Demostración automática de teoremas (2004–05)

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"

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

set(binary_res).
```

Axiomas de igualdad

- Formalización en OTTER con axiomas de igualdad

"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).
```

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"

```
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).
```

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”

```
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).
```

Axiomas de sustitución funcionales

- Formalización en OTTER con un axioma de sustitución

”ej-2b1.in”

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

list(sos).
 x!=y | opuesta(x)=opuesta(y).
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"

```
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"

```
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 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"

```
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"

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

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

```
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)).
```

Razonamiento con igualdad

- Formalización de los axiomas de igualdad 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

- 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"

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

set(para_into).
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	$\sigma($**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          σ = {x2/a, x1/b}
   Paramodulante       σ(P(x1,x2) | Q(x2) | R(x1))
                          ⟹ 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"

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

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

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"

```
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).
```

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"

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

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

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"

```
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

- Regla de demodulación

$$\frac{\begin{array}{c} \boldsymbol{C[t]} \\ \boldsymbol{t_1 = t_2} \end{array}}{\boldsymbol{\sigma(C[t_2])}} \qquad\qquad \boldsymbol{\sigma = umg(t, t_1)}$$

Demodulación

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

"ej-1e.in"

```
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).
```

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"

```
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).
```

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”

```
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(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

$$
\begin{array}{l}
(\forall x)[e.x = x] \\
(\forall x)[x.e = x] \\
(\forall x)[x.x^{-1} = e] \\
(\forall x)[x^{-1}.x = e] \\
(\forall x)(\forall y)(\forall z)[(x.y).z = x.(y.z)]
\end{array}
$$

  - Hipótesis

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

  - Conclusión

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

Operadores

- Formalización en OTTER

"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 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	$\sigma = \{\texttt{x2}/\texttt{x1},\ \texttt{y2}/\texttt{y1},\ \texttt{z1}/\texttt{y1}\}$
Paramodulante	$\sigma(\texttt{(x1*y1)*z1=x2})$
	$\Longrightarrow$ `(x1*y1)*y1=x1` $\{\texttt{x1}/\texttt{x},\ \texttt{y1}/\texttt{y}\}$
	$\Longrightarrow$ `(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	$\sigma = \{\texttt{x1}/\texttt{x2},\ \texttt{y1}/\texttt{x2},\ \texttt{z1}/\texttt{y2}\}$
Paramodulante	$\sigma(\texttt{x1*(y1*z1)=y2})$
	$\Longrightarrow$ `x2*(x2*y2)=y2` $\{\texttt{x2}/\texttt{x},\ \texttt{y2}/\texttt{y}\}$
	$\Longrightarrow$ `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        σ = {x2/x1, y2/x1, z2/y1}
Paramodulante     σ(y1=x2*(y2*z2))
                  ⟹ y1=x1*(x1*y1) {y1/x, x1/y}
                  ⟹ 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          σ = {x2/x1*y1, y2/y1}
Paramodulante       σ(x2*x1=y2)
                    ⟹ (x1*y1)*x1=y1 {x1/x, y1/y}
                    ⟹ (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"

```
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.
```

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

Operadores

- Mejora con demoduladores dinámicos

"ej-7c.in"

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

set(dynamic_demod).
```

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

Operadores

- Modo autónomo

"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.
```

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)