```
-----------------------------------------------
Teoría de Modelos. Clase del 22 de abril de 2020
-----------------------------------------------
-----------------------------------------------
Modelización de Bases de Conocimiento en ASP.
Jerarquías y Herencia.
-----------------------------------------------
([GK], páginas 91-97)
-----------------------------------------------

% Consideramos el siguiente ejemplo:

%  Los vehículos se clasifican en aéreos o de no_vuelo.
%  Los submarinos, los coches y las bicis son vehículos de no_vuelo.
%  Los aviones y los drones son vehículos aéreos.
%  Los submarinos son negros.
%  Los vehículos aéreos son blancos.
%  Las bicis son rojas.
%  Narval es un submarino.
%  Misterio1 es un vehículo de no_vuelo blanco.
%  Misterio2 es un vehículo rojo.

%  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Usaremos una representación que explota el carácter jerárquico
% de la información que queremos modelizar. Para ello:

% -) Organizaremos la información en una estructura de árbol con las
%    distintas clases que aparecen en el ejemplo (Jerarquía).
% -) Entenderemos que si una clase C1 es subclase de una clase C2
%    en la jerarquía, entonces los objetos de la clase C1 heredan
%    las propiedades de la clase C2 (Herencia).

% Veámoslo en nuestro ejemplo.

% Usamos el tipo clase/1 para enumerar las clases que conforman la
jerarquía
% y la relación es_subclase/2 para describir la estructura jerárquica
% (contenciones *inmediatas* entre clases).

clase(vehiculo).
clase(aire).
clase(no_vuelo).
clase(submarino).
clase(coche).
clase(bici).
clase(avion).
clase(dron).
```

```
es_subclase(aire,vehiculo).
es_subclase(no_vuelo,vehiculo).
es_subclase(submarino,no_vuelo).
es_subclase(coche,no_vuelo).
es_subclase(bici,no_vuelo).
es_subclase(avion,aire).
es_subclase(dron,aire).

% Definimos la relación subclase/2 como la *clausura transitiva* de
es_subclase/2.
% (Observa que la definición de subclase/2 es independiente de la
jerarquía particular
%  que queramos representar)

subclase(C1,C2) :- es_subclase(C1,C2).
subclase(C1,C2) :- es_subclase(C1,C3),
                   subclase(C3,C2).

% Añadimos CWA(=hipótesis de mundo cerrado) para subclase/2.

-subclase(C1,C2) :- clase(C1),clase(C2),
                    not subclase(C1,C2).

% Usamos el tipo objeto/1 para enumerar los objetos que aparecen en la
jerarquía y
% la relación es_un/2 para indicar que un objeto X pertenece a una clase
C.

objeto(narval).
objeto(misterio1).
objeto(misterio2).

es_un(narval,submarino).
es_un(misterio1,no_vuelo).
es_un(misterio2,vehiculo).

% Podemos ahora definir la relación principal entre objetos y clases de
la jerarquía,
% miembro/2, que extiende la relación es_un/2 a las superclases de una
clase dada.
% (Observa de nuevo que la definición de miembro/2 es independiente de la
jerarquía
% particular que queramos representar)

miembro(X,C) :- es_un(X,C).
miembro(X,C) :- es_un(X,C1),
                subclase(C1,C).

% Veamos en nuestro ejemplo qué hechos *miembro(X,C)* podemos deducir.
```

```
% Puesto que por un momento centramos nuestra atención en el predicado
% miembro/2, escribimos en el fichero la directiva

%#show miembro/2.

% que le indica a CLINGO que en el conjunto de respuestas solo muestre
los
% hechos de la forma  miembro(_,_).

%  clingo version 5.4.0
%  Reading from ejemplo1.lp
%  Solving...
%  Answer: 1
%  miembro(narval,no_vuelo) miembro(narval,vehiculo)
miembro(misterio1,vehiculo) miembro(narval,submarino)
%  miembro(misterio1,no_vuelo) miembro(misterio2,vehiculo)
%  SATISFIABLE

% Observad que la relación de pertenencia se ha extendido según la
jerarquía descrita. Por ejemplo,
% solo hemos añadido el hecho es_un(narval,submarino) pero en la base de
conocimiento aparecen
% miembro(narval,submarino)  miembro(narval,no_vuelo)
miembro(narval,vehiculo).

% Ahora bien, veamos qué ocurre si preguntamos al programa
%   ¿Es Narval un coche?
%   ¿Es Misterio1 un vehículo aéreo?

% En ambos casos, la respuesta sería *Desconocido* porque ni
miembro(narval,coche)
% ni -miembro(narval,coche) aparecen en el conjunto de respuestas y lo
mismo sucede
% para miembro(misterio1,aire) y -miembro(misterio1,aire).

% Ahora bien, aplicando el sentido común, la respuesta natural sería
*NO*:
% puesto que sabemos que Narval es un submarino y *los submarinos no son
coches*,
% podemos deducir que Narval no es un coche.

% Para incorporar este conocimiento implícito en nuestra jerarquía,
añadimos la
% siguiente regla:
%     "Clases hermanas de la jerarquía son disjuntas dos a dos"

hermanas(C1,C2) :- es_subclase(C1,C),
                   es_subclase(C2,C),
                   C1!=C2.
```

```
-miembro(X,C2) :- miembro(X,C1),
                  hermanas(C1,C2).

% Con estas nuevas reglas , ahora tenemos:

#show miembro/2.
#show -miembro/2.
#show hermanas/2.

%  clingo version 5.4.0
%  Reading from ejemplo1.lp
%  Solving...
%  Answer: 1
%  hermanas(no_vuelo,aire) hermanas(aire,no_vuelo)
hermanas(coche,submarino) hermanas(bici,submarino)
%  hermanas(submarino,coche) hermanas(bici,coche)
hermanas(submarino,bici) hermanas(coche,bici)
%  hermanas(dron,avion) hermanas(avion,dron) miembro(narval,submarino)
miembro(misterio1,no_vuelo)
%  miembro(misterio2,vehiculo) miembro(narval,no_vuelo)
miembro(narval,vehiculo) miembro(misterio1,vehiculo)
% -miembro(narval,coche) -miembro(narval,bici) -miembro(misterio1,aire)
-miembro(narval,aire)
%  SATISFIABLE

%
------------------------------------------------------------------------
------------


% Con ello, terminamos la descripción de la estructura de la jerarquía.
% Pasamos a describir distintas propiedades de los objetos de la
jerarquía.

% Para hablar del color de los objetos, introducimos un nuevo tipo
color/1
% y la relación color/2 que indica que el objeto X tiene el color C.

% *Reificación*: los colores no son meras propiedades de los objetos,
sino que
% se implementan como "ciudadanos de primera clase" de la jerarquía  y
después
% se relacionan con otros objetos mediante color/2.

color(negro).
color(blanco).
color(rojo).
```

```
% "Los submarinos son negros"
color(X,negro) :- miembro(X,submarino).

% "Los vehículos aéreos son blancos"
color(X,blanco) :- miembro(X,aire).

%  Las bicis son rojas.
color(X,rojo) :- miembro(X,bici).

% Misterio 1 es blanco y Misterio 2 es rojo.

color(misterio1,blanco).
color(misterio2,rojo).

% Conocimiento implícito: "Los objetos solo pueden ser de un color".

-color(X,C2) :- objeto(X),color(C2),
                color(X,C1),
                C1!=C2.

% Veámoslo en nuestro ejemplo:

#show color/2.
#show -color/2.
%  clingo version 5.4.0
%  Reading from ejemplo1.lp
%  Solving...
%  Answer: 1
%  color(misterio1,blanco) color(misterio2,rojo) color(narval,negro)
-color(misterio1,negro)
%  -color(misterio1,rojo) -color(misterio2,negro)
-color(misterio2,blanco) -color(narval,blanco)
%  -color(narval,rojo)
%  SATISFIABLE


% Por último, observa que a la pregunta
%   ¿Es Misterio 2 un submarino?
% nuestro programa respondería *Desconocido*, mientras que la
% respuesta esperada sería NO (los submarinos son negros y
% Misterio 2 no es negro por ser rojo). Esto es así porque solo
% henos añadido la regla "Los submarinos son negros" y no su
contrapositivo
% "Si un objeto no es negro, entonces no es un submarino".

% Ahora bien, si nuestra pregunta es en cambio:
%   ¿Es consistente con nuestra base de conocimiento que
%    Misterio 2 sea un submarino?
```

```
% entonces, la respuesta es "No es consistente" sin tener que añadir nada
más.

% miembro(misterio2,submarino).

%  clingo version 5.4.0
%  Reading from ejemplo1.lp
%  Solving...
%  UNSATISFIABLE


% Veamos ahora qué ocurre para la pregunta
%   ¿Es consistente con nuestra base de conocimiento que
%    Misterio 1 no sea ni un submarino, ni una bici, ni un coche?


% #show -miembro/2.
% #show miembro/2.

% -miembro(misterio1,coche).
% -miembro(misterio1,submarino).
% -miembro(misterio1,bici).

%  clingo version 5.4.0
%  Reading from ejemplo1.lp
%  Solving...
%  Answer: 1
%  miembro(narval,submarino) miembro(misterio1,no_vuelo)
miembro(misterio2,vehiculo)
%  miembro(narval,no_vuelo) miembro(narval,vehiculo)
miembro(misterio1,vehiculo)
% -miembro(misterio1,coche) -miembro(misterio1,submarino)
-miembro(misterio1,bici)
% -miembro(narval,coche) -miembro(narval,bici) -miembro(misterio1,aire)
-miembro(narval,aire)
% SATISFIABLE

% Esto es así porque en ningún momento hemos añadido la información de
que un vehículo de
% no vuelo de nuestra jerarquía es necesariamente un submarino, un coche
o una bici.
% Si nos interesara añadir esta información implícita, podríamos hacerlo
como sigue:


miembro(X,submarino) ; miembro(X,coche) ; miembro(X,bici) :-
miembro(X,no_vuelo).

% Observad que si añadimos esta regla, nuestro programa ya es capaz de
```

```
deducir que
% Misterio 1 es necesariamente un coche.

% #show miembro/2.

%  clingo version 5.4.0
%  miembro(narval,submarino) miembro(misterio1,no_vuelo)
miembro(misterio2,vehiculo) miembro(narval,no_vuelo)
%  miembro(narval,vehiculo)  miembro(misterio1,vehiculo)
miembro(misterio1,coche)
%  SATISFIABLE

%
------------------------------------------------------------------------
------------------------
% FIN DE LA CLASE
%
------------------------------------------------------------------------
----------------------
```