

Programación Lógica y Funcional

M. C. Efraín Solares

Ingeniería en Sistemas Computacionales
Instituto Tecnológico de Culiacán

Enero-Junio 2017



Tabla de verdad para la equivalencia

Tabla: Tabla de verdad \leftrightarrow

p	q	$p \leftrightarrow q$
V	V	V
V	F	F
F	V	F
F	F	V



equivalencias lógicas

$$p \wedge q \iff q \wedge p$$

$$p \vee q \iff q \vee p$$

$$(p \wedge q) \wedge r \iff p \wedge (q \wedge r)$$

$$(p \vee q) \vee r \iff p \vee (q \vee r)$$

$$p \wedge (q \vee r) \iff (p \wedge q) \vee (p \wedge r) \quad p \vee (q \wedge r) \iff (p \vee q) \wedge (p \vee r)$$

$$p \wedge T \iff p$$

$$p \vee F \iff p$$

$$p \vee \sim p \iff T$$

$$p \wedge \sim p \iff F$$

$$\sim(\sim p) \iff p$$

$$p \wedge p \iff p$$

$$p \vee p \iff p$$

$$p \vee T \iff T$$

$$p \wedge F \iff F$$

$$\sim(p \wedge q) \iff (\sim p) \vee (\sim q)$$

$$\sim(p \vee q) \iff (\sim p) \wedge (\sim q)$$

$$p \vee (p \wedge q) \iff p$$

$$p \wedge (p \vee q) \iff p$$

$$(p \implies q) \iff (\sim p \vee q)$$

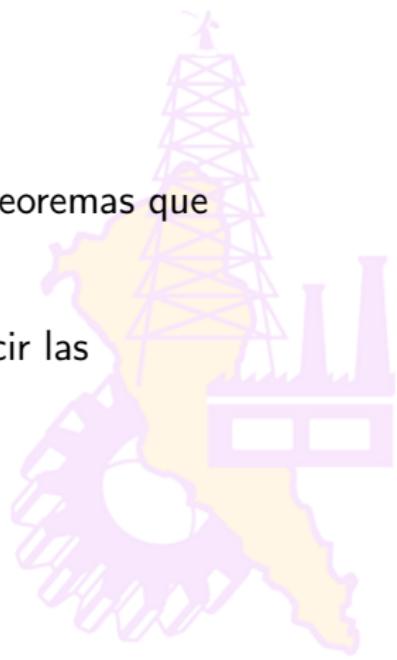
$$\sim(p \implies q) \iff (p \wedge \sim q)$$



Programación lógica con ProLog

Un programa lógico es un conjunto de axiomas y teoremas que definen relaciones entre objetos.

El cálculo de un programa lógico consiste en deducir las consecuencias del programa.



Componentes

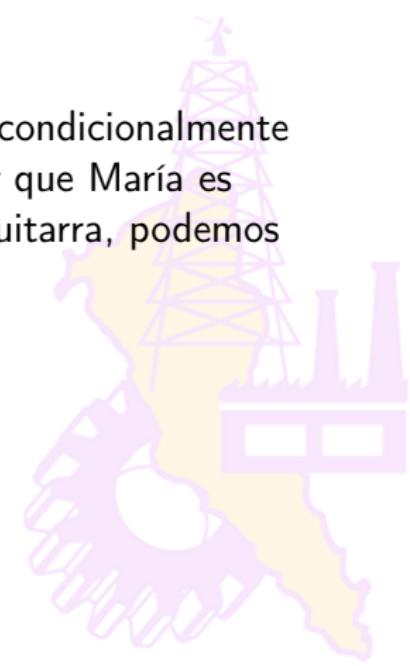
ProLog tiene solo tres componentes principales:

- Hechos.
- Reglas.
- Consultas.

Una colección de hechos y reglas es llamada *Base de Conocimiento* (o *Base de Datos*).



Hechos



Los hechos se usan para declarar cosas que son incondicionalmente verdaderas en el dominio de interés. Para expresar que María es mujer, que Juan es hombre y que María toca la guitarra, podemos expresar los hechos de la siguiente manera.

```
/*Base de conocimiento 1: bc1.pl*/
mujer(maria).
hombre(juan).
tocaGuitarra(maria).
```



?- mujer(maria).

?- tocaGuitarra(juan).

?- tocaGuitarra(pedro).

?- hombre(juan).



```
/*Base de conocimiento 2: bc2.pl*/
escuchaMusica(maria).
esFeliz(yolanda).
tocaGuitarra(maria) :- escuchaMusica(maria).
tocaGuitarra(yolanda) :- escuchaMusica(yolanda).
escuchaMusica(yolanda):- esFeliz(yolanda).
```

Las reglas indican información que es condicionalmente verdadera del dominio de interés. Por ejemplo, la primera regla dice que María toca la guitarra si escucha música, y la última regla dice que Yolanda escucha música si ella está feliz. Más generalmente, el :- debe leerse como "si", o "está implicado por". La parte en el lado izquierdo del :- se llama cabeza de la regla, la parte en el lado derecho se llama cuerpo. Así que en general las reglas dicen: si el cuerpo de la regla es verdad, entonces la cabeza de la regla es verdad también.

Importante: si una base de conocimiento contiene una regla cabeza :- cuerpo, y ProLog sabe que el cuerpo es verdadero, a partir de la información en la base de conocimientos, entonces ProLog puede inferir la cabeza.

```
?- tocaGuitarra(maria) .
```



?- tocaGuitarra(yolanda) .

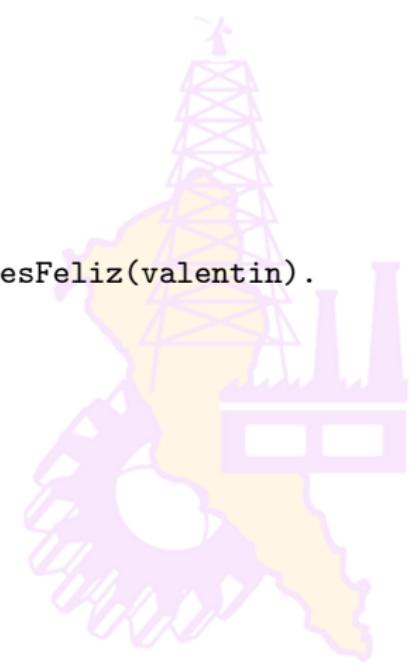


Los hechos y las reglas contenidas en una base de conocimiento se llaman cláusulas. Así la bc2 contiene cinco cláusulas: tres reglas y dos hechos.

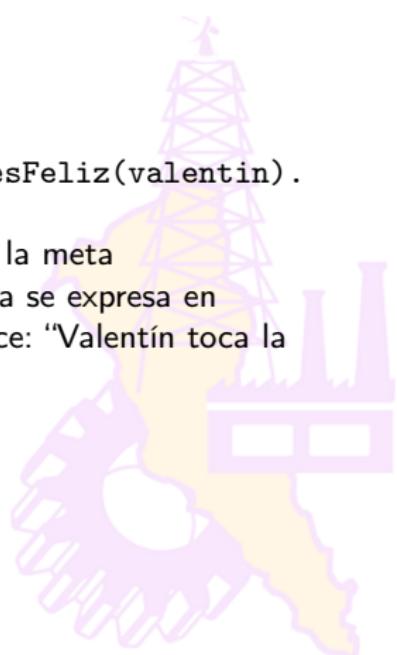
Otra manera de mirar la bc2 es decir que consiste de tres predicados:

- escuchaMusica
- esFeliz
- tocaGuitarra

El predicado esFeliz se define utilizando una sola cláusula (un hecho). Los predicados escuchaMusica y tocaGuitarra se definen cada uno mediante dos cláusulas (en ambos casos, dos reglas).



```
/*Base de conocimiento 3: bc3.pl*/
esFeliz(valentin).
escuchaMusica(ben).
tocaGuitarra(valentin):-escuchaMusica(valentin),esFeliz(valentin).
tocaGuitarra(ben):-esFeliz(ben).
tocaGuitarra(ben):-escuchaMusica(ben).
```



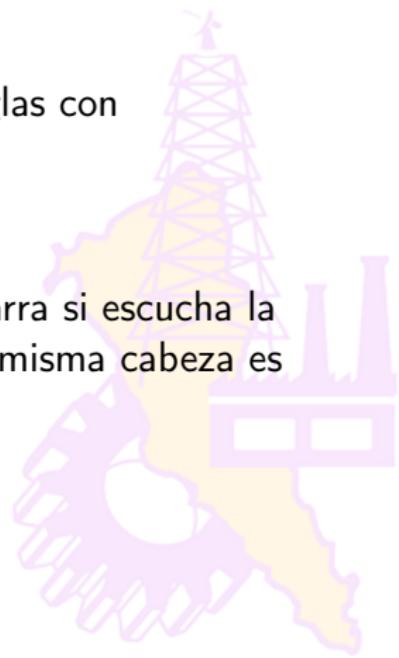
La regla

`tocaGuitarra(valentin):-escuchaMusica(valentin),esFeliz(valentin).`
contiene dos elementos en su cuerpo, llamados *metas*.

La coma que separa la meta `escuchaMusica(valentin)` y la meta `esFeliz(valentin)` es la forma en que la conjunción lógica se expresa en Prolog (es decir, la coma significa \wedge). Así que esta regla dice: “Valentín toca la guitarra si escucha música y está feliz”.

```
?- tocaGuitarra(valentin).
```





La base de conocimiento contiene también dos reglas con exactamente la misma cabeza, a saber:

`tocaGuitarra(ben) :- esFeliz(ben), ..`, y
`tocaGuitarra(ben) :- escuchaMusica(ben).`

Esta es una manera de decir que Ben toca la guitarra si escucha la música o si es feliz. (Listar múltiples reglas con la misma cabeza es una manera de expresar la disyunción, \vee .)



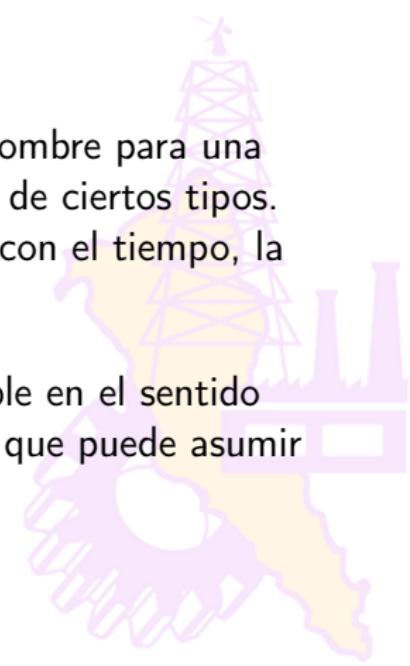
Hay otra manera de expresar la disyunción en Prolog. Podríamos reemplazar el par de reglas por la regla siguiente.

```
tocaGuitarra(ben) :- esFeliz(ben) ; escuchaMusica(ben).
```

```
?- tocaGuitarra(ben).
```



Variables



En los lenguajes imperativos, una variable es un nombre para una ubicación de memoria que puede almacenar datos de ciertos tipos. Aunque el contenido de la ubicación puede variar con el tiempo, la variable siempre apunta a la misma ubicación.

Una variable en programación lógica es una variable en el sentido matemático, es decir, es el nombre de un símbolo que puede asumir cualquier valor.

```
/*Base de conocimiento 4: bc4.pl*/
mujer(maria).
mujer(julia).
mujer(yolanda).
ama(valentin,maria).
ama(marcelo,maria).
ama(pedro,helena).
ama(helena,pedro).
```



?- mujer(X).

Prolog responde a esta consulta buscando a través de bc4, de arriba hacia abajo, intentando igualar (o unificar) la expresión mujer(X) con la información contenida en bc4.



```
?- ama(marcelo,X),mujer(X).
```



```
/*Base de conocimiento 5: bc5.pl*/
ama(valentin,maria).
ama(marcelo,maria).
ama(pedro,helena).
ama(helena,pedro).
cela(X,Y) :- ama(X,Z),ama(Y,Z).
```

El predicado cela expresa la regla “X cela a Y si X ama a Z y Y ama a Z”.



```
?- cela(marcelo,W).
```



Sintaxis



Símbolos disponibles:

- Mayúsculas (A, …, Z).
- Minúsculas (a, …, z).
- Dígitos (0,1, …, 9).
- Símbolos especiales (+, -, *, /, <, >, =, :, ., &, ~, espacio en blanco, _).

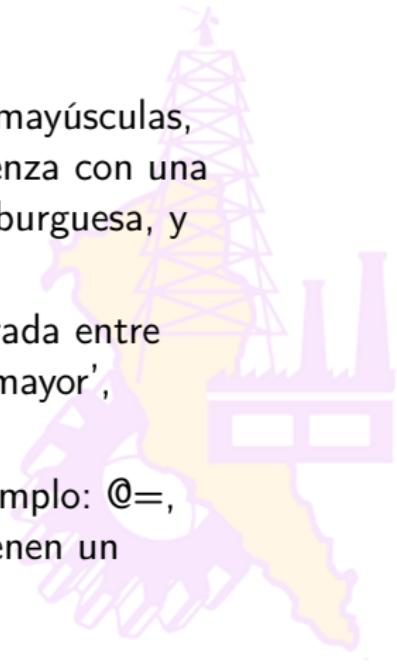
Los componentes de ProLog (hechos, reglas y consultas) están formados por *términos*. Existen cuatro tipos de términos:

- Átomos.
- Números.
- Variables.
- Términos complejos.

Los átomos y números se agrupan para formar las constantes del lenguaje, y las constantes y variables forman los términos simples .

Átomos

Un átomo es cualquiera de los siguientes.

- 
- ① Una cadena de caracteres formada por letras mayúsculas, minúsculas, dígitos o el guión bajo, que comienza con una letra minúscula. Por ejemplo: ben, gran_hamburguesa, y m_monroe2.
 - ② 2. Una secuencia arbitraria de símbolos encerrada entre comillas simples. Por ejemplo, 'Valentin', 'El mayor', 'Dos_Pesos', '& ^ % & \$ & *' y ''.
 - ③ 3. Una cadena de símbolos especiales. Por ejemplo: @=, ===>, ;, :, -. Algunos de estos átomos, tienen un significado predefinido.

Variables

Una variable es una cadena de letras mayúsculas, minúsculas, dígitos y guión bajo que comienza con una letra mayúscula o con un guión bajo. Por ejemplo, X, Y, Variable, _etiqueta, X_526, Lista.

La variable _ (un solo guión bajo) es llamada *variable anónima*.

Términos complejos

A menudo llamados estructuras, los términos complejos son formados a partir de átomos, constantes y variables. Un término complejo es construido a partir de un *funtor* o *functor* seguido por una secuencia de *argumentos*. Los argumentos se colocan entre corchetes, separados por comas, y después del functor. El functor debe ser un átomo. Es decir, las variables no pueden ser usadas como functores. Por otro lado, los argumentos pueden ser cualquier tipo de término. Algunos ejemplos de términos complejos son

- mujer(maria),
- tocaGuitarra(ben),
- ama(helena,pedro).

platica(X, padre(padre(padre(ben))))



El número de argumentos que un término complejo tiene se llama *aridad*. Por ejemplo, `mujer(maria)` es un término complejo con aridad 1, mientras que `ama(valentin, maria)` es un término complejo con aridad 2.

Cuando nos referimos a los predicados usaremos el sufijo / seguido de su aridad. Por ejemplo

`mujer/1,`
`ama/2.`

Tarea

- ① ¿Cuáles de las siguientes secuencias de caracteres son átomos, cuáles son variables, y cuáles no son ninguno de los anteriores?
- a) vICENTE
 - b) Mensaje
 - c) variable23
 - d) Variable2000
 - e) gran _ Hamburguesa
 - f) 'gran Hamburguesa'
 - g) gran Hamburguesa
 - h) 'Julio'
 - i) '_Julio'
 - j) '_ Julio'



Tarea

- ② ¿Cuáles de las siguientes secuencias de caracteres son átomos, cuáles son variables, cuáles son términos complejos, y cuáles no son términos en absoluto? Dé el functor y la aridad de cada término complejo.
- a) ama(Vicente,maria)
 - b) 'ama(Vicente,maria)'
 - c) Ben(boxeador)
 - d) boxeador(Ben)
 - e) y(gran(hamburguesa),deliciosa(hamburguesa))
 - f) y(gran(X),deliciosa(X))
 - g) _y(gran(X),deliciosa(X))
 - h) Ben cela Vicente)
 - i) cela(Ben Vicente)
 - j) cela(Ben,Vicente)



Tarea

③ ¿Cuántos hechos, reglas, cláusulas y predicados hay en la siguiente base de conocimiento? ¿Cuáles son las cabezas y cuáles son las metas de las reglas?

- a) mujer(vicente).
- b) mujer(maria).
- c) hombre(julio).
- d) persona(X) :- hombre(X); mujer(X).
- e) ama(X,Y) :- conoce(Y,X).
- f) padre(Y,Z) :- hombre(Y), hijo(Z,Y).
- g) padre(Y,Z) :- hombre(Y), hija(Z,Y).



Tarea

④ Represente lo siguiente en Prolog:

- a) Ben es un celoso.
- b) Maria y Marcelo están casados.
- c) Zed está dormido.
- d) Marcelo cela a todos los que le dan a María un masaje.
- e) María ama a todos los que son buenos bailarines.
- f) Julio come cualquier cosa nutritiva o sabrosa.



Tarea

5 Supongamos la siguiente base de conocimiento:

mago(ron).

tieneVarita(harry).

jugadorQuidditch(harry).

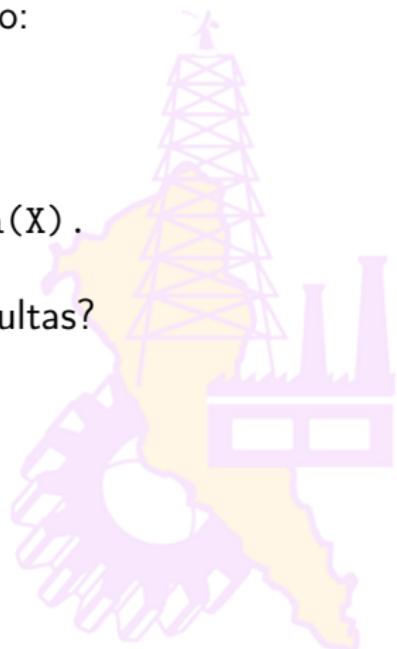
mago(X) :- tieneEscoba(X), tieneVarita(X).

hasBroom(X) :- quidditchPlayer(X).

¿Cómo responde ProLog a las siguientes consultas?

Argumente su respuesta.

- a) mago(ron).
- b) bruja(ron).
- c) mago(hermione).
- d) bruja(hermione).
- e) mago(harry).
- f) mago(Y).
- g) bruja(Y).

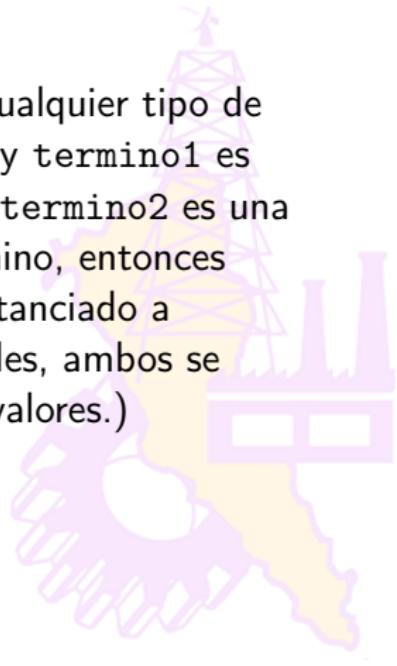


Coincidencia de términos

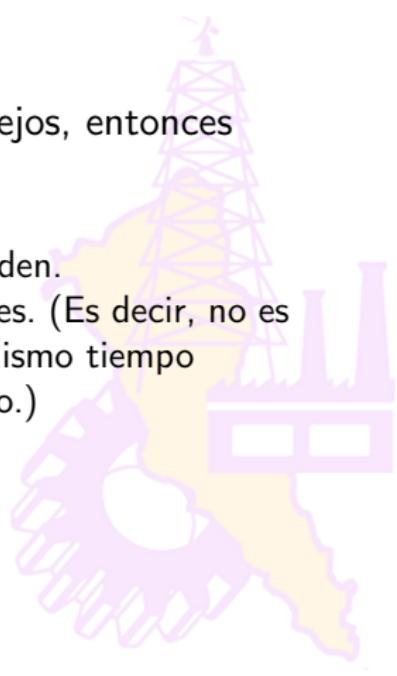
Dos términos coinciden si son iguales o si contienen variables que pueden ser instanciadas de tal manera que los términos resultantes sean iguales.

Formalmente, `termino1` y `termino2` coinciden si cumplen con las siguientes condiciones.

- ① Si `termino1` y `termino2` son constantes, entonces ambos términos coinciden si y solo si son el mismo átomo o el mismo número.

- 
- ② Si termino1 es una variable y termino2 es cualquier tipo de término, entonces ambos términos coinciden, y termino1 es instanciado a termino2. Del mismo modo, si termino2 es una variable y termino1 es cualquier tipo de término, entonces ambos términos coinciden, y termino2 es instanciado a termino1. (Por lo tanto, si ambos son variables, ambos se instancian entre sí y decimos que comparten valores.)



- 
- ③ Si termino1 y termino2 son términos complejos, entonces coinciden si y solo si
- a) Tienen el mismo functor y aridad.
 - b) Todos sus argumentos correspondientes coinciden.
 - c) Las instanciaciones de variables son compatibles. (Es decir, no es posible instanciar la variable X a maria y al mismo tiempo instanciarla a vicente. Ver ejemplo más abajo.)

- 
- ④ Dos términos coinciden si y solo si se deduce de las tres cláusulas anteriores que coinciden.



Algunos términos que coinciden son

`mujer(maria)`

`mujer(maria)`

`mujer(maria)`

`mujer(X)`

Algunos términos que no coinciden son

`mujer(maria)`

`mujer(mary)`

`ama(maria, X)`

`ama(X, vicente)`



Tarea

¿Cuáles de los siguientes pares de términos coinciden? Argumente su respuesta.

- ① pan = pan
- ② 'Pan' = pan
- ③ Pan = pan
- ④ pan = salchicha
- ⑤ comida(pan) = pan
- ⑥ comida(pan) = X
- ⑦ alimentos(X) = alimentos(pan)
- ⑧ comida(pan, X) = comida(Y, salchicha)
- ⑨ comida(pan, X, cerveza) = comida(Y, salchicha, X)
- ⑩ comida(pan, X, cerveza) = comida(Y, gran_hamburguesa)
- ⑪ alimento(X) = X
- ⑫ comida(desayuno(pan), bebida(cerveza)) = comida(X, Y)
- ⑬ comida(desayuno(pan), X) = comida(X, bebida(cerveza))



```
/*Base de conocimiento 6: bc6.pl*/
vertical(linea(punto(X,Y),punto(X,Z))).
horizontal(linea(punto(X,Y),punto(Z,Y))).
```





Tarea 03. Dadas las siguientes bases de conocimiento grafique el espacio de búsqueda para las consultas correspondientes

```
/*Base de conocimiento 6: bc6.pl*/
```

```
f(a).
```

```
f(b).
```

```
g(a).
```

```
g(b).
```

```
h(b).
```

```
k(X) :- f(X), g(X), h(X).
```

```
?- k(X).
```

```
/*Base de conocimiento 7: bc7.pl*/
ama(vicente,maria).
ama(marcelo,maria).
cela(X,Y) :- ama(X,Z),ama(Y,Z).

?- cela(X,Y).
```



Grafique el espacio de búsqueda para la consulta oracion(Palabra1, Palabra2, Palabra3, Palabra4, Palabra5) ., con respecto a la siguiente base de conocimiento. ¿Cuáles son las oraciones que esta “gramática” puede generar?

palabra(Articulo, a).

palabra(Articulo, cada).

palabra(Sustantivo, criminal).

palabra(Sustantivo, 'hamburguesa grande').

palabra(Verbo, come).

palabra(Verbo, gusta).

oracion(Palabra1, Palabra2, Palabra3, Palabra4, Palabra5) :-

palabra(Articulo, Palabra1),

palabra(Sustantivo, Palabra2),

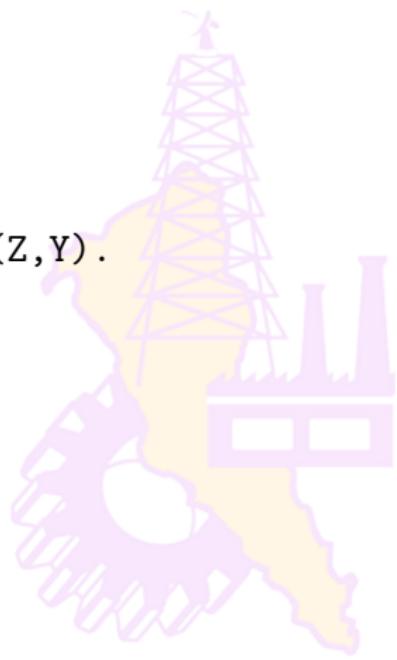
palabra(Verbo, Palabra3),

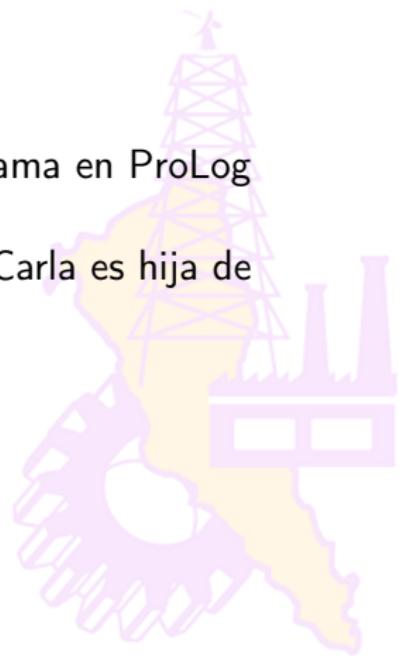
palabra(Articulo, Palabra4),

palabra(Sustantivo, Palabra5).

Recursividad

```
digiere(X,Y) :- acaba_comer(X,Y).  
digiere(X,Y) :- acaba_comer(X,Z), digiere(Z,Y).  
acaba_comer(mosquito,sangre(juan)).  
acaba_comer(rana,mosquito).  
acaba_comer(garza,rana).
```





Dado el siguiente árbol genealógico, cree un programa en ProLog que permita consultar si X es descendiente de Y.

Marta es hija de Carla. Carolina es hija de Laura. Carla es hija de Carolina. Laura es hija de Rosa.

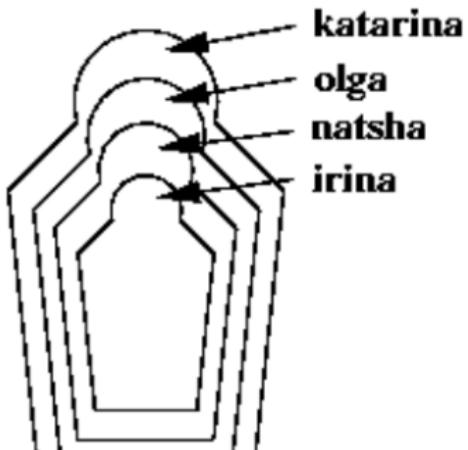


Tarea 04. ¿Cuál es el espacio de búsqueda para la siguiente consulta?

? - descendiente(X,Y) .

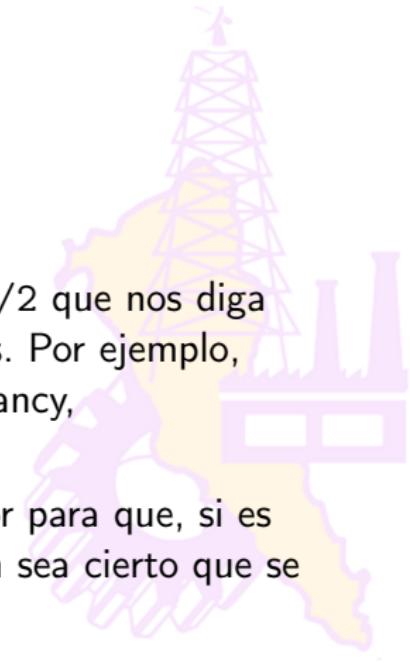
Tarea 05

Escriba una base de conocimiento usando el predicado `directamenteDentro/2` que represente qué muñeca está directamente contenida en qué otra muñeca. Luego, defina un predicado (recursivo) `dentro/2`, que nos diga qué muñeca está (directa o indirectamente) contenida en qué otra muñeca. Por ejemplo, la consulta `dentro(katarina, natsha)` debe evaluar a Falso, mientras que `dentro(olga, katarina)` debe evaluar a Ciento.



Considere la siguiente base de conocimiento.

```
viajarDirecto(forbach,saarbruecken).  
viajarDirecto(freyming,forbach).  
viajarDirecto(fahlquemont,stAvold).  
viajarDirecto(stAvold,forbach).  
viajarDirecto(saarbruecken,dudweiler).  
viajarDirecto(metz,fahlquemont).  
viajarDirecto(nancy,metz).
```

- 
- ① Escriba un predicado recursivo viajarEntre/2 que nos diga cuándo podemos viajar en entre dos ciudades. Por ejemplo, cuando se le da la consulta viajarEntre (nancy, saarbruecken) debería evaluar a Cierto.
 - ② Extienda la formulación del predicado anterior para que, si es cierto que se puede viajar de A a B, También sea cierto que se puede viajar de B a A.

Listas

[maria, vicente, julio, yolanda]

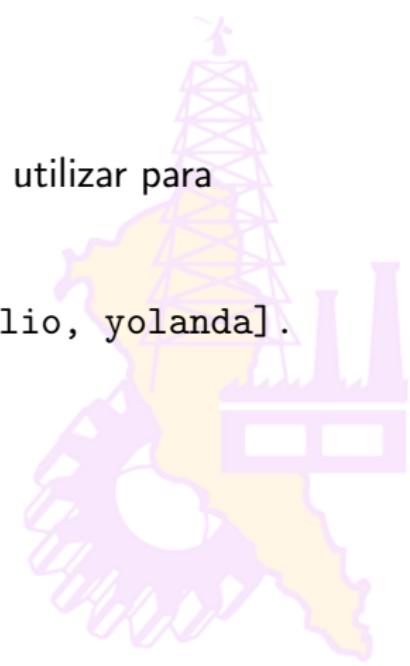
[maria, salta(conejo), X, 2, maria]

[]

[maria, [vicente, julio], [vicente, novia(vicente)]]

[[], dormido(vicente), [2, [b, julio]], [], Z, [2, [b, julio]]]]

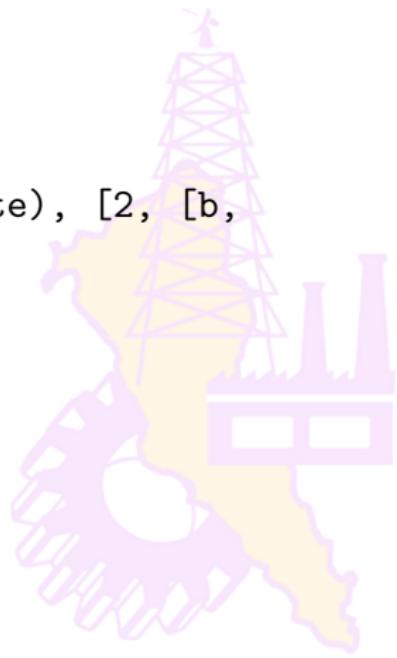




Prolog tiene un operador especial, |, que se puede utilizar para descomponer una lista en su cabeza y cola.

```
?- [Cabeza | Cola] = [maria, vicente, julio, yolanda].
```

```
?- [_, X, _, Y | _) = [[ ], dormido(vicente), [2, [b, julio]], [ ], Z, [2, [b, julio]]].
```



Escriba una base de conocimiento que permita conocer si X es miembro una lista dada. Por ejemplo, la consulta

? - esMiembro(maria, [maria, vicente, julio, yolanda]).

debe evaluar a Ciento.

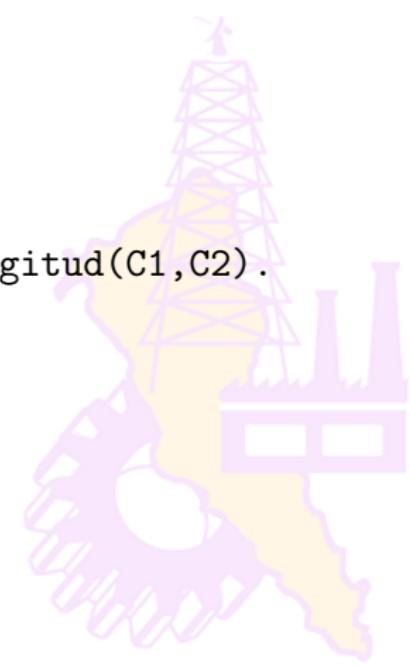
```
esMiembro(X, [X|_]).  
esMiembro(X, [_|C]) :- miembro(X,C).
```





Escriba una base de conocimiento que permita conocer si dos listas tienen la misma cantidad de elementos.

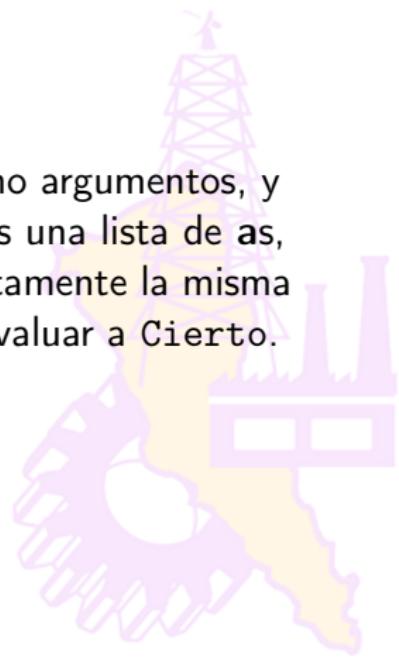
```
mismaLongitud([], []).  
mismaLongitud([_|C1], [_|C2]) :- mismaLongitud(C1, C2).
```



Tarea 06

Necesitamos un predicado que tome dos listas como argumentos, y evalúe a Cierto si y solo si el primer argumento es una lista de **as**, y el segundo argumento es una lista de **bs** de exactamente la misma longitud. Por ejemplo, la siguiente consulta debe evaluar a Cierto.

```
aYb([a,a,a,a],[b,b,b,b]).
```





X is 3+5.

X is 3+2*5.

X is mod(7,2).

cuadrado(X,Y) :- Y is X*X.

is(X,+(3,2)).



2 < 4. true

2 <= 4. true

4 <= 4. true

4=:=4. true

4=\=5. true

4=\=4. false

4 >= 4. true

Escriba una base de conocimiento que permita la longitud de una lista.



```
long([],0).  
long([_|Xs],N) :- long(Xs,X), N is X+1.
```



Tarea 07

Suponga que tenemos la siguiente base de conocimiento.

```
tran(eins,uno).  
tran(zwei,dos).  
tran(drei,tres).  
tran(vier,cuatro).  
tran(fuenf,cinco).  
tran(sechs,seis).  
tran(sieben,siete).  
tran(acht,ocho).  
tran(neun,nueve).
```

Escriba una cláusula que permita traducir una lista de números en alemán y letra a la lista de números en español y letra correspondiente. Por ejemplo:

```
traducir([zwei,acht,drei],X).  
X = [dos, ocho, tres].
```

El programa debería funcionar también en la otra dirección. Por ejemplo:

```
traducir(X, [dos, ocho, tres]).  
X = [zwei, acht, drei], X.
```



Tarea 08

Escriba un término complejo con aridad 2 cuyos argumentos son listas. La lista en el segundo argumento debe consistir en cada elemento de la primera lista escrito dos veces. Por ejemplo:

```
doble([1,2,3],X).  
X = [1,1,2,2,3,3].
```

