Metaprogramación

José A. Alonso Jiménez,
José L. Ruiz Reina y
Francisco J. Martín Mateos

Dpto. de Ciencias de la Computación e Inteligencia Artificial

Universidad de Sevilla

Aritmética en Prolog

- **Evaluación de expresiones aritméticas: `is`**

```
?- display(2+3*4).
+(2, *(3, 4))
?- X is 2+3*4.
X = 14
?- Y is (2+3)/4.
Y = 1.25
?- 5 is 2+3.
Yes
?- 5 is X+3.
[WARNING: Unbound variable in arithmetic expression]
```

- **Comparación de `is` con `=`**

```
?- X = 2+3.
X = 2 + 3
?- X is 2+3.
X = 5
?- 5 = 2+3.
No
?- 5 is 2+3.
Yes
?- 2+X = 2+3.
X = 3
?- X = 2+Y.
X = 2 + _G89
Y = _G89
```

Listas en Prolog

```
?- display([a]).
.(a, [])
Yes

?- display([a,b]).
.(a, .(b, []))
Yes

?- .(X,Y) = [a].
X = a
Y = []
Yes

?- X = .(a,[]).
X = [a]
Yes

?- .(X,Y) = [a,b].
X = a
Y = [b]
Yes

?- .(X,Y) = [a,b,c].
X = a
Y = [b, c]
Yes
```

```
?- [X|Y] = [a,b,c].
X = a
Y = [b, c]
Yes

?- [X,Y|Z] = [a,b,c].
X = a
Y = b
Z = [c]
Yes

?- [X,Y,Z] = [a,b,c].
X = a
Y = b
Z = c
Yes

?- [X,Y,Z] = [a,b].
No

?- [X,Y|Z] = [a,b].
X = a
Y = b
Z = []
Yes
```

Ayudas en SWI–Prolog

```
?- apropos(appe).
append/1            Append to a file
concat/3            Append two atoms
concat_atom/2       Append a list of atoms
concat_atom/3       Append a list of atoms with separator
append/3            Concatenate lists
protocola/1         Append log of the user interaction to
Yes
?- help(append).
append(?List1, ?List2, ?List3)
     Succeeds when List3 unifies with the concatenation
     of List1 and List2. The predicate can be used with
     any instantiation pattern (even three variables).
Yes
?- listing(append).
append([], A, A).
append([A|B], C, [A|D]) :-
        append(B, C, D).
Yes
?- append([a,b],[b,c],X).
X = [a, b, b, c]
Yes
?- append(X,Y,[a,b]).
X = []
Y = [a, b] ;
X = [a]
Y = [b] ;
X = [a, b]
Y = [] ;
No
```

Poda de la búsqueda mediante corte

- **Ejemplo** `nota`
  - **Programa sin corte**

```
nota(X,suspenso)      :- X < 5.
nota(X,aprobado)      :- 5 =< X, X < 7.
nota(X,notable)       :- 7 =< X, X < 9.
nota(X,sobresaliente) :- 9 =< X.
```

  - **Traza**

```
?- trace.
Yes
?- nota(6,X).
   Call:  (  7) nota(6, _G98) ?
   Call:  (  8) 6 < 5 ?
   Fail:  (  8) 6 < 5 ?
   Redo:  (  7) nota(6, _G98) ?
   Call:  (  8) 5 =< 6 ?
   Exit:  (  8) 5 =< 6 ?
   Call:  (  8) 6 < 7 ?
   Exit:  (  8) 6 < 7 ?
   Exit:  (  7) nota(6, aprobado) ?
X = aprobado ;
   Redo:  (  7) nota(6, _G98) ?
   Call:  (  8) 7 =< 6 ?
   Fail:  (  8) 7 =< 6 ?
   Redo:  (  7) nota(6, _G98) ?
   Call:  (  8) 9 =< 6 ?
   Fail:  (  8) 9 =< 6 ?
   Fail:  (  7) nota(6, _G98) ?
No
```

Poda de la búsqueda mediante corte

- **Programa con corte**

```
nota(X,suspenso)      :- X < 5, !.
nota(X,aprobado)      :- X < 7, !.
nota(X,notable)       :- X < 9, !.
nota(X,sobresaliente).
```

- **Traza**

```
?- trace.
Yes

?- nota(6,X).
   Call:  (  7) nota(6, _G98) ?
   Call:  (  8) 6 < 5 ?
   Fail:  (  8) 6 < 5 ?
   Redo:  (  7) nota(6, _G98) ?
   Call:  (  8) 6 < 7 ?
   Exit:  (  8) 6 < 7 ?
   Exit:  (  7) nota(6, aprobado) ?
X = aprobado ;
No
```

Poda de la búsqueda mediante corte

- **Ventajas e inconvenientes del uso del corte**
  - **Aumento de eficiencia**
  - **Pérdida de sentido declarativo. Ejemplo:**

```
?- nota(6,sobresaliente).
Yes
```

- **Ejemplo** `maximo`
  - **Programa sin corte**

```
maximo_1(X,Y,X) :- Y =< X.
maximo_1(X,Y,Y) :- X =< Y.
```

  - **Programa con corte**

```
maximo_2(X,Y,X) :- Y =< X, !.
maximo_2(X,Y,Y).
```

  - **Sesión**

```
?- maximo_1(3,5,X).
X = 5
Yes
?- maximo_2(3,5,X).
X = 5
Yes
?- maximo_1(3,2,2).
No
?- maximo_2(3,2,2).
Yes
```

Metaintérpretes

• Metaintérprete 1

• Programa objeto

```
:- op(1200,xfx,<-).

es_mamifero        <- [tiene_pelos].
es_mamifero        <- [da_leche].
es_ungulado        <- [es_mamifero, tiene_pezuñas].
es_ungulado        <- [es_mamifero, rumia].
es_jirafa          <- [es_ungulado, tiene_cuello_largo].
es_cebra           <- [es_ungulado, tiene_rayas_negras].
tiene_pelos        <- [].
tiene_pezuñas      <- [].
tiene_rayas_negras <- [].

alumno(A,P)       <- [estudia(A,C), enseña(P,C)].
estudia(ana,ia)   <- [].
estudia(ana,pl)   <- [].
estudia(eva,ra)   <- [].
enseña(jose_a,ia) <- [].
enseña(jose_a,ra) <- [].
enseña(rafael,pl) <- [].
```

• Sistema de razonamiento

```
prueba([]).
prueba([A|L]) :-
        prueba(A),
        prueba(L).
prueba(A) :-
        (A <- B),
        prueba(B).
```

Metaintérpretes

- **Sesión**

```
?- prueba(alumno(A,jose_a)).
A = ana ;
A = eva ;
No

?- prueba(es_cebra).
Yes

?- prueba(es_jirafa).
No
```

Metaintérpretes

- **Metaintérprete 2**
  - **Programas objetos**

```
:- op(1200,xfx,<-).

es_mamifero        <- tiene_pelos.
es_mamifero        <- da_leche.
es_ungulado        <- es_mamifero, tiene_pezuñas.
es_ungulado        <- es_mamifero, rumia.
es_jirafa          <- es_ungulado, tiene_cuello_largo.
es_cebra           <- es_ungulado, tiene_rayas_negras.
tiene_pelos        <- verdad.
tiene_pezuñas      <- verdad.
tiene_rayas_negras <- verdad.

alumno(A,P)       <- estudia(A,C), enseña(P,C).
estudia(ana,ia)   <- verdad.
estudia(ana,pl)   <- verdad.
estudia(eva,ra)   <- verdad.
enseña(jose_a,ia) <- verdad.
enseña(jose_a,ra) <- verdad.
enseña(rafael,pl) <- verdad.
```

  - **Sistema de razonamiento**

```
prueba(verdad).
prueba((A,L)) :-
        prueba(A),
        prueba(L).
prueba(A) :-
        (A <- B),
        prueba(B).
```

Metaintérpretes

- **Metaintérprete 3**
  - **Programas objetos**

```
es_mamifero :- tiene_pelos.
es_mamifero :- da_leche.
es_ungulado :- es_mamifero, tiene_pezuñas.
es_ungulado :- es_mamifero, rumia.
es_jirafa   :- es_ungulado, tiene_cuello_largo.
es_cebra    :- es_ungulado, tiene_rayas_negras.
tiene_pelos.
tiene_pezuñas.
tiene_rayas_negras.

alumno(A,P) :- estudia(A,C), enseña(P,C).
estudia(ana,ia).
estudia(ana,pl).
estudia(eva,ra).
enseña(jose_a,ia).
enseña(jose_a,ra).
enseña(rafael,pl).
```

  - **Sistema de razonamiento (I)**

```
prueba(true).
prueba((A,L)) :-
        prueba(A),
        prueba(L).
prueba(A) :-
        clause(A,B),
        prueba(B).
```

Metaintérpretes

- **Sesión**

```
?- prueba(alumno(A,jose_a)).
A = ana ;
A = eva ;
A = ana ;
A = eva
Yes

?- prueba(es_cebra).
Yes

?- prueba(es_jirafa).
% Action (h for help) ? a
abort
Execution Aborted
```

- **Sistema de razonamiento (II)**

```
prueba(true) :-
        !.
prueba((A,L)) :-
        !,
        prueba(A),
        prueba(L).
prueba(A) :-
        clause(A,B),
        prueba(B).
```

Metaintérpretes

- **Sesión**

```
?- prueba(alumno(A,jose_a)).
A = ana ;
A = eva ;
No
?- prueba(es_cebra).
Yes
?- prueba(es_jirafa).
No
```

- **Traza**

```
?- trace(prueba).
        prueba/1: call redo exit fail
Yes
?- prueba(alumno(A,jose_a)).
T Call:  (  8) prueba(alumno(_G226, jose_a))
T Call:  (  9) prueba( (estudia(_G226, _G326),
                        enseña(jose_a, _G326)))
T Call:  ( 10) prueba(estudia(_G226, _G326))
T Call:  ( 11) prueba(true)
T Exit:  ( 11) prueba(true)
T Exit:  ( 10) prueba(estudia(ana, ia))
T Call:  ( 10) prueba(enseña(jose_a, ia))
T Call:  ( 11) prueba(true)
T Exit:  ( 11) prueba(true)
T Exit:  ( 10) prueba(enseña(jose_a, ia))
T Exit:  (  9) prueba( (estudia(ana, ia),
                        enseña(jose_a, ia)))
T Exit:  (  8) prueba(alumno(ana, jose_a))
A = ana ;
```

Metaintérpretes

```
T Fail:  ( 10) prueba(enseña(jose_a, ia))
T Call:  ( 11) prueba(true)
T Exit:  ( 11) prueba(true)
T Exit:  ( 10) prueba(estudia(ana, pl))
T Call:  ( 10) prueba(enseña(jose_a, pl))
T Fail:  ( 10) prueba(enseña(jose_a, pl))
T Call:  ( 11) prueba(true)
T Exit:  ( 11) prueba(true)
T Exit:  ( 10) prueba(estudia(eva, ra))
T Call:  ( 10) prueba(enseña(jose_a, ra))
T Call:  ( 11) prueba(true)
T Exit:  ( 11) prueba(true)
T Exit:  ( 10) prueba(enseña(jose_a, ra))
T Exit:  (  9) prueba( (estudia(eva, ra),
                        enseña(jose_a, ra)))
T Exit:  (  8) prueba(alumno(eva, jose_a))
A = eva ;
No
```