertos para la *misma instanciación de las variables*.

- operadores:

P :- Q , R. **, conjunción** *and*

P :- Q ; R. **; disyunción** *or*

- Orden de precedencia:

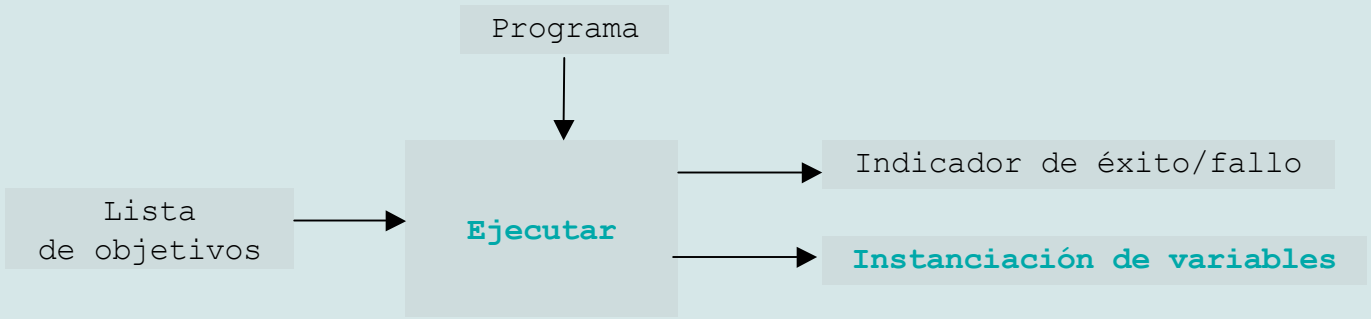
P :- Q, R; S, T, U P:- (Q, R); (S,T,U)

orden de precedencia

Un objetivo G es cierto si y sólo si:
(1) existe una cláusula C en el programa tal que
(2) existe una cláusula instancia I de C tal que
(a) la cabecera de I es idéntica a G, y
(b) todos los objetivos en el cuerpo de I son ciertos.

Sentido procedimental.

- El sentido procedimental especifica cómo Prolog contesta cuestiones.
- Para contestar una cuestión Prolog intenta satisfacer una *lista de objetivos*.
- Serán *satisfechos* si las *variables* presentes en los objetivos pueden ser *instanciadas* de modo que los objetivos se sigan del programa.
- *Ejecutar* objetivos significa satisfacerlos: *procedimiento* “ejecutar”



Vista I /O del **procedimiento** que ejecuta una lista de objetivos

Cómo Prolog contesta cuestiones

Satisfacer lista de objetivos: G1, G2, …, Gm

- Si la lista de objetivos está vacía, terminar con éxito.
- Si la lista no está vacía, continuar con la operación “escanear”:
  - escanear a través de las cláusulas desde el principio del programa al final hasta que se encuentra la primera cláusula C tal que C empareja con el primer objetivo G1.
    - Sea C de la forma: H :- B1, …, Bn.
      entonces renombrar las variables en C para obtener una variante C’ de C tal que C’ y la lista G1, …, Gm no posean variables comunes. Sea C’: H’:- B1’, …, Bn’.
    - Emparejar G1 y H’. Sea S el resultado de la instanciación de variables.
    - En la lista de objetivos G1, G2, ,,,. Gm reemplazar G1 con la lista B1’, …, Bn’ para obtener una nuva lista de objetivos: B1’, …, Bn’, G2, …, Gm.
    - Sustituir las variables de esta nueva lista de objetivos con los valores especificados en la instanciación S: B1’’, …, Bn’’, G2’, …, Gm’
  - Si no existe tal cláusula, terminar con fallo.
- Ejecutar recursivamente este procedimiento en la nueva lista de objetivos:
  - Si la ejecución de esta lista termina en éxito entonces terminar la ejecución de la lista original de objetivos con éxito.
  - Si la ejecución termina con fallo, entonces abandonar esta lista y escanear a partir de la cláusula siguiente a la cláusula C.

Cómo Prolog contesta cuestiones: ejemplo

```
grande(oso).                % cláusula 1
grande(elefante).           % cláusula 2
pequeño(gato).              % cláusula 3

marron(oso).                % cláusula 4
negro(gato).                % cláusula 5
gris(elefante).             % claúsula 6

oscuro(Z):- negro(Z).       % cláusula 7: todo negro es oscuro
oscuro(Z):- marron(Z).      % cláusula 8: todo marrón es oscuro
```

```
?- oscuro(X), grande(X).
```

resultado: `X = oso`

¿cómo?:

sustitución objetivo: `oscuro(Z):- negro(Z).`

instanciación: `X = gato`

backtracking: `grande(gato)` *fallo!*

sustitución objetivo: `oscuro(Z):- marron(Z).`

instanciación: `X = oso`

segundo objetivo: `grande(oso)` *cierto!*

respuesta: `X=oso`

Orden de cláusulas y objetivos

Consideremos la cláusula:

```
p :- p
```

- Declarativamente es correcto especificar que “p es cierto si *p* es cierto”.
- Procedimentalmente conduce a un bucle infinito cuando preguntamos al sistema por la cuestión ?-p.

[Un programa Prolog puede ser correcto declarativamente hablando pero incorrecto desde el punto de vista procedimental]

  - Un orden adecuado de las cláusulas previene la existencia de bucles infinitos y mejora la eficiencia del programa.
  - La reordenación de las cláusulas de un programa varía su sentido procedimental aunque mantenga el mismo sentido declarativo.
  - Existen también otras técnicas más robustas que el reordenamiento de cláusulas: usadas en programas que tratan sobre la resolución de problemas, búsquedas y búsqueda de caminos.

Efectos del reordenamiento de cláusulas:
cuatro versiones de un mismo programa

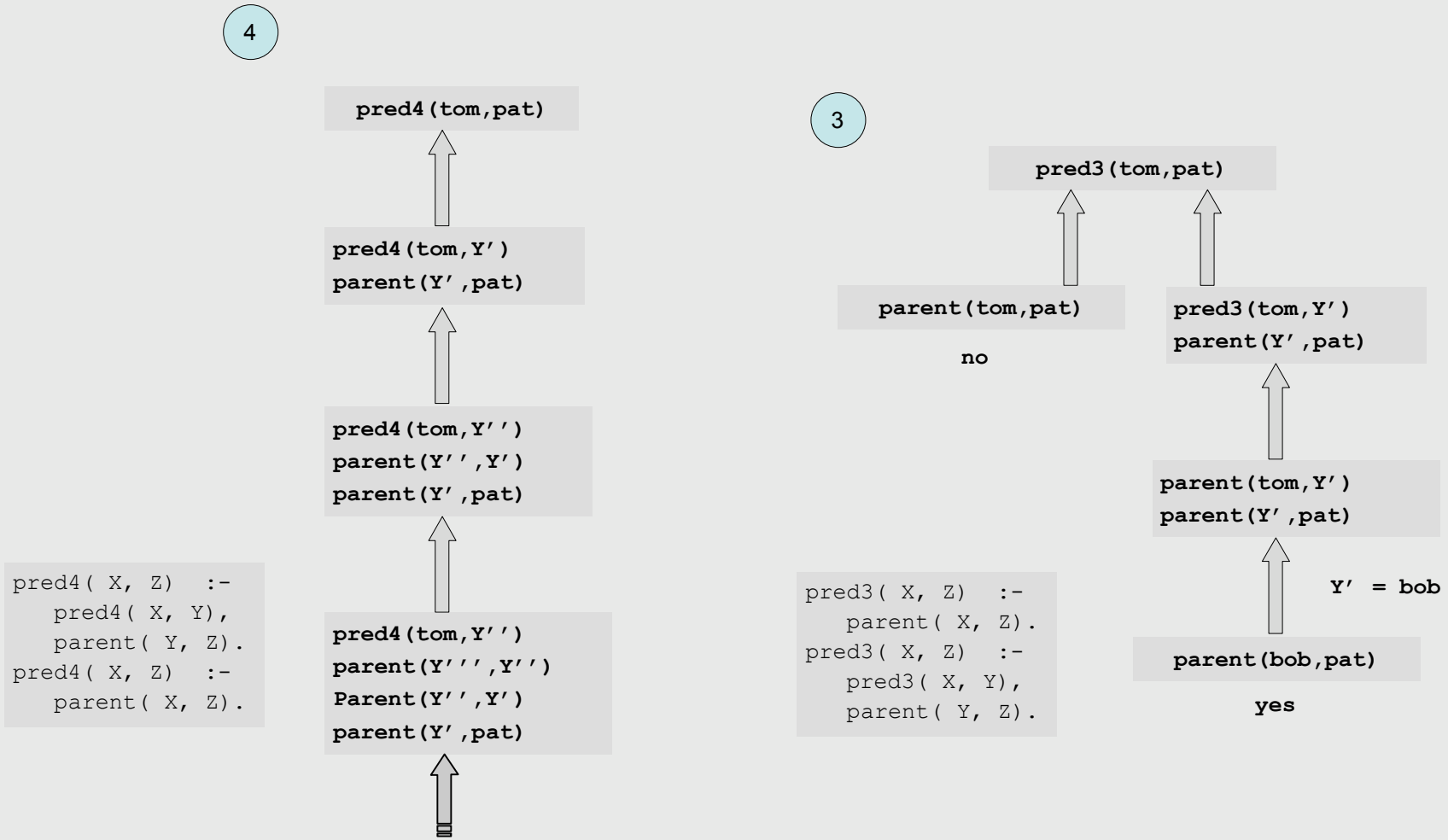
```
% The original version
pred1( X, Z)  :-
   parent( X, Z).
pred1( X, Z)  :-
   parent( X, Y),
   pred1( Y, Z).

% Variation a: swap clauses of the original version
pred2( X, Z)  :-
   parent( X, Y),
   pred2( Y, Z).
pred2( X, Z)  :-
   parent( X, Z).

% Variation b: swap goals in second clause of the original version
pred3( X, Z)  :-
   parent( X, Z).
pred3( X, Z)  :-
   pred3( X, Y),
   parent( Y, Z).

% Variation c: swap goals and clauses of the original version
pred4( X, Z)  :-
   pred4( X, Y),
   parent( Y, Z).
pred4( X, Z)  :-
   parent( X, Z).
```

Efectos del reordenamiento de cláusulas



Ejercicios propuestos

1. Haz una propuesta de representación de círculos, rectángulos y cuadrados como objetos Prolog estructurados. Usa una aproximación similar a la de la transparencia 8.
2. Las siguientes operaciones de emparejamiento (matching), ¿ tendrán éxito o fracasarán?. En caso de éxito, indica la instanciación de las variables:

```
punto (A,B) = punto (1,2)
punto (A,B) = punto (X,Y,Z)
triangulo(punto(-1,0), P2, P3) = triangulo(P1, punto(1,0), punto(0,Y))
```

3. El programa que aparece a continuación indica que dos personas son parientes si

   uno es predecesor del otro

   tienen un predecesor en común

   tienen un sucesor en común.

   ¿Se podría acortar el programa usando una notación con punto y coma? ¿Cómo?

```
parientes( X, Y) :- predecesor (X,Y)
parientes( X, Y) :- predecesor (Y,X)
parientes( X, Y) :-
   predecesor (Z,X)
   predecesor (Z,Y)
parientes( X, Y) :-
   predecesor (X, Z)
   predecesor (Y, Z)
```