



Práctica 1: PROLOG Avanzado

CE Inteligencia Artificial y Big Data
Modelos de Inteligencia Artificial
2024/2025

Daniel Marín López

Índice

1. Problema 1	3
2. Problema 2	6

1. Problema 1

Partiendo de la [Actividad 4](#), del árbol genealógico de una familia. Además de dar de alta a los miembros de la familia, se crearon algunas relaciones de parentesco pero faltan otras tales como: suegro, suegra, yerno, nuera, cuñado, cuñada, tío, tía, primo y prima. Programar dichas reglas en lenguaje PROLOG y comprobar su correcto funcionamiento.

Partiendo del ejercicio anterior, se pueden establecer las relaciones en PROLOG siguiendo este esquema:

- X es suegro de Y **SI** Y está casado/a con un hijo/a de X.
- Y es yerno/nuera de X **SI** Y está casado/a con un hijo/a de X.
- Y es cuñado de Z **SI** Y está casado/a con su hermano/a.
- T es tío/a de Y **SI** T es hermano del padre/madre de Y.
- Y y A son primos/as **SI** sus padres son hermanos.

Partiendo de eso vamos a realizar las relaciones en PROLOG:

1. Suegro/a

```
suegro(Suegro, Persona):-
    padre(Suegro, Esposo),
    son_pareja(Persona, Esposo).
```

```
suegra(Suegra, Persona):-
    madre(Suegra, Esposo),
    son_pareja(Persona, Esposo).
```

The screenshot shows a Prolog interpreter interface with two query windows. The first window shows the query `suegro(miguelII, isabelIII).` with the result `true` and a green '1' indicating one solution. The second window shows the query `suegra(carmenII, isabelIII).` with the result `true` and a green '1'. At the bottom, there is a third window with the query `?- suegra(carmenII, isabelIII).` and buttons for 'Examples', 'History', 'Solutions', 'table results', and a 'Run!' button.

Vemos que efectivamente, Miguel II y Carmen II son suegros de Isabel III.

2. Yerno/Nuera

```

yerno(Yerno, Persona):-
    son_pareja(Yerno, Hijo),
    progenitor(Persona, Hijo),
    hombre(Yerno).

```

```

nuera(Nuera, Persona):-
    son_pareja(Nuera, Hijo),
    progenitor(Persona, Hijo),
    mujer(Nuera).

```

The screenshot shows a Prolog interpreter window with two queries entered and executed successfully. The first query is `yerno(miguelIII, carmenI).` and the second is `nuera(idoiaIII, miguelIII).`. Both return `true` with a count of 1. The interface includes a toolbar with a gear icon, a dropdown arrow, a minus sign, and a close button. Below the queries, there are tabs for 'Examples', 'History', and 'Solutions'. At the bottom right, there is a checkbox for 'table results' and a blue 'Run!' button.

3. Cuñado/Cuñada

```

cuñado(Cuñado, Persona):-
    (son_pareja(Persona, Pareja), hermanos(Cuñado, Pareja); hermanos(Hermano, Persona),
    son_pareja(Cuñado, Hermano)),
    hombre(Cuñado).

```

```

cuñada(Cuñada, Persona):-
    (son_pareja(Persona, Pareja), hermanos(Cuñada, Pareja); hermanos(Hermana, Persona),
    son_pareja(Cuñada, Hermana)),
    mujer(Cuñada).

```

The screenshot shows a Prolog interpreter window with two queries entered and executed successfully. The first query is `cuñada(idoiaIII, fernandolIII).` and the second is `cuñado(juanIII, isabelIII).`. Both return `true` with a count of 1. The interface includes a toolbar with a gear icon, a dropdown arrow, a minus sign, and a close button. Below the queries, there are tabs for 'Examples', 'History', and 'Solutions'. At the bottom right, there is a checkbox for 'table results' and a blue 'Run!' button.

4. Tío/Tía

```
tio(Tio, Sobrino):-
    es_padre(Padre, Sobrino), hermano(Tio, Padre).
```

```
tia(Tia, Sobrino):-
    es_padre(Padre, Sobrino), hermana(Tia, Padre).
```

The screenshot shows a Prolog interpreter interface. The top panel contains two queries, each with a gear icon, a downward arrow, a minus sign, and a close button. The first query is `tia(victoriall, fernandolll).` and the second is `tio(fernandoll, juanlll).`. Both queries return `true` and are marked with a green '1' on the right. The bottom panel shows a prompt `?- tio(fernandoII, juanIII).` followed by a large empty text area. At the bottom, there are buttons for 'Examples▲', 'History▲', and 'Solutions▲', a checkbox for 'table results', and a 'Run!' button.

5. Primo/Prima

```
primo(Primo, Persona):-
    es_padre(P1, Persona), hermanos(P1, P2), es_padre(P2, Primo), hombre(Primo).
```

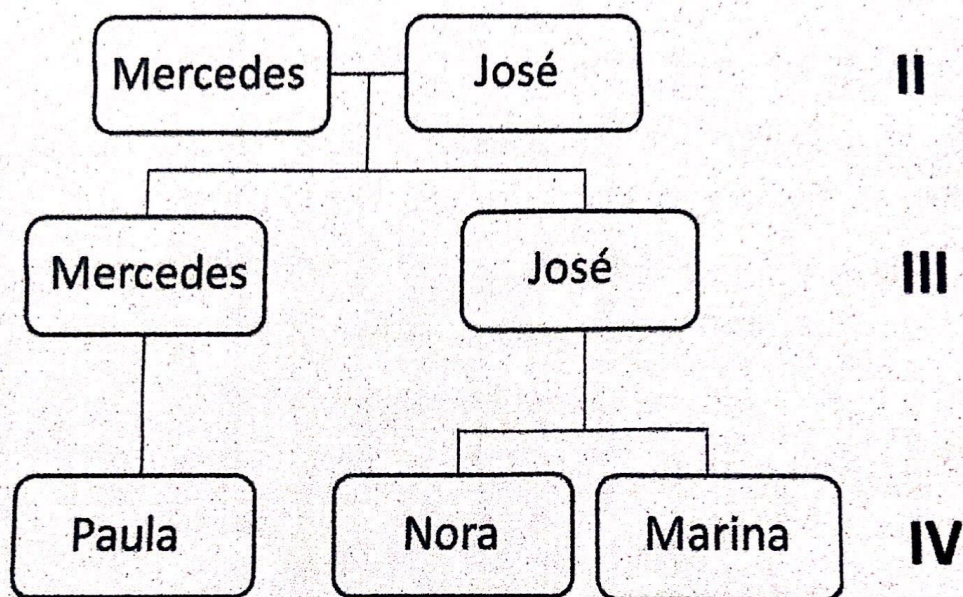
```
prima(Prima, Persona):-
    es_padre(P1, Persona), hermanos(P1, P2), es_padre(P2, Prima), mujer(Prima).
```

The screenshot shows a Prolog interpreter interface. The top panel contains a query with a gear icon, a downward arrow, a minus sign, and a close button. The query is `primo(miguellV, juanIV).` and it returns `true`, marked with a green '1' on the right. The bottom panel shows a prompt `?- primo(miguelIV, juanIV).` followed by a large empty text area. At the bottom, there are buttons for 'Examples▲', 'History▲', and 'Solutions▲', a checkbox for 'table results', and a 'Run!' button.

2. Problema 2

Siguiendo con el árbol genealógico propuesto en la [Actividad 4](#), resulta posible no solo la implementación de nuevas reglas tal y como se ha hecho en el punto anterior, sino también dar de alta a más miembros de la familia. Así, el siguiente árbol presenta a una nueva rama de la familia, en la que la persona de nombre

Mercedes, perteneciente a la generación II, es la misma que se encuentra en el árbol de la [Actividad 4](#). Partiendo de dicha información, añade los individuos que aparecen en el siguiente árbol al mismo programa y ejecutar sobre ellas tanto las funciones implementadas en dicha Actividad como las programadas en el punto 1 anterior.



Se han añadido las siguientes relaciones:

```

hombre(joseII).
hombre(joseIII).
mujer(mercedesIII).
mujer(paulaIV).
mujer(noraIV).
mujer(marinaIV).

```

```

progenitor(mercedesII, mercedesIII).
progenitor(joseII, mercedesIII).
progenitor(mercedesII, joseIII).
progenitor(joseII, joseIII).
progenitor(mercedesIII, paulaIV).
progenitor(joseIII, noraIV).
progenitor(joseIII, marinaIV).

















































```

















```

matrimonio(mercedesII, joseII).

```

Ahora procedemos con realizar algunas de las reglas que hay definidas previamente:

 <i>padre</i> (josell, mercedesIII).	  
true	1
 <i>madre</i> (mercedesII, joseIII).	  
true	1
 <i>hermano</i> (joseIII, mercedesIII).	  
true	1
 <i>hermana</i> (mercedesIII, joseIII).	  
true	1
 <i>esposo</i> (mercedesII, josell).	  
true	1
 <i>esposa</i> (mercedesII, josell).	  
true	1
 <i>nieto</i> (joseIII, leandrol).	  
true	1
 <i>nieta</i> (mercedesIII, leandrol).	  
true	1
 <i>suegro</i> (leandrol, josell).	  
true	1
 <i>suegra</i> (mercedesI, josell).	  
true	1
 <i>yerno</i> (josell, mercedesI).	  
true	1
 <i>cuñado</i> (josell, migueIII).	  
true	1

 <i>tio</i> (joseIII, paulaIV).	  
true	1
 <i>tia</i> (mercedesIII, marinaIV).	  
true	1
 <i>prima</i> (paulaIV, marinaIV).	  
true	1
 <i>primo</i> (juanIII, mercedesIII).	  
true	1