LÓGICA PARA COMPUTACIÓN

Año 2026

PRÁCTICO 6: APLICACIONES DE LA LÓGICA

Lenguaje de Programación Lógico: PROLOG

Ejercicio 1:

Dado el siguiente programa Prolog:

```
pais(argentina).
pais(alemania).
pais(uruguay).
pais(francia).
pais(senegal).
pertenece(argentina,america).
pertenece(uruguay,america).
pertenece(alemania,europa).
pertenece(senegal,africa).
pertenece(francia,europa).
ranking_fifa(uruguay,16).
ranking_fifa(argentina,1).
ranking_fifa(senegal,18).
ranking_fifa(francia,2).
ranking_fifa(alemania,15).
```

Explique cual considera es el objetivo de cada una de las siguientes consultas, ejecutando manualmente los pasos realizados para responder cada una de ellas. En todos los casos, muestre todas las respuestas diferentes que devuelve el intérprete Prolog si se usa el **;**:

```
a) ?- pais(P).
b) ?- pais(P) , pertenece(P,america).
c) ?- pais(P) , ranking_fifa(P,R) , R < 17.
d) ?- pertenece(senegal,europa).
```

Ejercicio 2:

Dado el siguiente programa Prolog:

```
clasificado(argentina).
clasificado(italia).
clasificado(colombia).
no_clasificado(chile).
no_clasificado(peru).
grupo(h,colombia).
grupo(g,argentina).
```

```
grupo(g,italia).
pais(X) :- clasificado(X).
pais(X) :- no_clasificado(X).
```

1. Defina una consulta que permita conocer todos los países que están el grupo G de la Copa Mundial Femenina 2023.
2. Ejecute manualmente los pasos realizados para responder dicha consulta.

Ejercicio 3:

Suponga definidas las siguientes relaciones:

`padre(X,Y).`	X es padre de Y.
`madre(X,Y).`	X es madre de Y.
`hombre(X).`	X es hombre.
`mujer(X).`	X es mujer.
`progenitor(X,Y).`	X es padre o madre de Y.

Se pide definir en base a ellas las siguientes relaciones:

`esprogenitor(X).`	X es padre o madre.
`hija(X,Y).`	X es hija de Y.
`hermano_a(X,Y)`	X es hermano o hermana de Y.
`sobrino(X,Y).`	X es sobrino de Y.

Ejercicio 4

Dado el siguiente programa Prolog:

```
hombre(raul).
hombre(juan).
hombre(pedro).
hermano(raul,maria).
hermano(raul,juan).
hermano(raul,carlos).
madre(maria,luis).
madre(maria,hugo).
padre(carlos,jose).
tio(marcelo,hugo).
tio(X,Y) :- hombre(X), hermano(X,Z), progenitor(Z,Y).
```

```
progenitor(pedro,juan).
progenitor(X,Y) :- padre(X,Y).
progenitor(X,Y) :- madre(X,Y).
```

Ejecute manualmente los pasos realizados para responder las siguientes consultas, para el item `b)` provea al menos 3 respuestas diferentes que devolvería el intérprete Prolog (si se usara el **;** al menos 2 veces):

```
a) ?- tio(X,hugo).
b) ?- tio(R,S).
```

En el archivo `tio.pdf` se encuentra la ejecución paso a paso de una consulta similar al item a.

Ejercicio 5

Realice un programa Prolog que implemente el "mundo de los bloques" definido en el Ejercicio 6 del Práctico 5. Además, definir como reglas Prolog los enunciados que se pedían en dicho ejercicio.

Ejercicio 6

Realizar en Prolog las consultas correspondientes al Ejercicio 17 del Práctico 5.

Ejercicio 7

Implementar como un programa Prolog el vocabulario presentado en el Ejercicio 18 del Práctico 5, junto con los enunciado y propiedades que allí se describen.

Ejercicio 8

La función $exp(A, B)$ retorna el número A elevado a la potencia B. Recursivamente se define como:

$$exp(A,B) = \begin{cases} 1 \quad si \quad B = 0 \\ A * exp(A, B-1) \quad si \;\; B \geq 1 \end{cases}$$

El predicado **exp(A,B,C)** que se muestra a continuación implementa dicha función. El primer argumento es la base A, el segundo argumento es el exponente B (valor entero) y C es el resultado de la exponenciación.

```
exp(_,0,1).
exp(N,M,R) :- M \= 0, N1 is M - 1, exp(N,N1,R1),R is R1 * N.
```

Verifique que el predicado funciona correctamnte con el siguiente ejemplo:

```
?- exp(2,3,R).          R = 8
```

Ejercicio 9

La función $fact(N)$ retorna el *factorial* de N (es decir, el $N!$). Recursivamente se define como:

$$fact(N) = \begin{cases} 1 \quad si \quad N = 1 \\ fact(N-1) * N \quad si \quad N \geq 1 \end{cases}$$

Defina el predicado **fact** que implemente dicha función, donde el primer argumento es el número N, y el segundo argumento es el valor factorial correspondiente. Ejemplo:

```
?- fact(2,X).          X = 2
?- fact(4,24).         yes
```

Ejercicio 10

Defina el predicado **coFact** el cual, dado como primer argumento un número N, devuelve en el segundo argumento el cociente entre el factorial de N y N. Ejemplo:

```
?- coFact(3,X).           X = 2
?- coFact(5,24).           yes
```

Ejercicio 11

Responda cual es el resultado de las siguientes consultas:

```
a) ?-  persona(nombre(juan_carlos), edad(47), X) =  persona(Y, edad(Z), dni(20304645)).
b) ?-  nacimiento(16,11,1964) = nacimiento(_,_,X), Y = 2023 - X, Edad is 2023 - X.
c) ?-  alumno(Reg, ingre(fecha(1,03,2023))) = alumno(302722, X), ingre(fecha(_,M,_)) = X
```

Ejercicio 12

Asumamos que para representar *árboles binarios*, usamos una estructura (recursiva) `nodo(Valor,Si,Sd)`, donde `Valor` es la información contenida en el nodo, y `Si` y `Sd` son los árboles binarios que constituyen los subárboles izquierdo y derecho del nodo. Por otra parte, un árbol vacío lo denotaremos con la constante `nil`. Dado el siguiente programa:

```
res(nil, 0).
res(nodo(V,SI,SD), R) :- res(SI, R1),
                         res(SD, R2),
                         R is V + R1 + R2.
```

Evalúe las siguientes invocaciones:

```
a) ?- res(nil,R).
b) ?- res(nodo(7,nil,nil),R).
c) ?- res(nodo(7,nil,nodo(9,nil,nil)),R).
```

Ejercicio 13

Escriba el resultado de las siguientes consultas:

```
a) ?- [a,b,c] = [a|[b,c]].
b) ?- [a,b,c] = [a,b|[c]].
c) ?- [a,b,c] = [a|[c]].
d) ?- [a,b,c] = [a,b,c|[]].
e) ?- [5,2,3,7] = [X,2,3,Y].
f) ?- [1,6,2,4] = [X,3,2,Y].
g) ?- [d,g,f] = [_|_].
h) ?- [6,8,10,12] = [6,8,10|Y].
i) ?- [[1],2,[3]] = [X|Y].
j) ?- [[1],2,[3]] = [X,_|Z].
k) ?- [9,[3,X,2],Q] = [Y,[R,[2],T],[4|[]]].
```

Ejercicio 14

A partir del siguiente programa Prolog:

```
p([],_,0).
p([X|L],X,R):- p(L,X,R1),
               R is 1 + R1.
p([X|L],Y,R):- X \= Y,
               p(L,Y,R).
```

Responda cual es el resultado de la siguiente consulta, mostrando claramente la ejecución paso a paso:

```
p([a,b,a],a,R).
```

Ejercicio 15

Responda cual es el resultado de las siguientes consultas en Prolog:

```
a) ?-  d(a,[1,2,3],p(3)) =  d(a,A,B).
b) ?- [f(5,[]),f(3,1)] = [f(A,B),C].
```

Ejercicio 16

Dada la siguiente definición del predicado **concate**:

```
concate([], L2, L2).
concate([X|L1], L2, [X|L3]) :- concate(L1, L2, L3).
```

Evalúe las siguientes invocaciones:

```
a) ?- concate([x,y],Y,[x,y,t,u]).
b) ?- concate(X,Y,[u,x,z,t]).
```

Ejercicio 17

Defina el predicado **longi** que calcula la longitud de la lista. Ejemplos:

```
?- longi([],0).                    yes
?- longi([4,3,2,1],X).            X = 4
?- longi([pepe,f(4),[5]],X).      X = 3
```

NOTA: Tenga en cuenta que para la programación de los siguientes predicados, puede utilizar predicados definidos en ejercicios anteriores.

Ejercicio 18

Defina el predicado **prom**, el cual dado como primer argumento una lista de enteros, obtiene en su segundo argumento el promedio de los elementos del primer argumento. Ejemplos:

```
?- prom([8,1,5,2],R).
R = 4
?- prom([1,2,3,4,5,6],R).
R = 3.5
```

Ejercicio 19

Defina el predicado **elimE** que permita eliminar de una lista, el elemento que se encuentra en la posición suministrada. Ejemplos:

```
?- elimE(1,[3,2,1,1],X).          X = [2,1,1]
?- elimE(3,[3,2,1,2],[3,2,2]).       yes
```

Ejercicio 20

Defina el predicado `sum_int_pre` el cual toma como primer argumento una lista de precios de productos, en su segundo y tercer argumentos son precios. Este predicado obtendrá en su cuarto argumento la suma de todos los precios, con valor entre el intevalos de los valores del segundo y tercer argumento incluyendo los mismos.

Ejemplos:

```
?- sum_int_pre([47.30,28.10,50.24,80.0],45.0,70.0,R).
R = 97.54
```

```
?- sum_int_pre([72.50,15.10,80],20.0,80,R).
R = 152.5
```

Ejercicio 21

Defina un predicado `incre_imp` que reciba como parámetro una lista L de valores enteros. Este predicado retorna en su segundo argumento una lista R. La lista R contiene los valores de las posiciones impares de L incrementado en 1 y los valores de las posiciones pares sin modificar.

Ejemplos:

```
?- incre_imp([],R).
?- R = []
?- incre_imp([1],R).
?- R = [2]
?- incre_imp([1,9,4],R).
?- R = [2,9,5]
?- incre_imp([2,6,5,1,2,2],R).
?- R = [3,6,6,1,3,2]
```

Ejercicio 22

Dado el siguiente programa, diga cual es su funcionalidad:

```
p([],[]).
p([f(Y,_)],Y).
p([f(Y,_),f(_,Z)|T],[Y,Z|T1]) :- p(T,T1).
```

Diga cual es el resultado cuando se ejecuta la siguiente consulta:

```
p([f(8,5),f(2,5),f(9,1),f(5,6)],R).
```

Ejercicio 23

Defina el predicado **sum_comp**, el cual dado como primer argumento una lista de estructuras $par(X, Y)$, obtiene en su segundo argumento la suma de todos los valores del primer componente de las estructuras. Ejemplos:

```
?- sum_comp([par(-3,5),par(7,4)],R).
R = 4
?- sum_comp([par(9,6),par(5,5),par(1,7)],R).
R = 15
```

Ejercicio 24

Suponga una lista que contiene estructuras `o(T,C)`, donde el T es una letra y el C es la cantidad de ocurrencias de la letra T. Defina el predicado `may` que toma como primer argumento una lista con el formato definido más arriba, como segundo argumento un entero V. Este predicado devolverá en su tercer argumento una lista `R`, con las letras de la lista del primer argumento cuya cantidad de ocurrencias supera el valor V.

Ejemplo:

```
?- may([o(a,7),o(b,2),o(c,5)],4,R).
R = [a,c]

?- may([o(d,2),o(a,1),o(f,2)],3,R).
R = []

?- may([o(e,5),o(c,2),o(d,3),o(a,3)],2,R).
R = [e,d,a]
```