

Tema 1: Introducción a Prolog

José A. Alonso Jiménez
Miguel A. Gutiérrez Naranjo

Dpto. de Ciencias de la Computación e Inteligencia Artificial

UNIVERSIDAD DE SEVILLA

Objetivos del curso

- Lógica como especificación y lenguaje de programación
- Prolog = Programming in Logic
- Relaciones con otros campos:
 - Bases de datos
 - Sistemas basados en el conocimiento
 - Procesamiento del lenguaje natural
 - Inteligencia artificial
- Pensar declarativamente

Declarativo vs. imperativo

- Paradigmas
 - Imperativo: Se describe *cómo* resolver el problema
 - Declarativo: Se describe *qué* es el problema
- Programas
 - Imperativo: Una sucesión de instrucciones
 - Declarativo: Un conjunto de sentencias
- Lenguajes
 - Imperativo: Pascal, C, Fortran
 - Declarativo: Prolog, Lisp puro, ML, Haskell
- Ventajas
 - Imperativo: Programas rápidos y especializados
 - Declarativo: Programas generales, cortos y legibles

Historia

- -350: Grecia clásica (Aristóteles,...)
- 1930: Edad de oro de la lógica (Gödel)
- 1960: Demostración automática de teoremas
- 1965: Resolución y unificación (Robinson)
- 1969: QA3, obtención de respuesta (Green)
- 1972: Implementación de Prolog (Colmerauer)
- 1974: Programación lógica (Kowalski)
- 1977: Prolog de Edimburgo (Warren)
- 1981: Proyecto japonés de Quinta Generación
- 1986: Programación lógica con restricciones
- 1995: Estándar ISO de Prolog

Un ejemplo simple: divisibilidad

- **Problema:** Escribir un programa para declarar que 2 divide a 6 y utilizarlo para responder a las siguientes cuestiones:
 - ¿2 divide a 6?.
 - ¿3 divide a 12?.
 - ¿Cuáles son los múltiplos de 2?.
 - ¿Cuáles son los divisores de 6?.
 - ¿Cuáles son los elementos X e Y tales que X divide a Y?.
- **Programa:** divisibilidad-1.pl
divide(2,6).
- **Sesión**
?- divide(2,6).
Yes
?- divide(3,12).
No
?- divide(2,X).
X = 6
Yes
?- divide(X,Y).
X=2
Y=6
Yes

Un ejemplo simple: divisibilidad

- **Conceptos**
 - Relación: divide/2
 - Nombre de una relación: divide
 - Argumentos de una relación
 - Hecho
 - Programa
 - Constante: 2, 6
 - Variable: X, Y
 - Objetivo
 - Objetivo satisfacible
 - Respuesta

Unificación

- Ejemplos de unificación

$\text{divide}(2,6)$ $\text{divide}(2,X)$ $\{X/6\}$ $\text{divide}(2,6)$

$\text{divide}(X,6)$ $\text{divide}(2,Y)$ $\{X/2, Y/6\}$ $\text{divide}(2,6)$

$\text{divide}(X,6)$ $\text{divide}(2,X)$ No unif.

$\text{divide}(2,6)$ $\text{divide}(X,X)$ No unif.

$\text{divide}(X,6)$ $\text{divide}(Y,Z)$ $\{X/4, Y/4, Z/6\}$ $\text{divide}(4,6)$

$\text{divide}(X,6)$ $\text{divide}(Y,Z)$ $\{X/Y, Z/6\}$ $\text{divide}(4,6)$

- Unificador de máxima generalidad (u.m.g.)

Ampliación del programa

- **Problema:** Ampliar el programa anterior, añadiéndole que 2 divide a 12 y que 3 divide a 6 y a 12 y utilizarlo para responder a las siguientes cuestiones:
 - ¿Cuáles son los elementos X e Y tales que X divide a Y?
 - ¿Cuáles son los múltiplos de 2 y de 3?
- **Programa:** divisibilidad-2.pl

```
divide(2,6).  
divide(2,12).  
divide(3,6).  
divide(3,12).
```

- **Sesión**

```
?- divide(X,Y).  
X = 2    Y = 6 ;  
X = 2    Y = 12 ;  
X = 3    Y = 6 ;  
X = 3    Y = 12 ;  
No  
?- divide(2,X), divide(3,X).  
X = 6 ;  
X = 12 ;  
No
```


Ampliación del programa

- Conceptos:
 - Pregunta cerrada
 - Pregunta abierta
 - Pregunta compuesta
 - Respuesta
 - Respuestas múltiples
 - Literal seleccionado

Reglas

- **Problema:** Ampliar el programa anterior añadiéndole que los números divisibles por 2 y por 3 son divisibles por 6 y utilizarlo para responder a las siguientes cuestiones:

- ¿Cuáles son los múltiplos de 6?
- ¿Cuáles son los elementos X e Y tales que X divide a Y?

- **Programa:** divisibilidad-3.pl

```
divide(2,6).
divide(2,12).
divide(3,6).
divide(3,12).
divide(6,X) :-
    divide(2,X),
    divide(3,X).
```

- **Interpretación de cláusulas**

- **Cláusula:**

`divide(6,X) :- divide(2,X), divide(3,X).`

- **Fórmula:**

$(\forall X)[\textit{divide}(2, X) \wedge \textit{divide}(3, X) \rightarrow \textit{divide}(6, X)]$

- Interpretación declarativa
- Interpretación procedimental

Reglas

- **Sesión**

?- divide(6,X).

X = 6 ;

X = 12 ;

No

?- divide(X,Y).

X = 2 Y = 6 ;

X = 2 Y = 12 ;

X = 3 Y = 6 ;

X = 3 Y = 12 ;

X = 6 Y = 6 ;

X = 6 Y = 12 ;

No

- **Conceptos:**

- Cláusulas: hechos, reglas y preguntas
- Reglas: cabeza y cuerpo

Resolución en lógica proposicional

- Programa: leche.pl

```
es_leche :-  
    parece_leche,  
    lo_da_la_vaca.  
parece_leche :-  
    es_blanco,  
    hay_una_vaca_en_la_etiqueta.  
lo_da_la_vaca.  
es_blanco.  
hay_una_vaca_en_la_etiqueta.
```

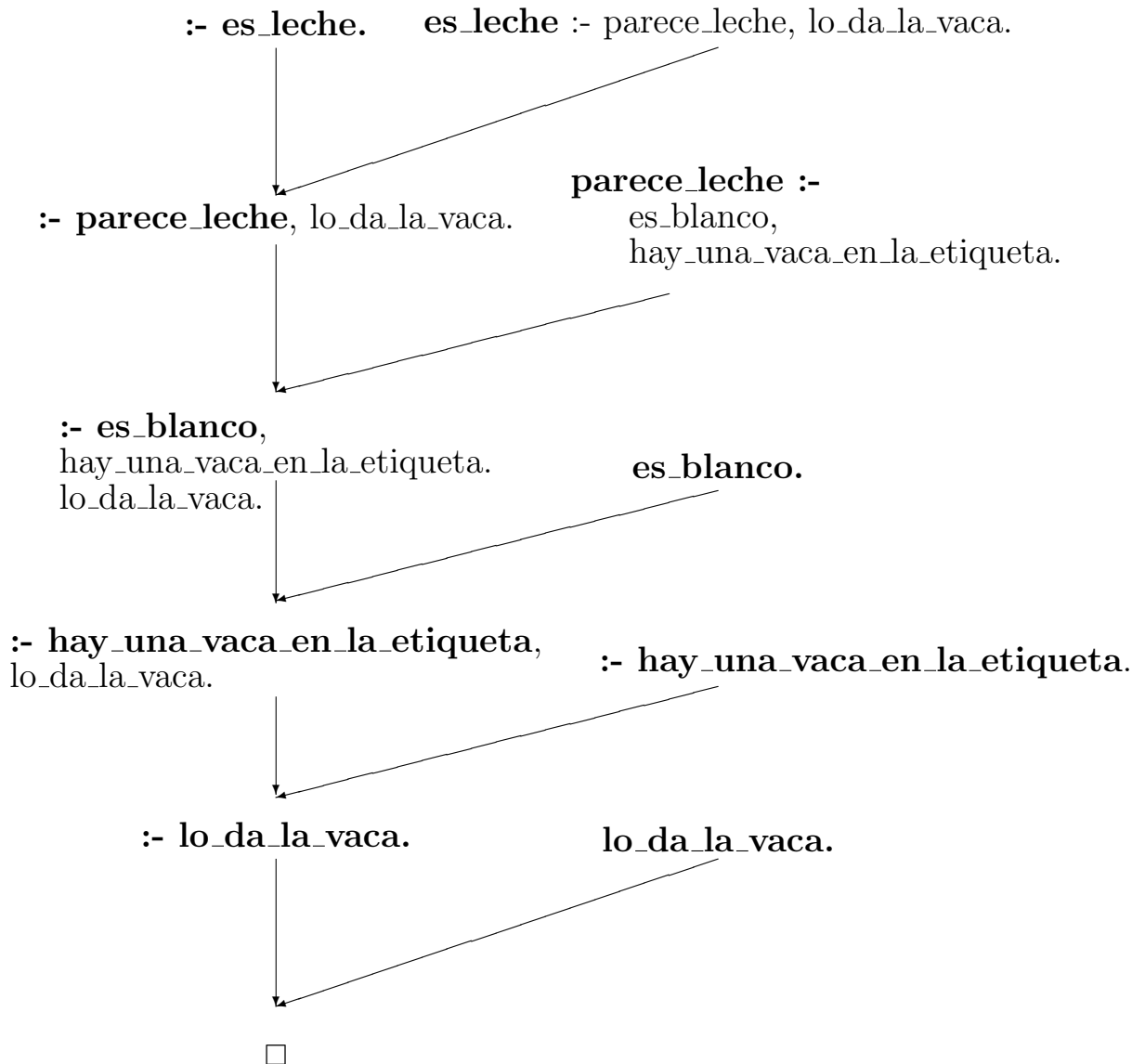
- Sesión

```
?- es_leche.  
yes
```

- Traza

```
(1) 0 CALL es_leche?  
(2) 1 CALL parece_leche?  
(3) 2 CALL es_blanco?  
(3) 2 EXIT es_blanco  
(4) 2 CALL hay_una_vaca_en_la_etiqueta?  
(4) 2 EXIT hay_una_vaca_en_la_etiqueta  
(2) 1 EXIT parece_leche  
(5) 1 CALL lo_da_la_vaca?  
(5) 1 EXIT lo_da_la_vaca  
(1) 0 EXIT es_leche
```

Demostración SLD



- SLD:
 - S: regla de Selección
 - L: resolución Lineal
 - D: cláusulas Definidas

Traza

- Problema: Utilizar el programa anterior para calcular los divisores de 6 con el dispositivo trace y construir el árbol de deducción.

```
?- trace(divide).
```

```
    divide/2: call redo exit fail
```

```
Yes
```

```
?- divide(6,X).
```

```
T Call: ( 7) divide(6, _G260)
```

```
T Call: ( 8) divide(2, _G260)
```

```
T Exit: ( 8) divide(2, 6)
```

```
T Call: ( 8) divide(3, 6)
```

```
T Exit: ( 8) divide(3, 6)
```

```
T Exit: ( 7) divide(6, 6)
```

```
X = 6 ;
```

```
T Redo: ( 8) divide(3, 6)
```

```
T Fail: ( 8) divide(3, 6)
```

```
T Redo: ( 8) divide(2, _G260)
```

```
T Exit: ( 8) divide(2, 12)
```

```
T Call: ( 8) divide(3, 12)
```

```
T Exit: ( 8) divide(3, 12)
```

```
T Exit: ( 7) divide(6, 12)
```

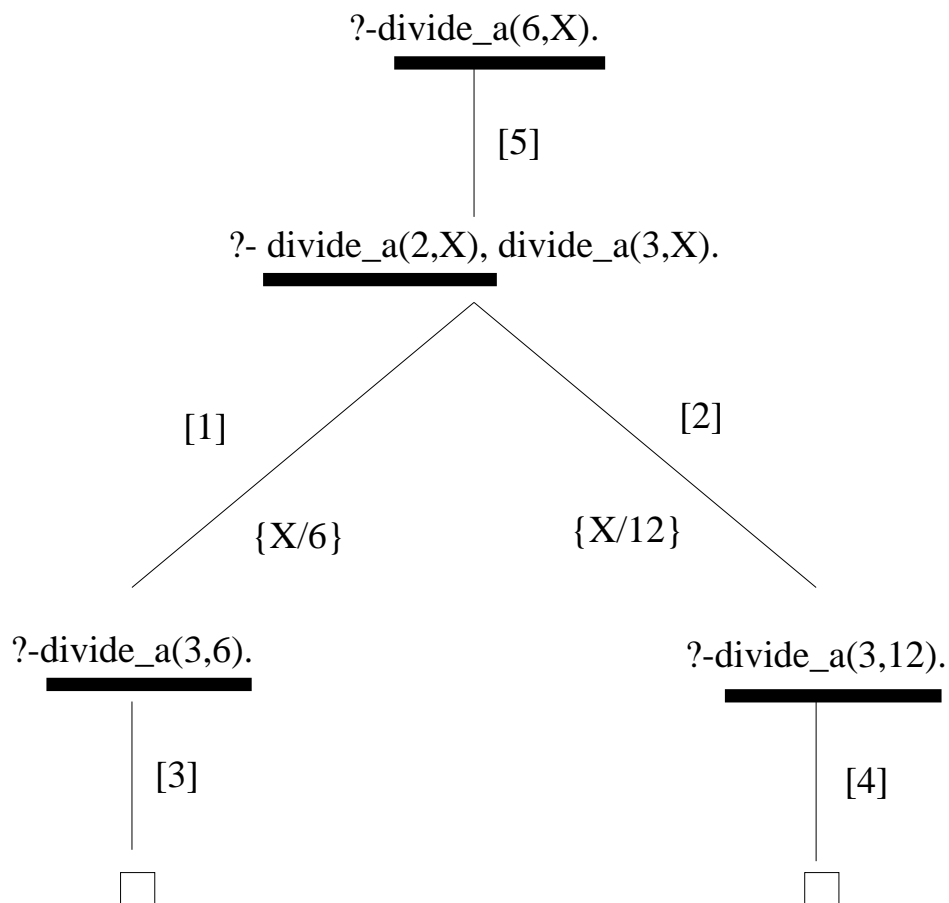
```
X = 12 ;
```

```
No
```

Traza

● Arbol de resolución SLD

```
[1]    divide(2,6).  
[2]    divide(2,12).  
[3]    divide(3,6).  
[4]    divide(3,12).  
[5]    divide(6,X) :-  
        divide(2,X),  
        divide(3,X).
```



Reglas recursivas: naturales.pl

- **Problema:** Los números naturales se forman a partir del cero y la función sucesor. De forma más precisa:
 - * El 0 es un número natural
 - * Si n es un número natural, $s(n)$ también lo esEscribir un programa para decidir si una expresión es un número natural y utilizarlo para responder a las siguientes cuestiones:
 - ¿Es $s(s(0))$ un número natural?
 - ¿Es 2 un número natural?
 - ¿Cuáles son los números naturales?
- **Programa:** naturales.pl

```
nat(0).  
nat(s(X)) :-  
    nat(X).
```


Reglas recursivas: naturales.pl

- **Sesión**

```
?- nat(s(s(0))).
```

Yes

```
?- nat(dos).
```

No

```
?- nat(X).
```

```
X = 0 ;
```

```
X = s(0) ;
```

```
X = s(s(0)) ;
```

```
X = s(s(s(0))) ;
```

```
X = s(s(s(s(0))))
```

Yes

- **Conceptos:**

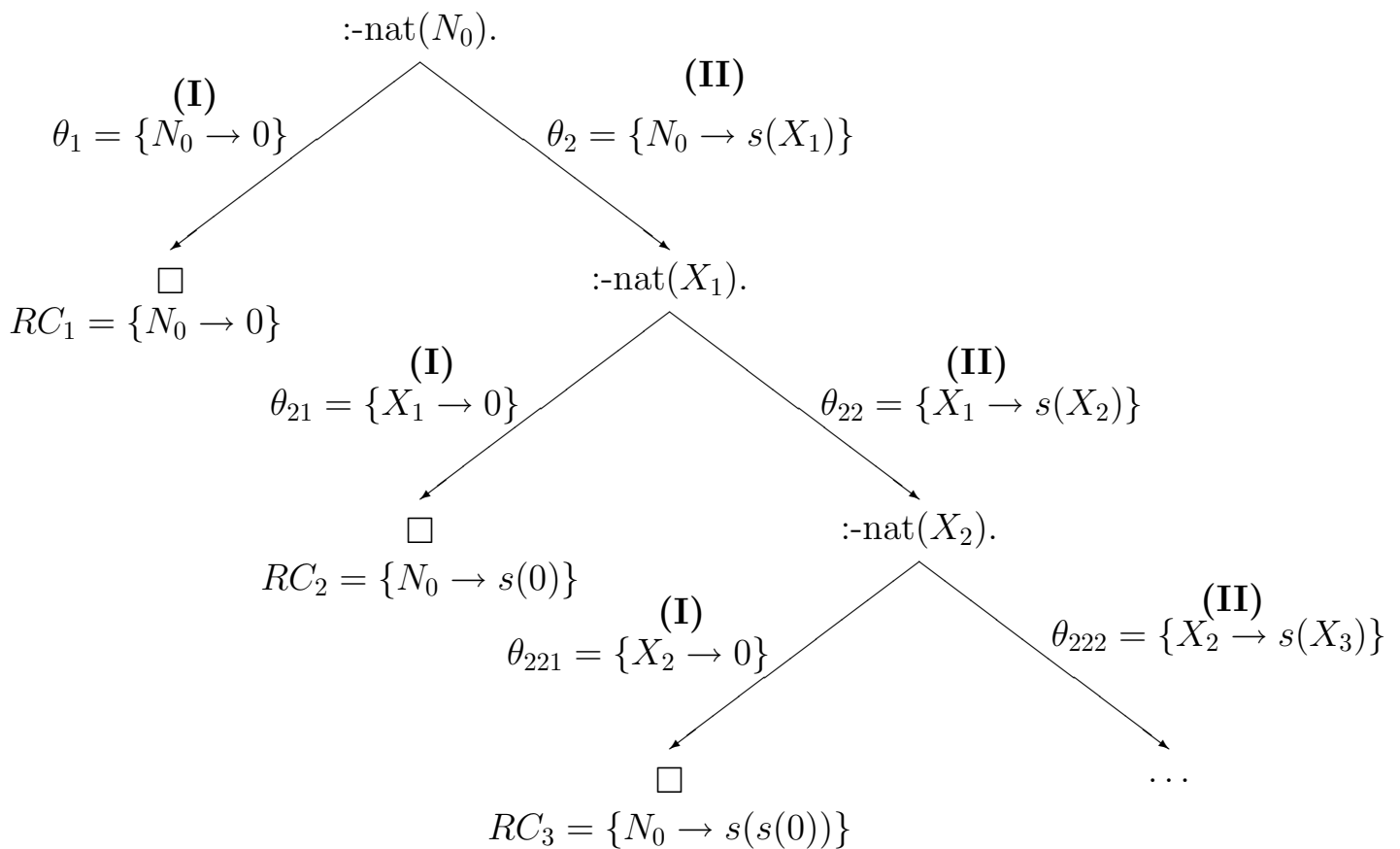
- Regla recursiva
- Símbolos de función
- Términos
- Respuestas infinitas

Reglas recursivas: naturales.pl

● Arbol de resolución SLD

(I) $\text{nat}(0).$

(II) $\text{nat}(s(X)):-\text{nat}(X).$



Reglas recursivas: suma.pl

- **Problema:** Definir el predicado `suma(X,Y,X)` de forma que si X e Y son dos números naturales con la representación del programa `naturales.pl`, entonces Z es el resultado de sumar X e Y . Por ejemplo,

$$\text{suma}(s(0),s(s(0)),X) \Rightarrow X=s(s(s(0)))$$

Utilizarlo para responder a las siguientes cuestiones:

- ¿Cuál es la suma de $s(0)$ y $s(s(0))$?
 - ¿Cuál es la resta de $s(s(s(0)))$ y $s(0)$?
 - ¿Cuáles son las soluciones de la ecuación $X + Y = s(s(0))$?
- **Programa:** `suma.pl`

```
suma(0,X,X).
```

```
suma(s(X),Y,s(Z)) :- suma(X,Y,Z).
```

Reglas recursivas: suma.pl

● Sesión

?- suma(s(0),s(s(0)),X).

X = s(s(s(0)))

Yes

?- suma(X,s(0),s(s(s(0)))).

X = s(s(0))

Yes

?- suma(X,Y,s(s(0))).

X = 0

Y = s(s(0)) ;

X = s(0)

Y = s(0) ;

X = s(s(0))

Y = 0 ;

No

Bibliografía

- Bratko, I. *Prolog Programming for Artificial Intelligence (2nd ed.)* (Addison–Wesley, 1990)
 - Cap. 1: “An overview of Prolog”
 - Cap. 2: “Syntax and meaning of Prolog programs”
- Clocksin, W.F. y Mellish, C.S. *Programming in Prolog (Fourth Edition)* (Springer Verlag, 1994)
 - Cap. 1: “Tutorial introduction”
 - Cap. 2: “A closer look”
- Shapiro, S.C. *Encyclopedia of Artificial Intelligence* (John Wiley, 1990)
 - “Logic programming” (por R.A. Kowalski y C.J. Hogger)
- Van Le, T. *Techniques of Prolog Programming* (John Wiley, 1993)
 - Cap. 1: “Introduction to Prolog”.