

# Taller 7

Luis Felipe Garzón Bonilla 1089931169

Steven Grisales López 1034289634

Se propuso realizar los mismos ejercicios de los talleres 2 y 3 pero esta vez se planteaban los hechos con listas y las reglas igualmente se realizaban con listas. Sin embargo, del taller 2 solo se realiza el primer punto.

## Primera parte (Taller 2).

### Definición de hechos:

```
es_mujer([mona, marge, lisa, maggie, jacqueline, patty, selma, ling]).
es_hombre([abraham, herbert, homero, bart, clancy]).

es_padre(abraham,[herbert,homero]).
es_padre(homero,[bart,lisa,maggie]).
es_padre(clancy,[patty, selma, marge]).


es_madre(mona,[herbert,homero]).
es_madre(marge,[bart,lisa,maggie]).
es_madre(jacqueline,[patty,selma,marge]).
es_madre(selma,[ling]).
```

### Puntos propuestos:

1. Abuelo / Abuela.

```
es_abuelo(Abuelo,Nieto):-
    es_padre(Abuelo, Lhijos),
    member(HijoA, Lhijos),
    (es_padre(HijoA,Lnietos); es_madre(HijoA,Lnietos)),
    member(Nieto,Lnietos).

es_abuela(Abuela,Nieto):-
    es_madre(Abuela, Lhijos),
    member(HijoA, Lhijos),
    (es_madre(HijoA,Lnietos); es_padre(HijoA,Lnietos)),
    member(Nieto,Lnietos).
```


	<code>es_abuela(jacqueline,Nieto).</code>
	<b>Nieto = ling</b>
	<b>Nieto = bart</b>
	<b>Nieto = lisa</b>
	<b>Nieto = maggie</b>
	<b>false</b>
	<code>es_abuelo(abraham,Nieto).</code>
	<b>Nieto = bart</b>
	<b>Nieto = lisa</b>
	<b>Nieto = maggie</b>
	<b>false</b>

Hay varias formas de hacer la consulta, podemos preguntar si alguien es abuelo/abuela de alguien o podemos preguntar que abuelos existen y sus respectivos nietos (incluye nietas), en este caso la consulta sería `es_abuelo(Abuelo, Nieto)` o `es_abuela(Abuela, Nieto)`.

## 2. Hijo / Hija.

```
es_hijo(Hijo,Padres):-  
    es_hombre(Lhombres),  
    member(Hijo,Lhombres),  
    (es_padre(Padres,Lhijos); es_madre(Padres,Lhijos)),  
    member(Hijo,Lhijos).
```


```
es_hija(Hijo,Padres):-  
    es_mujer(Lmujeres),  
    member(Hijo,Lmujeres),  
    (es_padre(Padres,Lhijos); es_madre(Padres,Lhijos)),  
    member(Hijo,Lhijos).
```

 es\_hijo(bart,Padres).

**Padres = homero**

**Padres = marge**

**false**

 es\_hija(ling,Padres).


**Padres = selma**

Hay varias formas de hacer la consulta, podemos preguntar si alguien es hijo/hija de alguien o podemos preguntar que hijos existen y sus respectivos padres (incluye a mamás), en este caso la consulta sería `es_hijo(Hijo, Padres)` o `es_hija(Hija, Padres)`.

## 3. Nieto / Nieta.

```
es_nieto(Nieto,Abuelos) :-  
    es_hombre(Lhombres),  
    member(Nieto,Lhombres),  
    (es_padre(Abuelos,Lhijos); es_madre(Abuelos,Lhijos)),  
    member(HijoA,Lhijos),  
    (es_padre(HijoA,Lnietos); es_madre(HijoA,Lnietos)),  
    member(Nieto,Lnietos).
```

```
es_nieta(Nieta,Abuelos) :-  
    es_mujer(Lmujeres),  
    member(Nieta,Lmujeres),  
    (es_padre(Abuelos,Lhijos); es_madre(Abuelos,Lhijos)),  
    member(HijoA,Lhijos),  
    (es_padre(HijoA,Lnietos); es_madre(HijoA,Lnietos)),  
    member(Nieta,Lnietos).
```

 es\_nieto(bart,Abuelos).


**Abuelos = abraham**

**Abuelos = clancy**

**Abuelos = mona**

**Abuelos = jacqueline**

**false**

 es\_nieta(ling,Abuelos).

**Abuelos = clancy**

**Abuelos = jacqueline**


**false**

Hay varias formas de hacer la consulta, podemos preguntar si alguien es nieto/nieta de alguien o podemos preguntar que nietos existen y sus respectivos abuelos (incluye a abuelas), en este caso la consulta sería `es_nieto(Nieto, Abuelos)` o `es_nieta(Nieta, Abuelos)`.

#### 4. Hermano / Hermana.


```
es_hermano(Hermano,Hermanos):-  
    es_hombre(Lhombres),  
    member(Hermano,Lhombres),  
    (es_padre(_,Lhijos); es_madre(_,Lhijos)),  
    (member(Hermano,Lhijos), member(Hermanos,Lhijos)),  
    Hermano \= Hermanos.
```

```
es_hermana(Hermana,Hermanos):-  
    es_mujer(Lmujeres),  
    member(Hermana,Lmujeres),  
    (es_padre(_,Lhijos); es_madre(_,Lhijos)),  
    (member(Hermana,Lhijos), member(Hermanos,Lhijos)),  
    Hermana \= Hermanos.
```

 `es_hermano(bart,Hermanos).`

**Hermanos** = lisa

**Hermanos** = maggie

 `es_hermana(lisa,Hermanos).`

**Hermanos** = bart


**Hermanos** = maggie

Hay varias formas de hacer la consulta, podemos preguntar si alguien es hermano/hermana de alguien o podemos preguntar que hermanos existen y sus respectivos hermanos (incluye a hermanas), en este caso la consulta sería `es_hermano(Hermano, Hermanos)` o `es_hermana(Hermana, Hermanos)`.

#### 5. Tío / Tía.

```
es_tio(Tio,Sobrinos):-  
    es_hermano(Tio,Padres),  
    (es_padre(Padres,Lhijos); es_madre(Padres,Lhijos)),  
    member(Sobrinos, Lhijos).
```


```
es_tia(Tia,Sobrinos):-  
    es_hermana(Tia,Padres),  
    (es_padre(Padres,Lhijos); es_madre(Padres,Lhijos)),  
    member(Sobrinos, Lhijos).
```

 `es_tio(herbert,Sobrinos).`

**Sobrinos** = bart

**Sobrinos** = lisa

**Sobrinos** = maggie

 `es_tia(selma,Sobrinos).`

**Sobrinos** = bart

**Sobrinos** = lisa

**Sobrinos** = maggie

Hay varias formas de hacer la consulta, podemos preguntar si alguien es tío/tía de alguien o podemos preguntar que tíos existen y sus respectivos sobrinos (incluye a sobrinas), en este caso la consulta sería `es_tio(Tio, Sobrinos)` o `es_tia(Tia, Sobrinos)`.

## 6. Primo / Prima.

```
es_primo(Primo,Primos):-
    es_hombre(Lhombres),
    member(Primo,Lhombres),
    (es_tio(Tios,Primo); es_tia(Tios,Primo)),
    (es_padre(Tios, Lhijos); es_madre(Tios,Lhijos)),
    member(Primos, Lhijos).

es_prima(Prima,Primas):-
    es_mujer(Lmujeres),
    member(Prima,Lmujeres),
    (es_tio(Tios,Prima); es_tia(Tios,Prima)),
    (es_padre(Tios, Lhijos); es_madre(Tios,Lhijos)),
    member(Primas, Lhijos).
```

```
es_primo(bart,Primos).
Primos = ling

es_prima(ling,Primos).
Primos = bart
Primos = lisa
Primos = maggie
```

Hay varias formas de hacer la consulta, podemos preguntar si alguien es primo/prima de alguien o podemos preguntar que primos existen y sus respectivos primos (incluye a primas), en este caso la consulta sería `es_primo(Primo1, Primo2)` o `es_prima (Prima1, Prima2)`.

## 7. Sobrino / Sobrina.

```
es_sobrino(Sobrino,Tios) :-
    es_hombre(Lhombres),
    member(Sobrino,Lhombres),
    (es_hermano(Tios,Padres); es_hermana(Tios,Padres)),
    (es_padre(Padres, Lhijos); es_madre(Padres, Lhijos)),
    member(Sobrino, Lhijos).

es_sobrina(Sobrina,Tios) :-
    es_mujer(Lmujeres),
    member(Sobrina,Lmujeres),
    (es_hermano(Tios,Padres); es_hermana(Tios,Padres)),
    (es_padre(Padres, Lhijos); es_madre(Padres, Lhijos)),
    member(Sobrina, Lhijos).
```

```
es_sobrino(bart,Tios).
Tios = herbert
Tios = herbert
Tios = patty
Tios = patty
Tios = selma
Tios = selma

es_sobrina(ling,Tios).
Tios = marge
Tios = marge
Tios = patty
Tios = patty
```

Hay varias formas de hacer la consulta, podemos preguntar si alguien es sobrino/sobrina de alguien o podemos preguntar que sobrinos existen y sus respectivos tíos (incluye a tías), en este caso la consulta sería `es_sobrino(Sobrino, Tios)` o `es_sobrina(Sobrina, Tios)`.

**ACLARACIÓN:** En los puntos 6 y 7 no sabemos dónde colocar el corte (;) para que no suelte todos los caminos que encuentre, así que lo dejamos de esa manera.

## Segunda parte (Taller 3).

### Definición de hechos:

```
/* Ejercicio: Mismos enunciados y consultas del taller 3 modelado mediante listas */

conexion(vancouver, [[edmonton,16], [calgary,13]]).
conexion(edmonton, [saskatoon,12]).
conexion(calgary, [[edmonton,4], [regina,14]]).
conexion(regina, [[saskatoon,7], [winnipeg,4]]).
conexion(saskatoon, [[calgary,9], [winnipeg,20]]).
```

### Puntos propuestos:

1. ¿Existe una conexión entre Saskatoon y Vancouver?

```
/* Preguntas */
% 1-¿Existe conexión entre Saskatoon y Vancouver
hay_conexion(Origen, Destino) :-
    conexion(Origen, LDestinosCosto),
    member([Destino|_], LDestinosCosto).
/* Consultas: hay_conexion(vancouver, saskatoon). -> false.
* hay_conexion(vancouver, Destino) -> Destino = edmonton, calgary. */
```

```
hay_conexion(saskatoon, vancouver).
false
hay_conexion(vancouver, Destino).
Destino = edmonton
Destino = calgary
```

En esta consulta, existen 2 formas de realizarlas para que arroje un resultado efectivo, la cual, la primera sería preguntar si hay una conexión directa de una ciudad a otra, en caso verdadero arrojará verdadero y en caso falso, arrojará el falso. La segunda forma de aplicar esta consulta es para preguntar cuantas conexiones directas tiene una ciudad, en este caso, mostrándonos que las conexiones directas de Vancouver son Edmonton y Calgary.

2. ¿Con qué nodos está conectado Regina y cuál es el costo de cada conexión?

```
hay_conexion(Origen, Destino) :-
    conexion(Origen, LDestinosCosto),
    member([Destino|_], LDestinosCosto).
/* Consultas: hay_conexion(vancouver, saskatoon). -> false.
* hay_conexion(vancouver, Destino) -> Destino = edmonton, calgary. */

costo(Origen, Destino, Costo) :- conexion(Origen, Intermedio), buscar(Destino, Intermedio, Costo).

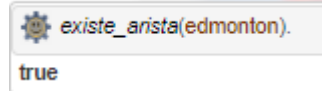
costo(Origen, Destino, Costo) :- costo(Origen, Intermedio, C1),
    costo(Intermedio, Destino, C2),
    Costo is C1 + C2, !.
```

```
hay_conexion(regina, Destino).
Destino = saskatoon
Destino = winnipeg
costo(regina, saskatoon, Costo).
Costo = 7
costo(regina, winnipeg, Costo).
Costo = 4
```

En esta pregunta, para responderla se recurre primeramente a verificar conexiones de Regina, con la regla de si existe conexión, luego, se recurre a la regla de costo, que nos muestra el costo que hay entre estas conexiones directas.

3. Construir una regla para determinar si un nodo tiene aristas.

```
% 3- Construir regla para determinar si un nodo tiene aristas.  
existe_arista(Origen):- conexion(Origen, [_]).
```



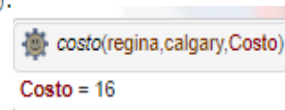
```
existe_arista(edmonton).  
true
```

Esta regla, nos verifica si una ciudad tiene alguna conexión, mediante una comprobación de listas en este caso, el ejemplo de Edmonton siendo verdad, hay otra consulta, `existe_arista(Origen)`, la cual nos indica todos los nodos que poseen aristas.

4. Construir una regla para determinar cuál es el costo para ir de un nodo X a un Z pasando por Y.

```
costo(Origen, Destino, Costo):- conexion(Origen, Intermedio), buscar(Destino, Intermedio, Costo).
```

```
costo(Origen, Destino, Costo):- costo(Origen, Intermedio, C1),  
    costo(Intermedio, Destino, C2),  
    Costo is C1 + C2, !.
```



```
costo(regina, calgary, Costo)  
Costo = 16
```

Esta regla es la misma usada anteriormente, ya que funciona de forma recursiva y recorre las ciudades sumando el costo general, ya que Regina y Calgary no tienen conexión directa se suma el costo que tiene por pasar por Saskatoon, luego de este a Calgary.