Introducción a la Programación Lógica

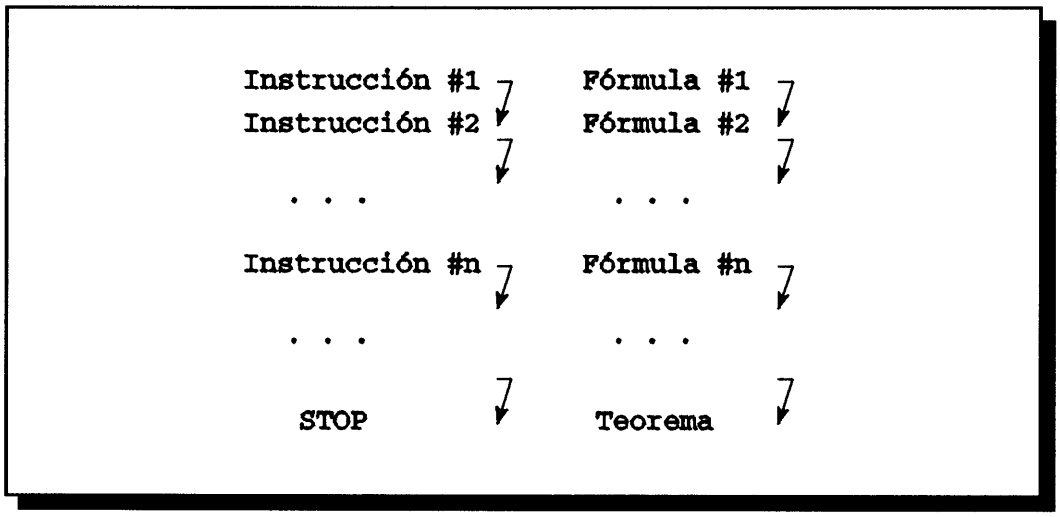
- Revisión histórica, ideas centrales, Prolog.
- Aspectos centrales de un lenguaje de programación lógica.
- Rasgos distintivos de Prolog respecto de otros lenguajes de programación.

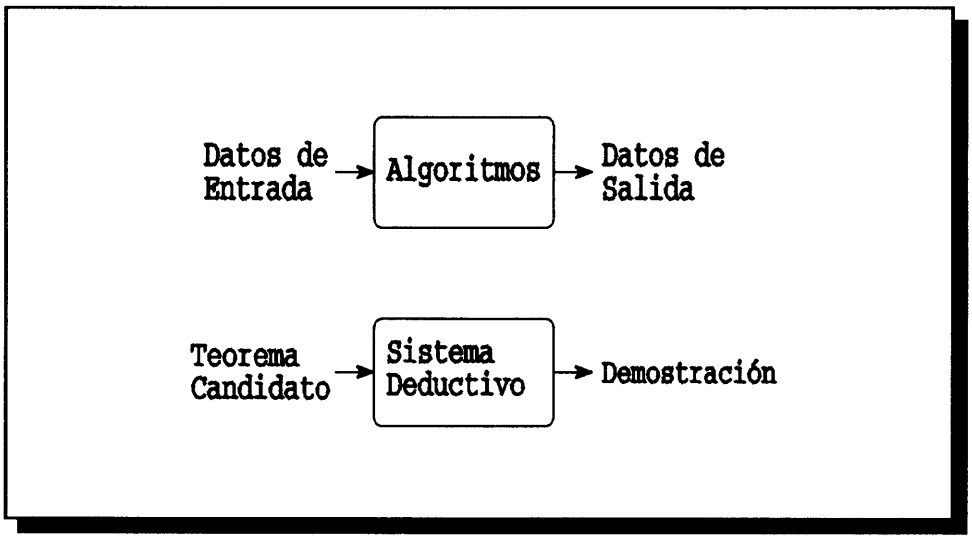
Orígenes históricos

- Comienzos de los años 70, R. Kowalski y A. Colmerauer
- Dos usos de la Lógica:
  - representación rigurosa del conocimiento
  - demostración como algoritmo: Algoritmo = Lógica + Control
- Resultados anteriores de Herbrand, Horn y Robinson (demostración automática de teoremas)

Programa = Algoritmos + Datos

Programa = Reglas + Teoría





- Desarrollo de implementaciones eficientes de Prolog a partir de la de Warren
- Normalización de Prolog: ISO 95
- Prolog como lenguaje de proposito general: Prolog puro e impuro.
- Constraint Logic Programming, PARLOG, Datalog, ...

EJEMPLO NO. 1

Conocimiento a representar:

1. Todos los hombres son mortales.
2. Sócrates es un hombre.

A deducir:

Sócrates es mortal.

Representación PROLOG:

```
mortal(X):-
        hombre(X).

hombre(socrates).

?- mortal(socrates).
```

Representación LP1:

```
∀x(Hx -> Mx)

Ha

⊢ Ma
```

EJEMPLO NO. 2

Conocimiento a representar:

1. El Sol es una estrella.
2. Mercurio orbita en torno al Sol.
3. Venus orbita en torno al Sol.
4. La Luna orbita en torno a la Tierra.
5. La Tierra orbita en torno al Sol.
6. Marte orbita en torno al Sol.
7. Fobos orbita en torno a Marte.
8. Deimos orbita en torno a Marte.
9. Son planetas aquellos cuerpos que orbitan en torno a una estrella.
10. Todo cuerpo que orbita en torno a un planeta es un satélite.
11. Pertenecen a un sistema solar la estrella misma y todo cuerpo que orbite en torno a otro que pertenezca.

Representación PROLOG:

```
1. estrella(sol).
2. orbita(mercurio,sol).
3. orbita(venus,sol).
4. orbita(luna,tierra).
5. orbita(tierra,sol).
6. orbita(marte,sol).
7. orbita(fobos,marte).
8. orbita(deimos,marte).

9. planeta(X):-
        orbita(X,Y),
        estrella(Y).

10. satelite(X):-
        orbita(X,Y),
        planeta(Y).

11. pertenece(X,X):-
        estrella(X).

  pertenece(X,Y):-
        orbita(X,Z),
        pertenece(Z,Y).
```

Representación LP1:

```
1. Ea     (a : Sol)
2. Oba    (b : Mercurio)
3. Oca    (c : Venus)
4. Ode    (d : Luna)
5. Oea    (e : Tierra)
6. Ofa    (f : Marte)
7. Ogf    (g : fobos)
8. Ohf    (h : deimos)

9. ∀xy(Oxy & Ey -> Px)

10. ∀xy(Oxy & Py-> Sx)

11. ∀x(Ex -> Txx)

    ∀xyz(Oxz & Tzy -> Txy)
```

EJEMPLO NO. 3

Conocimiento a representar:

1. '0' es el numeral que representa al numero *cero*.
2. La expresión 's(n)' es el numeral del numero *sucesor* al representado por la expresión 'n'.
3. Cualquier numero *sumado* a cero es igual a ese mismo numero.
4. La *suma* de dos numeros 'n' y 's(m)' es igual a 's(j)', siendo 'j' el resultado de sumar 'n' y 'm'. [n + s(m) = s(n+m)]

Representación Prolog:

```
1. numero(0).
2. numero(s(N)):-
        numero(N).

3. suma(N,0,N):-
        numero(N).
4. suma(N,s(M),s(J)):-
        suma(N,M,J).
```

Representación LP1:

```
1. Na     (a: cero)
2. ∀x(Nx -> Ns(x))

3. ∀x(Nx -> Sxax)

4. ∀xyz(Sxyz -> Sxs(y)s(z))
```

Algunos usos de la teoría:

```
?- numero(s(s(0))).
yes.

?- numero(X).
X = 0 ->
X = s(0) ->
X = s(s(0) ->
...

?- numero(2).
no.
```

```
?- suma(0,0,0).
yes.

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

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

?- suma(X,Y,s(0)).
X = s(0), Y = 0 ->
X = 0, Y = s(0) ->
no.
```

EJEMPLO NO. 4

```
mortal(X):- hombre(X).
```
hombre(X) -> mortal(X)

```
?- mortal(socrates).
```
mortal(socrates) -> □

{X/socrates}

```
?- hombre(socrates).
```
hombre(socrates) -> □

```
hombre(socrates).
```
■ -> hombre(socrates)

```
?- hombre(socrates).
```
hombre(socrates) -> □

{ }

```
yes.
```
■ -> □

Dada la teoría (programa) T :

{ hombre(X)-> mortal(X), ■ -> hombre(socrates) }

y el candidato a teorema (objetivo) C :

mortal(socrates)-> □

ha sido demostrado que T ∪ { C } ├ ■ -> □

Luego T ├ ¬C

¬(mortal(socrates)-> □)
mortal(socrates) ∧ ¬□
mortal(socrates) ∧ ■

mortal(socrates)

Términos

CONSTANTES

Nombran individuos (objetos) de la teoría:

```
12      marte      'Alejandro Iglesias'
```

VARIABLES

Toman como valor cualquier expresión:

```
X      Valor      A1
```

Una *sustitución* θ es un conjunto finito de pares:

$$\{ v_1/t_1, v_2/t_2, \ldots, v_n/t_n \}$$

donde cada v_i es una variable distinta del resto, y cada t_i es un término en el que no aparece v_i.

EXPRESIONES FUNCIONALES

Nombran objetos mediante la composición de otros más simples

```
3+2   padre('Alejandro Iglesias')   derivada(2*x^2 + 3*x)
```

En general, siendo 'f' un símbolo funcional n-ádico y siendo $t_1 \ldots t_n$ términos, la expresión:

$$f(t_1,\ldots,t_n)$$

es un término.

Cláusulas

Fórmulas compuestas a partir de fórmulas atómicas según el esquema:

$$\forall x_1 \ldots x_m \; ((A_1 \wedge \ldots \wedge A_n) \rightarrow B)$$

donde B, A_1 ... A_n son fórmulas atómicas y $x_1 \ldots x_m$ son <u>todas</u> las variables que aparecen en ellas.

REGLAS

$$A_1 \wedge \ldots \wedge A_n \rightarrow B$$

B :- A_1 , ... , A_n.

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

HECHOS

$$\blacksquare \rightarrow B$$

```
planeta(tierra).    divisible(N,1).    casados(bill,hilary).
```

OBJETIVOS

$$A_1 \wedge \ldots \wedge A_n \rightarrow \square$$

```
:- divisible(3,1).          ?- planeta(X).
```

Fórmulas atómicas

Reciben un valor de verdad

```
mayor(3,2)      casados(bill,hilary)      planeta(marte)
```

En general, siendo 'p' un símbolo predicativo n-ádico y siendo $t_1 \ldots t_n$ términos, la expresión:

$$p(t_1,\ldots,t_n)$$

es una fórmula atómica. Conviene incluir tambien como fórmulas atómicas los signos ■ y □, que reciben siempre los valores verdadero falso respectivamente.

Unificación

Dos fórmulas atómicas A y B unifican si y sólo si hay una sustitución θ tal que $(A)\theta$ es idéntica a $(B)\theta$.

Ejemplos:

p(X) unifica con p(a) bajo θ = { X/a } : $(p(X))\theta$ es idéntico a $(p(a))\theta$.

p(a) no unifica con p(b)

p(X,a) unifica con p(b,Y) bajo θ = { X/b, Y/a }

p(X) unifica con p(Y) bajo θ = { X/Y }
bajo θ' = { X/Y }
bajo θ'' = { X/a, Y/a }
bajo θ''' = { X/b, Y/b }
. . .

Regla de Resolución

Unica regla deductiva empleada en una demostración / computación. En su forma general establece:

$$\frac{A_1 \wedge \ldots \wedge A_n \rightarrow B \qquad B_1 \wedge \ldots \wedge B_{i-1} \wedge B_i \wedge B_{i+1} \wedge \ldots B_m \rightarrow \square}{(B)\theta = (B_i)\theta \qquad (B_1 \wedge \ldots \wedge B_{i-1} \wedge A_1 \wedge \ldots \wedge A_n \wedge B_{i+1} \wedge \ldots B_m)\theta \rightarrow \square}$$

tal y como la aplica Prolog:

$$\frac{A_1 \wedge \ldots \wedge A_n \rightarrow B \qquad B_1 \wedge B_2 \wedge \ldots \wedge B_m \rightarrow \square}{(B)\theta = (B_1)\theta \qquad (A_1 \wedge \ldots \wedge A_n \wedge B_2 \wedge \ldots B_m)\theta \rightarrow \square}$$

casos particulares son:

$$\frac{\blacksquare \rightarrow B \qquad B_1 \wedge B_2 \wedge \ldots \wedge B_m \rightarrow \square}{(B)\theta = (B_1)\theta \qquad (B_2 \wedge \ldots B_m)\theta \rightarrow \square}$$

$$\frac{\blacksquare \rightarrow B \qquad B_1 \rightarrow \square}{(B)\theta = (B_1)\theta \qquad \blacksquare \rightarrow \square}$$

EJEMPLO NO. 5

```
planeta(X):-orbita(X,Y),estrella(Y).
```
orbita(X,Y) ∧ estrella(Y) -> planeta(X)

```
?- planeta(P).
```
planeta(P) -> □

{X/P}

```
  ?- orbita(P,Y),estrella(Y).
```
orbita(P,Y) ∧ estella(Y) -> □

```
orbita(tierra,sol).
```
■ -> orbita(tierra,sol)

```
?- orbita(P,Y),estrella(Y).
```
orbita(P,Y) ∧\estrella(Y) -> □

{P/tierra,Y/sol}

```
  ?- estrella(sol).
```
■ ∧ estrella(sol) -> □

```
  estrella(sol).
```
■ -> estrella(sol)

```
  ?- estrella(sol).
```
estrella(sol) -> □

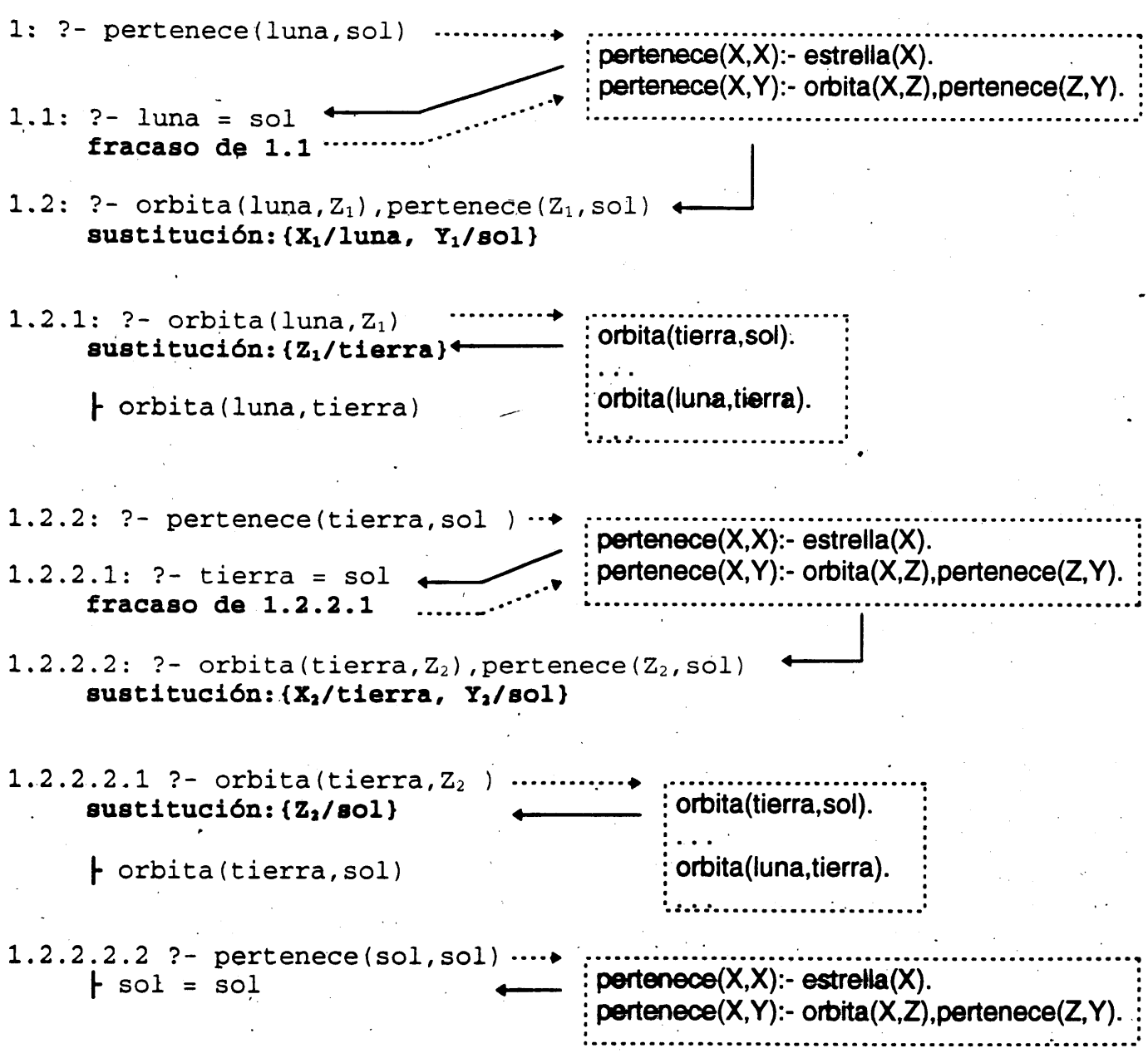
{ }

```
  yes.
  P = tierra
```
■ -> □

T = { orbita(X,Y) ∧ estrella(Y) -> planeta(X), ■ -> orbita(tierra,sol), ■ -> estrella(sol) }

T ∪ { planeta(P) -> □ } ├ ■ -> □

T ├ ¬ ∀P (planeta(P) -> □)
T ├ ∃P¬(planeta(P) -> □)
T ├ ∃P(planeta(P) ∧ ¬□)
T ├ ∃P planeta(P)



Si ├ estrella(sol) **entonces** ├ pertenece(sol,sol)

Si ├ orbita(tierra,sol) **y** ├ pertenece(sol,sol)
entonces ├ pertenece(tierra,sol)

Si ├ orbita(luna,tierra) **y** ├ pertenece(tierra,sol)
entonces ├ pertenece(luna,sol)

Control

Algoritmo = Lógica + Control: exploración del árbol de prueba.

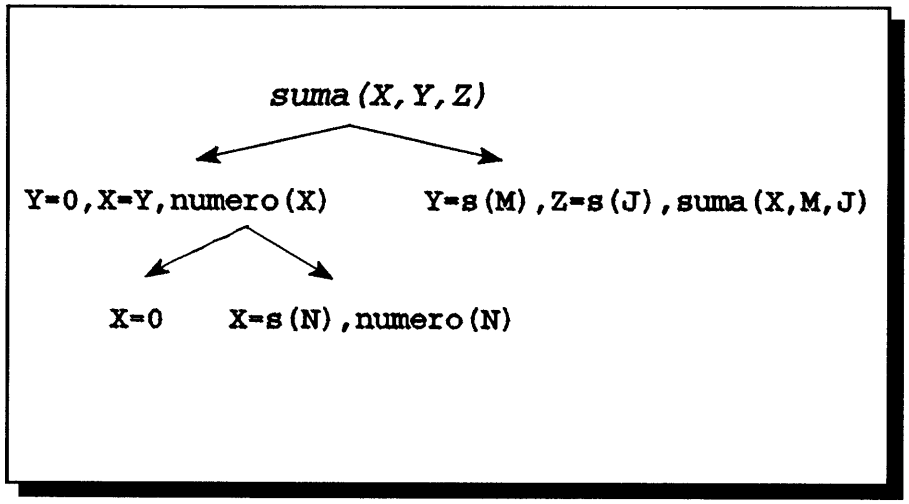
```
numero(0).

numero(s(N)):-
     numero(N).

suma(N,0,N):-
     numero(N).

suma(N,s(M),s(J)):-
     suma(N,M,J).

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



- Convenciones arriba/abajo e izquierda/derecha.
- Busqueda en profundidad (extensión) por la izquierda.
- Backtraking

Computación Lógica

Dado un programa lógico P y un objetivo G, la computación lógica de G en P es una secuencia de aplicaciones de la regla de resolución, inicialmente sobre G y una de las cláusulas de P y continuando sobre los sucesivos G_i resultantes de aplicar la regla sobre el objetivo anterior G_{i-1}.

Una computación *finaliza* cuando:

1. la regla de resolución rinde □ (■ –> □). En este caso tiene éxito en la *refutación* del objetivo inicialmente propuesto.

Si $P \cup \{G\} \vdash \square$ entonces $P \vdash \neg G$.

2. no puede aplicarse la regla de resolución entre ninguna de las cláusulas de P y el resultado de la anterior aplicación. En este caso *fracasa* el intento de refutar el objetivo inicial.

Si $P \cup \{G\} \not\vdash \square$ entonces $P \not\vdash \neg G$.

No dándose 1 ni 2, la computación prosigue indefinidamente.

Validez y Completud

Aplicada a cláusulas de Horn, la Regla de Resolución produce siempre y únicamente resultados correctos, por lo que es una regla *válida*.

Además, todo objetivo lógicamente deducible de un programa lógico puede serlo aplicando la Regla de Resolución, por lo que es una regla *completa*.

Rasgos distintivos de Prolog

- La "crisis del software"
  - incremento de los costes de desarrollo del *software*
  - falta de fiabilidad

  Dos aproximaciones al problema: POO y PL.
- Simplicidad sintáctica.
- Simplicidad en su ejecución, resolución como única regla.
- Capacidad simbólica: el término como única estructura de datos.
- recursivos.
- (Casi) total ausencia de instrucciones de control.
- Multiuso
- Meta-programación
- Programación dinámica
- Paralelización transparente.

Limitaciones

- Mecanismo de control del árbol de prueba.
- Uso del procedimiento de corte.
- Negación.
- Efectos laterales no cancelables bajo backtraking.
- Carencia de aritmética multiuso.