

Lógica Computacional

Actividad 2: Lógica de predicados

Jordi Rafael Lazo Florensa
Grado en Ingeniería Informática
Universidad de Lleida
2018/2019

1. Modelización

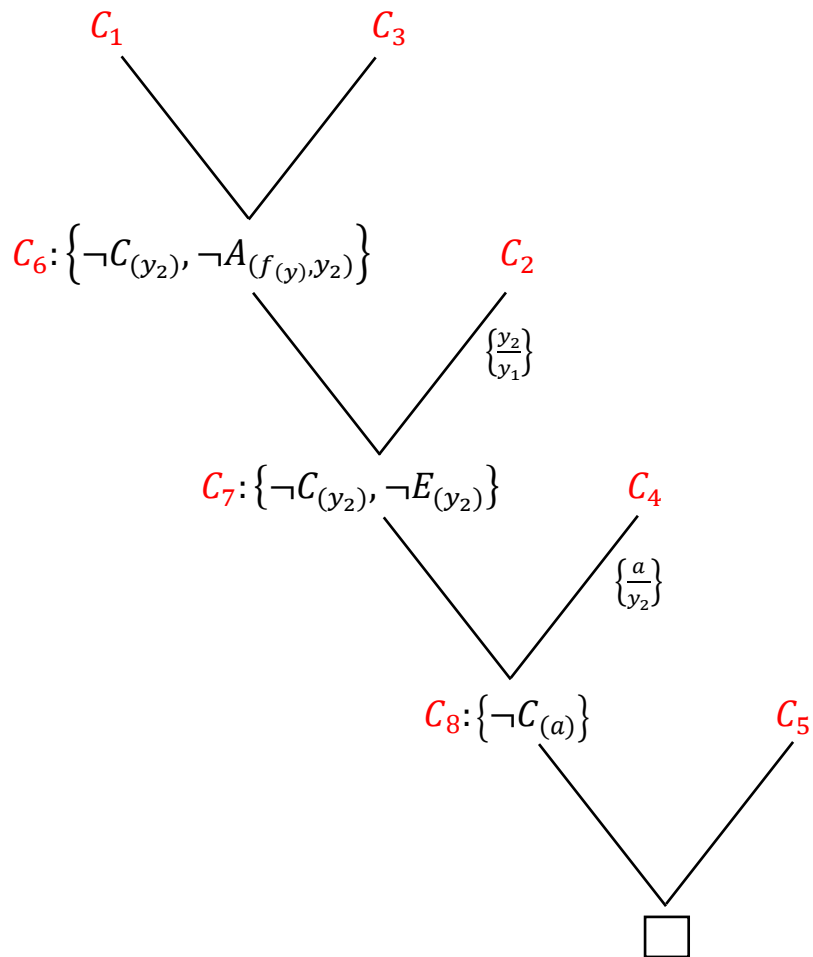
• Algunos franceses son amigos de todos los españoles. Ningún francés es amigo de los aficionados al cricket. Por lo tanto, ningún español es aficionado al cricket.

– Predicados: $E(x)$: x es español; $F(x)$: x es francés; $C(x)$: x es aficionado al cricket; $A(x, y)$: x es amigo de y .

$$\begin{aligned} 1: \forall_y (\exists_x F(x) \wedge (E(y) \rightarrow A(x, y))) \\ \equiv \\ C_1: \{F(f(y))\} \quad C_2: \{\neg E(y_1), A(f(y), y_1)\} \end{aligned}$$

$$\begin{aligned} 2: \forall_y (C(y) \rightarrow \neg \exists_x (F(x) \wedge A(x, y))) \\ \equiv \\ C_3: \{\neg C(y_2), \neg F(f(y)), \neg A(f(y), y_2)\} \end{aligned}$$

$$\begin{aligned} C: \models \neg \exists_x (E(x) \wedge C(x)) \\ \equiv \\ C_4: \{E(a)\} \quad C_5: \{C(a)\} \end{aligned}$$



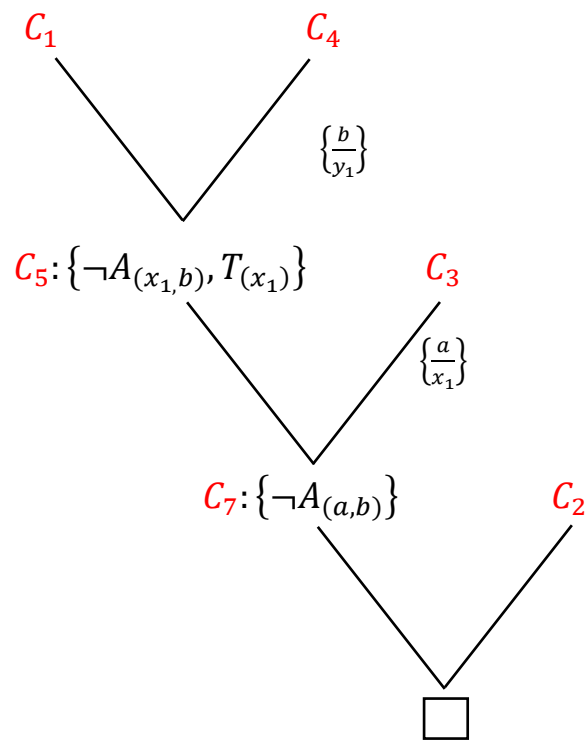
- Sólo los tontos alimentan a los osos salvajes. Cristina alimenta a Nicolás. Cristina no es tonta. Por lo tanto, Nicolás no es un oso salvaje.
- Predicados: $T(x)$: x es tonto; $O(x)$: x es un oso salvaje; $A(x, y)$: x alimenta a y .
- Constantes: a Cristina; b : Nicolás.

$$\begin{aligned}
 1: & \forall_x \forall_y (O_{(y)} \wedge A_{(x,y)} \rightarrow T_{(x)}) \\
 & \equiv \\
 C_1: & \{ \neg O_{(y_1)}, \neg A_{(x_1,y_1)}, T_{(x_1)} \}
 \end{aligned}$$

$$\begin{aligned}
 2: & (A_{(a,b)}) \\
 & \equiv \\
 C_2: & \{ A_{(a,b)} \}
 \end{aligned}$$

$$\begin{aligned}
 3: & (\neg T_{(a)}) \\
 & \equiv \\
 C_3: & \{ \neg T_{(a)} \}
 \end{aligned}$$

$$\begin{aligned}
 C: & \models (\neg O_{(b)}) \\
 & \equiv \\
 C_4: & \{ O_{(b)} \}
 \end{aligned}$$



• Todos los que ayudan a Juan viven en casa de Manolo. Antonio ayuda a todos los que trabajan con él. Juan trabaja con todos los amigos de Carlos. Antonio es amigo de Carlos. Por lo tanto, Antonio vive en casa de Manolo.

– Predicados: $AY(x, y)$: x ayuda a y ; $V(x, y)$: x vive en casa de y ; $T(x, y)$: x trabaja con y ; $A(x, y)$: x es amigo de y .

– Constantes: j : Juan; m : Manolo; a : Antonio; c : Carlos.

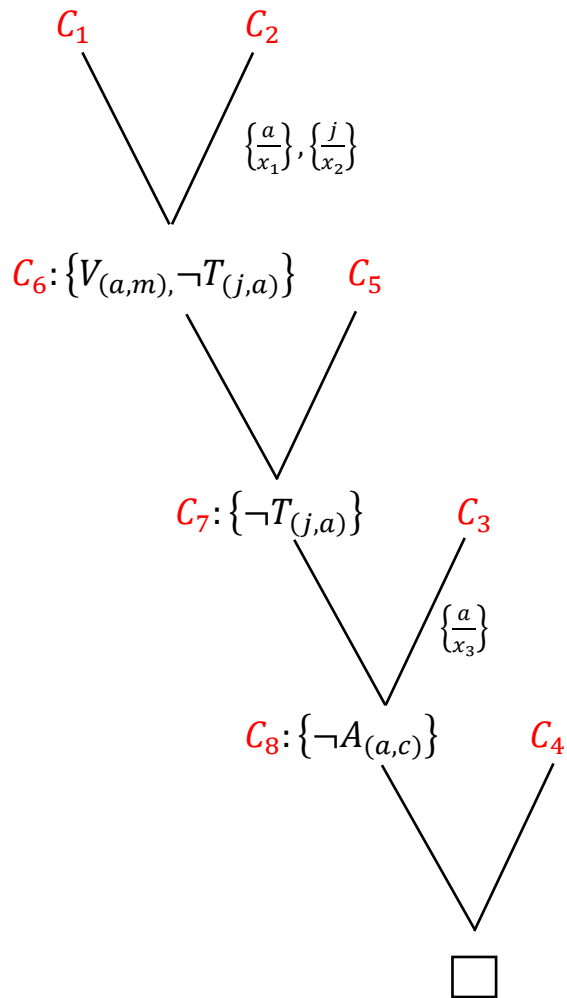
$$\begin{aligned} 1: & \forall_x (AY_{(x,j)} \rightarrow V_{(x,m)}) \\ & \equiv \\ C_1: & \{ \neg AY_{(x_1,j)}, V_{(x_1,m)} \} \end{aligned}$$

$$\begin{aligned} 2: & \forall_x (T_{(x,a)} \rightarrow AY_{(a,x)}) \\ & \equiv \\ C_2: & \{ \neg T_{(x_2,a)}, AY_{(a,x_2)} \} \end{aligned}$$

$$\begin{aligned} 3: & \forall_x (A_{(x,c)} \rightarrow T_{(j,x)}) \\ & \equiv \\ C_3: & \{ \neg A_{(x_3,c)}, T_{(j,x_3)} \} \end{aligned}$$

$$\begin{aligned} 4: & (A_{(a,c)}) \\ & \equiv \\ C_4: & \{ A_{(a,c)} \} \end{aligned}$$

$$\begin{aligned} C: & \models (V_{(a,m)}) \\ & \equiv \\ C_5: & \{ \neg V_{(a,m)} \} \end{aligned}$$

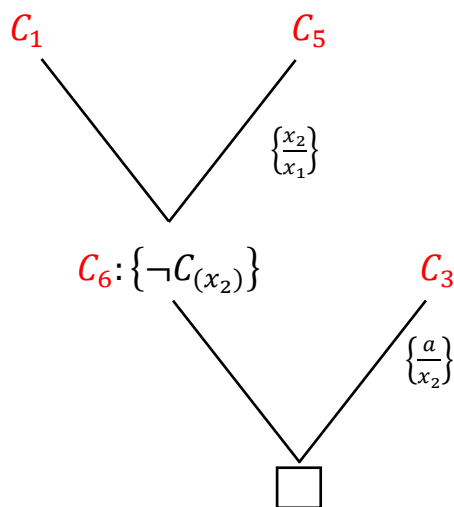


- Ningún aristócrata debe ser condenado a galeras a menos que sus crímenes sean vergonzosos y lleve una vida silenciosa. Hay aristócratas que han cometido crímenes vergonzosos, aunque su forma de vida no es silenciosa. Por lo tanto, hay algún aristócrata que no debe ser condenado a galeras.
- Predicados: $G(x)$: x debe ser condenado a galeras; $C(x)$: x ha cometido crímenes vergonzosos; $V(x)$: x lleva una vida silenciosa

$$\begin{aligned}
 1: & \forall x (G(x) \rightarrow C(x) \wedge V(x)) \\
 & \equiv \\
 C_1: & \{\neg G(x_1), \neg C(x_1)\} \quad C_2: \{\neg G(x_1), \neg V(x_1)\}
 \end{aligned}$$

$$\begin{aligned}
 2: & \exists x (C(x) \wedge \neg V(x)) \\
 & \equiv \\
 C_3: & \{C(a)\} \quad C_4: \{\neg V(a)\}
 \end{aligned}$$

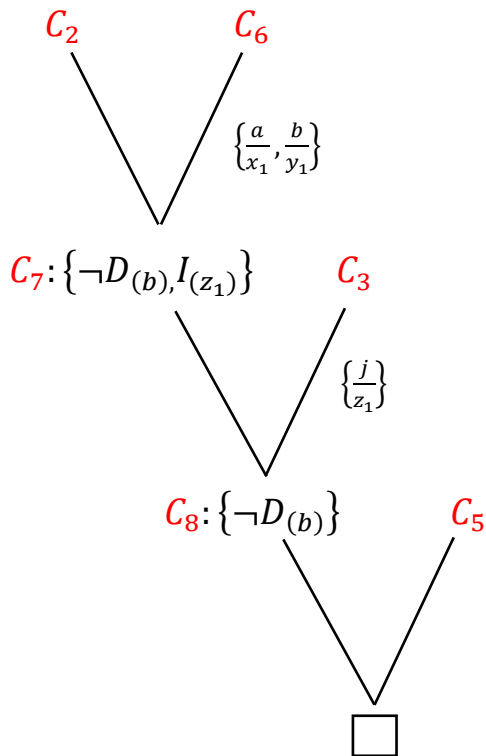
$$\begin{aligned}
 C: & \models \exists x (\neg G(x)) \\
 & \equiv \\
 C_5: & \{G(x_2)\}
 \end{aligned}$$



- Cualquiera que ahorra dinero gana interés. Por lo tanto, si no hay interés, entonces nadie ahorra dinero.
- Predicados: $A(x, y)$: x ahorra y ; $D(x)$: x es dinero; $I(x)$: x es interés; $G(x, y)$: x gana y .

$$\begin{aligned}
 1: & \forall_x \forall_y \forall_z ((A_{(x,y)} \wedge D_{(y)}) \rightarrow (G_{(x,z)} \wedge I_{(z)})) \\
 & \equiv \\
 C_1: & \{\neg A_{(x_1,y_1)}, \neg D_{(y_1)}, G_{(x_1,z_1)}\} \quad C_2: \{\neg A_{(x_1,y_1)}, \neg D_{(y_1)}, I_{(z_1)}\}
 \end{aligned}$$

$$\begin{aligned}
 C: & \models \forall_x \forall_y \forall_z ((\neg I_{(y)} \wedge \neg G_{(x,z)} \wedge \neg D_{(z)}) \rightarrow \neg A_{(x,z)}) \\
 & \equiv \\
 C_3: & \{\neg I_{(j)}\} \quad C_4: \{\neg G_{(a,b)}\} \quad C_5: \{D_{(b)}\} \quad C_6: \{A_{(a,b)}\}
 \end{aligned}$$

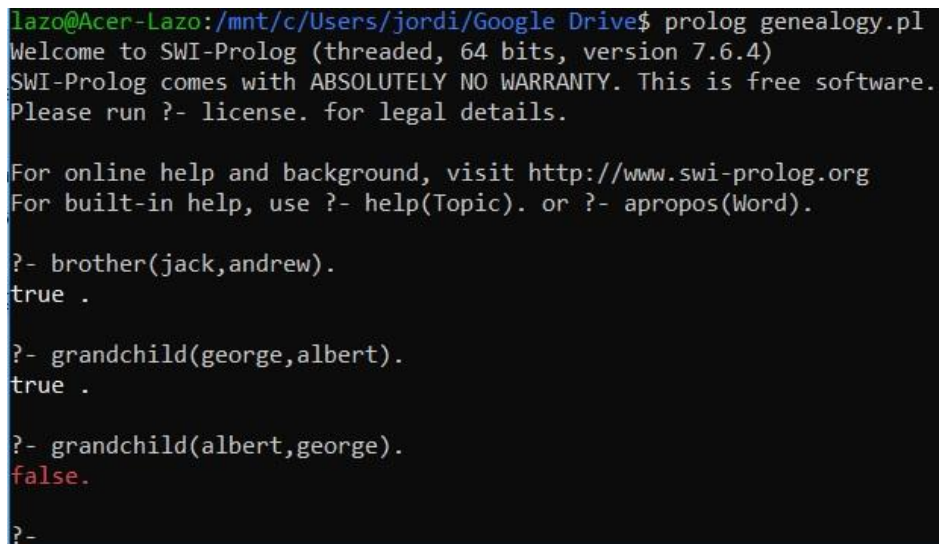


2. Programación Lógica

Para desarrollar esta práctica he nombrado los 11 predicados que aparecen en las instrucciones de la práctica. Además, he añadido 1 variable extra que es *siblings* para que no haya confusiones entre *brother* y *sister*. No obstante, he decidido no usarla para crear los otros predicados para evitar posibles errores. He intentado usar la menor cantidad de predicados programados por mí para programar los predicados de esta práctica.

Tal y como se puede observar en la siguiente foto ejecutando el *prolog* se puede observar, con ejemplos, como devuelve el resultado. Si es *true* significa que la relación familiar es cierta y si devuelve *false* es que no existe tal relación.

Además, he realizado el árbol genealógico para comprobar que relaciones son verdaderas y cuales otras son falsas.

A screenshot of a terminal window with a black background and white text. The prompt is 'lazo@Acer-Lazo:/mnt/c/Users/jordi/Google Drive\$'. The user has entered 'prolog genealogy.pl'. The output shows the Prolog version (7.6.4) and a license notice. The user then enters three queries: 'brother(jack, andrew).', 'grandchild(george, albert).', and 'grandchild(albert, george).'. The first two return 'true .' and the third returns 'false.'.

```
lazo@Acer-Lazo:/mnt/c/Users/jordi/Google Drive$ prolog genealogy.pl
Welcome to SWI-Prolog (threaded, 64 bits, version 7.6.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit http://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- brother(jack, andrew).
true .

?- grandchild(george, albert).
true .

?- grandchild(albert, george).
false.

?-
```

Ejemplo: `grandchild(george, albert).` true. (Y así con todas los predicados escritos).

```
1- father(X,Y) :- male(X), parent(X, Y).
   father(X,Y) :- male(X), married(X, Z), parent(Z, Y).
2- mother(X,Y) :- female(X), parent(X, Y).
   mother(X,Y) :- female(X), married(X, Z), parent(Z, Y).
3- son(X,Y) :- male(X), parent(Y, X).
4- daughter(X,Y) :- female(X), parent(Y, X).
5- brother(X,Y) :- male(X), parent(Z, X), parent(Z, Y).
6- sister(X,Y) :- female(X), parent(Z, X), parent(Z, Y).
7- uncle(X,Y) :- brother(X, Z), parent(Z, Y).
   uncle(X,Y) :- married(X, Z), brother(Z, A), parent(A, Y).
8- aunt(X,Y) :- sister(X, Z), parent(Z, Y).
   aunt(X,Y) :- married(X, Z), sister(Z, A), parent(A, Y).
9- grandparent(X,Y) :- parent(X, Z), parent(Z, Y).
10- grandchild(X,Y) :- parent(Y, Z), parent(Z, X).
11- cousin(X,Y) :- son(X, Z), uncle(Z, Y).
   cousin(X,Y) :- daughter(X, Z), uncle(Z, Y).
   cousin(X,Y) :- son(X, Z), aunt(Z, Y).
   cousin(X,Y) :- daughter(X, Z), aunt(Z, Y).
12- siblings(X,Y) :- parent(Z, X), parent(Z, Y), X \= Y.
```

Árbol genealógico

