



# **Actividad 4: Árboles Genealógicos**

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

Daniel Marín López

## 1. Problema

Se pide realizar una serie de consultas en Prolog en SWISH sobre este árbol genealógico:



**Figura 5.10** Árbol genealógico.

Para ello tenemos el siguiente conocimiento:

```

hombre(fernandoI).
hombre(leandroI).
hombre(fernandoII).
hombre(miguelII).
hombre(mercedesII).
hombre(fernandoIII).
hombre(juanIII).
hombre(miguelIV).
hombre(tomasIV).
hombre(juanIV).
hombre(ignacioIV).
  
```

```

mujer(carmenI).
mujer(mercedesI).
mujer(palomaII).
mujer(carmenII).
mujer(victoriaII).
mujer(montserratII).
mujer(isabelIII).
mujer(idoiaIII).
  
```

```

% relaciones de progenitura
progenitor(fernandoI, fernandoII).
progenitor(fernandoI, palomaII).
progenitor(fernandoI, carmenII).
progenitor(fernandoI, victoriaII).
progenitor(fernandoI, montserratII).
  
```

```

progenitor(carmenI, fernandoII).
progenitor(carmenI, palomaII).
progenitor(carmenI, carmenII).
progenitor(carmenI, victoriaII).
progenitor(carmenI, montserratII).
progenitor(leandroI, miguelII).
progenitor(leandroI, mercedesII).
progenitor(mercedesI, miguelII).
progenitor(mercedesI, mercedesII).
progenitor(carmenII, fernandoIII).
progenitor(carmenII, juanIII).
progenitor(miguelII, fernandoIII).
progenitor(miguelII, juanIII).
progenitor(fernandoIII, miguelIV).
progenitor(fernandoIII, tomasIV).
progenitor(isabelIII, miguelIV).
progenitor(isabelIII, tomasIV).
progenitor(juanIII, juanIV).
progenitor(juanIII, ignacioIV).
progenitor(idoiaIII, juanIV).
progenitor(idoiaIII, ignacioIV).

% relaciones de matrimonio
matrimonio(carmenI, fernandoI).
matrimonio(mercedesI, leandroI).
matrimonio(carmenII, miguelII).
matrimonio(isabelIII, fernandoIII).
matrimonio(idoiaIII, juanIII).

% relaciones de parentesco
padre(Padre, Hijo):-hombre(Padre), progenitor(Padre, Hijo).
madre(Madre, Hijo):-mujer(Madre), progenitor(Madre, Hijo).

hermanos(Hermano1, Hermano2):-progenitor(Progenitor, Hermano1),
    progenitor(Progenitor, Hermano2), not(Hermano1==Hermano2).

hermano(Hermano1, Hermano2):-hombre(Hermano1), hermanos(Hermano1, Hermano2).
hermana(Hermano1, Hermano2):-mujer(Hermano1), hermanos(Hermano1, Hermano2).

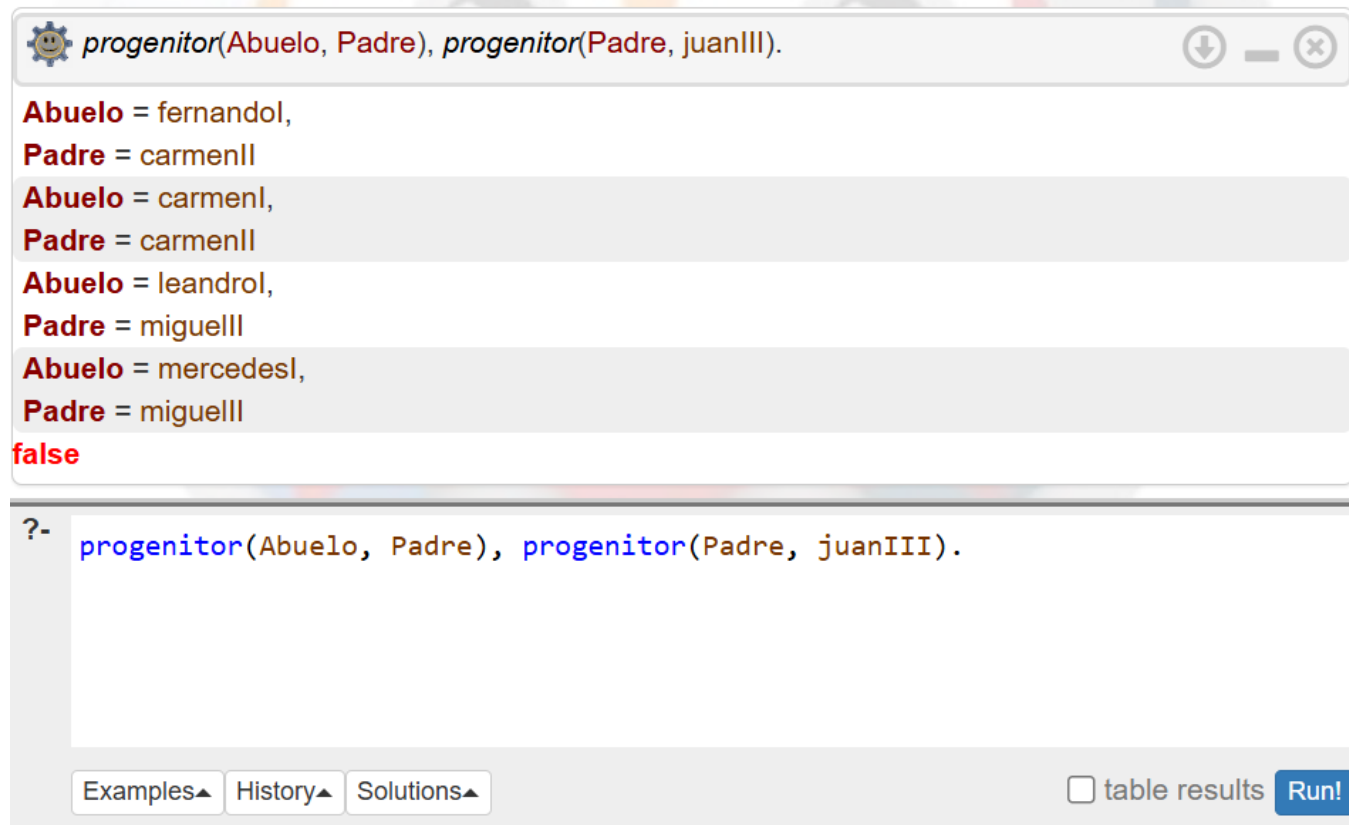
esposo(Mujer, Marido):-matrimonio(Mujer, Marido), hombre(Marido).
esposa(Mujer, Marido):-matrimonio(Mujer, Marido), mujer(Mujer).

nieto(Nieto, Abuelos):-progenitor(Abuelos, Padres), progenitor(Padres, Nieto), hombre(Nieto).
nieta(Nieta, Abuelos):-progenitor(Abuelos, Padres), progenitor(Padres, Nieta), mujer(Nieta).

```

Se plantean las siguientes consultas:

a) Obtener el listado de los cuatro abuelos de Juan, persona que corresponde a la tercera generación de la familia. Tenemos que llamar a la función `progenitor` 2 veces, 1 para sacar al padre de Juan III y otra para sacar al abuelo.



The screenshot shows a Prolog IDE window with the following content:

```
progenitor(Abuelo, Padre), progenitor(Padre, juanIII).
```

The output area displays the following results:

```
Abuelo = fernandol,
Padre = carmenII
Abuelo = carmenI,
Padre = carmenII
Abuelo = leandrol,
Padre = miguellI
Abuelo = mercedesI,
Padre = miguellI
false
```

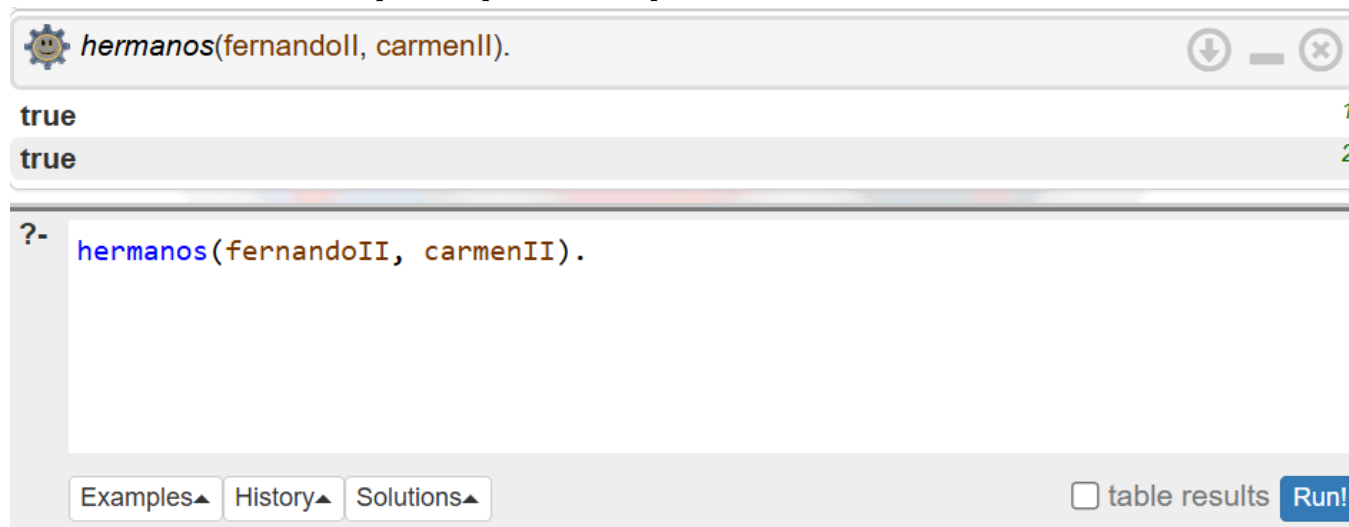
Below the output, there is a query input field with the text: `?- progenitor(Abuelo, Padre), progenitor(Padre, juanIII).`

At the bottom, there are buttons for "Examples", "History", "Solutions", a checkbox for "table results", and a "Run!" button.

Vemos que primero se imprimen los abuelos maternos y luego los paternos.

b) Comprobar si Fernando y Carmen, pertenecientes a la segunda generación, son hermanos.

Usaremos la función `hermanos` para comprobar si estas personas son hermanos



The screenshot shows a Prolog IDE window with the following content:

```
hermanos(fernandoII, carmenII).
```

The output area displays the following results:

```
true 1
true 2
```

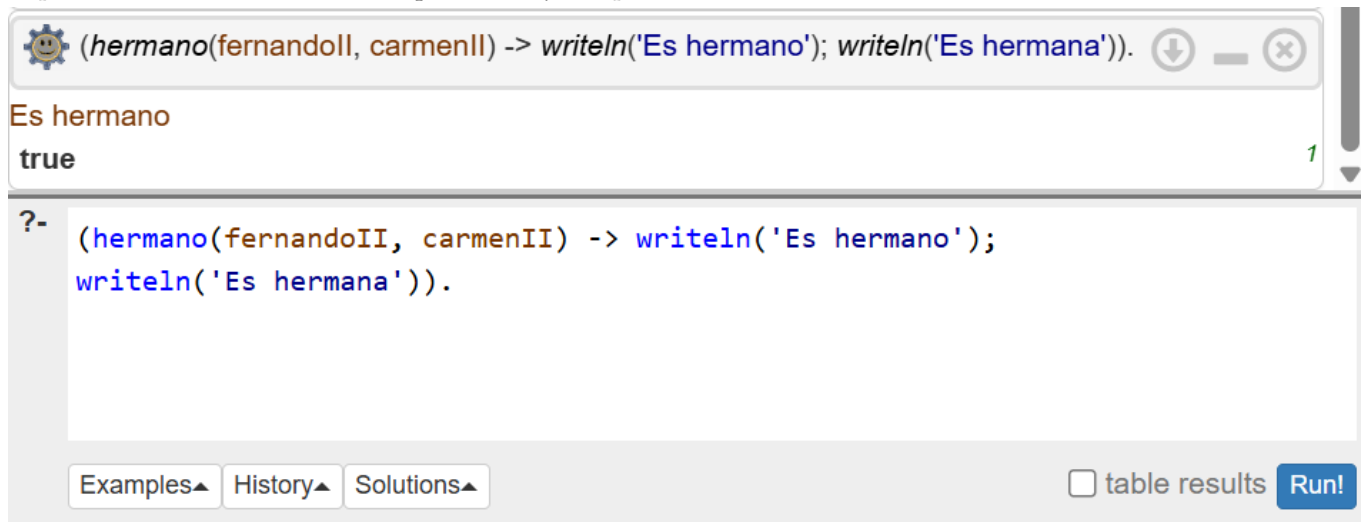
Below the output, there is a query input field with the text: `?- hermanos(fernandoII, carmenII).`

At the bottom, there are buttons for "Examples", "History", "Solutions", a checkbox for "table results", and a "Run!" button.

Vemos que sale True dos veces, es posible que se haga la comprobación desde los dos lados y por eso salga duplicada la respuesta.

c) Cómo escribirías la consulta para saber el sexo de si Fernando es hermano de Carmen o es hermana.

Usaremos la función `hermano` que comprueba si el primer pariente es hermano del segundo, si ese pariente es hombre imprimirá “Es hermano” mientras que si es mujer imprimirá “Es hermana”.

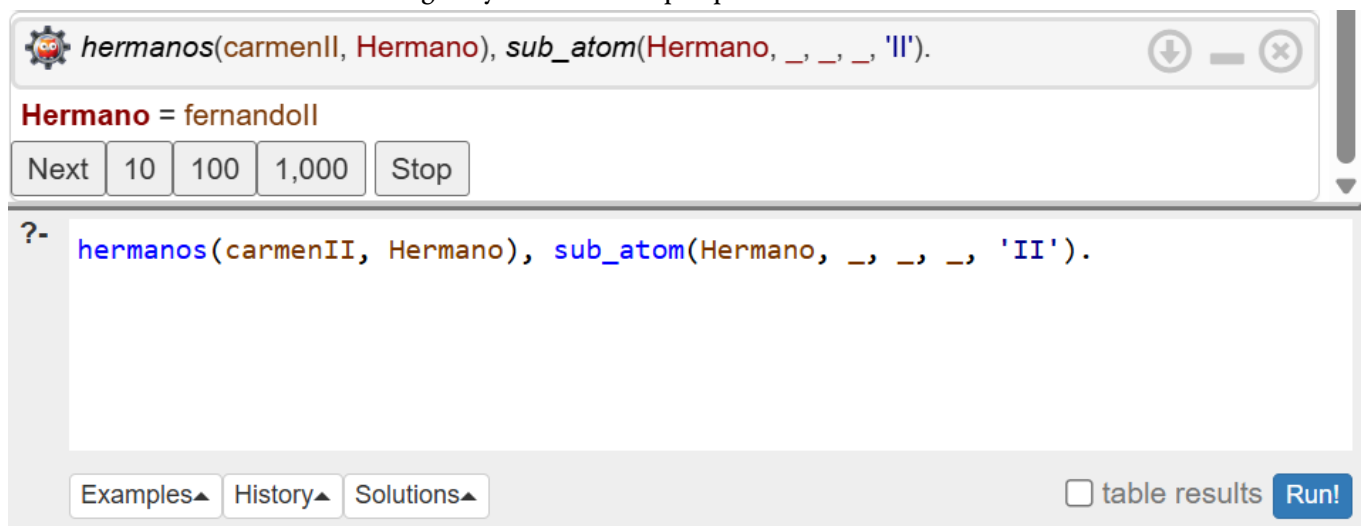


The screenshot shows a Prolog IDE interface. At the top, a query is entered: `(hermano(fernandoll, carmenll) -> writeln('Es hermano'); writeln('Es hermana')).` Below the query, the result is displayed: `Es hermano` followed by `true` on the next line. At the bottom, there are buttons for 'Examples', 'History', 'Solutions', a checkbox for 'table results', and a 'Run!' button.

Vemos que aparte de decir que son hermanos, se indica que Fernando es **hermano** de Carmen. Si Carmen y Fernando se invierten en la función saldría que Carmén es **hermana** de Fernando.

d) Listar todos los hermanos de Carmen de la segunda generación de la familia.

Como Carmen pertenece a la segunda generación, usaremos aquellos parientes que sean hermanos y que además tengan en su nombre indicado que son de dicha generación. Podemos usar `sub_atom` para partir los nombres, solamente indicaremos la cadena original y la subcadena que queremos buscar.



The screenshot shows a Prolog IDE interface. At the top, a query is entered: `hermanos(carmenll, Hermano), sub_atom(Hermano, _, _, _, 'II').` Below the query, the result is displayed: `Hermano = fernandoll`. Below the result, there are buttons for 'Next', '10', '100', '1,000', and 'Stop'. At the bottom, there are buttons for 'Examples', 'History', 'Solutions', a checkbox for 'table results', and a 'Run!' button.

Vemos que el resultado devuelve que Fernando II es hermano de Carmen.