Tema DA–5: Razonamiento automático con igualdad

José A. Alonso Jiménez
Miguel A. Gutiérrez Naranjo

Dpto. de Ciencias de la Computación e Inteligencia Artificial

Universidad de Sevilla

Axiomas de igualdad

- **Problema 1:**
  - **Enunciado: Demostrar que si Francisco es igual a Curro y a Paco, entonces Curro y Paco son iguales**
- **Problema 1: Primer intento**
  - **Entrada** `ej-1a.in`

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

set(binary_res).
```

  - **Salida**

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

Axiomas de igualdad

- **Problema 1b: Con axiomas de igualdad**
  - **Entrada** `ej-1b.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).
```

  - **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.
```

  - **Estadísticas**

Analiz.	Gener.	Reten.	Sub. adel.	Sub. atrás	Seg.
10	35	18	10	0	0.63

Axiomas de igualdad

- **Problema 1c: Con soporte y resolución UR**
  - **Entrada** `ej-1c.in`

```
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).
```

  - **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.
```

  - **Estadísticas**

Analiz.	Gener.	Reten.	Sub. adel.	Sub. atrás	Seg.
2	7	3	4	0	0.00

Axiomas de sustitución funcionales

- **Problema 2:**
  - **Enunciado: 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**
- **Problema 2a: Con axiomas de igualdad**
  - **Entrada** `ej-2a.in`

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

set(binary_res).
```

  - **Salida**

```
KEPT 35: opuesta(opuesta(derecha))!=opuesta(izquierda).
```

  - **No para**

Axiomas de sustitución funcionales

- **Problema 2b1: Con axioma de sustitución**
  - **Entrada** `ej-2b1.in`

```
include('ej-2a.in').

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

  - **Prueba**

```
 2 [] x!=y|y=x.
 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 [binary,2.1,6.1] derecha=opuesta(izquierda).
 9 [binary,2.1,5.1] izquierda=opuesta(derecha).
10 [binary,2.2,7.1] derecha!=opuesta(opuesta(derecha)).
11 [binary,4.1,9.1] opuesta(izquierda)
                    =opuesta(opuesta(derecha)).
36 [binary,3.1,8.1] opuesta(izquierda)!=x|derecha=x.
81 [binary,36.1,11.1] derecha=opuesta(opuesta(derecha)).
82 [binary,81.1,10.1] $F.
```

  - **Estadísticas**

Analiz.	Gener.	Reten.	Sub. adel.	Sub. atrás	Seg.
23	**130**	**74**	**49**	**0**	**0.02**

Axiomas de sustitución funcionales

- **Problema 2b2: Con soporte y resolución UR**
  - **Entrada** `ej-2b2.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).
assign(stats_level,1).
```

  - **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.
```

  - **Estadísticas**

Analiz.	Gener.	Reten.	Sub. adel.	Sub. atrás	Seg.
3	8	3	5	0	0.00

Axiomas de sustitución relacionales

- **Problema 3**
  - **Demostrar que si los padres son mayores que los hijos y Luis es el padre de Juan, entonces Luis es mayor que Juan.**
- **Problema 3a: Con axiomas de sustitución relacionales**
  - **Entrada** `ej-3a.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
Mayor(Padre(x),x).
Padre(Juan)=Luis.
-Mayor(Luis,Juan).
end_of_list.

set(binary_res).
```

Axiomas de sustitución relacionales

- **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 [] Mayor(Padre(x),x).
  8 [] Padre(Juan)=Luis.
  9 [] -Mayor(Luis,Juan).
 26 [binary,3.1,8.1] 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,7,1] $F.
```

- **Estadísticas**

Analiz.	Gener.	Reten.	Sub. adel.	Sub. atrás	Seg.
55	**671**	**302**	**351**	**0**	**0.23**

Axiomas de sustitución relacionales

- **Problema 3b: Con soporte y resolución UR**
  - **Entrada** `ej-3b.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
Mayor(Padre(x),x).
Padre(Juan)=Luis.
end_of_list.

list(sos).
-Mayor(Luis,Juan).
end_of_list.

set(ur_res).
```

  - **Prueba**

```
 5 [] x1!=x2| -Mayor(x1,y)|Mayor(x2,y).
 7 [] Mayor(Padre(x),x).
 8 [] Padre(Juan)=Luis.
 9 [] -Mayor(Luis,Juan).
10 [ur,9,5,7] Padre(Juan)!=Luis.
11 [binary,10.1,8.1] $F.
```

  - **Estadísticas**

Analiz.	Gener.	Reten.	Sub. adel.	Sub. atrás	Seg.
1	2	1	1	0	0.00

Axiomas de sustitución relacionales

- **Problema 3c: Sin funciones ni igualdad**
  - **Entrada** `ej-3c.in`

```
list(sos).
-Padre(x,y) | Mayor(x,y).
Padre(Luis,Juan).
-Mayor(Luis,Juan).
end_of_list.

set(binary_res).
```

  - **Prueba**

```
1 [] -Padre(x,y)|Mayor(x,y).
2 [] Padre(Luis,Juan).
3 [] -Mayor(Luis,Juan).
4 [binary,1.1,2.1] Mayor(Luis,Juan).
5 [binary,4.1,3.1] $F.
```

  - **Estadísticas**

Analiz.	Gener.	Reten.	Sub. adel.	Sub. atrás	Seg.
3	**1**	**1**	**0**	**0**	**0.00**

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

% Axiomas de sustitución de la función f/3
all x1 x2 x3 y (x1=y -> f(x1,x2,x3) = f(y,x2,x3)).
all x1 x2 x3 y (x2=y -> f(x1,x2,x3) = f(x1,y,x3)).
all x1 x2 x3 y (x3=y -> f(x1,x2,x3) = f(x1,x2,y)).

% Axiomas de sustitución de la relación P/3
all x1 x2 x3 y (x1=y & P(x1,x2,x3) -> P(y.x2.x3)).
all x1 x2 x3 y (x2=y & P(x1,x2,x3) -> P(x1.y.x3)).
all x1 x2 x3 y (x3=y & P(x1,x2,x3) -> P(x1.x2.y)).
```

  - **Mediante cláusulas**

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

% Axiomas de sustitución de la función f/3
x1 != y | f(x1,x2,x3) = f(y,x2,x3)).
x2 != y | f(x1,x2,x3) = f(x1,y,x3)).
x3 != y | f(x1,x2,x3) = f(x1,x2,y)).

% Axiomas de sustitución de la relación P/3
x1 != y | -P(x1,x2,x3) | P(y.x2.x3)).
x2 != y | -P(x1,x2,x3) | P(x1.y.x3)).
x3 != y | -P(x1,x2,x3) | P(x1.x2.y)).
```

Paramodulación

- **Problema 4:**
  - **Enunciado:** $\{P(a), a = b\} \models P(b)$
  - **Entrada:** `ej-4.in`

```
list(sos).
P(a).
a=b.
-P(b).
end_of_list.

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

  - **Búsqueda**

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

given clause #2: (wt=2) 3 [] -P(b).

given clause #3: (wt=3) 2 [] a=b.
** KEPT (pick-wt=3): 4 [para_into,2.1.1,2.1.1] b=b.
** KEPT (pick-wt=3): 5 [para_into,2.1.2,2.1.2] a=a.
** KEPT (pick-wt=2): 6 [para_from,2.1.1,1.1.1] P(b).
```

  - **¿Cómo evitar conocimiento "evidente":** `a=a`, `b=b`**?**

Paramodulación

• Problema 5: Ejemplos de paramodulación

- **Entrada:** `ej-5.in`

```
list(sos).
P(f(x,b),x) | Q(x).
f(a,x)=x | R(x).
end_of_list.

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

- **Búsqueda**

```
given clause #1: (wt=7) 1 [] P(f(x,b),x)|Q(x).

given clause #2: (wt=7) 2 [] f(a,x)=x|R(x).
** KEPT 3 [para_into,2.1.1,2.1.1] x=x|R(x).
** KEPT 4 [para_from,2.1.1,1.1.1] P(b,a)|Q(a)|R(b).
```

- **Explicaciones**

```
3 [para_into,2.1.1,2.1.1] x=x | R(x).

Into 2.1.1    ‘f(a,x1)’=x1 | R(x1).
from 2.1.1    ‘f(a,x2)’=x2 | R(x2).
Unificador    s = {x2/x1}
Paramodulante ((x2=x1) | R(x2) | R(x1))s
              => x1=x1 | R(x1)
              => x=x | R(x)
```

Paramodulación

```
4 [para_from,2.1.1,1.1.1] P(b,a) | Q(a) | R(b).

From 2.1.1    'f(a,x1)'=x1 | R(x1).
into 1.1.1    P('f(x2,b)',x2) | Q(x2).
Unificador    s = {x2/a, x1/b}
Paramodulante (P(x1,x2) | Q(x2) | R(x1))s
              => P(b) | Q(a) | R(b).
```

- **Regla de paramodulación**
  - **Izquierda**

$$\frac{\begin{array}{c} s_1 = t \cup C \\ L[s_2] \cup D \end{array}}{L[t]\sigma \cup C\sigma \cup D\sigma}$$

    donde σ es un unificador de máxima generalidad de s_1 y s_2
  - **Derecha**

$$\frac{\begin{array}{c} t = s_1 \cup C \\ L[s_2] \cup D \end{array}}{L[t]\sigma \cup C\sigma \cup D\sigma}$$

    donde σ es un unificador de máxima generalidad de s_1 y s_2

Paramodulación

- **Problema 3d: Mediante paramodulación**
  - **Entrada** `ej-3d.in`

```
x=x.                    % Reflexividad
Mayor(Padre(x),x).
Padre(Juan)=Luis.
end_of_list.

list(sos).
-Mayor(Luis,Juan).
end_of_list.

set(para_into).
```

  - **Prueba**

```
2 [] Mayor(Padre(x),x).
3 [] Padre(Juan)=Luis.
4 [] -Mayor(Luis,Juan).
5 [para_into,4.1.1,3.1.2] -Mayor(Padre(Juan),Juan).
6 [binary,5.1,2.1] $F.
```

  - **Cláusula 5**

```
into 4.1.1                  -Mayor('Luis',Juan)
from 4.1.2                  Padre(Juan)='Luis'
[para_into,4.1.1,3.1.2]     -Mayor(Padre(Juan),Juan)
```

  - **Estadísticas**

Analiz.	Gener.	Reten.	Sub. adel.	Sub. atrás	Seg.
1	1	1	0	0	0.02

Paramodulación

- **Problema 1d mediante paramodulación**
  - **Entrada** `ej-1d.in`

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

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

set(para_into).
```

  - **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**

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

  - **Estadísticas**

Analiz.	Gener.	Reten.	Sub. adel.	Sub. atrás	Seg.
1	1	1	0	0	0.00

Paramodulación

- **Problema 2c mediante paramodulación**
  - **Entrada** `ej-2c.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).
set(para_from).
```

  - **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** `[para_into,4.1.1.1,2.1.1]`

```
into 4.1.1 opuesta(‘opuesta(derecha)’)!=derecha.
from 2.1.1 ‘opuesta(derecha)’=izquierda
6          opuesta(izquierda)!=derecha.
```

  - **Estadísticas**

Analiz.	Gener.	Reten.	Sub. adel.	Sub. atrás	Seg.
1	2	2	0	0	0.01

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: $C[t_2\sigma]$
- **Problema 1e mediante demodulación**
  - Entrada `ej-1e.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(process_input).
```

  - 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 2d mediante demodulación

- **Entrada** `ej-2d.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(process_input).
```

- **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 6**
  - **Enunciado: Demostrar que si Juan está casado y es el tío de Pepe, entonces el hermano del padre de Juan está casado**
  - **Entrada** `ej-6a.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).
```

  - **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 6b: mediante paramodulación**
  - **Entrada** `ej-6b.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).
```

  - **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.
```

Operadores

- **Problema 7**
  - **Enunciado: 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**

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

Operadores

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

Operadores

- **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.
```

- **Estadísticas**

Analiz.	Gener.	Reten.	Sub. adel.	Sub. atrás	Seg.
72	**5741**	**747**	**4994**	**45**	**0.26**

- **Cláusula 19** [para_from,7.1.2,3.1.1.2]

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

- **Cláusula 20** [para_from,7.1.2,2.1.1.1]

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

Operadores

- **Cláusula 31** [para_into,19.1.1,6.1.2]

```
into 19.1.1     X2*(Y2*Y2)=x2
from 6.1.2      (x1*y1)*z1=X1*(Y1*Z1)
Unificador      s = {x2/x1, y2/y1, z1/y1}
Paramodulante   ((x1*y1)*z1=x2)s
                => (x1*y1)*y1=x1     {x1/x, y1/y}
                => (x*y)*y=x
```

- **Cláusula 167** [para_into,20.1.1,6.1.1]

```
into 20.1.1     (X2*X2)*Y2=y2
from 6.1.1      (X1*Y1)*Z1=x1*(y1*z1)
Unificador      s = {x1/x2, y1/x2, z1/y2}
Paramodulante   (x1*(y1*z1)=y2)s
                => x2*(x2*y2)=y2          {x2/x, y2/y}
                => x*(x*y)=y
```

- **Cláusula 496** [para_into,167.1.1.2,31.1.1]

```
into 167.1.1.2  x2*(X2*Y2)=y2
from 31.1.1     (X1*Y1)*Y1=x1
Unificador      s = {x2/x1*y1, y2/y1}
Paramodulante   (x2*x1=y2)s
                => (x1*y1)*x1=y1  {x1/x, y1/y}
                => (x*y)*x=y
```

- **Cláusula 755** [para_into,496.1.1.1,170.1.2]

```
into 496.1.1.1  (X2*Y2)*x2=y2
from 170.1.2    x1=Y1*(Y1*X1)
Unificador      s = {x2/y1, y2/y1*x1}
Paramodulante   ((x1*x2)=y2)s
                => x1*y1=y1*x1    {x1/x, y1/y}
                => x*y=y*x
```

Demoduladores

- **Mejora con demoduladores**
  - **Entrada** `ej-7b.in`

```
include('ej-7a.in').

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.
```

  - **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.
```

  - **Estadísticas**

Analiz.	Gener.	Reten.	Sub. adel.	Sub. atrás	Seg.
68	**5562**	**527**	**5035**	**7**	**0.26**

Demoduladores dinámicos

• Mejora con demoduladores dinámicos

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

```
include('ej-7b.in').

set(dynamic_demod).
```

- **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.
```

- **Estadísticas**

Analiz.	Gener.	Reten.	Sub. adel.	Sub. atrás	Seg.
10	219	13	206	1	0.01

Modo autónomo

• Problema 7d: con modo autónomo

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

```
set(auto2).

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
x * x = e.
a * b != b * a.
end_of_list.
```

- **Prueba**

```
1 [] a*b!=b*a.
2 [copy,1,flip.1] b*a!=a*b.
4,3 [] e*x=x.
6,5 [] x*e=x.
11 [] (x*y)*z=x*y*z.
14 [] x*x=e.
18 [para_into,11.1.1.1,14.1.1,demod,4,flip.1] x*x*y=y.
24 [para_into,11.1.1,14.1.1,flip.1] x*y*x*y=e.
34 [para_from,24.1.1,18.1.1.2,demod,6,flip.1] x*y*x=y.
38 [para_from,34.1.1,18.1.1.2] x*y=y*x.
39 [binary,38.1,2.1] $F.
```

- **Estadísticas**

Analiz.	Gener.	Reten.	Sub. adel.	Sub. atrás	Seg.
12	90	20	87	8	0.18

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)
  - Cap. 8 "The equality relation"
- 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)