

# Introducción a Prolog

# El Lenguaje Prolog

- Idea general:
  - Se describe una situación: Se listan una serie de hechos, relaciones.
  - El compilador (o intérprete) responde preguntas/consultas (queries) con base en estos datos.
- Los hechos y las reglas conforman la "base de datos" del entorno de prolog.

# Compiladores/Intérpretes de Prolog

- SWI-Prolog: Versión recomendada para los ejercicios. Disponible en

<http://www.swi-prolog.org/>

- Una lista muy completa de versiones gratuitas para varias plataformas se encuentra aquí:

<http://www.thefreecountry.com/compilers/prolog.shtml>

# Filosofia General

- Es un lenguaje *declarativo* (Lenguajes como C, Pascal y Java son procedimentales).
- Imperativo/Procedimental: Serie de comandos
- Declarativo: Se definen las condiciones lógicas que debe satisfacer la respuesta. El compilador se encarga de buscarla.
- El nombre Prolog resulta de la contracción "**P**rogramming in **L**ogic"

## Información adicional:

Una descripción muy completa de los muchos paradigmas de programación existentes:  
[http://en.wikipedia.org/wiki/Comparison\\_of\\_programming\\_paradigms](http://en.wikipedia.org/wiki/Comparison_of_programming_paradigms)

# Un paréntesis: Paradigmas de programación



# Base de datos

Compuesta de hechos y reglas.

- **Los hechos** son las premisas básicas. Expresan un dato verdadero para el entorno. Ejemplos de hechos son
  - El león es un mamífero
  - Maria es la madre de Juan
  - Bogotá es la capital de Colombia
- **Las reglas** en prolog son el análogo de las reglas de inferencia en lógica. Una regla expresa las condiciones bajo las cuales una conclusión es verdadera.

# Consultas

- Una consulta es una pregunta a resolver a partir de los hechos y las reglas existentes en la base de datos. Por ejemplo,
  - Quién es hijo de Maria?
  - Cuál es la capital de Colombia?

# Sintaxis

- Los hechos, reglas y consultas se expresan como *cláusulas*.
- Un programa es una colección de *cláusulas*.
- Las *cláusulas* se componen de *fórmulas atómicas*: Un predicado seguido por una lista de argumentos.
- Todas las cláusulas se terminan con un punto al final.



# Sintaxis - Hechos

- Así es la sintaxis de los hechos:  
    mamifero(león).  
    capital(colombia, bogotá).  
    madre(juan, maria).
- Todo lo que no haya sido definido como un hecho es una falsedad.

# Creación de la base de datos

- El programa se salva en un archivo de texto plano, por ejemplo

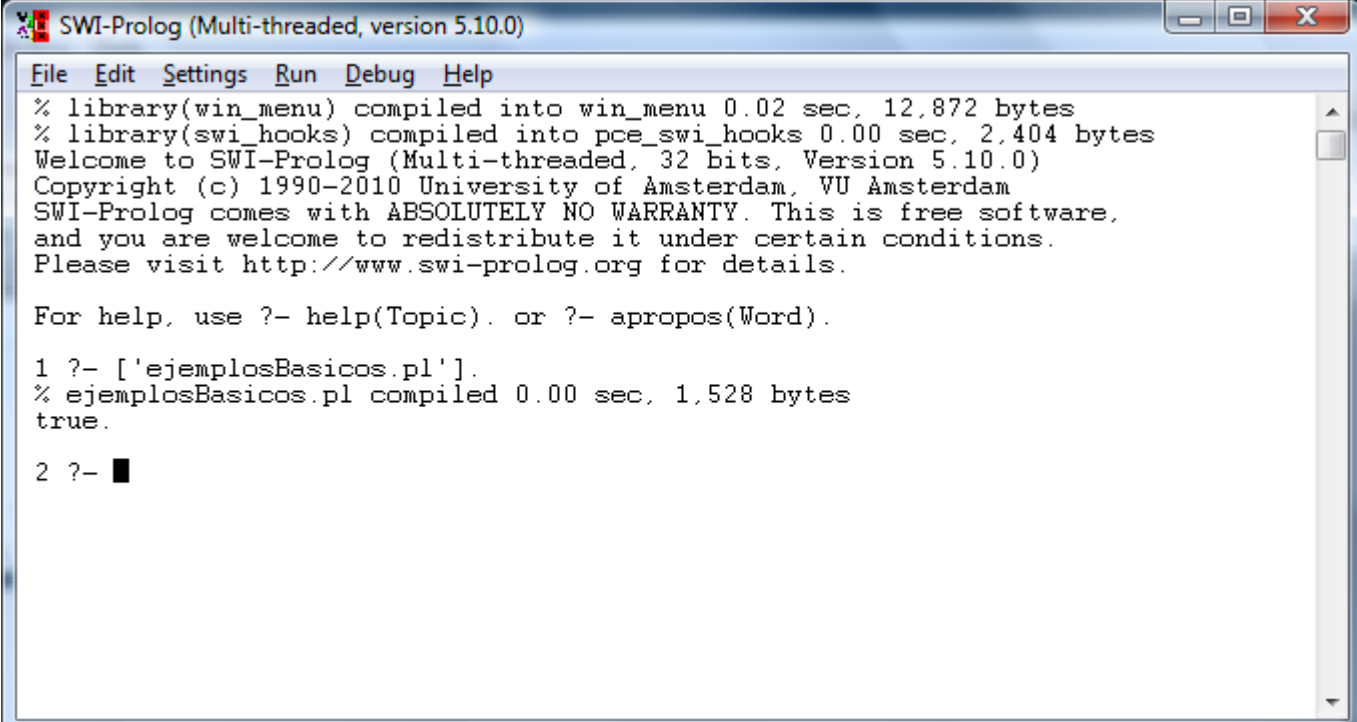
**ejemplosBasicos.pl**

```
mamifero(león).  
capital(colombia, bogotá).  
madre(juan, maria).
```

# Carga de la base de datos

- El entorno "lee y compila" la base de datos mediante el comando consultar, que tiene la sintaxis: `[ 'nombreArchivo' ]`.

**Ejemplo:**



```
SWI-Prolog (Multi-threaded, version 5.10.0)
File Edit Settings Run Debug Help
% library(win_menu) compiled into win_menu 0.02 sec, 12,872 bytes
% library(swi_hooks) compiled into pce_swi_hooks 0.00 sec, 2,404 bytes
Welcome to SWI-Prolog (Multi-threaded, 32 bits, Version 5.10.0)
Copyright (c) 1990-2010 University of Amsterdam, VU Amsterdam
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to redistribute it under certain conditions.
Please visit http://www.swi-prolog.org for details.

For help, use ?- help(Topic). or ?- apropos(Word).

1 ?- ['ejemplosBasicos.pl'].
% ejemplosBasicos.pl compiled 0.00 sec, 1,528 bytes
true.

2 ?- ■
```

# Sintaxis – Consultas simples

- Se puede verificar si un hecho aparece en la base de datos (es cierto) simplemente escribiendo la cláusula correspondiente en la interfaz de consultas:

```
11 ?- mamifero(león).  
true.  
12 ?- madre(juan, maria).  
true.  
13 ?- capital(colombia, bogotá).  
true.  
14 ?- mamifero(perro).  
false.  
15 ?- capital(peru, bolivia).  
false.
```

# Sintaxis - Hechos

**Nombre de predicado:** Comienzan en minúscula y van seguidos por la lista de argumentos.

**Términos:** Argumentos, que pueden ser átomos, números, variables o estructuras.



mamifero(león).

# Átomos

Un átomo identifica un miembro del dominio.

Empiezan por letra minúscula:

- león, maria, juan, bogotá, colombia

Alternativamente, para cadenas alfanuméricas arbitrarias (con espacios, caracteres especiales, o que empiezan por mayúscula), se pueden definir átomos encerrando la cadena entre comillas simples:

- 'Albert Einstein'

El predicado `atom/1` determina si el argumento es un átomo:

- `atom(león).`
- `atom('María Rodríguez').`

# Variables

- Empiezan por letra mayúscula, por ejemplo

X

Y

País

Animal

# Comentarios

- Todo lo escrito a continuación del signo '%' hasta el final de línea es un comentario

```
% Base de datos ejemplo  
mamifero(león). % Sobre el león  
  
% Los paises y sus capitales  
capital(colombia, bogotá).  
  
% La madre de juan es maria  
madre(juan, maria).
```



# Consultas (2)

- Cuando aparece una variable en una consulta, Prolog busca términos que hagan verdadera la cláusula. Por ejemplo:

```
1 ?- ['ejemplosBasicos.pl'].  
% ejemplosBasicos.pl compiled 0.00 sec,  
1,528 bytes  
true.
```

```
2 ?- madre(juan,X).  
X = maria.
```

```
3 ?- capital(colombia,Y).  
Y = bogotá.
```

```
4 ?- capital(Z, bogotá).  
Z = colombia ;  
false.
```

# Reglas

- Las reglas son sentencias condicionales. El lado izquierdo es el *encabezado*, el cual es verdadero si se cumplen las condiciones del lado derecho. El lado derecho se denomina el cuerpo de la regla.
- Ejemplo: La proposición "Si juanito estudia, gana la materia" se expresa así  
`ganaLaMateria(juanito) :- estudia(juanito).`

## Reglas (2)

- Se pueden tener múltiples condiciones, separadas por comas. Las comas se entienden como conjunciones: Todos los predicados del cuerpo deben ser verdaderos para satisfacer el encabezado.
- Ejemplo:  
Si juanito viene a clase, estudia y hace las tareas, gana la materia.

```
ganaLaMateria(juanito) :- vieneAClase(juanito),  
                           estudia(juanito),  haceLasTareas(juanito).
```

# Reglas (3)

- Se pueden definir reglas con variables

```
ganaLaMateria(X) :- vieneAClase(X),  
                    estudia(X),  haceLasTareas(X).
```

Cuando se hace una consulta como

```
?- ganaLaMateria(juanito).
```

las fórmulas atómicas que aparecen en el cuerpo se denominan *metas*, el Prolog intenta satisfacer todas las metas. Si lo logra, el encabezado es verdadero.

# Como resuelve el Prolog las Consultas?

Prolog se centra en el uso de un motor de inferencia basados en 2 técnicas: La *resolución* y la *unificación*.

- **Resolución:** Generalización del Modus Ponens: Si las metas son todas verdaderas, el encabezado es verdadero.
- **Unificación:** Cuando hay variables libres en una cláusula, asignar valores a las variables, buscando satisfacer las metas.

# Resolución

Tomemos la siguiente base de datos

```
pais(colombia).  
pais(ecuador).  
pais(españa).  
capital(colombia, bogotá).  
capital(ecuador, quito).  
capital(españa, madrid).
```

# Resoluciones simples

- Consultas en la forma de hechos se resuelven inmediatamente contra la base de datos:

```
3 ?- pais(colombia).  
true.
```

```
4 ?- capital(ecuador, quito).  
true.
```

```
5 ?- pais(nigeria).  
false.
```

- Una consulta satisfecha se llama un *éxito*.
- Cuando no es posible satisfacerla con la base de datos se denomina un *fracaso*.

# Unificación

- Si la consulta contiene variables, se compara la consulta con los hechos y las reglas en la base de datos. La consulta tiene éxito si es posible unificar las variables con hechos/reglas existentes.

```
6 ?- pais(X).  
X = colombia ;  
X = ecuador ;  
X = españa.
```

- Prolog reporta cada unificación exitosa.



# Unificación – Ejemplos adicionales

```
7 ?- capital(colombia, Y).  
Y = bogotá.
```

```
8 ?- capital(Z, madrid).  
Z = españa.
```

```
9 ?- capital(A, B).  
A = colombia,  
B = bogotá ;  
A = ecuador,  
B = quito ;  
A = españa,  
B = madrid.
```

# Consultas que implican reglas

- Tomemos la siguiente base de datos

```
humano(tatiana).  
humano(ramiro).  
mortal(X) :- humano(X).
```

- Analicemos las consultas

```
10 ?- mortal(ramiro).  
true.  
  
11 ?- mortal(john).  
false.
```

```
12 ?- mortal(Z).  
Z = tatiana ;  
Z = ramiro.
```

# Resolución con reglas compuestas

- Sea la base de datos

```
% b,c son hijos de a
padre(b, a). % El padre de b es a
padre(c, a).

% e,f son hijos de d
padre(e, d).
padre(f, d).

% regla para definir hermanos
hermanos(X,Y) :- padre(X,Z), padre(Y,Z).
```

- Analizar las siguientes consultas:

```
?- hermanos(a,b).
?- hermanos(X,c).
?- hermanos(Y,Z).
```

# Corrección a la regla

- Una persona no se considera hermana de si misma.

```
% regla corregida para definir hermanos  
hermanos(X,Y) :- padre(X,Z), padre(Y,Z), X\=Y.
```

# Operadores básicos

Operadores sobre la unificación (No se evalúa la expresión)

Operador	Significado
$X=Y$	Las variables unifican
$X\neq Y$	Las variables no unifican
$X==Y$	Términos idénticos (son la misma variable)
$X\neq Y$	Términos no idénticos

Poco usadas  
en la práctica

Operadores de comparación aritmética (Evalúan la expresión)

Operador	Significado
$E1:=E2$	Las expresiones son iguales
$E1\neq E2$	Las expresiones no son iguales
$E1<E2$	$E1$ es menor que $E2$
$E1>E2$	$E1$ es mayor que $E2$
$E1\leq E2$	$E1$ es menor o igual que $E2$
$E1\geq E2$	$E1$ es mayor o igual que $E2$

# Retroceso (Backtracking)

- Es el proceso por el cual se abandona una solución fallida, para intentar otras alternativas.
- En principio Prolog puede explorar todas las posibles asignaciones de las variables cuando intenta resolver una consulta.