

TUTORIAL PROLOG

Lenguajes de programación
Universidad Nacional de Colombia
Santiago Carvajal Castañeda
Mario Andrés Moreno Norato

Prolog es un lenguaje de programación lógica de propósito general que se fundamenta en la lógica de predicados o de primer orden.

A diferencia de la mayoría de lenguajes, PROLOG es un lenguaje declarativo. Esto quiere decir que describe su lógica computacional sin necesidad de una serie de pasos específica (control de flujo), sino que lo hace mediante hechos y relaciones lógicas entre objetos que permiten realizar preguntas sobre esas relaciones.

En general se podría decir que es un lenguaje donde se especifican hechos y relaciones para responder consultas.

La palabra PROLOG proviene del francés ***PRO**grammation en **LOG**ique*,


Breve historia


Prolog fue creado en la Universidad de Aix-Marseille en Francia por los profesores Alain Colmerauer y Philippe Roussel como herramienta práctica de programación lógica tomando como base la interpretación de Robert Kowalski de las cláusulas de Horn. Una primera versión preliminar fue lanzada en 1971 pero en 1972 apareció la primera versión definitiva. La primera versión era totalmente interpretada y fue implementada en un lenguaje de programación llamado ALGOL.


En 1983 David Warren desarrolló un compilador capaz de traducir Prolog en el conjunto de instrucciones de una máquina abstracta llamada Warren Abstract Machine que permitió mayor eficiencia, haciendo a prolog un lenguaje semi-interpretado.






Instalación del IDE - SWI Prolog

Link de instalación: <http://www.swi-prolog.org/download/stable>

 Linux versions are often available as a package for your distribution. We collect information about available packages and issues for building on specific distros [here](#). We provide a [PPA](#) for [Ubuntu](#)

 Please check the [windows release notes](#) (also in the SWI-Prolog startup menu of your installed version) for details.

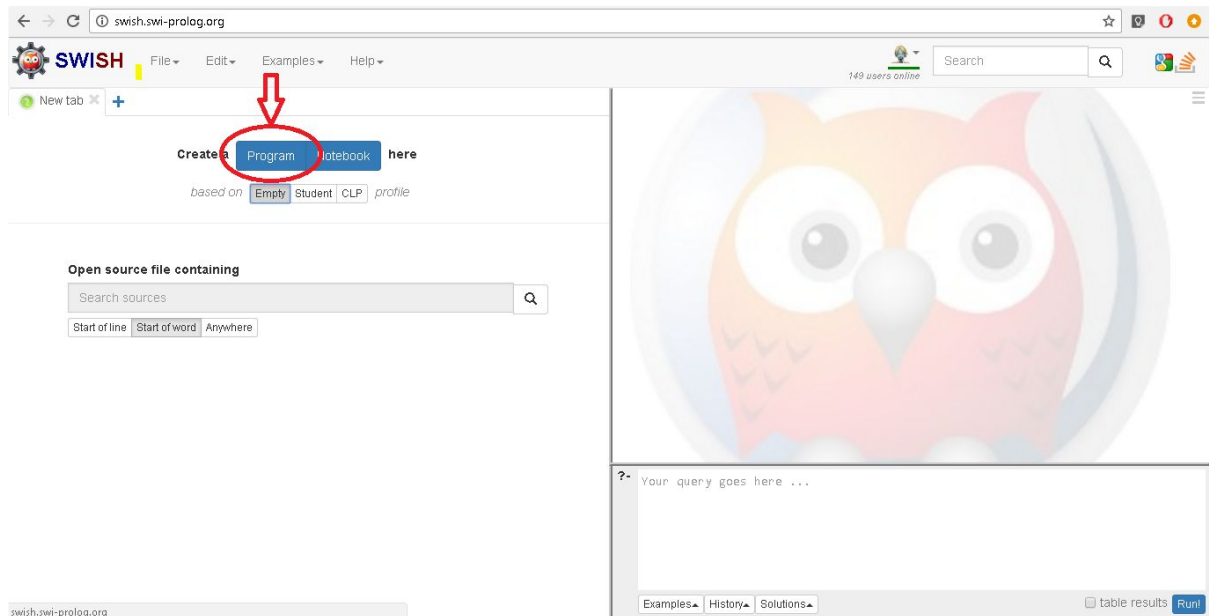
 Examine the [ChangeLog](#).

Binaries		
	20,420,987 bytes	SWI-Prolog 7.4.2 for Microsoft Windows (64 bit) Self-installing executable for Microsoft's Windows 64-bit editions. Requires at least Windows 7. See the reference manual for deciding on whether to use the 32- or 64-bits version. This binary is linked against GMP 6.1.1 which is covered by the LGPL license.
	19,121,158 bytes	SWI-Prolog 7.4.2 for Microsoft Windows (32 bit) Self-installing executable for MS-Windows. Requires at least Windows 7. Installs swipl-win.exe and swipl.exe . This binary is linked against GMP 6.1.1 which is covered by the LGPL license.
	23,576,581 bytes	SWI-Prolog 7.4.2 for MacOSX 10.6 (Snow Leopard) and later on intel Mac OS X disk image with relocatable application bundle . Needs xquartz (X11) installed for running the development tools . Currently, version 2.7.11 is required. You can check the version by opening an X11 application and then checking 'about' in the X11 menu. The bundle also provides the commandline tools in Contents/MacOS. The command line tools need at least MacOS 10.6 (Snow Leopard). The graphical application needs at least MacOS 10.7 (Lion).
Sources		
	16,496,738 bytes	SWI-Prolog source for 7.4.2 Sources in .tar.gz format, including packages and generated documentation files. See build instructions .
Documentation		
	2,376,271 bytes	SWI-Prolog 7.4.2 reference manual in PDF SWI-Prolog reference manual as PDF file. This does <i>not</i> include the package documentation.

IDE Online: <http://swish.swi-prolog.org/>

Para la realización de este tutorial usaremos el IDE Online.

Luego de ingresar a la página web, hacemos click en "Program"



El IDE online tiene las siguientes secciones en donde vamos a trabajar:



Ya podemos empezar a escribir nuestros programas en prolog.

Estructura de un programa en Prolog

Un programa en prolog está compuesto por cláusulas. Las cláusulas son la base del programa, están conformadas por **hechos** y **reglas**. El conjunto de hechos y reglas forman el conocimiento base del programa, mediante el cual podremos realizar consultas e inferir datos.

Hechos: Un hecho, en PROLOG, es una relación entre objetos. Para nosotros es una afirmación directa entre uno o varios objetos, donde debemos tomar en cuenta la siguiente formato:

Predicado(sujeto(';'||sujeto)*)

Ejemplos:

le_gusta_a(juan,maria).

valioso(oro).

tiene(juan,libro).

da(juan,libro,maria).

Reglas: Una regla, en PROLOG, es una sentencia condicional. Es decir si un objeto cumple cierta característica, se obliga a que tenga otra relación y debemos tener en cuenta el siguiente formato:

Conclusión condicional ParteCondicional

*condicional : “:-”

Ejemplo:

Primer Ejemplo Básico

Como una primer aproximación al lenguaje vamos a realizar el siguiente ejemplo sencillo y básico.

```
1 %Primer ejemplo:
2 %hechos
3 mamifero(perro).
4 mamifero(gato).
5 %reglas
6 animal(X) :- mamifero(X).
7
8
```



Elementos básicos del lenguaje (Sintaxis)

Antes de empezar con nuestro primer programa en Prolog vamos a conocer los elementos básicos que componen el lenguaje.

En realidad Prolog tiene un solo tipo de dato llamado término.

Término: Único tipo de dato que tiene varios subtipos: átomos, números, variables y términos compuestos.

- Átomo:

Un átomo es un nombre de propósito general, se compone de una secuencia de caracteres que son analizados como una sola unidad.

Sintaxis: Estos siempre comienzan en minúscula (Nunca en número) seguido de una secuencia de caracteres sin ningún tipo de carácter especial. Si el átomo tiene espacios o cierto tipo de caracteres especiales, el átomo se debe poner entre comillas sencillas.

Un átomo es siempre una constante

Ejemplos:

```
1 isAtom(pedro).
2 isAtom('Camilo Perez').
3 isAtom('$&&/*').
4 isAtom(atOmMm12345).
5 isAtom(i_am_AToMm_23).
6 |
```

- Números:

Los números se usan para representar valores constantes numéricos, estos valores pueden ser enteros o reales.

Sintaxis: Se usa la misma notación decimal para escribir valores numéricos como en cualquier otro lenguaje, igualmente se pueden escribir los valores en notación exponencial.

Ejemplos:

```
1 isNumber(2).
2 isNumber(232423).
3 isNumber(-3454).
4 isNumber(-8.87).
5 isNumber(2.1416).
6 isNumber(0.234).
7 isNumber(2e10).
8 isNumber(3e-24).
9 isNumber(2e2).
10 isNumber(.87767).
11 |
```

^
Syntax error: Operator expected

- Variables:

Sintaxis: Las variables se indican mediante una cadena de caracteres que puede

constar de letras, números y guión bajo, adicionalmente debe comenzar con una letra mayúscula o con guión bajo, esto las diferencia de los átomos.

Existe una variable especial que solo consta de un guión bajo (_), a esta variable se le llama variable anónima. Esta se usa cuando no es importante el nombre de la variable o cuando la variable no puede unificar con otra, dentro de la misma cláusula. Es importante señalar que el alcance de una variable es la cláusula donde aparece, y el alcance de una constante es todo el programa PROLOG.

Ejemplos:

```
1 isVariable(Var).
2 isVariable(_var).
3 isVariable(Va2323sdf).
4 isVariable(V_a_w_3_).
5 isVariable(_).
6
```

- Estructuras:

Término compuesto por otros términos. Una estructura se compone de un átomo llamado "functor" y cierto número de argumentos. Cada argumento es un término.

Sintaxis: Las estructuras se escriben ordinariamente como un functor seguido de una lista de términos(argumentos) separados por comas, que está contenida entre paréntesis

Ejemplos:

```
1 name(arg1).
  Functor Argumentos
3 name(arg1, arg2, arg3).
  Functor Argumentos
5 name(Var).
  Functor Argumentos
7 name(2, 3, asd, _var).
  Functor Argumentos
```

Consultas

Luego de tener el conocimiento base, podemos realizar consultas sobre este conocimiento.

Para realizar consultas usamos ?- seguido de la consulta a realizar. La consulta retorna true o false dependiendo si es cierta o no.

En el siguiente ejemplo tenemos 2 hechos y una regla, vamos a realizar consultas sobre estos:

```
1 esposa(alicia,lucas). % alicia es la esposa de lucas
2 padre(lucas,juan). % lucas es el padre de Juan
3
4 %Reglas :
5 madre(M,E):-padre(P,E),esposa(M,P).
6
7
```

Consulta 1

```
esposa(alicia,lucas)
true
?- esposa(alicia,lucas)
```

Consulta 2

```
esposa(alicia,juan)
false
?- esposa(alicia,juan)
```

Consulta 3

```
madre(alicia,lucas)
false
?- madre(alicia,lucas)
```

Consulta 4

```
madre(alicia,juan)
true
?- madre(alicia,juan)
```

Uso de variable en consultas

Usando variables en las consultas podemos obtener información más allá de true o false. Cuando se pone una variable en una consulta Prolog busca en el conocimiento base los posibles resultados que puede tomar esa variable para que la consulta sea verdadera:

Consulta 5

```
padre(X,juan)
X = lucas
?- padre(X,juan)
```

Consulta 6

```
madre(X,juan)
X = alicia
?- madre(X,juan)
```

Consulta 7

```
madre(X,lucas)
false
?- madre(X,lucas)
```

Consulta 8

```
esposa(alicia,X)
X = lucas
?- esposa(alicia,X)
```

Consultas complejas

Las consultas pueden realizar varias “preguntas” al tiempo. Separando las diferentes preguntas con comas obtenemos una consulta compleja.

Se modificó el ejemplo anterior poniendo dos nuevos hechos, que servirán como criterio en la consulta compleja.

```
1 hombre(lucas).
2 hombre(juan).
3 esposa(alicia,lucas). % alicia es la esposa de Lucas
4 padre(lucas,juan). % Lucas es el padre de Juan
5 padre(lucas,camila). % Lucas es el padre de Juan
6
7 %Reglas :
8 madre(M,E) :- padre(P,E), esposa(M,P).
9
10
11
12
13
```


Consulta normal

```
Hijos = juan
Hijos = camila
?- madre(alicia, Hijos)
```

Consulta compleja (Hijos de alicia que también son hombres).

```
madre(alicia, Hijos), hombre(Hijos)
Hijos = juan
false
?- madre(alicia, Hijos), hombre(Hijos)
```

Ejemplo 2

Recursividad en Prolog

Prolog no tiene ciclos, por lo tanto hay que usar recursividad.

Ejemplos:

- Potencia

```
1 potencia(A,0,1):- A \=0.
2 potencia(X,Y,Res):- Y>0,Y1 is Y-1,
3                       potencia(X,Y1,Res1 ),
4                       Res is X*Res1.
5
```

Consultas

`potencia(3,1,X)`

X = 3

Next	10	100	1,000	Stop
------	----	-----	-------	------

`potencia(3,2,X)`

X = 9

Next	10	100	1,000	Stop
------	----	-----	-------	------

`potencia(0,0,X)`

false

`potencia(2,100,X)`

X = 1267650600228229401496703205376

Next	10	100	1,000	Stop
------	----	-----	-------	------

`potencia(2,3,X)`

X = 8

Next	10	100	1,000	Stop
------	----	-----	-------	------

`potencia(2,3,8)`

true

Next	10	100	1,000	Stop
------	----	-----	-------	------

`potencia(2,3,80)`

false

?- `potencia(2,3,80)`

- Fibonacci

```

1 fibonacci(0,0). %Caso base
2 fibonacci(1,1). %Caso base
3 fibonacci(N,Y):- N>1,
4                 N1 is N-1,
5                 fibonacci(N1,Y1),
6                 N2 is N-2,
7                 fibonacci(N2,Y2),
8                 Y is Y1+Y2.
9
10
11
12
13


```

Consultas


fibonacci(10,Resultado)

Resultado = 55

?- fibonacci(10,Resultado)


fibonacci(10,55)

true

?- fibonacci(10,55)



fibonacci(10,100)

false

?-

fibonacci(10,100)

Construcción de expresiones aritméticas en PROLOG

A continuación se muestran los operadores que se pueden utilizar para construir expresiones:

$X+Y$	Suma
$X - Y$	Resta
$X * Y$	Multiplicación
X / Y ó $X \text{ div } Y$	división real y entera
$X \text{ mod } Y$	Modulo
$X ^ Y$	X elevado a la Y
$-X$	negacion
$\text{abs}(X)$	Valos absoluto
$\text{acos}(X)$	arco coseno de x
$\text{asen}(X)$	arco seno de X
$\text{atan}(X)$	arco tangente de X
$\text{cos}(X)$	cos de X
$\text{exp}(X)$	exponencial de X
$\text{ln}(X)$	logaritmo natural de X
$\text{log}(X)$	logaritmo en base 2 de X
$\text{sin}(X)$	seno de X
$\text{sqrt}(X)$	rayz cuadrada de X
$\text{tan}(X)$	tangente de X
$\text{round}(X,N)$	redondeo del real X con N decimales

Comparación de términos en PROLOG.

Los siguientes operadores son los que permiten comparar términos

$X < Y$	X es menor que Y
$X > Y$	X es mayor que Y
$X \leq Y$	X es menor o igual que Y
$X \geq Y$	X es mayor o igual que Y
$X = Y$	X es igual que Y
$X \neq Y$	X es distinto que Y

Comparación de expresiones en PROLOG.

Una expresión es un conjunto de términos unidos por operadores aritméticos. Los siguientes predicados predefinidos comparan expresiones sin evaluarlas, mediante una comparación sintáctica.

$X == Y$	la expresión X es igual que la expresión Y
$X \neq Y$	la expresión X es distinta que la expresión Y
$X @< Y$	la expresión X es menor que la expresión Y
$X @> Y$	la expresión X es mayor que la expresión Y
$X @ \leq Y$	la expresión X es menor o igual que la expresión Y
$X @ \geq Y$	la expresión X es mayor o igual que la expresión Y





























Para obtener una asignación en PROLOG usamos el predicado "is" para instanciar la variable que aparece al lado izquierdo de "is", esta asignación solo se puede hacer si la variable de la izquierda no está instanciada, es decir no se puede cambiar el valor de una variable.

Ejemplo:

```

1 mayor(X,Y):- X>Y.
2
3 diferencia(X,Y,Z):- Z is X-Y.
4
5 producto(X,Y,Z):- Z is X*Y.
6
7 senoCuadrado(X,Z):- producto(sin(X),sin(X),Z1), Z is Z1.
8
9 comparacionIs(X,Y) :- X is Y. %Evalua antes de comparar
10
11 comparacion(X,Y) :- X == Y. %Comparación sintáctica
12

```

 mayor(10,-2)	  
true	1
 diferencia(9,3,Z)	  
Z = 6	
 diferencia(10,2,8)	  
true	1
 senoCuadrado(60,Z)	  
Z = 0.0929095147367191	
 comparacionIs(60,30*2)	  
true	1
 comparacion(60,30*2)	  
false	
 comparacion(30*2,30*2)	  
true	1

Los siguientes predicados predefinidos comparan términos haciendo una evaluación de expresiones:

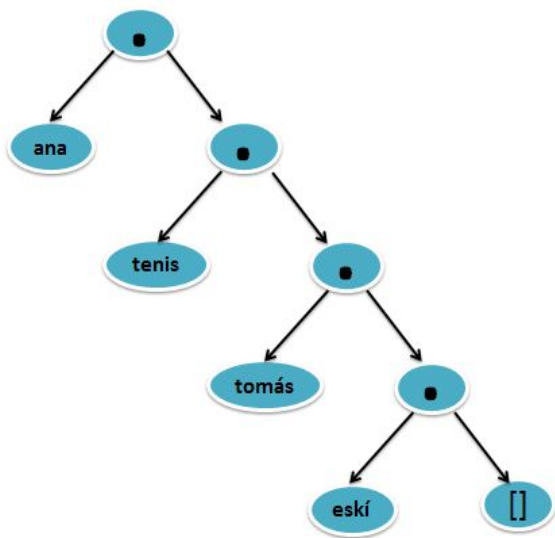
$X:=Y$	El resultado de evaluar la expresión X es igual al resultado de evaluar la expresión Y
$X\neq Y$	El resultado de evaluar la expresión X es distinto al resultado de evaluar la expresión Y

Listas en PROLOG

La sintaxis de PROLOG para las listas consiste en englobar las secuencias de elementos entre corchetes:

[ana,tenis,tomás,eski]

La representación interna de las listas en PROLOG es con árboles binarios, donde la rama de la izquierda es el primer elemento (cabeza de la lista) y la rama de la derecha es el resto (cola de la lista). De la rama que contiene el resto de la lista también se distinguen dos ramas: la cabeza y la cola. Y así sucesivamente hasta que la rama de la derecha contenga la lista vacía (representado por [])



Los elementos de una lista pueden ser de cualquier tipo, incluso otra lista.

ejemplo 2

[ana,[tenis,tomás],eski]

