

Prolog



Jaime Andres Vargas
Jonathan Alberto Ortiz

Contenido

1. Que es prolog
2. Porque prolog
3. Introduccion a prolog
 - a. Operadores logicos
 - b. Otros operadores
 - c. Variables
 - d. Busquedas
 - e. Recursion
 - f. Listas
 - g. Arboles
 - h. Grafos
 - i. automatas
4. Aplicaciones
5. Ventajas / Desventajas

¿Qué es Prolog?

Es un lenguaje declarativo para programación lógica de propósito general , se basa en la representación de relaciones mediante reglas y hechos . La principal diferencia con el imperativo es que buscamos describir el problema.

Prolog es un diminutivo de PROgrammation en LOGique



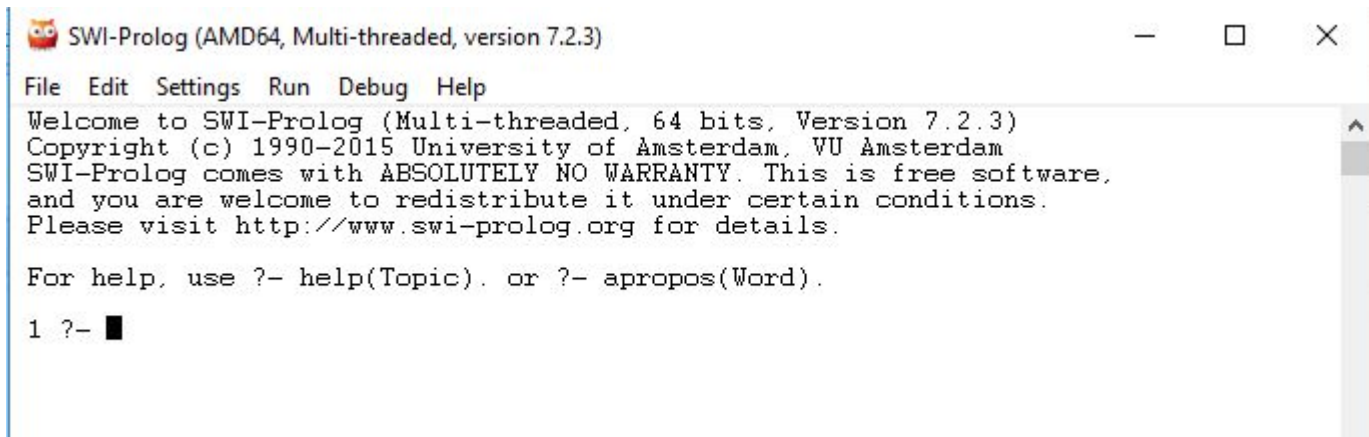
SWI Prolog

¿Porqué Prolog?

- Inteligencia Artificial (inferencia y representación del conocimiento)
- Procesamiento de lenguaje(Autómatas)
- La recursion.

IDE

Existen múltiples IDE para prolog, en este caso usaremos SWI-PROLOG
[enlace de descarga](#)



```
SWI-Prolog (AMD64, Multi-threaded, version 7.2.3)
File Edit Settings Run Debug Help
Welcome to SWI-Prolog (Multi-threaded, 64 bits, Version 7.2.3)
Copyright (c) 1990-2015 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 ?- █
```

También existe un version en linea de [prolog](#)

Como usar el IDE

Para crear un archivo en prolog.

file-> new->"nombre_del_archivo"

Para "ejecutar"

file-> consult->"nombre_del_archivo"

Características de sintaxis

En prolog los elementos más importantes del lenguaje son los términos,compuestos en :

Variables: se representan con una letra mayúscula en su inicio, una excepción es la variable anonima ‘_’.

Constantes:

Números: prolog soporta números reales y enteros.

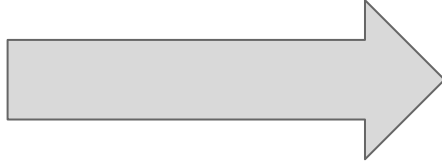
Átomos o functor: nombrar objetos y propiedades

Estructuras complejas: son términos compuestos por otros términos.

Propiedades y relaciones

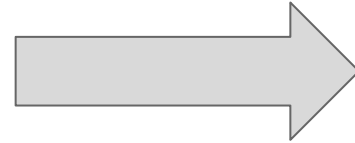
filosofo(aristoteles).

↓
predicado



Es un hecho que es
Propiedad

realMadridBBC(bale,benzema,cristiano).



Es un hecho que es
Relacion

Cuando se tiene un hecho con un solo argumento se dice que es monádico, de lo contrario es poliádico.

Nota: el nombre de un hecho empieza por minúscula y siempre termina en punto.

Operadores logicos

Prolog	Logica
A :- B	A -> B
A , B	A && B
A ; B	A B

Otros operadores

Prolog	Significado
$X \backslash == Y$	X diferente de Y
$X == Y$	X igual Y
$X < Y$	X menor a Y
$X > Y$	X mayor a Y
$X >= Y$	X mayor igual a Y
$X <= Y$	X menor igual a Y
$X ::= Y$	X es igual a Y (evalúa X y Y primero).

Cuando se quiere comparar expresiones(ej:predicados, términos complejos) se añade un @ inmediatamente después dela primera expresión ($X@<Y$)

a.

88 ?-

|

|

| abc@<e.

true.

89 ?- 5@<3.

false.

90 ?- 5@<6.

true.

91 ?- 5<6

|

| .

true.

92 ?- X is 3*4 .

X = 12.

93 ?- Y is 8/2.

Y = 4.

94 ?- X<Y.

ERROR: </2: Arguments are not sufficiently instantiated

95 ?- X@<Y.

true.

Introduccion a prolog

Prolog se basa en un conocimiento base para solucionar consultas así como inferir información .

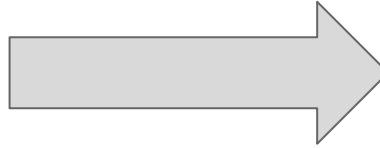
El conocimiento lo representamos mediante cláusulas. Las cláusulas son **hechos** o **reglas** , están compuestas usualmente de predicados.

```
filosofo(aritoteles).  
filosofo(platon).
```

```
fisico(faraday).  
fisico(feynman).
```

```
matematico(gauss).  
matematico(wiles).
```

```
sabe_matematicas(pepe):-matematico(pepe).  
sabe_fisica(pepe):- fisico(pepe).  
sabe_filosofia(pepe):- filosofo(pepe).
```



Hechos



Reglas



Clausulas	9
Preposiciones	6

```
introprolog.pl
File Edit Browse Compile Prolog Pce Help
introprolog.pl
filosofo(aritoteles).
filosofo(platon).

fisico(faraday).
fisico(feynman).

matematico(gauss).
matematico(wiles).

sabe_matematicas(pepe):-matematico(pepe).
sabe_fisica(pepe):- fisico(pepe).
sabe_filosofia(pepe):- filosofo(pepe).
```

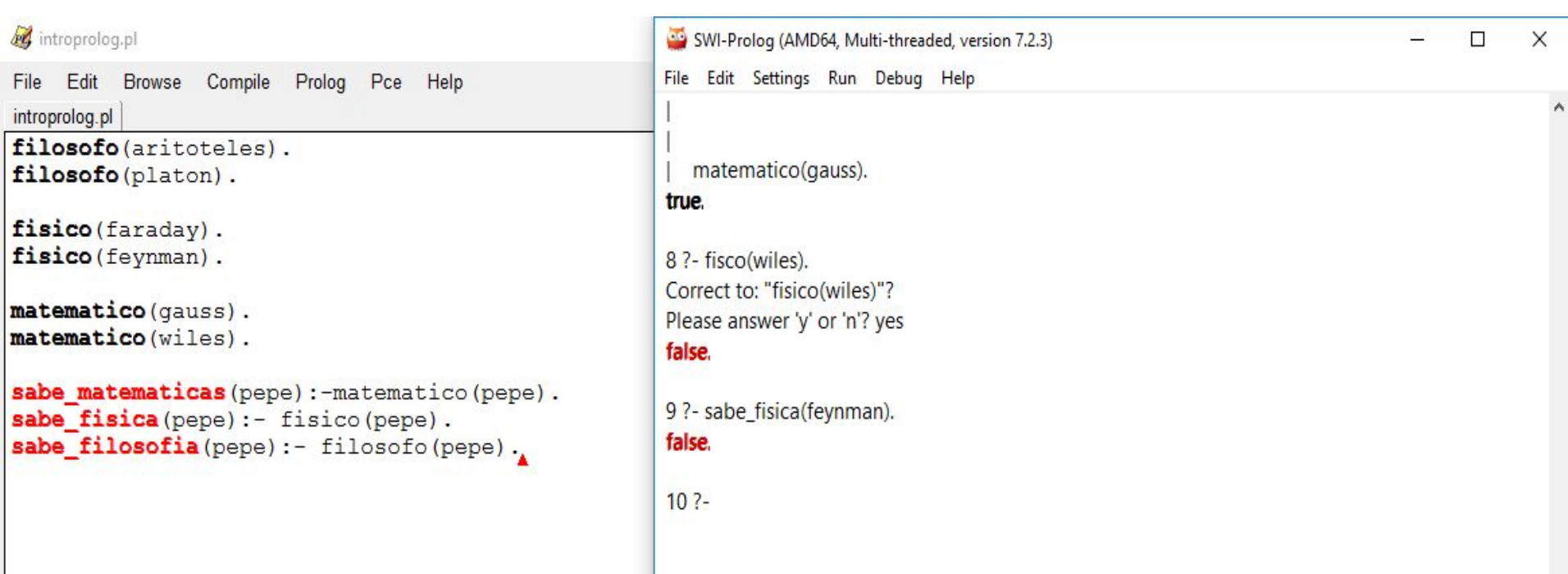
```
SWI-Prolog (AMD64, Multi-threaded, version 7.2.3)
File Edit Settings Run Debug Help
|
|
| matematico(gauss).
true.

8 ?- fisico(wiles).
Correct to: "fisico(wiles)"?
Please answer 'y' or 'n'? yes
false.

9 ?- sabe_fisica(feynman).
false.

10 ?-
```

Las reglas se pueden “leer” de la siguiente forma: **pepe sabe matemáticas si es matemático.**



```
introprolog.pl
File Edit Browse Compile Prolog Pce Help
introprolog.pl
filosofo(aritoteles).
filosofo(platon).

fisico(faraday).
fisico(feynman).

matematico(gauss).
matematico(wiles).

sabe_matematicas(pepe):-matematico(pepe).
sabe_fisica(pepe):- fisico(pepe).
sabe_filosofia(pepe):- filosofo(pepe).

SWI-Prolog (AMD64, Multi-threaded, version 7.2.3)
File Edit Settings Run Debug Help
|
|
| matematico(gauss).
true.

8 ?- fisico(wiles).
Correct to: "fisico(wiles)"?
Please answer 'y' or 'n'? yes
false.

9 ?- sabe_fisica(feynman).
false.

10 ?-
```

Pero Feynman es fisico, luego debería retornar verdadero la consulta 9, ¿Por que?

Variables


En prolog se representan las variables dinámicas si se empieza por mayúscula, note gauss, aristoteles, ...etc, son constantes.

 SWI-Prolog (AMD64, Multi-threaded, version 7.2.3)

File Edit Settings Run Debug Help

```
2 ?-  
|  sabe_fisica(feynman)  
|  .  
true.
```

```
3 ?-
```

 introprolog.pl

File Edit Browse Compile Prolog Pce Help

introprolog.pl

```
filosofo(aristoteles).  
filosofo(platon).  
  
fisico(faraday).  
fisico(feynman).  
  
matematico(gauss).  
matematico(wiles).  
  
sabe_matematicas(Pepe):-matematico(Pepe).  
sabe_fisica(Pepe):- fisico(Pepe).  
sabe_filosofia(Pepe):- filosofo(Pepe).
```



```
2 ?-  
|  sabe_fisica(feynman)  
|  .  
true.
```

```
3 ?- filosofo(X).  
X = aritoteles |
```

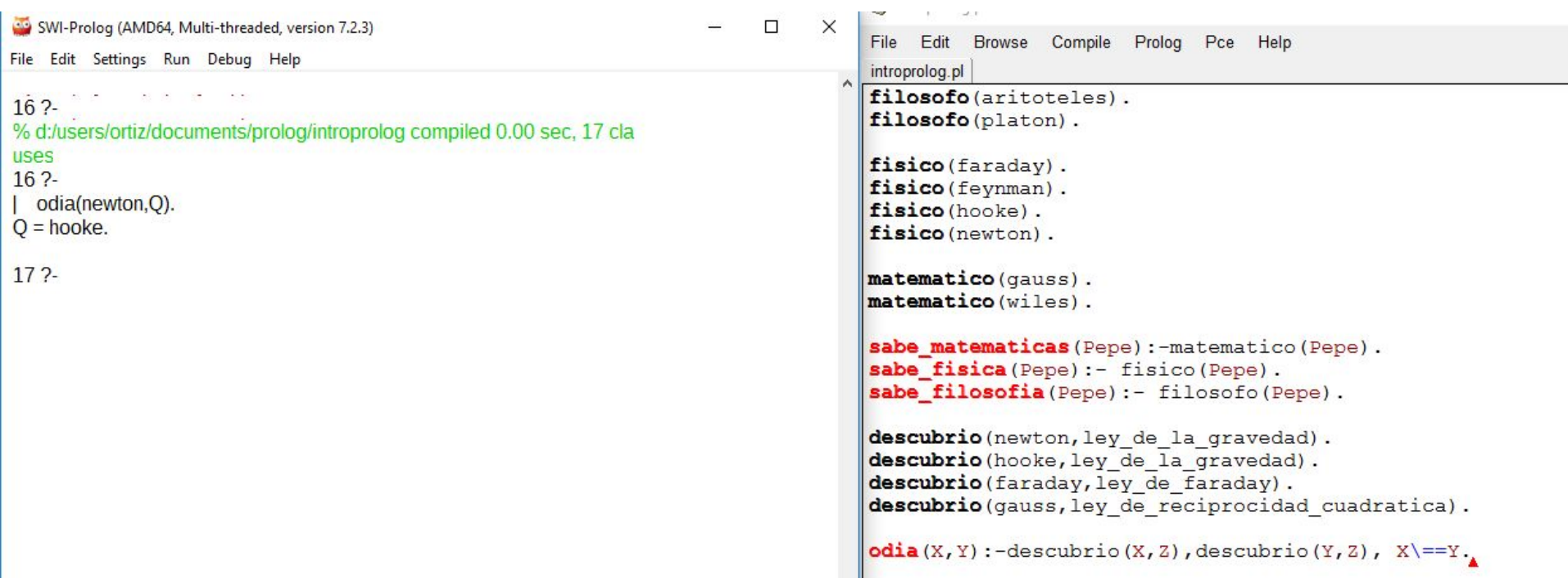
```
2 ?-  
|  sabe_fisica(feynman)  
|  .  
true.
```

```
3 ?- filosofo(X).  
X = aritoteles ;  
X = platon.
```

```
4 ?- |
```

Cuando en una consulta se tiene mas de una alternativa Prolog devuelve la primera ocurrencia, se obtienen las demás insertando el token punto y coma.

Note que X es una variable dinámica.



The image shows a screenshot of the SWI-Prolog IDE. The title bar indicates 'SWI-Prolog (AMD64, Multi-threaded, version 7.2.3)'. The interface is split into two main panes. The left pane shows the Prolog prompt and the execution of a query. The right pane shows the source code of the Prolog program.

Left Pane (Execution Output):

```
16 ?-  
% d:/users/ortiz/documents/prolog/introprolog compiled 0.00 sec, 17 clauses  
uses  
16 ?-  
| odia(newton,Q).  
Q = hooke.  
17 ?-
```

Right Pane (Source Code: introprolog.pl):

```
filosofo(aritoteles).  
filosofo(platon).  
  
fisico(faraday).  
fisico(feynman).  
fisico(hooke).  
fisico(newton).  
  
matematico(gauss).  
matematico(wiles).  
  
sabe_matematicas(Pepe):-matematico(Pepe).  
sabe_fisica(Pepe):- fisico(Pepe).  
sabe_filosofia(Pepe):- filosofo(Pepe).  
  
descubrio(newton,ley_de_la_gravedad).  
descubrio(hooke,ley_de_la_gravedad).  
descubrio(faraday,ley_de_faraday).  
descubrio(gauss,ley_de_reciprocidad_cuadratica).  
  
odia(X,Y):-descubrio(X,Z),descubrio(Y,Z), X\==Y.
```

¿Qué sucede si se elimina la comparación en la regla de odia?

Prolog Strings / Aritmetica

```
2 ?- write("asd")._
asd
true.
```

```
3 ?- print("asd").
"asd"
true.
```

```
16 ?- X is 5+4.
X = 9.
```

```
17 ?- X is 9-21.
X = -12.
```

```
18 ?- X is 8*9.
X = 72.
```

```
19 ?- X is 7/8.
X = 0.875.
```

```
20 ?- X is cos(pi/2).
X = 6.123031769111886e-17.
```

```
21 ?- X is cos(0).
X = 1.0.
```

```
22 ?- X is sin(pi/2).
X = 1.0.
```

unificacion

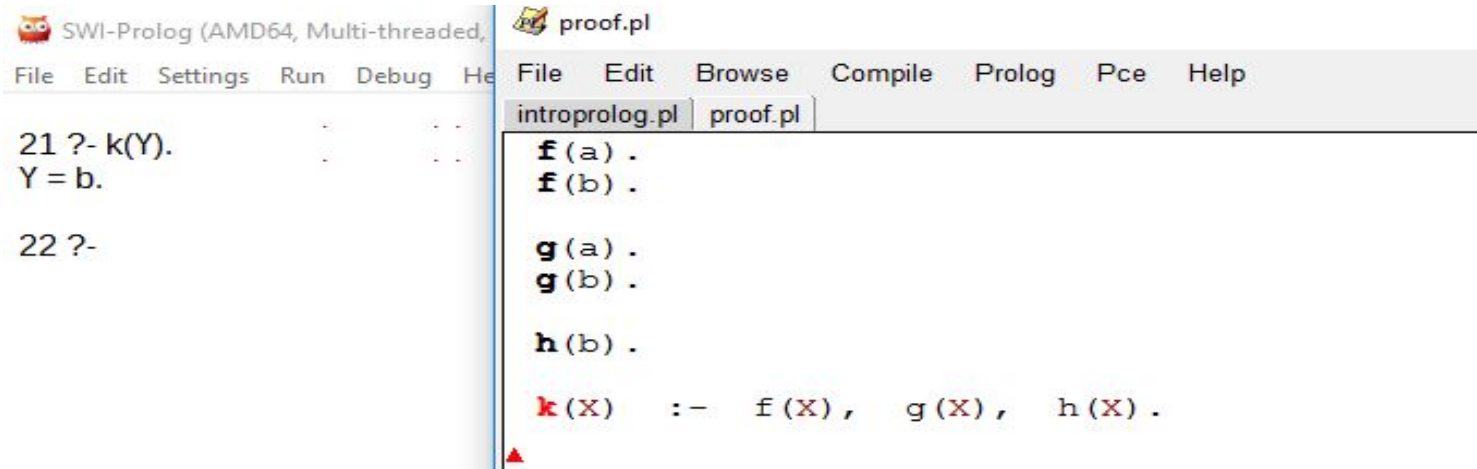
- Se pueden unificar listas y variables no instanciadas.
- Para unificar una variable, simplemente asignamos el objeto con el operador =
- Para asignar una lista, podemos asignar variables a la cabecera y la cola o asignar los elementos uno por uno.

```
?- X=[ elemento1,elemento2,elemento3 ].  
X = [elemento1, elemento2, elemento3].
```

```
?- [a,b,c] = [ Head | Tail ].  
Head = a,  
Tail = [b, c].
```

```
?- [a,b,c,d,e]=[X,Y | Tail].  
X = a,  
Y = b,  
Tail = [c, d, e].
```

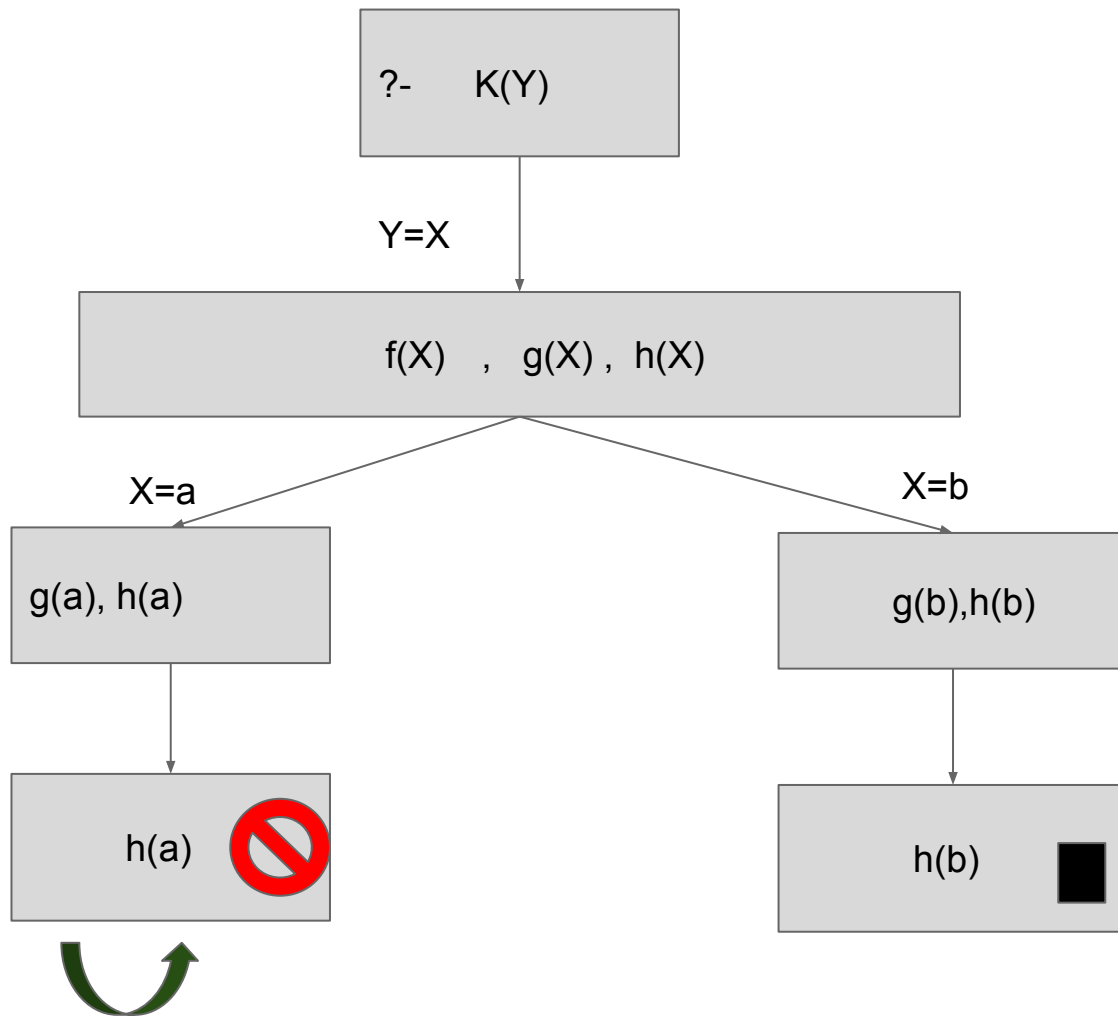
Busquedas/Backtracking



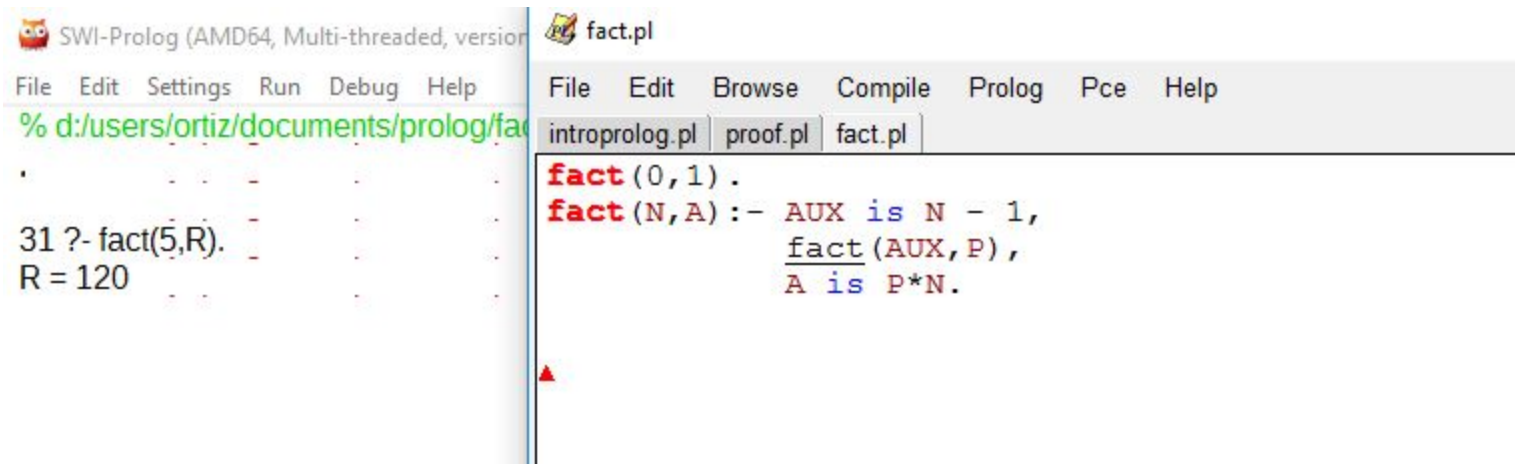
The screenshot shows the SWI-Prolog IDE interface. The title bar indicates 'SWI-Prolog (AMD64, Multi-threaded, ...)' and 'proof.pl'. The menu bar includes 'File', 'Edit', 'Settings', 'Run', 'Debug', 'Help', 'File', 'Edit', 'Browse', 'Compile', 'Prolog', 'Pce', and 'Help'. The 'File' menu is open, showing 'introprolog.pl' and 'proof.pl'. The main editor window displays the following Prolog code:

```
21 ?- k(Y).  
Y = b.  
  
22 ?-  
  
f(a).  
f(b).  
  
g(a).  
g(b).  
  
h(b).  
  
k(X) :- f(X), g(X), h(X).
```

¿Como prolog realiza la búsqueda en su “knowledge base”?



Recursion



The screenshot shows the SWI-Prolog IDE with a file named `fact.pl` open. The code defines a recursive factorial function. The left pane shows the execution of the query `?- fact(5,R).`, which results in `R = 120`.

```
SWI-Prolog (AMD64, Multi-threaded, version 8.6.2) fact.pl
File Edit Settings Run Debug Help
% d:/users/ortiz/documents/prolog/fact.pl
introprolog.pl proof.pl fact.pl

fact(0,1).
fact(N,A):- AUX is N - 1,
            fact(AUX,P),
            A is P*N.

?- fact(5,R).
R = 120
```

```
fact(5,R).  
fact(4,R).  
fact(3,R).  
fact(2,R).  
fact(1,R).  
fact(0,R). -> R=1
```

```
fact(5,R). -> R=120 , AUX = 24 * 5  
fact(4,R). -> R=24 , AUX = 6 * 4  
fact(3,R). -> R=6 , AUX = 2 * 3  
fact(2,R). -> R=2 , AUX = 1 * 2  
fact(1,R). -> R=1 , AUX = 1 * 1  
fact(0,R). -> R=1
```


Ejemplo de familias

```
%papas
padre(jose,jesus).
padre(jesus,sofia).
padre(jesus,alejandro).
padre(jose,juan).
padre(jose,alvaro).
padre(jose,gustavo).
padre(gustavo,william).
padre(gustavo,felipe).
padre(gustavo,oscar).
padre(gustavo,luís).
padre(gustavo,sandra).
padre(juan,manuel).
padre(manuel,esteban).
padre(manuel,antonio).
padre(manuel,martin).
padre(alvaro,tomas).
padre(danilo,anabel). %agrego
```

```
%mamas
madre(maria,jesus).
madre(magdalena,sofia).
madre(magdalena,alejandro).
madre(maria,juan).
madre(maria,alvaro).
madre(maria,gustavo).
madre(josefina,anabel). %agrego
madre(anabel,manuel).
madre(clementina,esteban).
madre(clementina,antonio).
madre(clementina,martin).
```

%hijo o hija

`hijo(X,Y):-padre(Y,X).`

`hijo(X,Y):-madre(Y,X).`

%abuelo o abuela

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

`abuelo(X,Y):-madre(Z,Y),madre(X,Z).`

%hermano o hermana

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

`hermano(X,Y):-madre(Z,X),hijo(Y,Z),X\=Y.`

%tio o tia

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

`tio(X,Y):-madre(Z,Y),hermano(Z,X).`

%esposos o esposas

`esposos(X,Y):-padre(X,Z),madre(Y,Z).`

`esposos(X,Y):-madre(X,Z),padre(Y,Z).`

%sobrino o sobrina

`sobrino(X,Y):-tio(Y,X).`

%primo o prima

`primo(X,Y):-padre(Z,Y),tio(Z,X).`

`primo(X,Y):-madre(Z,Y),tio(Z,X).`

%%cuñado o cuñada

`cunado(X,Y):-hermano(Z,Y),esposos(Z,X).`

`cunado(X,Y):-hermano(Z,X),esposos(Z,Y).`

%nieto o nieta

`nieto(X,Y):-abuelo(Y,X).`

%bisabuelo o bisabuela

`bisabuelo(X,Y):-abuelo(X,Z),padre(Z,Y).`

`bisabuelo(X,Y):-abuelo(X,Z),madre(Z,Y).`

%bisnieto o bisnieta

`bisnieto(X,Y):-bisabuelo(Y,X).`

%yerno o nuera

`yerno_nuera(X,Y):-hijo(Z,Y),esposos(X,Z).`

%suegro o suegra

`suegro(X,Y):-yerno_nuera(Y,X).`

Listas

- Son una secuencia finita de ítems.
- Se componen por una cabeza y una cola
- La cabeza siempre es el primer elemento , la cola es lo que sobra
- Estan encerradas por paréntesis cuadrados.
- [] es la lista vacia.

[HEAD | TAIL]=[item1,item2,...,item_N].

- En prolog las listas se definen recursivamente.

[a,b,c]=[a | [b,c]]=[a | [b | [c]]] = [a|[b|[c|[]]]]

```
50 ?- [ H | T ]=[a,b,c,D,filosofo(aristoteles),[1,2,3]].
```

```
H = a,
```

```
T = [b, c, D, filosofo(aristoteles), [1, 2, 3]].
```

```
51 ?- [ H,H2,H3 | T ]=[a,b,c,D,filosofo(aristoteles),[1,2,3]].
```

```
H = a,
```

```
H2 = b,
```

```
H3 = c,
```

```
T = [D, filosofo(aristoteles), [1, 2, 3]].
```

```
52 ?- [ H,_,H3,H4,_ | _ ]=[a,b,c,D,filosofo(aristoteles),[1,2,3]]
```

```
.
```

```
H = a,
```

```
H3 = c,
```

```
H4 = D.
```

```
53 ?- |
```

La variable '_' se denomina variable anónima ,se usa cuando es nesario usar una variable pero no se esta interesado en saber su contenido.

Funciones predefinidas para listas

- `is_List()`.
- `memberchk()`.
- `length()`.
- `sort()`

Entre otros . ver [funciones](#)

 SWI-Prolog (AMD64, Multi-threaded, version 7.2.3)

File Edit Settings Run Debug Help

```
46 ?- length([a,b,c,[e,d]],L).  
L = 4.
```

```
47 ?- memberchk([a,b],[1,2,3,[a,b]]).  
true.
```

```
48 ?- memberchk(a,[1,2,3,[a,b]]).  
false.
```

```
49 ?- sort([3,0,-1,8,0,69],L).  
L = [-1, 0, 3, 8, 69].
```

```
50 ?-
```

Append

Esta funcion permite

- Concatenar dos listas
- Obtener sublistas
- Generar todas las combinaciones de listas.

 SWI-Prolog (AMD64, Multi-threaded, version 7.2.3)

File Edit Settings Run Debug Help

```
|  
|  
| append([a,b],[x,y],L).  
L = [a, b, x, y].
```

```
65 ?- append(X,[1,2,3],[-1,0,1,2,3]).  
X = [-1, 0].
```

```
66 ?- append(X,Y,[a,b,c,d]).  
X = [],  
Y = [a, b, c, d] ;  
X = [a],  
Y = [b, c, d] ;  
X = [a, b],  
Y = [c, d] ;  
X = [a, b, c],  
Y = [d] ;  
X = [a, b, c, d],  
Y = [] ;  
false.
```

```
67 ?- |
```

Member



SWI-Prolog (AMD64, Multi-threaded, version 7.2.3)

File Edit Settings Run Debug Help

```
|  
|  
| member(a,[1,2,3,[a,b]]).  
false.  
  
62 ?- member(3,[1,2,3,[a,b]]).  
true.  
  
63 ?- member(X,[1,2,3,[a,b]]).  
X = 1 ;  
X = 2 ;  
X = 3 ;  
X = [a, b].  
  
64 ?-
```

Esta función es muy útil para “iterar” sobre la lista así como revisar si un item pertenece a la lista.

```
llista([],0).
llista([_|T],N):- llista(T,NT),N is NT +1.

sumlista([],0).
sumlista([H|T],S):- sumlista(T,NT), S is NT +H.

maxlista([H],H).
maxlista([H|T],N):- maxlista(T,NT), N is max(NT,H).

minlista([H],H).
minlista([H|T],N):- minlista(T,NT), N is min(NT,H).

invlista([],[]).
invlista([X|T],N):-invlista([X|T],[],N).
invlista([],Zs,Zs).
invlista([X|Xs],Ys,Zs):- invlista(Xs,[X|Ys],Zs).
```



```
invlista( [4,3,2,1],[],S).  
inlista([4|3,2,1],[],S).  
invlista([3,2,1],[4|[]],S).  
invlista([3|2,1],[4],S).  
invlista([2,1],[3|4],S).  
invlista([2|1],[3,4],S).  
invlista([1],[2|3,4],S).  
invlista([1|[]],[2,3,4],S).  
invlista([], [1|2,3,4],S).
```

```
invlista( [4,3,2,1],[],S). -> S=[1,2,3,4]  
inlista([4|3,2,1],[],S). -> S=[1,2,3,4]  
invlista([3,2,1],[4|[]],S). -> S=[1,2,3,4]  
invlista([3|2,1],[4],S). -> S=[1,2,3,4]  
invlista([2,1],[3|4],S). -> S=[1,2,3,4]  
invlista([2|1],[3,4],S). -> S=[1,2,3,4]  
invlista([1],[2|3,4],S). -> S=[1,2,3,4]  
invlista([1|[]],[2,3,4],S). -> S=[1,2,3,4]  
invlista([], [1|2,3,4],S). -> S=[1,2,3,4]
```

Merge Sort

```
llist([],0).
llist([_|T],N):- llist(T,NT),N is NT +1.

divHalf(List,A,B) :- llsplit(List,List,A,B).
llsplit(S,[],[],S).
llsplit(S,[_],[],S).
llsplit([H|T],[_,_|T2],[H|A],S) :-llsplit(T,T2,A,S).

merge([], H, H).
merge(H, [], H).
merge([H|T], [Y|K], [H|S]) :- H <= Y, merge(T, [Y|K], S).
merge([H|T], [Y|K], [Y|S]) :- Y <= H, merge([H|T], K, S).

mergesort( [],[] ).
mergesort([H], [H]).
mergesort([H|T], S) :-
    llist(T,LEN),
    LEN > 0,
    divHalf([H|T], Y, Z),
    mergesort(Y, L1),
    mergesort(Z, L2),
    merge(L1, L2, S).
```

```
List = [a,b,c,d]
divHalf(List,A,B).
llsplit([a,b,c,d],[a,b,c,d],A,B).
llsplit([a|b,c,d],[_,_|c,d],[a|A],S).
llsplit([b,c,d],[c,d],A,S).
llsplit([b|c,d],[_,_|[]],[b|A],S).
llsplit([c,d],[],A,S).
```

```
List = [a,b,c,d]
divHalf(List,A,B).
llsplit([a,b,c,d],[a,b,c,d],A,B). -> A=[a,b] , B=[c,d]
llsplit([a|b,c,d],[_,_|c,d],[a|A],S). -> [a|A]=[a,b] , [c,d]
llsplit([b,c,d],[c,d],A,S). -> A=[b] , S=[c,d]
llsplit([b|c,d],[_,_|[]],[b|A],S). -> [b|A] = [b|[]] , S=[c,d]
llsplit([c,d],[],A,S). -> A=[] , S=[c,d]
```

```

L1=[1,7], L2=[-1,8]
merge([1,7],[-1,8],S ).
merge( [1,7],[-1,8],[-1|S] ).
merge([1|7],[8],S).
merge([1,7],[8],[1|S] ).
merge([7],[8],S). S=[7,8]
merge([7],[8],[7|S]).
merge( [], [8], S ).

```

```

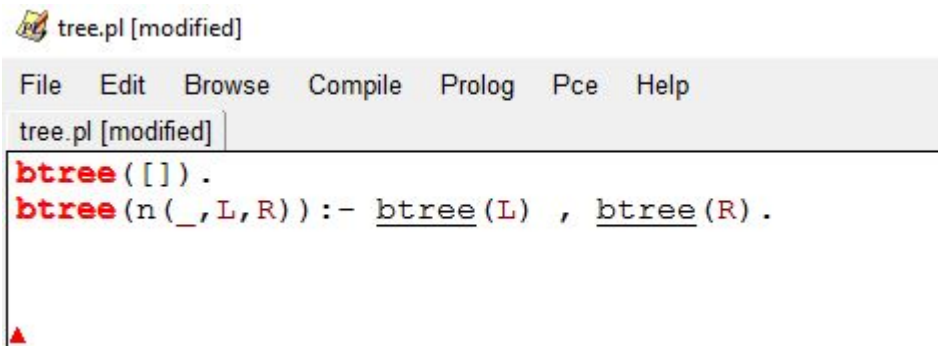
L1=[1,7], L2=[-1,8]
merge([1,7],[-1,8],S ) S=[-1,1,7,8]
merge( [1,7],[-1,8],[-1|S] ). [-1|S] =[-1| 1,7,8] % Y<= H
merge([1|7],[8],S). S=[1,7,8]
merge([1,7],[8],[1|S] ). [1|S]=[1|7,8] % H<=Y
merge([7],[8],S). S=[7,8]
merge([7],[8],[7|S]). [7|S]=[7|8] % H<=Y
merge( [], [8], S ). S=[8]

```

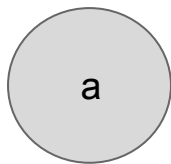
Arboles

Gracias a que Prolog está basado en recursividad , la definición de un árbol no es muy compleja.

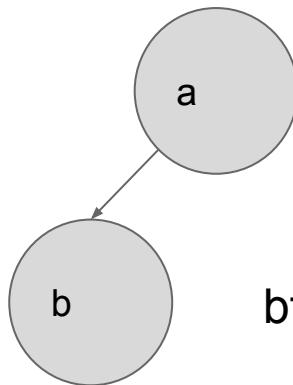
Representamos los árboles por medio de listas



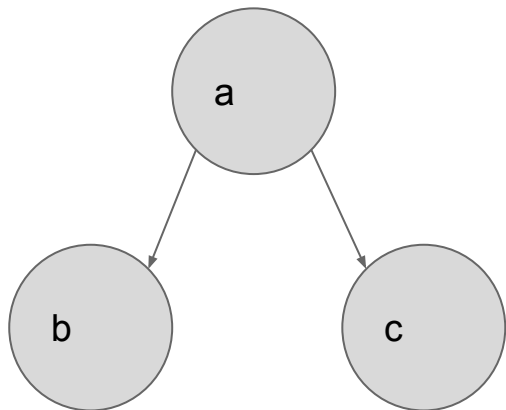
```
tree.pl [modified]
File Edit Browse Compile Prolog Pce Help
tree.pl [modified]
btree([ ]).
btree(n(_,L,R)):- btree(L) , btree(R).
```



`btree(n(a,[],[])).`

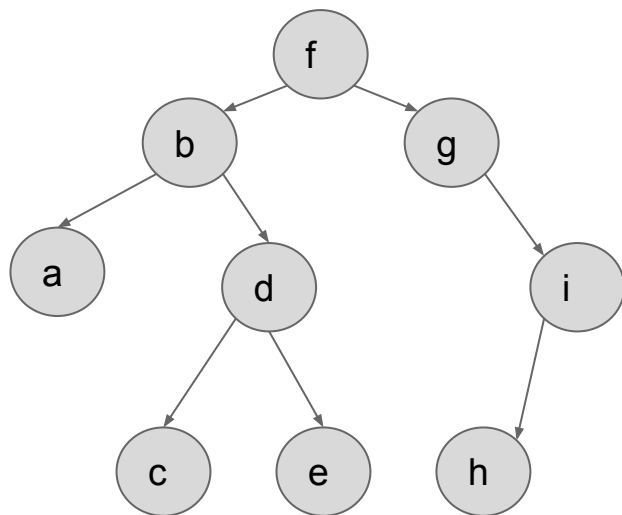


`btree(n(a,n(b,[],[]),[]))`



`btree(n(a, n(b,[],[]), n(c,[],[]))).`

Recorriendo un árbol



```
inorder([], []).  
inorder(n(X,L,R), Traversal) :-
```

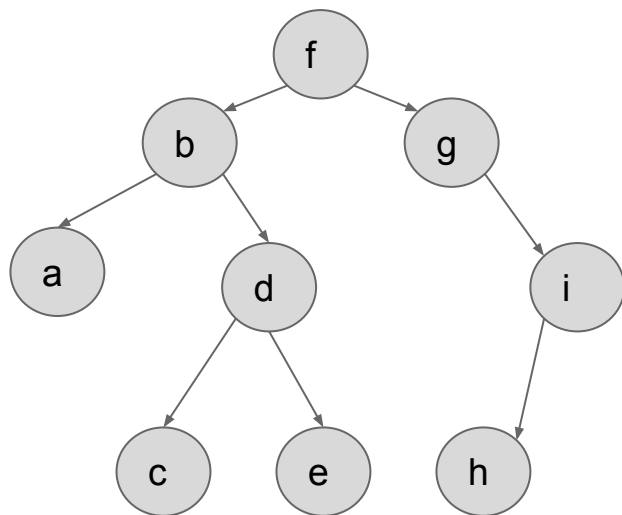
```
    inorder(L, L2),  
    inorder(R, L3),  
    append(L2, [X|L3], Traversal).
```

67 ?-

```
|  
|  
|   inorder(n(f,n(b,n(a,[],[]),n(d,n(c,[],[]),n(e,[],[]))),n(g,[],n(i,  
n(h,[],[],[]))),Traversal).  
Traversal = [a, b, c, d, e, f, g, h, i].
```

68 ?- |

Operaciones dentro de un arbol



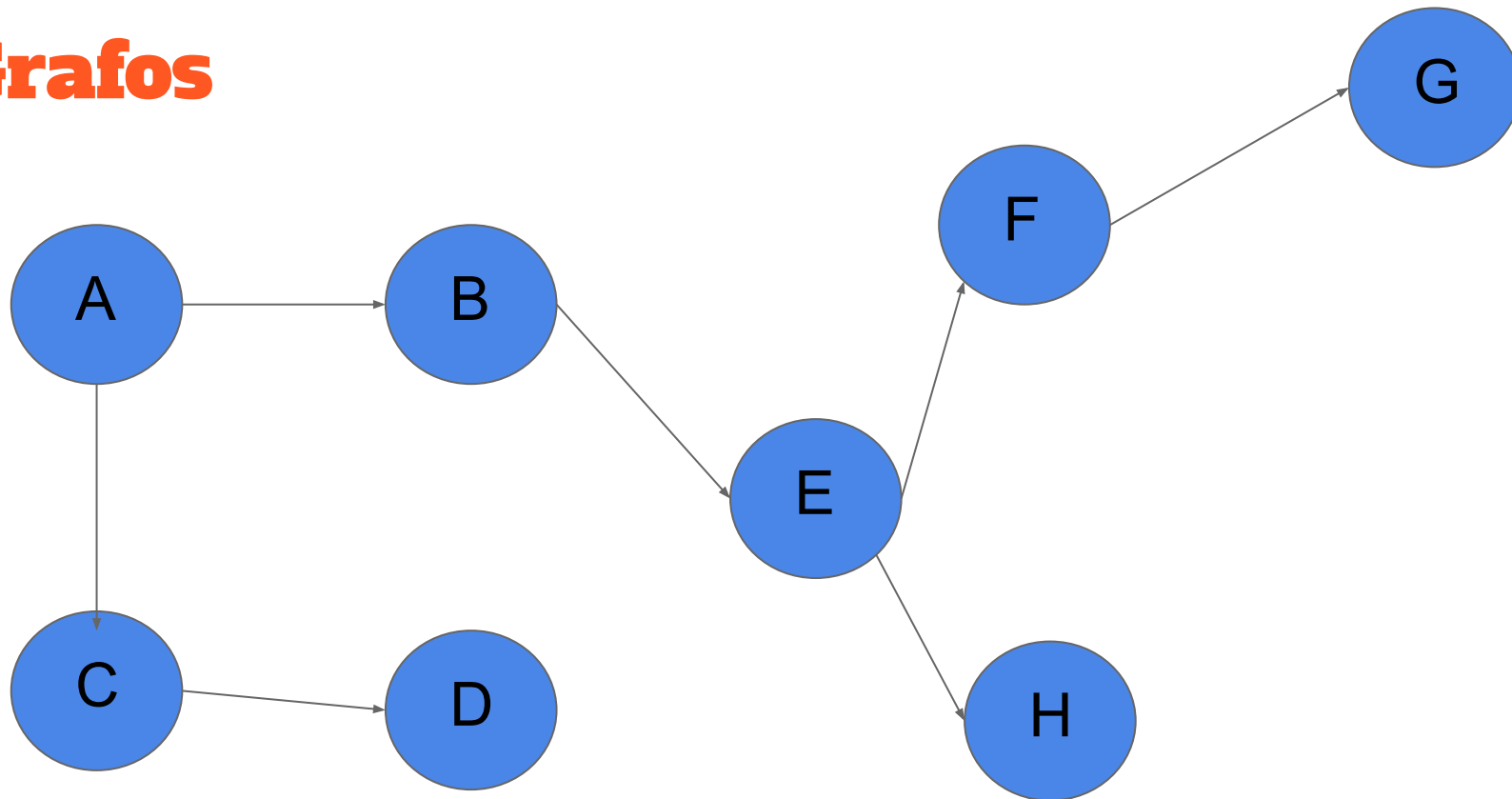
71 ?-

```
| count_leaves(n(f,n(b,n(a,[],[]),n(d,n(c,[],[]),n(e,[],[]))),n(g,[]  
| ,n(i,n(h,[],[]),[]))),N).
```

N = 4

```
count_leaves([],0).  
count_leaves(n(_,[],[]),1).  
count_leaves(n(_ ,L,R),N):-count_leaves(L,AUX1),count_leaves(R,AUX2),N is AUX1+AUX2.
```


Grafos



```
edge(a,b).
```

```
edge(a,c).
```

```
edge(c,d).
```

```
edge(b,e).
```

```
edge(e,h).
```

```
edge(e,f).
```

```
edge(f,g).
```

```
path(X,Y):-edge(X,Y).
```

```
path(X,Y):-edge(X,Z),path(Z,Y).
```

```
?- path(a,d).
```

```
true.
```

```
?- path(d,a).
```

```
false.
```

```
?- findall(X,path(a,X), S ).
```

```
S = [b, c, e, h, f, g, d].
```

```
?- findall(X,path(d,X), S ).
```

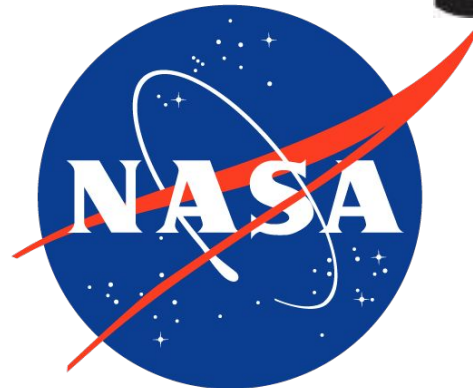
```
S = [].
```

```
?- findall(X,path(e,X), S ).
```

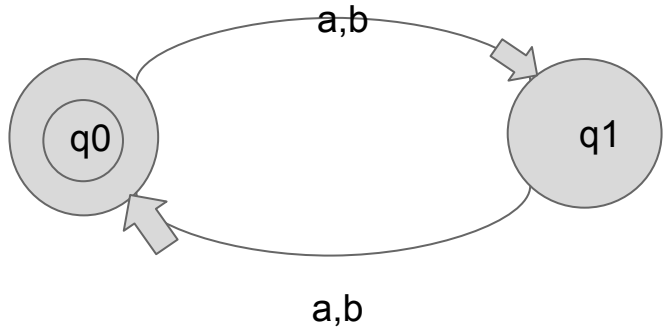
```
S = [h, f, g].
```

Aplicaciones

- Sistemas expertos que emulan la habilidad de un humano para la toma de decisiones.
- Software “clarissa” construido por la NASA para la ISS. Es una interfaz de voz que busca los procesos de la estación espacial.
- SICStus Prolog se encarga de la logística de la reservación de boletos de aerolíneas y ayudar a los ferrocarriles a operar mejor



Aplicaciones-Autómatas



automata.pl

File Edit Browse Compile Prolog Pce Help

tree.pl automata.pl

```
automata(S, [], S) .  
automata(S, [H|T], F) :- d(S, H, AUX), automata(AUX, T, F) .  
d(q0, a, q1) .  
d(q0, b, q1) .  
d(q1, a, q0) .  
d(q1, b, q0) .
```

82 ?- automata(q0,[a,b,a,a],q0).
true.

83 ?- automata(q0,[a,b,a],q0).
false.

84 ?- |

Cadenas de longitud par con el alfabeto a,b

Ventajas / Desventajas

Desventajas

- Curva de aprendizaje
- Se debe establecer muy bien los hechos o la representación del conocimiento, porque pueden haber soluciones erróneas.
- Forma de pensar es diferente a como estamos acostumbrados.

Ventajas

- El código tiende a ser mucho más corto, luego es fácil de modificar
- Facilidad para programar

Referencias

- SICS. SICStus Prolog TOmado del link
<https://www.sics.se/projects/sicstus-prolog-leading-prolog-technology>
- Stack Overflow. prolog applications. Tomado del link
<http://stackoverflow.com/questions/130097/real-world-prolog-usage>
- Rodney's Corner. Prolog and graphs. Tomado del link
<http://rlgomes.github.io/work/prolog/2012/05/22/19.00-prolog-and-graphs.html>
- SwiProlog. Built-in list operations. Tomado del link
<http://www.swi-prolog.org/pldoc/man?section=builtinlist>