Demostración automática de teoremas

Tema 5: Razonamiento automático con igualdad

J.A. Alonso, J. Borrego, A. Chávez y F.J. Martín

Dpto. Ciencias de la Computación e Inteligencia Artificial

Universidad de Sevilla

Axiomas de igualdad

- Demostrar que si Francisco es igual a Curro y a Paco, entonces Curro y Paco son iguales
- Formalización en OTTER

”ej-1a.in”

```
1  list(sos).
2   francisco = curro.
3   francisco = paco.
4   paco != curro.
5  end_of_list.
6
7  set(binary_res).
```

Axiomas de igualdad

- Formalización en OTTER con axiomas de igualdad

”ej-1b.in”

```
1  list(sos).
2   x=x.                      % Reflexividad
3   x!=y | y=x.               % Simetría
4   x!=y | y!=z | x=z.        % Transisitividad
5   francisco = curro.
6   francisco = paco.
7   paco != curro.
8  end_of_list.
9
10 set(binary_res).
```

Axiomas de igualdad

- Formalización en OTTER con axiomas de igualdad

”ej-1b.in”

```
1  list(sos).
2   x=x.                      % Reflexividad
3   x!=y | y=x.               % Simetría
4   x!=y | y!=z | x=z.        % Transisitividad
5   francisco = curro.
6   francisco = paco.
7   paco != curro.
8  end_of_list.
9
10 set(binary_res).
```

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

Axiomas de igualdad

- Formalización en OTTER con soporte y resolución UR

”ej-1c.in”

```
1  list(usable).
2   x=x.                      % Reflexividad
3   x!=y | y=x.               % Simetría
4   x!=y | y!=z | x=z.        % Transisitividad
5   francisco = curro.
6   francisco = paco.
7  end_of_list.
8
9  list(sos).
10  paco != curro.
11 end_of_list.
12
13 set(ur_res).
```

Axiomas de igualdad

- Formalización en OTTER con soporte y resolución UR

”ej-1c.in”

```
1  list(usable).
2   x=x.                      % Reflexividad
3   x!=y | y=x.               % Simetría
4   x!=y | y!=z | x=z.        % Transisitividad
5   francisco = curro.
6   francisco = paco.
7  end_of_list.
8
9  list(sos).
10  paco != curro.
11 end_of_list.
12
13 set(ur_res).
```

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

Axiomas de sustitución funcionales

- 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
- Formalización en OTTER con axiomas de igualdad

”ej-2a.in”

```
1   list(sos).
2    x=x.                          % Reflexividad
3    x!=y | y=x.                   % Simetría
4    x!=y | y!=z | x=z.            % Transisitividad
5    opuesta(derecha) = izquierda.
6    opuesta(izquierda) = derecha.
7    opuesta(opuesta(derecha)) != derecha.
8   end_of_list.
9
10  set(binary_res).
```

Axiomas de sustitución funcionales

- Formalización en OTTER con un axioma de sustitución

”ej-2b1.in”

```
1  include('ej-2a.in').
2
3  list(sos).
4   x!=y | opuesta(x)=opuesta(y).
5  end_of_list.
```

Axiomas de sustitución funcionales

- Formalización en OTTER con un axioma de sustitución

”ej-2b1.in”

```
1  include('ej-2a.in').
2
3  list(sos).
4   x!=y | opuesta(x)=opuesta(y).
5  end_of_list.
```

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

Axiomas de sustitución funcionales

- Formalización en OTTER con soporte y resolución UR

”ej-2b2.in”

```
1  list(usable).
2   x=x.                                % Reflexividad
3   x!=y | y=x.                         % Simetría
4   x!=y | y!=z | x=z.                  % Transisitividad
5   x!=y | opuesta(x)=opuesta(y).       % Sustitución
6   opuesta(derecha) = izquierda.
7   opuesta(izquierda) = derecha.
8  end_of_list.
9
10 list(sos).
11  opuesta(opuesta(derecha)) != derecha.
12 end_of_list.
13
14 set(ur_res).
```

Axiomas de sustitución funcionales

- Formalización en OTTER con soporte y resolución UR

”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).
```

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

Axiomas de sustitución relacionales

- Demostrar que si los padres son mayores que los hijos y Luis es el padre de Juan, entonces Luis es mayor que Juan.
- Formalización en OTTER con axiomas de sustitución relacionales

”ej-3a.in”

```
1   list(sos).
2    x=x.                                        % Reflexividad
3    x!=y | y=x.                                 % Simetría
4    x!=y | y!=z | x=z.                          % Transisitividad
5    x!=y | Padre(x)=Padre(y).                   % Sustitución
6    x1!=x2 | -Mayor(x1,y) | Mayor(x2,y). % Sustitución
7    y1!=y2 | -Mayor(x,y1) | Mayor(x,y2). % Sustitución
8    Mayor(Padre(x),x).
9    Padre(Juan)=Luis.
10   -Mayor(Luis,Juan).
11  end_of_list.
12
13  set(binary_res).
```

Axiomas de sustitución relacionales

- Prueba obtenida

```
  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.
```

Axiomas de sustitución relacionales

- Prueba obtenida

```
  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.
```

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

Axiomas de sustitución relacionales

- Formalización en OTTER con soporte y resolución UR

”ej-3b.in”

```
1  list(usable).
2   x=x.                                          % Reflexividad
3   x!=y | y=x.                                   % Simetría
4   x!=y | y!=z | x=z.                            % Transisitividad
5   x!=y | Padre(x)=Padre(y).                     % Sustitución
6   x1!=x2 | -Mayor(x1,y) | Mayor(x2,y).          % Sustitución
7   y1!=y2 | -Mayor(x,y1) | Mayor(x,y2).          % Sustitución
8   Mayor(Padre(x),x).
9   Padre(Juan)=Luis.
10 end_of_list.
11
12 list(sos).
13  -Mayor(Luis,Juan).
14 end_of_list.
15
16 set(ur_res).
```

Axiomas de sustitución relacionales

- Formalización en OTTER con soporte y resolución UR

”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).
```

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

Axiomas de sustitución relacionales

- Formalización en OTTER mediante cláusulas

”ej-3c.in”

```
1  list(sos).
2   -Padre(x,y) | Mayor(x,y).
3   Padre(Luis,Juan).
4   -Mayor(Luis,Juan).
5  end_of_list.
6
7  set(binary_res).
```

Axiomas de sustitución relacionales

- Formalización en OTTER mediante cláusulas

”ej-3c.in”

```
1  list(sos).
2   -Padre(x,y) | Mayor(x,y).
3   Padre(Luis,Juan).
4   -Mayor(Luis,Juan).
5  end_of_list.
6
7  set(binary_res).
```

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

Razonamiento con igualdad

- Formalización de los axiomas de igualdad mediante fórmulas

```
1   all x (x=x).                          % Reflexividad
2   all x y (x=y -> y=x).                 % Simetría
3   all x y z (x=y & y=z -> x=z).         % Transitividad
4
5   % Axiomas de sustitución de la función f/3
6   all x1 x2 x3 y (x1=y -> f(x1,x2,x3) = f(y,x2,x3)).
7   all x1 x2 x3 y (x2=y -> f(x1,x2,x3) = f(x1,y,x3)).
8   all x1 x2 x3 y (x3=y -> f(x1,x2,x3) = f(x1,x2,y)).
9
10  % Axiomas de sustitución de la relación P/3
11  all x1 x2 x3 y (x1=y & P(x1,x2,x3) -> P(y.x2.x3)).
12  all x1 x2 x3 y (x2=y & P(x1,x2,x3) -> P(x1.y.x3)).
13  all x1 x2 x3 y (x3=y & P(x1,x2,x3) -> P(x1.x2.y)).
```

Razonamiento con igualdad

- Formalización de los axiomas de igualdad mediante cláusulas

```
1   x = x.                          % Reflexividad
2   x != y | y = x.                 % Simetría
3   x != y | y! = z | x=z.          % Transitividad
4
5   % Axiomas de sustitución de la función f/3
6   x1 != y | f(x1,x2,x3) = f(y,x2,x3)).
7   x2 != y | f(x1,x2,x3) = f(x1,y,x3)).
8   x3 != y | f(x1,x2,x3) = f(x1,x2,y)).
9
10  % Axiomas de sustitución de la relación P/3
11  x1 != y | -P(x1,x2,x3) | P(y.x2.x3)).
12  x2 != y | -P(x1,x2,x3) | P(x1.y.x3)).
13  x3 != y | -P(x1,x2,x3) | P(x1.x2.y)).
```

Paramodulación

- Regla de paramodulación
  - Izquierda

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

  - Derecha

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

Paramodulación

- Ejemplos de paramodulación

”ej-5.in”

```
1  list(sos).
2   P(f(x,b),x) | Q(x).
3   f(a,x)=x | R(x).
4  end_of_list.
5
6  set(para_into).
7  set(para_from).
```

Paramodulación

- Ejemplos de paramodulación: 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).
```

Paramodulación

- Ejemplos de paramodulación: explicación

```
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	$\sigma = \{x2/x1\}$
Paramodulante	σ(x2=x1 | R(x2) | R(x1))
	$\Longrightarrow$ x1=x1 | R(x1)
	$\Longrightarrow$ x=x | R(x)

Paramodulación

- Ejemplos de paramodulación: explicación

```
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	$\sigma = \{x2/a,\ x1/b\}$
Paramodulante	σ(P(x1,x2) | Q(x2) | R(x1)) $\Longrightarrow$ P(b) | Q(a) | R(b)

Paramodulación

- Demostrar que si Francisco es igual a Curro y a Paco, entonces Curro y Paco son iguales

”ej-1d.in”

```
1   list(usable).
2    x=x.                    % Reflexividad
3    francisco = curro.
4    francisco = paco.
5   end_of_list.
6
7   list(sos).
8    paco != curro.
9   end_of_list.
10
11  set(para_into).
```

Paramodulación

- Explicación de la cláusula

```
[para_into,4.1.1,3.1.2] francisco!=curro.

   Into 4.1.1     'paco'!=curro
   From 3.1.2     francisco='paco'
```

- Estadísticas

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

Paramodulación

- 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

”ej-2c.in”

```
1   list(usable).
2    x=x.                              % Reflexividad
3    opuesta(derecha) = izquierda.
4    opuesta(izquierda) = derecha.
5   end_of_list.
6
7   list(sos).
8    opuesta(opuesta(derecha)) != derecha.
9   end_of_list.
10
11  set(para_into).
12  set(para_from).
```

Paramodulación

- Explicación de la cláusula

```
[para_into,4.1.1.1,2.1.1] opuesta(izquierda)!=derecha.

   Into 4.1.1     opuesta(‘opuesta(derecha)’)!=derecha
   From 2.1.1     ‘opuesta(derecha)’=izquierda
```

- Estadísticas

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

Paramodulación

- Demostrar que si los padres son mayores que los hijos y Luis es el padre de Juan, entonces Luis es mayor que Juan.

”ej-3d.in”

```
1   list(usable).
2    x=x.                    % Reflexividad
3    Mayor(Padre(x),x).
4    Padre(Juan)=Luis.
5   end_of_list.
6
7   list(sos).
8    -Mayor(Luis,Juan).
9   end_of_list.
10
11  set(para_into).
```

Paramodulación

- Explicación de la cláusula

```
[para_into,4.1.1,3.1.2] -Mayor(Padre(Juan),Juan).

   Into 4.1.1     -Mayor('Luis',Juan)
   From 4.1.2     Padre(Juan)='Luis'
```

- Estadísticas

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

Paramodulación

- Demostrar que si Juan está casado y es el tío de Pepe, entonces el hermano del padre de Juan está casado

”ej-6b.in”

```
1   list(usable).
2    x=x.
3    casado(juan).
4    hermano(padre(x)) = tio(x).
5    tio(pepe)=juan.
6   end_of_list.
7
8   list(sos).
9    -casado(hermano(padre(pepe))).
10  end_of_list.
11
12  set(para_into).
```

Demodulación

- Regla de demodulación

$$\frac{C[t] \quad t_1 = t_2}{\sigma(C[t_2])} \qquad \sigma = umg(t, t_1)$$

Demodulación

- Demostrar que si Francisco es igual a Curro y a Paco, entonces Curro y Paco son iguales

”ej-1e.in”

```
1  list(usable).
2   x=x.                    % Reflexividad
3  end_of_list.
4
5  list(demodulators).
6   curro = francisco.
7   paco = francisco.
8  end_of_list.
9
10 list(sos).
11  paco != curro.
12 end_of_list.
13
14 set(process_input).
```

Demodulación

- 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

”ej-2d.in”

```
1  list(usable).
2   x=x.
3  end_of_list.
4
5  list(demodulators).
6   opuesta(derecha) = izquierda.
7   opuesta(izquierda) = derecha.
8  end_of_list.
9
10 list(sos).
11  opuesta(opuesta(derecha)) != derecha.
12 end_of_list.
13
14 set(process_input).
```

Demodulación

- Demostrar que si Juan está casado y es el tío de Pepe, entonces el hermano del padre de Juan está casado

”ej-6a.in”

```
1  list(usable).
2   x=x.
3   casado(juan).
4  end_of_list.
5
6  list(demodulators).
7   hermano(padre(x)) = tio(x).
8   tio(pepe)=juan.
9  end_of_list.
10
11 list(sos).
12  -casado(hermano(padre(pepe))).
13 end_of_list.
14
15 set(process_input).
```

Operadores

- 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

- Formalización en OTTER

”ej-7a.in”

```
1  op(400, xfy, *).
2  op(300, yf, ^).
3
4  list(usable).
5   x = x.                                  % Reflexividad
6   e * x = x.                              % Ax. 1
7   x * e = x.                              % Ax. 2
8   x^ * x = e.                             % Ax. 3
9   x * x^ = e.                             % Ax. 4
10  (x * y) * z = x * (y * z).              % Ax. 5
11 end_of_list.
12
13 list(sos).
14  x * x = e.
15  a * b != b * a.
16 end_of_list.
17
18 set(para_into).
19 set(para_from).
```

Operadores

- Prueba obtenida

```
  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.
```

Operadores

- Cláusula 19 **[para_from,7.1.2,3.1.1.2] x*y*y=x.**

  From 7.1.2 x1*x1=‘e’
  Into 3.1.1.2 x2*‘e’=x2
  $\Longrightarrow$ x2*(x1*x1)=x2 {x2/x, x1/y}
  $\Longrightarrow$ x*(y*y)=x

- Cláusula 20 **[para_from,7.1.2,2.1.1.1] (x*x)*y=y.**

  From 7.1.2 x1*x1=‘e’
  Into 2.1.1.1 ‘e’*x2=x2
  $\Longrightarrow$ (x1*x1)*x2=x2 {x1/x, x2/y}
  $\Longrightarrow$ (x*x)*y=y

Operadores

- Cláusula 31 **[para_into,19.1.1,6.1.2] (x*y)*y=x.**

  Into 19.1.1 **‘x2*(y2*y2)’=x2**
  From 6.1.2 **(x1*y1)*z1=‘x1*(y1*z1)’**
  Unificador $\sigma = \{x2/x1,\ y2/y1,\ z1/y1\}$
  Paramodulante $\sigma((x1*y1)*z1=x2)$
  $\Longrightarrow$ **(x1*y1)*y1=x1** $\{x1/x,\ y1/y\}$
  $\Longrightarrow$ **(x*y)*y=x**

- Cláusula 167 **[para_into,20.1.1,6.1.1] x*x*y=y.**

  Into 20.1.1 **‘(x2*x2)*y2’=y2**
  From 6.1.1 **‘(x1*y1)*z1’=x1*(y1*z1)**
  Unificador $\sigma = \{x1/x2,\ y1/x2,\ z1/y2\}$
  Paramodulante $\sigma(x1*(y1*z1)=y2)$
  $\Longrightarrow$ **x2*(x2*y2)=y2** $\{x2/x,\ y2/y\}$
  $\Longrightarrow$ **x*(x*y)=y**

Operadores

- Cláusula 170 **[para_from,20.1.1,6.1.1] x=y*y*x.**

  From 20.1.1 ‘(x1*x1)*y1’=y1
  Into 6.1.1 ‘(x2*y2)*z2’=x2*(y2*z2)
  Unificador $\sigma = \{x2/x1,\ y2/x1,\ z2/y1\}$
  Paramodulante $\sigma(y1=x2*(y2*z2))$
  $\Longrightarrow$ y1=x1*(x1*y1) $\{y1/x,\ x1/y\}$
  $\Longrightarrow$ x=y*(y*x)

- Cláusula 496 **[para_into,167.1.1.2,31.1.1] (x*y)*x=y.**

  Into 167.1.1.2 x2*‘(x2*y2)’=y2
  From 31.1.1 ‘(x1*y1)*y1’=x1
  Unificador $\sigma = \{x2/x1*y1,\ y2/y1\}$
  Paramodulante $\sigma(x2*x1=y2)$
  $\Longrightarrow$ (x1*y1)*x1=y1 $\{x1/x,\ y1/y\}$
  $\Longrightarrow$ (x*y)*x=y

Operadores

- Cláusula 755 **[para_into,496.1.1.1,170.1.2] x*y=y*x.**

Into 496.1.1.1 ‘(x2*y2)’*x2=y2
From 170.1.2 x1=‘y1*(y1*x1)’
Unificador $\sigma = \{x2/y1,\ y2/y1*x1\}$
Paramodulante $\sigma((x1*x2)=y2)$
$\Longrightarrow$ x1*y1=y1*x1 $\{x1/x,\ y1/y\}$
$\Longrightarrow$ x*y=y*x

- Estadísticas

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

Operadores

- Mejora con demoduladores

”ej-7b.in”

```
1  include('ej-7a.in').
2
3  list(demodulators).
4   e * x = x.                          % Ax. 1
5   x * e = x.                          % Ax. 2
6   x^ * x = e.                         % Ax. 3
7   x * x^ = e.                         % Ax. 4
8   (x * y) * z = x * (y * z).          % Ax. 5
9  end_of_list.
```

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

Operadores

- Mejora con demoduladores dinámicos

”ej-7c.in”

```
1  include('ej-7b.in').
2
3  set(dynamic_demod).
```

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

Operadores

- Modo autónomo

”ej-7d.in”

```
1   set(auto2).
2
3   op(400, xfy, *).
4   op(300, yf, ^).
5
6   list(usable).
7    e * x = x.                          % Ax. 1
8    x * e = x.                          % Ax. 2
9    x^ * x = e.                         % Ax. 3
10   x * x^ = e.                         % Ax. 4
11   (x * y) * z = x * (y * z).          % Ax. 5
12   x = x.                              % Ax. 6
13   x * x = e.
14   a * b != b * a.
15  end_of_list.
```

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. y Lee, R.C.T. *Symbolic logic and mechanical theorem proving*. (Academic Press, 1973)
  - Cap. 8 “The equality relation”
- Genesereth, M.R. y Nilsson, N.J. *Logical foundations of Artificial Intelligence* (Morgan Kaufmann, 1987)
  - Cap. 9 “Relational resolution”

Bibliografía

- 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)