Razonamiento automático con OTTER

José A. Alonso Jiménez
José L. Ruiz Reina

Dpto. de Ciencias de la Computación e Inteligencia Artificial
Universidad de Sevilla

Inconsistencia proposicional y resolución binaria

- **Problema 1: Determinar si el conjunto**
  $$\{P \vee Q, \neg P \vee Q, P \vee \neg Q, \neg P \vee \neg Q\}$$
  es inconsistente.

- **Resolución binaria proposicional**

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

Inconsistencia proposicional y resolución binaria

- **Entrada** `ej1.in`

```
list(sos).
P | Q.
-P | Q.
P | -Q.
-P | -Q.
end_of_list.

set(binary_res).
```

- **Demostración con OTTER**

```
otter <ej1.in >ej1.out
```

Inconsistencia proposicional y resolución binaria

- **Salida** `ej1.out`
  - **Procesamiento de la entrada**

```
----- Otter 3.0.5, Feb 1998 -----

list(sos).
1 [] P|Q.
2 [] -P|Q.
3 [] P| -Q.
4 [] -P| -Q.
end_of_list.
set(binary_res).
   dependent: set(factor).
   dependent: set(unit_deletion).

======= end of input processing =======
```

Inconsistencia proposicional y resolución binaria

- **Búsqueda de la prueba**

```
========== start of search ==========
given clause #1: (wt=2) 1 [] P|Q.

given clause #2: (wt=2) 2 [] -P|Q.
** KEPT (pick-wt=1): 5 [binary,2.1,1.1,factor_simp] Q.
5 back subsumes 2.
5 back subsumes 1.

given clause #3: (wt=1) 5 [binary,2.1,1.1,factor_simp] Q.

given clause #4: (wt=2) 3 [] P| -Q.
** KEPT (pick-wt=1): 6 [binary,3.2,5.1] P.
6 back subsumes 3.

given clause #5: (wt=1) 6 [binary,3.2,5.1] P.

given clause #6: (wt=2) 4 [] -P| -Q.
** KEPT (pick-wt=1): 7 [binary,4.1,6.1] -Q.

----> UNIT CONFLICT at   0.06 sec ----> 8 [binary,7.1,5.1] $F.
```

Inconsistencia proposicional y resolución binaria

- **Demostración**

```
Length of proof is 3.  Level of proof is 3.

---------------- PROOF ----------------

1 [] P|Q.
2 [] -P|Q.
3 [] P| -Q.
4 [] -P| -Q.
5 [binary,2.1,1.1,factor_simp] Q.
6 [binary,3.2,5.1] P.
7 [binary,4.1,6.1] -Q.
8 [binary,7.1,5.1] $F.

------------ end of proof -------------
```

Inconsistencia de cláusulas de primer orden y unificación

- **Problema 2: Demostrar que el conjunto**
  $$\{\neg P(x) \vee Q(x), P(a), \neg Q(z)\}$$
  es inconsistente.

- **Resolución binaria de primer orden**

```
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)
```

Inconsistencia de cláusulas de primer orden y unificación

- **Entrada** `ej2.in`

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

set(binary_res).
```

Inconsistencia de cláusulas de primer orden y unificación

- **Salida** `ej2.out`
  - **Prepocesamiento**

```
----- Otter 3.0.5, Feb 1998 -----

list(sos).
1 [] -P(x)|Q(x).
2 [] P(a).
3 [] -Q(z).
end_of_list.
set(binary_res).
   dependent: set(factor).
   dependent: set(unit_deletion).

======= end of input processing =======
```

Inconsistencia de cláusulas de primer orden y unificación

- **Búsqueda**

```
=========== 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 at   0.07 sec ----> 5 [binary,4.1,3.1] $F.
```

Inconsistencia de cláusulas de primer orden y unificación

- **Prueba**

```
Length of proof is 1.  Level of proof is 1.

---------------- 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 -------------
```

Inconsistencia de fórmulas de primer orden y skolemización

- **Problema 3: 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).
```

Inconsistencia de fórmulas de primer orden y skolemización

- **Salida**
  - **Prepocesamiento**

```
-------> sos clausifies to:

list(sos).
1 [] -P(x)|Q(x).
2 [] P(a).
3 [] -Q(z).
end_of_list.
set(binary_res).
   dependent: set(factor).
   dependent: set(unit_deletion).
```

Inconsistencia de fórmulas de primer orden y skolemización

- **Búsqueda**

```
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 at   0.07 sec ----> 5 [binary,4.1,3.1] $F.
```

- **Prueba**

```
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.
```

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 determinar si la fórmula 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 y representación del conocimiento

- Nuevos problemas en la decisión de la validez de una argumentación:
  - Problema de la representación del conocimiento
  - Problema de la explicitación del conocimiento implícito

- Problema 4: 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.

Argumentaciones y representación del conocimiento

- **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 y representación del conocimiento

- **Entrada** `ej-4a1.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).
```

Argumentaciones y representación del conocimiento

- **Salida**
  - **Preprocesamiento**

```
-------> sos clausifies to:

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.
set(binary_res).
   dependent: set(factor).
   dependent: set(unit_deletion).
```

Argumentaciones y representación del conocimiento

- **Búsqueda**

```
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 y representación del conocimiento

- **Búsqueda de modelos con MACE**

```
mace -n2 -p -m1 <ej-4a1.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
```

Argumentaciones y representación del conocimiento

- **Entrada** `ej-4a2.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).

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

set(binary_res).
```

- **Salida**

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

- **Búsqueda de modelos con MACE**

```
mace -n2 -p -m1 <ej-4a2.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
```

Argumentaciones y representación del conocimiento

- **Entrada** `ej-4a3.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)).

% 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 y representación del conocimiento

- **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.
```

La estrategia del conjunto soporte

- **Entrada** `ej-4b.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

- **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.
```

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 = M'_j: j en {1,...,i-1,i+1,...n}}
      M'_i es el complementario de M_i
```

Resolución UR

- **Entrada** `ej-4c.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.
```

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}
```

Hiper–resolución

- **Entrada** `ej-4d.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.
```

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**
    $$S \cup \{(\forall x_1) \ldots (\forall x_n)[F(x_1, \ldots, x_n) \rightarrow \$ANS(x_1, \ldots, x_n)]\}$$
  - **Aplica 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 5a: 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).
```

Obtención de respuestas

- **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).
```

- **Problema 5b: 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-5b1.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).
```

Obtención de respuestas

- **Pruebas**

```
----> UNIT CONFLICT at   0.06 sec ----> 6 [binary,5.1,4.1] $ANS(b).
---------------- PROOF ----------------
1 [] -P(x)|Q(x).
3 [] P(b).
4 [] -Q(z)|$ANS(z).
5 [binary,1.1,3.1] Q(b).
6 [binary,5.1,4.1] $ANS(b).
------------ end of proof -------------

----> UNIT CONFLICT at   0.07 sec ----> 8 [binary,7.1,4.1] $ANS(a).
---------------- PROOF ----------------
1 [] -P(x)|Q(x).
2 [] P(a).
4 [] -Q(z)|$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-5b2.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-5b3.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).
```

Obtención de respuestas

- **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).
```

Obtención de respuestas

- **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.**
  - **Si x es hijo de y e y es hijo de z, entonces x es nieto de z.**

  Obtener la respuesta a la siguiente pregunta:
  - **Si a es nieto de x, ¿quién es x?.**

Obtención de respuestas

- **Entrada**

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

% Si x es hijo de y e y es hijo de z, entonces x es nieto de z:
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).
```

Obtención de respuestas

- **Respuesta**

```
----> UNIT CONFLICT at   0.05 sec ----> 5 [binary,4.1,1.1] $Ans($f1($f1(a))).

---------------- 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 -------------
```

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

Obtención de respuestas

- **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).
```

Razonamiento con igualdad

- **Problema 6: Demostrar que si Francisco es igual a Curro y a Paco, entonces Curro y Paco son iguales**
  - **Entrada** `ej-6a.in`

```
list(sos).
francisco = curro.
francisco = paco.
paco != curro.
end_of_list.

set(binary_res).
```

  - **Salida**

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

Razonamiento con igualdad

- **Axiomas de igualdad** `ej-6b.in`

```
list(sos).
x=x.                  % Reflexividad
x!=y | y=x.           % Simetría
x!=y | y!=z | x=z.    % Transisitividad
francisco = curro.
francisco = paco.
paco != curro.
end_of_list.

set(binary_res).
```

Razonamiento con igualdad

- **Prueba**

```
2 [] x!=y|y=x.
3 [] x!=y|y!=z|x=z.
4 [] francisco=curro.
5 [] francisco=paco.
6 [] paco!=curro.
8 [binary,2.1,4.1] curro=francisco.
9 [binary,2.2,6.1] curro!=paco.
10 [binary,3.1,8.1] francisco!=x|curro=x.
24 [binary,10.1,5.1] curro=paco.
25 [binary,24.1,9.1] $F.
```

Razonamiento con igualdad

- **Mejora con soporte y resolución UR**

```
list(usable).
x=x.                    % Reflexividad
x!=y | y=x.             % Simetría
x!=y | y!=z | x=z.      % Transisitividad
francisco = curro.
francisco = paco.
end_of_list.

list(sos).
paco != curro.
end_of_list.

set(ur_res).
```

Razonamiento con igualdad

- **Prueba**

```
2 [] x!=y|y=x.
3 [] x!=y|y!=z|x=z.
4 [] francisco=curro.
5 [] francisco=paco.
6 [] paco!=curro.
7 [ur,6,3,4] paco!=francisco.
9 [ur,7,2] francisco!=paco.
10 [binary,9.1,5.1] $F.
```

- **Problema 7: Demostrar que si la opuesta de la derecha es la izquierda y la opuesta de la izquierda es la derecha, entonces la opuesta a la opuesta de la derecha es la derecha**

Razonamiento con igualdad

- **Entrada** `ej-7a.in`

```
list(usable).
x=x.                              % Reflexividad
x!=y | y=x.                       % Simetría
x!=y | y!=z | x=z.                % Transisitividad
opuesta(derecha) = izquierda.
opuesta(izquierda) = derecha.
end_of_list.

list(sos).
opuesta(opuesta(derecha)) != derecha.
end_of_list.

set(ur_res).
```

Razonamiento con igualdad

- **Salida**

```
1 [] x=x.
2 [] x!=y|y=x.
3 [] x!=y|y!=z|x=z.
4 [] opuesta(derecha)=izquierda.
5 [] opuesta(izquierda)=derecha.
6 [] opuesta(opuesta(derecha))!=derecha.
** KEPT 7 [ur,6,3,5] opuesta(opuesta(derecha))!=opuesta(izquierda).
** KEPT 8 [ur,6,2] derecha!=opuesta(opuesta(derecha)).
** KEPT 9 [ur,7,2] opuesta(izquierda)!=opuesta(opuesta(derecha)).
```

Razonamiento con igualdad

- **Axioma de sustitución** `ej-7b.in`

```
list(usable).
x=x.                              % Reflexividad
x!=y | y=x.                       % Simetría
x!=y | y!=z | x=z.                % Transisitividad
x!=y | opuesta(x)=opuesta(y).     % Sustitución
opuesta(derecha) = izquierda.
opuesta(izquierda) = derecha.
end_of_list.

list(sos).
opuesta(opuesta(derecha)) != derecha.
end_of_list.

set(ur_res).
```

Razonamiento con igualdad

- **Prueba**

```
3 [] x!=y|y!=z|x=z.
4 [] x!=y|opuesta(x)=opuesta(y).
5 [] opuesta(derecha)=izquierda.
6 [] opuesta(izquierda)=derecha.
7 [] opuesta(opuesta(derecha))!=derecha.
8 [ur,7,3,6] opuesta(opuesta(derecha))!=opuesta(izquierda).
10 [ur,8,4] opuesta(derecha)!=izquierda.
11 [binary,10.1,5.1] $F.
```

- **Problema 8: Demostrar que si Luis es el padre de Juan, entonces Luis es mayor que Juan.**

Razonamiento con igualdad

- **Entrada** `ej-8a.in`

```
list(sos).
x=x.                                    % Reflexividad
x!=y | y=x.                             % Simetría
x!=y | y!=z | x=z.                      % Transisitividad
x!=y | padre(x)=padre(y).               % Sustitución
x1!=x2 | -MAYOR(x1,y) | MAYOR(x2,y).   % Sustitución
y1!=y2 | -MAYOR(x,y1) | MAYOR(x,y2).   % Sustitución
padre(juan)=luis.
MAYOR(padre(x),x).
-MAYOR(luis,juan).
end_of_list.

set(binary_res).
```

Razonamiento con igualdad

- **Prueba**

```
1 [] x=x.
2 [] x!=y|y=x.
3 [] x!=y|y!=z|x=z.
5 [] x1!=x2| -MAYOR(x1,y)|MAYOR(x2,y).
7 [] padre(juan)=luis.
8 [] MAYOR(padre(x),x).
9 [] -MAYOR(luis,juan).
26 [binary,3.1,7.1] luis!=x|padre(juan)=x.
53 [binary,26.1,2.2] padre(juan)=x|x!=luis.
303 [binary,5.3,9.1] x!=luis| -MAYOR(x,juan).
312 [binary,303.1,53.1,unit_del,8,1] $F.
```

Razonamiento con igualdad

- **Estadísticas**

```
Length of proof is 3.  Level of proof is 2.

clauses given                 55
clauses generated            671
  binary_res generated       628
  factors generated           43
tautologies deleted           17
clauses forward subsumed     351
unit deletions                66
clauses kept                 302
usable size                   55
sos size                     256
user CPU time                  2.27 sec
```

Razonamiento con igualdad

- **Estrategia del conjunto soporte** `ej-8b.in`

```
list(usable).
x=x.                                        % Reflexividad
x!=y | y=x.                                 % Simetría
x!=y | y!=z | x=z.                          % Transisitividad
x!=y | padre(x)=padre(y).                   % Sustitución
x1!=x2 | -MAYOR(x1,y) | MAYOR(x2,y).       % Sustitución
y1!=y2 | -MAYOR(x,y1) | MAYOR(x,y2).       % Sustitución
padre(juan)=luis.
MAYOR(padre(x),x).
end_of_list.

list(sos).
-MAYOR(luis,juan).
end_of_list.

set(binary_res).
```

Razonamiento con igualdad

- **Prueba**

```
 5 [] x1!=x2| -MAYOR(x1,y)|MAYOR(x2,y).
 7 [] padre(juan)=luis.
 8 [] MAYOR(padre(x),x).
 9 [] -MAYOR(luis,juan).
11 [binary,9.1,5.3] x!=luis| -MAYOR(x,juan).
15 [binary,11.1,7.1] -MAYOR(padre(juan),juan).
16 [binary,15.1,8.1] $F.
```

Razonamiento con igualdad

- **Estadísticas**

```
Length of proof is 2.  Level of proof is 2.
clauses given                  3
clauses generated             12
  binary_res generated        11
  factors generated            1
tautologies deleted            0
clauses forward subsumed       4
unit deletions                 1
clauses kept                   8
usable size                   11
sos size                       6
user CPU time                  0.06 sec
```

Razonamiento con igualdad

- **Regla de resolución UR** `ej-8c1.in`

```
list(sos).
x=x.                                        % Reflexividad
x!=y | y=x.                                 % Simetría
x!=y | y!=z | x=z.                          % Transisitividad
x!=y | padre(x)=padre(y).                   % Sustitución
x1!=x2 | -MAYOR(x1,y) | MAYOR(x2,y).       % Sustitución
y1!=y2 | -MAYOR(x,y1) | MAYOR(x,y2).       % Sustitución
padre(juan)=luis.
MAYOR(padre(x),x).
-MAYOR(luis,juan).
end_of_list.

set(ur_res).
```

Razonamiento con igualdad

- **Prueba**

```
5 [] x1!=x2| -MAYOR(x1,y)|MAYOR(x2,y).
7 [] padre(juan)=luis.
8 [] MAYOR(padre(x),x).
9 [] -MAYOR(luis,juan).
17 [ur,5,8,9] padre(juan)!=luis.
18 [binary,17.1,7.1] $F.
```

Razonamiento con igualdad

- **Estadísticas**

```
Length of proof is 1.  Level of proof is 1.

clauses given                 13
clauses generated             34
tautologies deleted            0
clauses forward subsumed      26
unit deletions                 0
clauses kept                   8
usable size                   13
sos size                       4
user CPU time                  0.09
```

Razonamiento con igualdad

- **Regla de resolución UR y estrategia soporte** `ej-8c2.in`

```
list(usable).
x=x.                                        % Reflexividad
x!=y | y=x.                                 % Simetría
x!=y | y!=z | x=z.                          % Transisitividad
x!=y | padre(x)=padre(y).                   % Sustitución
x1!=x2 | -MAYOR(x1,y) | MAYOR(x2,y).       % Sustitución
y1!=y2 | -MAYOR(x,y1) | MAYOR(x,y2).       % Sustitución
padre(juan)=luis.
MAYOR(padre(x),x).
end_of_list.

list(sos).
-MAYOR(luis,juan).
end_of_list.

set(ur_res).
```

Razonamiento con igualdad

- **Prueba**

```
5 [] x1!=x2| -MAYOR(x1,y)|MAYOR(x2,y).
7 [] padre(juan)=luis.
8 [] MAYOR(padre(x),x).
9 [] -MAYOR(luis,juan).
10 [ur,9,5,8] padre(juan)!=luis.
11 [binary,10.1,7.1] $F.
```

Razonamiento con igualdad

- **Estadísticas**

```
Length of proof is 1.  Level of proof is 1.

clauses given                  1
clauses generated              2
tautologies deleted            0
clauses forward subsumed       1
unit deletions                 0
clauses kept                   1
usable size                    9
sos size                       1
user CPU time                  0.05 sec
```

Razonamiento con igualdad

- **Axiomas de igualdad**
  - **Mediante fórmulas**

```
all x (x=x).                        % Reflexividad
all x y (x=y -> y=x).               % Simetría
all x y z (x=y & y=z -> x=z).       % Transitividad

% Sustitución:
all x1 ... xn z (xj=z & P(x1,...,xj,...,xn) -> P(x1,...,z,...,xn)).
all x1 ... xn z (xj=z -> f(x1,...,xj,...,xn) = f(x1,...,z,...,xn)).
```

Razonamiento con igualdad

- **Mediante cláusulas**

```
x = x.                      % Reflexividad
x != y | y = x.             % Simetría
x != y | y! = z | x=z.      % Transitividad

% Sustitución:
xj != z | -P(x1,...,xj,...,xn) | P(x1,...,z,...,xn).
xj != z | f(x1,...,xj,...,xn) = f(x1,...,z,...,xn)).
```

Paramodulación

- **Regla de paramodulación**

```
L1 |...| Li[t1] |...| Ln.
M1 |...| M{j-1} | t2=t3 | M{j+1} |...| Mk.
-------------------------------------------------------------
(L1 |...|Li[t3] |...|Ln | M1 |...| M{j-1}| t2=t3 | M{j+1}|...| Mk)s

donde s es u.m.g. de t1 y t2
```

Paramodulación

- **Problema 8 mediante paramodulación**
  - **Entrada** `ej-8d.in`

```
x=x.                    % Reflexividad
padre(juan)=luis.
MAYOR(padre(x),x).
end_of_list.

list(sos).
-MAYOR(luis,juan).
end_of_list.

set(para_into).
```

Paramodulación

- **Prueba**

```
2 [] padre(juan)=luis.
3 [] MAYOR(padre(x),x).
4 [] -MAYOR(luis,juan).
5 [para_into,4.1.1,2.1.2] -MAYOR(padre(juan),juan).
6 [binary,5.1,3.1] $F.
```

- **Cláusula 5**

```
from 2.1.2                 padre(juan)='luis'
into 4.1.1                 -MAYOR('luis',juan)
[para_into,4.1.1,2.1.2] -MAYOR(padre(juan),juan)
```

Paramodulación

- **Estadísticas**

```
Length of proof is 1.  Level of proof is 1.

clauses given                  1
clauses generated              1
tautologies deleted            0
clauses forward subsumed       0
unit deletions                 0
factor simplifications         0
clauses kept                   1
usable size                    4
sos size                       1
user CPU time                  0.04 sec
```

Paramodulación

- **Problema 6 mediante paramodulación**
  - **Entrada** `ej-6d.in`

```
list(usable).
x=x.                    % Reflexividad
francisco = curro.
francisco = paco.
end_of_list.

list(sos).
paco != curro.
end_of_list.

set(para_into).
```

Paramodulación

- **Prueba**

```
2 [] francisco=curro.
3 [] francisco=paco.
4 [] paco!=curro.
5 [para_into,4.1.1,3.1.2] francisco!=curro.
6 [binary,5.1,2.1] $F.
```

- **Cláusula 5**

```
from 3.1.2                 francisco=‘paco’
into 4.1.1                 ‘paco’!=curro
[para_into,4.1.1,3.1.2] francisco!=curro.
```

Paramodulación

- **Problema 7 mediante paramodulación**
  - **Entrada** `ej-7c.in`

```
list(usable).
x=x.                                % Reflexividad
opuesta(derecha) = izquierda.
opuesta(izquierda) = derecha.
end_of_list.

list(sos).
opuesta(opuesta(derecha)) != derecha.
end_of_list.

set(para_into).
```

Paramodulación

- **Prueba**

```
2 [] opuesta(derecha)=izquierda.
3 [] opuesta(izquierda)=derecha.
4 [] opuesta(opuesta(derecha))!=derecha.
6 [para_into,4.1.1.1,2.1.1] opuesta(izquierda)!=derecha.
7 [binary,6.1,3.1] $F.
```

- **Cláusula 6**

```
from 2.1.1                 ‘opuesta(derecha)’=izquierda
into 4.1.1                 opuesta(‘opuesta(derecha)’)!=derecha.
[para_into,4.1.1.1,2.1.1] opuesta(izquierda)!=derecha.
```

Demodulación

- **Def. de demodulación:**
  - **$C[t]$ es una clásula que contiene el término t**
  - **$t_1 = t_2$**
  - **σ es un unificador de máxima generalidad de t y t_1**
  - **Demodulación: $C[t_2\sigma]$**

Demodulación

- **Problema 6 mediante demodulación**
  - **Entrada** `ej-6e.in`

```
list(usable).
x=x.                    % Reflexividad
end_of_list.

list(demodulators).
curro = francisco.
paco = francisco.
end_of_list.

list(sos).
paco != curro.
end_of_list.

set(binary_res).
set(process_input).
```

Demodulación

- **Prueba**

```
1 [] x=x.
2 [] curro=francisco.
3 [] paco=francisco.
4 [copy,5,demod,3,2] francisco!=francisco.
5 [] paco!=curro.
6 [binary,4.1,1.1] $F.
```

Demodulación

- **Problema 7 mediante demodulación**
  - **Entrada** `ej-07d.in`

```
list(usable).
x=x.
end_of_list.

list(demodulators).
opuesta(derecha) = izquierda.
opuesta(izquierda) = derecha.
end_of_list.

list(sos).
opuesta(opuesta(derecha)) != derecha.
end_of_list.

set(binary_res).
set(process_input).
```

Demodulación

- **Prueba**

```
1 [] x=x.
2 [] opuesta(derecha)=izquierda.
3 [] opuesta(izquierda)=derecha.
4 [copy,5,demod,2,3] derecha!=derecha.
5 [] opuesta(opuesta(derecha))!=derecha.
6 [binary,4.1,1.1] $F.
```

Demodulación

- **Problema 9: Demostrar que si Juan está casado y es el tío de Pepe, entonces el hermano del padre de Juan está casado**
  - **Entrada** `ej-9a.in`

```
list(usable).
x=x.
casado(juan).
end_of_list.

list(demodulators).
hermano(padre(x)) = tio(x).
tio(pepe)=juan.
end_of_list.

list(sos).
-casado(hermano(padre(pepe))).
end_of_list.

set(binary_res).
set(process_input).
```

Demodulación

- **Prueba**

```
2 [] casado(juan).
3 [] hermano(padre(x))=tio(x).
4 [] tio(pepe)=juan.
5 [copy,6,demod,3,4] -casado(juan).
6 [] -casado(hermano(padre(pepe))).
7 [binary,5.1,2.1] $F.
```

Demodulación

- **Problema 9 mediante paramodulación**
  - **Entrada** `ej-09b.in`

```
list(usable).
x=x.
casado(juan).
hermano(padre(x)) = tio(x).
tio(pepe)=juan.
end_of_list.

list(sos).
-casado(hermano(padre(pepe))).
end_of_list.

set(para_into).
```

Demodulación

- **Prueba**

```
2 [] casado(juan).
3 [] hermano(padre(x))=tio(x).
4 [] tio(pepe)=juan.
5 [] -casado(hermano(padre(pepe))).
6 [para_into,5.1.1,3.1.1] -casado(tio(pepe)).
7 [para_into,6.1.1,4.1.1] -casado(juan).
8 [binary,7.1,2.1] $F.
```

Problema de grupos

- **Problema 10: Sea G un grupo y e su elemento neutro. Demostrar que si, para todo x de G, $x^2 = e$, entonces G es conmutativo.**
- **Formalización**
  - **Axiomas de grupo**

    $(\forall x)[e.x = x]$
    $(\forall x)[x.e = x]$
    $(\forall x)[x.x^{-1} = e]$
    $(\forall x)[x^{-1}.x = e]$
    $(\forall x)(\forall y)(\forall z)[(x.y).z = x.(y.z)]$
  - **Hipótesis**

    $(\forall x)[x.x = e]$
  - **Conclusión**

    $(\forall x)(\forall y)[x.y = y.x]$

Problema de grupos

- **Entrada** `ej-10a.in`

```
op(400, xfy, *).
op(300, yf, ^).

list(usable).
x = x.                              % Reflexividad
e * x = x.                          % Ax. 1
x * e = x.                          % Ax. 2
x^ * x = e.                         % Ax. 3
x * x^ = e.                         % Ax. 4
(x * y) * z = x * (y * z).          % Ax. 5
end_of_list.

list(sos).
x * x = e.
a * b != b * a.
end_of_list.

set(para_into).
set(para_from).
```

Problema de grupos

- **Prueba**

```
2 [] e*x=x.
3 [] x*e=x.
6 [] (x*y)*z=x*y*z.
7 [] x*x=e.
8 [] a*b!=b*a.
19 [para_from,7.1.2,3.1.1.2] x*y*y=x.
20 [para_from,7.1.2,2.1.1.1] (x*x)*y=y.
31 [para_into,19.1.1,6.1.2] (x*y)*y=x.
167 [para_into,20.1.1,6.1.1] x*x*y=y.
170 [para_from,20.1.1,6.1.1] x=y*y*x.
496 [para_into,167.1.1.2,31.1.1] (x*y)*x=y.
755 [para_into,496.1.1.1,170.1.2] x*y=y*x.
756 [binary,755.1,8.1] $F.
```

Problema de grupos

- **Cláusula 19**

```
from 7.1.2                  x1*x1=E
into 3.1.1.2                x2*E=x2
[para_from,7.1.2,3.1.1.2] x2*(x1*x1)=x2   {x2/x, x1/y}
                            x*(y*y)=x
```

- **Cláusula 20**

```
from 7.1.2                  x1*x1=E
into 2.1.1.1                E*x2=x2.
[para_from,7.1.2,2.1.1.1] (x1*x1)*x2=x2  {x1/x, x2/y}
                            (x*x)*y=y
```

- **Cláusula 31**

```
from 6.1.2                (x1*y1)*z1=X1*(Y1*Z1)
into 19.1.1               X2*(Y2*Y2)=x2          {x2/x1, y2/y1, z1/y1}
[para_into,2.1.1,6.1.2]   (x1*y1)*y1=x1          {x1/x, y1/y}
                          (x*y)*y=x
```

Problema de grupos

- **Cláusula 167**

```
from 6.1.1                  (X1*Y1)*Z1=x1*(y1*z1)
into 20.1.1                 (X2*X2)*Y2=y2          {x1/x2, y1/x2, z1/y2}
[para_into,20.1.1,6.1.1] x2*(x2*y2)=y2             {x2/x, y2/y}
                            x*(x*y)=y
```

- **Cláusula 170**

```
from 20.1.1                 (X1*X1)*Y1=y1
into 6.1.1                  (X2*Y2)*Z2=x2*(y2*z2) {x2/x1, y2/x1, z2/y1}
[para_from,20.1.1,6.1.1] y1=x1*(x1*y1)            {y1/x, x1/y}
                            x=y*(y*x)
```

- **Cláusula 496**

```
from 31.1.1                    (X1*Y1)*Y1=x1
into 167.1.1.2                 x2*(X2*Y2)=y2  {x2/x1*y1, y2/y1}
[para_into,167.1.1.2,31.1.1] (x1*y1)*x1=y1    {x1/x, y1/y}
                               (x*y)*x=y
```

Problema de grupos

- **Cláusula 755**

```
from 170.1.2                    x1=Y1*(Y1*X1)
into 496.1.1.1                  (X2*Y2)*x2=y2  {x2/y1, y2/y1*x1}
[para_into,496.1.1.1,170.1.2] x1*y1=y1*x1     {x1/x, y1/y}
                                x*y=y*x
```

- **Estadísticas**

```
Length of proof is 7.  Level of proof is 4.

clauses given                  72
clauses generated            5741
clauses forward subsumed     4994
clauses kept                  747
clauses back subsumed          45
usable size                    72
sos size                      638
user CPU time                   3.76 sec.
```

Problema de grupos

- **Mejora con demoduladores**
  - **Entrada** `ej-10b.in`

```
op(400, xfy, *).
op(300, yf, ^).

list(usable).
e * x = x.                          % Ax. 1
x * e = x.                          % Ax. 2
x^ * x = e.                         % Ax. 3
x * x^ = e.                         % Ax. 4
(x * y) * z = x * (y * z).          % Ax. 5
x = x.
end_of_list.

list(sos).
x * x = e.
a * b != b * a.
end_of_list.
```

Problema de grupos

```
list(demodulators).
e * x = x.                          % Ax. 1
x * e = x.                          % Ax. 2
x^ * x = e.                         % Ax. 3
x * x^ = e.                         % Ax. 4
(x * y) * z = x * (y * z).          % Ax. 5
end_of_list.

set(para_into).
set(para_from).
```

Problema de grupos

- **Prueba**

```
1 [] e*x=x.
5 [] (x*y)*z=x*y*z.
7 [] x*x=e.
8 [] a*b!=b*a.
10 [] x*e=x.
13 [] (x*y)*z=x*y*z.
14 [para_into,7.1.1,5.1.2,demod,13,13,13] x*y*x*y=e.
20 [para_from,7.1.2,1.1.1.1,demod,13] x*x*y=y.
494 [para_from,14.1.1,20.1.1.2,demod,10] x=y*x*y.
540 [para_from,494.1.2,20.1.1.2] x*y=y*x.
541 [binary,540.1,8.1] $F.
```

Problema de grupos

- **Estadísticas**

```
clauses given                  68
clauses generated            5562
demod & eval rewrites         727
clauses forward subsumed     5035
clauses kept                  527
usable size                    70
sos size                      458
demodulators size               5
user CPU time                   4.51 sec
```

Problema de grupos

- **Mejora con demoduladores dinámicos**
  - **Entrada** `ej-10c.in`

```
op(400, xfy, *).
op(300, yf, ^).

list(usable).
e * x = x.                          % Ax. 1
x * e = x.                          % Ax. 2
x^ * x = e.                         % Ax. 3
x * x^ = e.                         % Ax. 4
(x * y) * z = x * (y * z).          % Ax. 5
x = x.                              % Ax. 6
end_of_list.

list(sos).
x * x = e.
a * b != b * a.
end_of_list.
```

Problema de grupos

```
list(demodulators).
e * x = x.                          % Ax. 1
x * e = x.                          % Ax. 2
x^ * x = e.                         % Ax. 3
x * x^ = e.                         % Ax. 4
(x * y) * z = x * (y * z).          % Ax. 5
end_of_list.

set(para_into).
set(para_from).
set(dynamic_demod).
```

Problema de grupos

- **Prueba**

```
5 [] (x*y)*z=x*y*z.
7 [] x*x=e.
8 [] a*b!=b*a.
9 [] e*x=x.
10 [] x*e=x.
13 [] (x*y)*z=x*y*z.
14 [para_into,7.1.1,5.1.2,demod,13,13,13] x*y*x*y=e.
19 [para_from,7.1.1,5.1.1.1,demod,9,flip.1] x*x*y=y.
31 [para_from,14.1.1,19.1.1.2,demod,10,flip.1] x*y*x=y.
36 [para_from,31.1.1,19.1.1.2] x*y=y*x.
37 [binary,36.1,8.1] $F.
```

Problema de grupos

- **Estadísticas**

```
clauses given                 10
clauses generated            219
demod & eval rewrites        255
clauses forward subsumed     206
clauses kept                  13
new demodulators              10
clauses back subsumed          1
usable size                   15
sos size                       5
demodulators size             15
user CPU time                  0.22 sec
```

Bibliografía

- Alonso, J.A.; Fernández, A. y Pérez, M.J. *Razonamiento automático* (en *Lógica formal (Orígenes, métodos y aplicaciones*, Ed. Kronos, 1995)
- Baj, F. *Logic with PAIL* (Technical Report, 1991)
- Chang, C.L.; Lee, R.C.T. *Symbolic logic and mechanical theorem proving.* (Academic Press, 1973)
- Duffy, D. *Principles of Automated Theorem Proving* (John Wiley, 1991)

Bibliografía

- **Genesereth, M.R. y Nilsson, N.J. *Logical foundations of Artificial Intelligence* (Morgan Kaufmann, 1987)**
  - **Cap. 4: "Resolution"**
  - **Cap. 5: "Resolution strategies"**
- **McCune, W. OTTER *3.0 Reference Manual and Guide.* (Technical Report ANL–94/6, Argone National Laboratory, 1994)**
- **Wos, L.; Overbeek, R.; Lusk, E. y Boyle, J. *Automated Reasoning: Introduction and Applications, (2nd ed.)* (McGraw–Hill, 1992)**