Tema 6: Resolución relacional

José A. Alonso Jiménez
Miguel A. Gutiérrez Naranjo

Dpto. de Ciencias de la Computación e Inteligencia Artificial

Universidad de Sevilla

Resolución binaria

- **Problema 1: Demostrar que el conjunto**
$$\{\neg P(x) \vee Q(x), P(a), \neg Q(z)\}$$
es inconsistente.

- **Entrada** `ej-res-bin.in`

```
list(sos).
-P(x) | Q(x).
P(a).
-Q(z).
end_of_list.

set(binary_res).
```

- **Salida:**

```
========== start of search ==========

given clause #1: (wt=2) 2 [] P(a).

given clause #2: (wt=2) 3 [] -Q(z).

given clause #3: (wt=4) 1 [] -P(x)|Q(x).
** KEPT (pick-wt=2): 4 [binary,1.1,2.1] Q(a).

----> UNIT CONFLICT: 5 [binary,4.1,3.1] $F.

---------------- PROOF ----------------
1 [] -P(x)|Q(x).
2 [] P(a).
3 [] -Q(z).
4 [binary,1.1,2.1] Q(a).
5 [binary,4.1,3.1] $F.
------------ end of proof -------------
```

Resolución binaria

- **Sintaxis**
  - **Variables: Cadenas alfanuméricas que empiezan por** `u,v,w,x,y,z`
  - **Constantes, relaciones: Cadenas alfanuméricas que no empiezan por** `u,v,w,x,y,z`
  - **Cláusula**
  - **Cuantificación universal implícita**
  - **Entorno** `list`
- **Inferencia**
  - **Regla de resolución binaria**

```
L1 |...| Li |   A | L{i+1} |...| Ln.
M1 |...| Mj | - B | M{j+1} |...| Mk.
-----------------------------------
(L1 |...| Li | L{i+1} |...| Ln |
 M1 |...| Mj | M{j+1} |...| Mk)s.

s = u.m.g.(A,B)
```

  - **Unificación**
  - **Separación de variables**

Skolemización

- **Problema 2: Demostrar que el conjunto de fórmulas**
$$\{(\forall x)[P(x) \rightarrow Q(x)], P(a), \neg(\exists z)Q(z)\}$$
es inconsistente.

- **Entrada**

```
formula_list(sos).
all x (P(x) -> Q(x)).
P(a).
-(exists z Q(z)).
end_of_list.

set(binary_res).
```

- **Salida**

```
list(sos).
1 [] -P(x)|Q(x).
2 [] P(a).
3 [] -Q(z).
end_of_list.

---------------- PROOF ----------------
1 [] -P(x)|Q(x).
2 [] P(a).
3 [] -Q(z).
4 [binary,1.1,2.1] Q(a).
5 [binary,4.1,3.1] $F.
------------ end of proof -------------
```

- **Comentarios:**
  * **Cuantificadores** `all`, `exists`
  * **Skolemización**

Consecuencia lógica

- **El problema de la inconsistencia:**
  Dado un conjunto de fórmulas S
  determinar si es inconsistente
- **El problema de consecuencia lógica:**
  Dado un conjunto de fórmulas S
  y una fórmula F
  determinar si F es consecuencia lógica de S
- **Reducción de problemas: Son equivalentes**
  1. **F es consecuencia lógica de S**
  2. **$S \cup \{\neg F\}$ es inconsistente**

Argumentaciones

- **Problema 3: Demostrar la validez del siguiente argumento:**
  Los caballos son más rápidos que los perros. Algunos galgos son más rápidos que los conejos. Lucero es un caballo y Orejón es un conejo. Por tanto, Lucero es más rápido que Orejón.
- **Nuevos problemas en la decisión de la validez de una argumentación:**
  - **Representación del conocimiento**
  - **Explicitación del conocimiento implícito**
- **Lenguaje del problema:**

Símbolos:	Significado:
Lucero	Lucero
Orejon	Orejón
CABALLO(x)	x es un caballo
CONEJO(x)	x es un conejo
GALGO(x)	x es un galgo
PERRO(x)	x es un perro
MAS_RAPIDO(x,y)	x es más rápido que y

Argumentaciones

- **Entrada** `ej-3a1.in`

```
formula_list(sos).
% Los caballos son más rápidos que los perros.
all x y (CABALLO(x) & PERRO(y) -> MAS_RAPIDO(x,y)).

% Algunos galgos son más rápidos que los conejos
exists x (GALGO(x) &
          (all y (CONEJO(y) -> MAS_RAPIDO(x,y)))).

% Lucero es un caballo
CABALLO(Lucero).

% Orejón es un conejo.
CONEJO(Orejon).

% Lucero no es más rápido que Orejón
-MAS_RAPIDO(Lucero,Orejon).
end_of_list.

set(binary_res).
```

- **Salida**

```
list(sos).
1 [] -CABALLO(x)| -PERRO(y)|MAS_RAPIDO(x,y).
2 [] GALGO($c1).
3 [] -CONEJO(y)|MAS_RAPIDO($c1,y).
4 [] CABALLO(Lucero).
5 [] CONEJO(Orejon).
6 [] -MAS_RAPIDO(Lucero,Orejon).
end_of_list.
```

Argumentaciones

```
given clause #1: (wt=2) 2 [] GALGO($c1).
given clause #2: (wt=2) 4 [] CABALLO(Lucero).
given clause #3: (wt=2) 5 [] CONEJO(Orejon).
given clause #4: (wt=3) 6 [] -MAS_RAPIDO(Lucero,Orejon).
given clause #5: (wt=5) 3 [] -CONEJO(y)|MAS_RAPIDO($c1,y).
** KEPT (pick-wt=3): 7 [binary,3.1,5.1] MAS_RAPIDO($c1,Orejon).

given clause #6: (wt=3) 7 [binary,3.1,5.1] MAS_RAPIDO($c1,Orejon).

given clause #7: (wt=7) 1 [] -CABALLO(x)| -PERRO(y)|MAS_RAPIDO(x,y).
** KEPT (pick-wt=5): 8 [binary,1.1,4.1] -PERRO(x)|MAS_RAPIDO(Lucero,x).
** KEPT (pick-wt=2): 9 [binary,1.3,6.1,unit_del,4] -PERRO(Orejon).

given clause #8: (wt=2) 9 [binary,1.3,6.1,unit_del,4] -PERRO(Orejon).

given clause #9: (wt=5) 8 [binary,1.1,4.1] -PERRO(x)|MAS_RAPIDO(Lucero,x).

Search stopped because sos empty.
```

Argumentaciones

- **Búsqueda de modelos con MACE**

```
mace -n2 -p -m1 <ej-3a1.in
```

- **Modelo encontrado**

```
======================= Model #1 at 0.03 seconds:

 Lucero: 1     Orejon: 0    $c1: 0

 CABALLO :     PERRO :      GALGO :      CONEJO :
       0 1           0 1          0 1          0 1
    -------       -------      -------      -------
       F T           F F          T F          T F

 MAS_RAPIDO :
     | 0 1
    --+----
    0 | T F
    1 | F F
end_of_model
```

- **Entrada** `ej-3a2.in`

```
include('ej-03a1.in').

formula_list(sos).
% Los galgos son perros
all x (GALGO(x) -> PERRO(x)).
end_of_list.

set(binary_res).
```

- **Salida**

```
Search stopped because sos empty.
```

Argumentaciones

- **Búsqueda de modelos con MACE**

```
mace -n2 -p -m1 <ej-3a2.in
```

- **Modelo encontrado**

```
Lucero: 1    Orejon: 1    $c1: 0

CABALLO :    PERRO :      GALGO :      CONEJO :
       0 1          0 1          0 1          0 1
   -------      -------      -------      -------
       F T          T F          T F          F T

MAS_RAPIDO :
     | 0 1
   --+----
   0 | F T
   1 | T F
```

- **Entrada** `ej-3a3.in`

```
include('ej-03a2.in').

formula_list(sos).
% Si x es más rápido que y e y es más rápido que z,
% entonces x es más rápido que z.
all x y z (MAS_RAPIDO(x,y) & MAS_RAPIDO(y,z)
           -> MAS_RAPIDO(x,z)).
end_of_list.

set(binary_res).
```

Argumentaciones

- **Prueba**

```
 1 [] -CABALLO(x)| -PERRO(y)|MAS_RAPIDO(x,y).
 2 [] GALGO($c1).
 3 [] -CONEJO(y)
      | MAS_RAPIDO($c1,y).
 4 [] CABALLO(Lucero).
 5 [] CONEJO(Orejon).
 6 [] -MAS_RAPIDO(Lucero,Orejon).
 7 [] -GALGO(x)|PERRO(x).
 8 [] -MAS_RAPIDO(x,y)
       | -MAS_RAPIDO(y,z)
       | MAS_RAPIDO(x,z).
 9 [binary,7.1,2.1] PERRO($c1).
10 [binary,3.1,5.1] MAS_RAPIDO($c1,Orejon).
11 [binary,1.1,4.1] -PERRO(x)
                     | MAS_RAPIDO(Lucero,x).
16 [binary,11.1,9.1] MAS_RAPIDO(Lucero,$c1).
19 [binary,8.1,16.1] -MAS_RAPIDO($c1,x)
                      | MAS_RAPIDO(Lucero,x).
36 [binary,19.1,10.1] MAS_RAPIDO(Lucero,Orejon).
37 [binary,36.1,6.1] $F.
```

- **Estadísticas**

```
clauses given                 18
clauses generated             43
clauses kept                  28
clauses forward subsumed      12
clauses back subsumed          0
```

La estrategia del conjunto soporte

- **Entrada** `ej-3b.in`

```
formula_list(usable).
all x y (CABALLO(x) & PERRO(y) -> MAS_RAPIDO(x,y)).
exists x (GALGO(x) &
          (all y (CONEJO(y) -> MAS_RAPIDO(x,y)))).
CABALLO(Lucero).
CONEJO(Orejon).
all x (GALGO(x) -> PERRO(x)).
all x y z (MAS_RAPIDO(x,y) & MAS_RAPIDO(y,z)
           -> MAS_RAPIDO(x,z)).
end_of_list.

formula_list(sos).
-MAS_RAPIDO(Lucero,Orejon).
end_of_list.

set(binary_res).
```

La estrategia del conjunto soporte

- **Salida**

```
list(usable).
1 [] -CABALLO(x)| -PERRO(y)|MAS_RAPIDO(x,y).
2 [] GALGO($c1).
3 [] -CONEJO(y)|MAS_RAPIDO($c1,y).
4 [] CABALLO(Lucero).
5 [] CONEJO(Orejon).
6 [] -GALGO(x)|PERRO(x).
7 [] -MAS_RAPIDO(x,y)| -MAS_RAPIDO(y,z)|MAS_RAPIDO(x,z).
end_of_list.

list(sos).
8 [] -MAS_RAPIDO(Lucero,Orejon).
end_of_list.

========== start of search ==========

given clause #1: (wt=3) 8 [] -MAS_RAPIDO(Lucero,Orejon).
** KEPT: 9 [binary,8.1,7.3] -MAS_RAPIDO(Lucero,x)| -MAS_RAPIDO(x,Orejon).
** KEPT: 10 [binary,8.1,1.3,unit_del,4] -PERRO(Orejon).
```

La estrategia del conjunto soporte

```
given clause #2: (wt=2) 10 [binary,8.1,1.3,unit_del,4] -PERRO(Orejon).
** KEPT: 11 [binary,10.1,6.2] -GALGO(Orejon).

given clause #3: (wt=2) 11 [binary,10.1,6.2] -GALGO(Orejon).

given clause #4: (wt=6) 9 [binary,8.1,7.3] -MAS_RAPIDO(Lucero,x)| -MAS_RAPIDO(x,Orejon
** KEPT: 12 [binary,9.1,7.3] -MAS_RAPIDO(x,Orejon)| -MAS_RAPIDO(Lucero,y)| -MAS_RAPIDO
** KEPT: 13 [binary,9.1,1.3,unit_del,4] -MAS_RAPIDO(x,Orejon)| -PERRO(x).
** KEPT: 14 [binary,9.2,3.2,unit_del,5] -MAS_RAPIDO(Lucero,$c1).

given clause #5: (wt=3) 14 [binary,9.2,3.2,unit_del,5] -MAS_RAPIDO(Lucero,$c1).
** KEPT: 15 [binary,14.1,7.3] -MAS_RAPIDO(Lucero,x)| -MAS_RAPIDO(x,$c1).
** KEPT: 16 [binary,14.1,1.3,unit_del,4] -PERRO($c1).

given clause #6: (wt=2) 16 [binary,14.1,1.3,unit_del,4] -PERRO($c1).
** KEPT: 17 [binary,16.1,6.2] -GALGO($c1).

----> UNIT CONFLICT at   0.01 sec ----> 18 [binary,17.1,2.1] $F.
```

La estrategia del conjunto soporte

- **Prueba**

```
 1 [] -CABALLO(x)
      | -PERRO(y)
      | MAS_RAPIDO(x,y).
 2 [] GALGO($c1).
 3 [] -CONEJO(y)
      | MAS_RAPIDO($c1,y).
 4 [] CABALLO(Lucero).
 5 [] CONEJO(Orejon).
 6 [] -GALGO(x)
      | PERRO(x).
 7 [] -MAS_RAPIDO(x,y)
      | -MAS_RAPIDO(y,z)
      | MAS_RAPIDO(x,z).
 8 [] -MAS_RAPIDO(Lucero,Orejon).
 9 [binary,8.1,7.3] -MAS_RAPIDO(Lucero,x)
                    | -MAS_RAPIDO(x,Orejon).
14 [binary,9.2,3.2,unit_del,5] -MAS_RAPIDO(Lucero,$c1).
16 [binary,14.1,1.3,unit_del,4] -PERRO($c1).
17 [binary,16.1,6.2] -GALGO($c1).
18 [binary,17.1,2.1] $F.
```

- **Estadísticas**

```
clauses given                  6
clauses generated             14
clauses kept                   9
clauses forward subsumed       5
clauses back subsumed          0
```

Resolución UR

- **Resolución UR**

```
L_1 |...| L_n.
M_1.
...
M_{i-1}.
M_{i+1}.
...
M_n.
-------------
(L_i)s.

donde s es u.m.g de L_j y el
 complementario de M_j
 (j en {1,...,i-1,i+1,...n}})
```

- **Entrada** `ej-3c.in`

```
formula_list(sos).
all x y (CABALLO(x) & PERRO(y) -> MAS_RAPIDO(x,y)).
exists x (GALGO(x) &
          (all y (CONEJO(y) -> MAS_RAPIDO(x,y)))).
CABALLO(Lucero).
CONEJO(Orejon).
-MAS_RAPIDO(Lucero,Orejon).
all x (GALGO(x) -> PERRO(x)).
all x y z (MAS_RAPIDO(x,y) & MAS_RAPIDO(y,z)
           -> MAS_RAPIDO(x,z)).
end_of_list.

set(ur_res).
```

Resolución UR

• Prueba

```
 1 [] -CABALLO(x)
      | -PERRO(y)
      | MAS_RAPIDO(x,y).
 2 [] GALGO($c1).
 3 [] -CONEJO(y)
      | MAS_RAPIDO($c1,y).
 4 [] CABALLO(Lucero).
 5 [] CONEJO(Orejon).
 6 [] -MAS_RAPIDO(Lucero,Orejon).
 7 [] -GALGO(x)|PERRO(x).
 8 [] -MAS_RAPIDO(x,y)
      | -MAS_RAPIDO(y,z)
      | MAS_RAPIDO(x,z).
 9 [ur,7,2] PERRO($c1).
10 [ur,3,5] MAS_RAPIDO($c1,Orejon).
12 [ur,1,4,9] MAS_RAPIDO(Lucero,$c1).
14 [ur,8,10,6] -MAS_RAPIDO(Lucero,$c1).
15 [binary,14.1,12.1] $F.
```

• Estadísticas

```
clauses given                 13
clauses generated              6
clauses kept                   6
clauses forward subsumed       0
clauses back subsumed          0
```

Hiper–resolución

- **Regla de hiper–resolución**

```
-A_1 |...| -A_n | B_1 |...| B_m.
M_1.
...
M_n.
-------------
(B_1 |...| B_m)s.

donde A_1, ..., A_n, B_1, ..., B_m son átomos
y s es un u.m.g. de {A_1 = M_1,...,A_n = M_n}
```

- **Entrada** `ej-3d.in`

```
formula_list(sos).
all x y (CABALLO(x) & PERRO(y) -> MAS_RAPIDO(x,y)).
exists x (GALGO(x) &
          (all y (CONEJO(y) -> MAS_RAPIDO(x,y)))).
CABALLO(Lucero).
CONEJO(Orejon).
-MAS_RAPIDO(Lucero,Orejon).
all x (GALGO(x) -> PERRO(x)).
all x y z (MAS_RAPIDO(x,y) & MAS_RAPIDO(y,z)
           -> MAS_RAPIDO(x,z)).
end_of_list.

set(hyper_res).
```

Hiper–resolución

- **Prueba**

```
 1 [] -CABALLO(x)
      | -PERRO(y)
      | MAS_RAPIDO(x,y).
 2 [] GALGO($c1).
 3 [] -CONEJO(y)
      | MAS_RAPIDO($c1,y).
 4 [] CABALLO(Lucero).
 5 [] CONEJO(Orejon).
 6 [] -MAS_RAPIDO(Lucero,Orejon).
 7 [] -GALGO(x)|PERRO(x).
 8 [] -MAS_RAPIDO(x,y)
      | -MAS_RAPIDO(y,z)
      | MAS_RAPIDO(x,z).
 9 [hyper,7,2] PERRO($c1).
10 [hyper,3,5] MAS_RAPIDO($c1,Orejon).
11 [hyper,1,4,9] MAS_RAPIDO(Lucero,$c1).
12 [hyper,8,11,10] MAS_RAPIDO(Lucero,Orejon).
13 [binary,12.1,6.1] $F.
```

- **Estadística**

```
clauses given                11
clauses generated             4
clauses kept                  4
clauses forward subsumed      0
clauses back subsumed         0
```

Obtención de respuestas

- **Problema: Dado un conjunto de fórmulas S y una fórmula $F(x_1, \ldots, x_n)$, cuyas variables libres son $x_1, \ldots, x_n$, encontrar términos $t_1, \ldots, t_n$ tales que $F(t_1, \ldots, t_n)$ sea consecuencia de S**
- **Procedimiento de solución**
  - **Introducir un nuevo símbolo de predicados (`$ANS`)**
  - **Considerar el conjunto de las cláusulas correspondientes a las fórmulas de**
    $$S \cup \{(\forall x_1) \ldots (\forall x_n)[F(x_1, \ldots, x_n) \rightarrow \$ANS(x_1, \ldots, x_n)]\}$$
  - **Aplicar el procedimiento de resolución hasta encontrar una cláusula cuyo único literal contenga el predicado `$ANS`**
  - **Los términos que aparecen en dicho literal forman una respuesta a la cuestión planteada.**

Obtención de respuestas

- **Problema 4a: Dado $\{\forall(P(x) \rightarrow Q(x)), P(a)\}$ determinar un z tal que $Q(z)$ sea consecuencia del conjunto.**
- **Entrada**

```
formula_list(sos).
all x (P(x) -> Q(x)).
P(a).
all z (Q(z) -> $ANS(z)).
end_of_list.

set(binary_res).
```

- **Prueba**

```
1 [] -P(x)|Q(x).
2 [] P(a).
3 [] -Q(z)|$ANS(z).
4 [binary,1.1,2.1] Q(a).
5 [binary,4.1,3.1] $ANS(a).
```

Obtención de respuestas

- **Problema 4b: Dado $\{\forall(P(x) \rightarrow Q(x)), P(a) \wedge P(b)\}$ determinar un z tal que $Q(z)$ sea consecuencia del conjunto.**
- **Entrada** `ej-4b1.in`

```
formula_list(sos).
all x (P(x) -> Q(x)).
P(a) & P(b).
all z (Q(z) -> $ANS(z)).
end_of_list.

set(binary_res).
assign(max_proofs,2).
```

- **Pruebas**

```
---------------- PROOF ----------------
1 [] -P(x)|Q(x).
3 [] P(b).
4 [] -Q(z)|$ANS(z).
5 [binary,1.1,3.1] Q(b).
6 [binary,5.1,4.1] $ANS(b).
------------ end of proof -------------
---------------- PROOF ----------------
1 [] -P(x)|Q(x).
2 [] P(a).
4 [] -Q(z)|$ANS(z).
7 [binary,1.1,2.1] Q(a).
8 [binary,7.1,4.1] $ANS(a).
------------ end of proof -------------

Search stopped by max_proofs option.
```

Obtención de respuestas

- **Entrada** `ej-4b2.in`

```
formula_list(sos).
all x (P(x) -> Q(x)).
P(a) & P(b).
all z (Q(z) -> $ANS(z)).
end_of_list.

set(binary_res).
assign(max_proofs,-1).
```

- **Respuestas**

```
- PROOF --- 6 [binary,5.1,4.1] $ANS(b).
- PROOF --- 8 [binary,7.1,4.1] $ANS(a).
- PROOF --- 10 [binary,9.1,3.1] $ANS(b).
- PROOF --- 11 [binary,9.1,2.1] $ANS(a).
- PROOF --- 12 [binary,5.1,4.1] $ANS(b).
- PROOF --- 13 [binary,7.1,4.1] $ANS(a).
- PROOF --- 14 [binary,9.1,3.1] $ANS(b).
- PROOF --- 15 [binary,9.1,2.1] $ANS(a).
Search stopped because sos empty.
```

Obtención de respuestas

- **Entrada** `ej-4b3.in`

```
formula_list(sos).
all x (P(x) -> Q(x)).
P(a) & P(b).
end_of_list.

formula_list(passive).
all z (Q(z) -> $ANS(z)).
end_of_list.

set(binary_res).
assign(max_proofs,-1).
```

- **Pruebas**

```
---------------- PROOF ----------------
1 [] -P(x)|Q(x).
3 [] P(b).
4 [] -Q(z)|$ANS(z).
5 [binary,1.1,3.1] Q(b).
6 [binary,5.1,4.1] $ANS(b).
------------ end of proof -------------

---------------- PROOF ----------------
1 [] -P(x)|Q(x).
2 [] P(a).
4 [] -Q(z)|$ANS(z).
7 [binary,1.1,2.1] Q(a).
8 [binary,7.1,4.1] $ANS(a).
------------ end of proof -------------

Search stopped because sos empty.
```

Obtención de respuestas

- **Problema 5–1: De las siguientes personas Juan, Jorge, Víctor, María, Agata y Carla se sabe que María, Jorge y Victor son ricos y que María y Juan se aman, que Víctor ama a María y que Jorge y Víctor se aman. Admitiendo que dos personas pueden casarse si son de distintos sexo y se aman o una es rica y ama a la otra, determinar las parejas que pueden casarse.**

- **Entrada**

```
list(usable).
Hombre(Juan).
Hombre(Jorge).
Hombre(Victor).
Mujer(Maria).
Mujer(Agata).
Mujer(Carla).
Rico(Maria).
Rico(Jorge).
Rico(Victor).
Ama(Maria, Juan).
Ama(Juan, Maria).
Ama(Victor, Maria).
Ama(Jorge, Victor).
Ama(Victor, Jorge).
```

Obtención de respuestas

```
% Los hombres y las mujeres son de sexos distintos:
-Mujer(x) | -Hombre(y) | Distinto_sexo(x,y).
-Mujer(x) | -Hombre(y) | Distinto_sexo(y,x).

% Dos personas de distintos sexo pueden casarse si
% se aman o una es rica y ama a la otra:
-Distinto_sexo(x,y)
 | -Ama(x,y)
 | -Ama(y,x)
 | Pueden_casarse(x,y).
-Distinto_sexo(x,y)
 | -Ama(x,y)
 | -Rico(x)
 | Pueden_casarse(x,y).
end_of_list.

list(sos).
-Pueden_casarse(x,y) | $ans(x,y).
end_of_list.

set(ur_res).
assign(max_proofs, -1).
```

Obtención de respuestas

- **Respuesta 1**

```
---------------- PROOF ----------------
 3 [] Hombre(Victor).
 4 [] Mujer(Maria).
 9 [] Rico(Victor).
12 [] Ama(Victor,Maria).
16 [] -Mujer(x)| -Hombre(y)|Distinto_sexo(y,x).
18 [] -Distinto_sexo(x,y)| -Ama(x,y)| -Rico(x)
      |Pueden_casarse(x,y).
19 [] -Pueden_casarse(x,y)|$ans(x,y).
22 [ur,19,18,12,9] $ans(Victor,Maria)
                   | -Distinto_sexo(Victor,Maria).
27 [ur,22,16,3] $ans(Victor,Maria)| -Mujer(Maria).
28 [binary,27.1,4.1] $ans(Victor,Maria).
```

- **Respuesta 2**

```
---------------- PROOF ----------------
 1 [] Hombre(Juan).
 4 [] Mujer(Maria).
 7 [] Rico(Maria).
10 [] Ama(Maria,Juan).
15 [] -Mujer(x)| -Hombre(y)|Distinto_sexo(x,y).
18 [] -Distinto_sexo(x,y)| -Ama(x,y)| -Rico(x)
      |Pueden_casarse(x,y).
19 [] -Pueden_casarse(x,y)|$ans(x,y).
23 [ur,19,18,10,7] $ans(Maria,Juan)
                   | -Distinto_sexo(Maria,Juan).
31 [ur,23,15,4] $ans(Maria,Juan)| -Hombre(Juan).
32 [binary,31.1,1.1] $ans(Maria,Juan).
```

Obtención de respuestas

- **Problema 5–2 A partir de las siguientes sentencias:**
  *** Toda persona es hijo de su padre.**
  *** Los hijos de los hijos son nietos.**
  Obtener la respuesta a la siguiente pregunta:
  *** Si a es nieto de x, ¿quién es x?.**

- **Entrada**

```
formula_list(usable).
% Toda persona es hijo de su padre:
all x exists y Hijo(x,y).

% Los hijos de los hijos son nietos:
all x y z (Hijo(x,y) & Hijo(y,z) -> Nieto(x,z)).
end_of_list.

formula_list(sos).
% Si a es nieto de x, ¿quién es x?:
all x (Nieto(a,x) -> $Ans(x)).
end_of_list.

set(ur_res).
```

- **Respuesta**

```
---------------- PROOF ----------------
1 [] Hijo(x,$f1(x)).
2 [] -Hijo(x,y)| -Hijo(y,z)|Nieto(x,z).
3 [] -Nieto(a,x)|$Ans(x).
4 [ur,3,2,1] $Ans($f1(x))| -Hijo(a,x).
5 [binary,4.1,1.1] $Ans($f1($f1(a))).
------------ end of proof -------------
```

Obtención de respuestas

- **Problema 5–3: Si Luna es una persona y todas las personas están solteras o casadas, ¿cómo está Luna?**
- **Entrada**

```
formula_list(usable).
Persona(Luna).
all x (Persona(x) -> Estado(x,Soltero)
                   | Estado(x,Casado)).
end_of_list.

formula_list(sos).
all x (Estado(Luna, x) -> $ans(x)).
end_of_list.

set(ur_res).
```

- **Respuesta**

```
1 [] Persona(Luna).
2 [] -Persona(x)
     | Estado(x,Soltero)
     | Estado(x,Casado).
3 [] -Estado(Luna,x)
     | $ans(x).
4 [ur,3,2,3] $ans(Casado)
             | -Persona(Luna)
             | $ans(Soltero).
5 [binary,4.1,1.1] $ans(Casado)|$ans(Soltero).
```

Bibliografía

- Alonso, J.A.; Fernández, A. y Pérez, M.J. *Razonamiento automático* (en *Lógica formal (Orígenes, métodos y aplicaciones*, Ed. Kronos, 1995)
- Chang, C.L.; Lee, R.C.T. *Symbolic logic and mechanical theorem proving.* (Academic Press, 1973)
- Genesereth, M.R. *Computational Logic* (27 March 2000)
  - Cap. 9 "Relational resolution"
- Genesereth, M.R. y Nilsson, N.J. *Logical foundations of Artificial Intelligence* (Morgan Kaufmann, 1987)
  - Cap. 4: "Resolution"
  - Cap. 5: "Resolution strategies"
- Wos, L.; Overbeek, R.; Lusk, E. y Boyle, J. *Automated Reasoning: Introduction and Applications, (2nd ed.)* (McGraw–Hill, 1992)