

Demostración automática de teoremas (2004–05)

Tema 4: Resolución de primer orden

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

Dpto. Ciencias de la Computación e Inteligencia Artificial

Universidad de Sevilla

Sintaxis de la lógica de primer orden

- Extensión de la lógica proposicional
- Alfabeto de primer orden:
 - Funciones: $f(x_1, \dots, x_n)$
 - Predicados: $P(x_1, \dots, x_m)$
 - Cuantificadores:
 - \forall (universal),
 - \exists (existencial).
- Fórmulas de primer orden
 - $\forall xF, \exists xF$.
 - Ámbito de una variable cuantificada.

- Sintaxis en OTTER/MACE

Usual	\forall	\exists
OTTER/MACE	all	exists

Resolución binaria

- Demostrar que el conjunto $\{\neg P(x) \vee Q(x), P(a), \neg Q(z)\}$ es inconsistente.
- Sintaxis en OTTER/MACE
 - Variables: Cadenas alfanuméricas que empiezan por u, v, w, x, y, z
 - Constantes relaciones: Cadenas alfanuméricas que no empiezan por u, v, w, x, y, z
 - Cláusula
 - Cuantificación universal implícita
 - Listas de cláusulas: `list`

Resolución binaria

- Inferencia
 - Regla de resolución binaria

$$\frac{L_1 \mid \dots \mid L_i \mid A \mid L_{i+1} \mid \dots \mid L_n}{M_1 \mid \dots \mid M_j \mid -B \mid M_{j+1} \mid \dots \mid M_k}$$
$$\sigma(L_1 \mid \dots \mid L_i \mid L_{i+1} \mid \dots \mid L_n \mid$$
$$M_1 \mid \dots \mid M_j \mid M_{j+1} \mid \dots \mid M_k)$$

$$\sigma = \text{umg}(A, B)$$
$$\sigma(A) = \sigma(B)$$

- Unificación
- Separación de variables

Resolución binaria

- Formalización en OTTER

"ej-1.in"

```
1 list(sos).  
2 -P(x) | Q(x).  
3 P(a).  
4 -Q(z).  
5 end_of_list.  
6  
7 set(binary_res).
```

Resolución binaria

- Búsqueda y prueba en OTTER

"ej-1.out"

```
===== start of search =====
given clause #1: (wt=2) 2 [] P(a).
given clause #2: (wt=2) 3 [] -Q(z).
given clause #3: (wt=4) 1 [] -P(x)|Q(x).
** KEPT (pick-wt=2): 4 [binary,1.1,2.1] Q(a).
----> UNIT CONFLICT: 5 [binary,4.1,3.1] $F.

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

Skolemización

- Demostrar que el conjunto $\{(\forall x)[P(x) \rightarrow Q(x)], P(a), \neg(\exists z)Q(z)\}$ es inconsistente
- Transformación a clausulas

1. Eliminación de equivalencias e implicaciones

2. Interiorización de la negación:

$$\neg(\forall x A) \equiv \exists x \neg A \quad (9)$$

$$\neg(\exists x A) \equiv \forall x \neg A \quad (10)$$

3. Renombrar variables cuantificadas

4. Skolemización: Eliminación de cuantificaciones existenciales

$$\begin{aligned} \forall x_1 \dots \forall x_n \exists y F(x_1, \dots, x_n, y) &\equiv \\ &\equiv \forall x_1 \dots \forall x_n F(x_1, \dots, x_n, f_y(x_1, \dots, x_n)) \end{aligned}$$

5. Agrupar cuantificaciones al principio

6. Interiorizar las disyunciones

Skolemización

- Formalización en OTTER

"ej-2.in"

```
1 formula_list(sos).  
2   all x (P(x) -> Q(x)).  
3   P(a).  
4   -(exists z Q(z)).  
5 end_of_list.  
6  
7 set(binary_res).
```


Resolución binaria

- Búsqueda y prueba en OTTER

"ej-2.out"

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

Consecuencia lógica

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

Argumentaciones

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

Lucero

Orejon

CABALLO(x)

CONEJO(x)

GALGO(x)

PERRO(x)

MAS_RAPIDO(x, y)

Lucero

Orejón

x es un caballo

x es un conejo

x es un galgo

x es un perro

x es más rápido que y

Argumentaciones

- Formalización en OTTER

"ej-3a1.in"

```
1 formula_list(sos).
2 % Los caballos son más rápidos que los perros.
3 all x y (CABALLO(x) & PERRO(y) -> MAS_RAPIDO(x,y)).
4 % Algunos galgos son más rápidos que los conejos
5 exists x (GALGO(x) &
6           (all y (CONEJO(y) -> MAS_RAPIDO(x,y)))).
7 % Lucero es un caballo
8 CABALLO(Lucero).
9 % Orejón es un conejo.
10 CONEJO(Orejon).
11 % Lucero no es más rápido que Orejón
12 -MAS_RAPIDO(Lucero,Orejon).
13 end_of_list.
14
15 set(binary_res).
```

Argumentaciones

- Clasificación y Skolemización en OTTER

_____ "ej-3a1.out" _____

```
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.
```

- ¿Cuál es la respuesta de OTTER ?

Argumentaciones

- Búsqueda de modelos con MACE: `mace2 -n2 -p -m1 < ej-3a1.in`
- Modelo encontrado

```
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
```

Argumentaciones

- Información implícita: Los galgos son perros.
- Formalización en OTTER

```
_____ "ej-3a2.in" _____  
1  include('ej-3a1.in').  
2  
3  formula_list(sos).  
4    % Los galgos son perros  
5    all x (GALGO(x) -> PERRO(x)).  
6  end_of_list.  
7  
8  set(binary_res).
```

- ¿Qué dice ahora OTTER ?

Argumentaciones

- Búsqueda de modelos con MACE: `mace2 -n2 -p -m1 < ej-3a2.in`
- Modelo encontrado

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

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

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


Argumentaciones

- Información implícita: Si X es más rápido que Y e Y es más rápido que Z, entonces X es más rápido que Z.
- Formalización en OTTER

```
_____ "ej-3a3.in" _____  
1  include('ej-3a2.in').  
2  
3  formula_list(sos).  
4    % "más rápido que" es transitiva.  
5    all x y z (MAS_RAPIDO(x,y) & MAS_RAPIDO(y,z)  
6              -> MAS_RAPIDO(x,z)).  
7  end_of_list.  
8  
9  set(binary_res).
```

Argumentaciones

- Prueba obtenida con OTTER :

```
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

- Formalización en OTTER

"ej-3b.in"

```
1 formula_list(usable).
2   all x y (CABALLO(x) & PERRO(y) -> MAS_RAPIDO(x,y)).
3   exists x (GALGO(x) &
4             (all y (CONEJO(y) -> MAS_RAPIDO(x,y)))).
5   CABALLO(Lucero).
6   CONEJO(Orejon).
7   all x (GALGO(x) -> PERRO(x)).
8   all x y z (MAS_RAPIDO(x,y) & MAS_RAPIDO(y,z)
9             -> MAS_RAPIDO(x,z)).
10 end_of_list.
11 formula_list(sos).
12   -MAS_RAPIDO(Lucero,Orejon).
13 end_of_list.
14
15 set(binary_res).
```

La estrategia del conjunto soporte

- Prueba obtenida con OTTER :

```
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

$$L_1 \mid \dots \mid L_n$$

$$M_1$$

$$\vdots$$

$$M_{i-1}$$

$$M_{i+1}$$

$$\vdots$$

$$M_n$$

$$\sigma(L_i)$$

donde $\sigma = umg(L_j, \overline{M_j})$ $j \in \{1, \dots, i-1, i+1, \dots, n\}$

Resolución UR

- Formalización en OTTER

"ej-3c.in"

```
1 formula_list(sos).
2   all x y (CABALLO(x) & PERRO(y) -> MAS_RAPIDO(x,y)).
3   exists x (GALGO(x) &
4             (all y (CONEJO(y) -> MAS_RAPIDO(x,y))))).
5   CABALLO(Lucero).
6   CONEJO(Orejon).
7   -MAS_RAPIDO(Lucero,Orejon).
8   all x (GALGO(x) -> PERRO(x)).
9   all x y z (MAS_RAPIDO(x,y) & MAS_RAPIDO(y,z)
10             -> MAS_RAPIDO(x,z)).
11 end_of_list.
12
13 set(ur_res).
```

Resolución UR

- Prueba obtenida con OTTER :

```
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

$$\neg A_1 \mid \dots \mid \neg A_n \mid B_1 \mid \dots \mid B_m$$

$$M_1$$

$$\vdots$$

$$M_n$$

$$\sigma(B_1) \mid \dots \mid \sigma(B_m)$$

donde $A_1, \dots, A_n, B_1, \dots, B_m$ son átomos y $\sigma = umg(A_i, M_i), i \in \{1, \dots, n\}$

Hiper-resolución

- Formalización en OTTER

"ej-3d.in"

```
1 formula_list(sos).
2   all x y (CABALLO(x) & PERRO(y) -> MAS_RAPIDO(x,y)).
3   exists x (GALGO(x) &
4             (all y (CONEJO(y) -> MAS_RAPIDO(x,y))))).
5   CABALLO(Lucero).
6   CONEJO(Orejon).
7   -MAS_RAPIDO(Lucero,Orejon).
8   all x (GALGO(x) -> PERRO(x)).
9   all x y z (MAS_RAPIDO(x,y) & MAS_RAPIDO(y,z)
10             -> MAS_RAPIDO(x,z)).
11 end_of_list.
12
13 set(hyper_res).
```

Hiper-resolución

- Prueba obtenida con OTTER :

```
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

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

Obtención de respuestas

- Dado $\{\forall x(P(x) \rightarrow Q(x)), P(a)\}$ determinar un z tal que $Q(z)$ sea consecuencia del conjunto.
- Formalización en OTTER :

```
_____ "ej-4a.in" _____  
1  formula_list(sos).  
2    all x (P(x) -> Q(x)).  
3    P(a).  
4    all z (Q(z) -> $ANS(z)).  
5  end_of_list.  
6  
7  set(binary_res).
```

Obtención de varias respuestas

- Dado $\{\forall x(P(x) \rightarrow Q(x)), P(a) \wedge P(b)\}$ determinar un z tal que $Q(z)$ sea consecuencia del conjunto.
- Formalización en OTTER

```
----- "ej-4b1.in" -----  
1  formula_list(sos).  
2    all x (P(x) -> Q(x)).  
3    P(a) & P(b).  
4    all z (Q(z) -> $ANS(z)).  
5  end_of_list.  
6  
7  set(binary_res).  
8  assign(max_proofs,2).
```

- Número máximo de pruebas encontradas: `assign(max_proofs, N)`

Obtención de todas las pruebas posibles

- Formalización en OTTER

```
_____ "ej-4b2.in" _____  
1  formula_list(sos).  
2    all x (P(x) -> Q(x)).  
3    P(a) & P(b).  
4    all z (Q(z) -> $ANS(z)).  
5  end_of_list.  
6  
7  set(binary_res).  
8  assign(max_proofs,-1).
```

- Obtención de todas las pruebas posibles: `assign(max_proofs,-1)`

Obtención de todas las respuestas posibles

- Formalización en OTTER

```
_____ "ej-4b3.in" _____  
1  formula_list(sos).  
2    all x (P(x) -> Q(x)).  
3    P(a) & P(b).  
4  end_of_list.  
5  
6  formula_list(passive).  
7    all z (Q(z) -> $ANS(z)).  
8  end_of_list.  
9  
10 set(binary_res).  
11 assign(max_proofs,-1).
```

- Fórmulas a utilizar en último lugar: `formula_list(passive)`

Obtención de respuestas

- 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.
- Formalización en OTTER : hechos

"ej-5.in"

```
1 list(usable).
2 Hombre(Juan). Hombre(Jorge). Hombre(Victor).
3 Mujer(Maria). Mujer(Agata). Mujer(Carla).
4 Rico(Maria). Rico(Jorge). Rico(Victor).
5 Ama(Maria, Juan). Ama(Juan, Maria).
6 Ama(Victor, Maria). Ama(Jorge, Victor).
7 Ama(Victor, Jorge).
```


Obtención de respuestas

- Formalización en OTTER : relaciones

"ej-5.in"

```
1      % Los hombres y las mujeres son de sexos distintos:
2      -Mujer(x) | -Hombre(y) | Distinto_sexo(x,y).
3      -Mujer(x) | -Hombre(y) | Distinto_sexo(y,x).
4      % Dos personas de distintos sexo pueden casarse si
5      % se aman o una es rica y ama a la otra:
6      -Distinto_sexo(x,y) | -Ama(x,y)
7      | -Ama(y,x) | Pueden_casarse(x,y).
8      -Distinto_sexo(x,y) | -Ama(x,y)
9      | -Rico(x) | Pueden_casarse(x,y).
10     end_of_list.
11
12     list(sos).
13     -Pueden_casarse(x,y) | $ANS(x,y).
14     end_of_list.
15
16     set(ur_res).
17     assign(max_proofs, -1).
```

Obtención de respuestas

- Primera respuesta:

```
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

- Segunda respuesta:

```
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

- Si sabemos que toda persona es hijo de su padre y que los hijos de los hijos son nietos entonces, si Luis es nieto de X, ¿quién es X?
- Formalización en OTTER :

"ej-6.in"

```
1 formula_list(usable).
2   % Toda persona es hijo de su padre:
3   all x exists y Hijo(x,y).
4   % Los hijos de los hijos son nietos:
5   all x y z (Hijo(x,y) & Hijo(y,z) -> Nieto(x,z)).
6 end_of_list.
7
8 formula_list(sos).
9   % Si Luis es nieto de x, quién es x:
10  all x (Nieto(Luis,x) -> $ANS(x)).
11 end_of_list.
12
13 set(ur_res).
```

Obtención de respuestas

- Si Luna es una persona y todas las personas están solteras o casadas, ¿cuál es el estado civil de Luna?
- Formalización en OTTER :

```
_____ "ej-7.in" _____  
1  formula_list(usable).  
2  Persona(Luna).  
3  all x (Persona(x) -> Estado(x,Soltero)  
4          | Estado(x,Casado)).  
5  end_of_list.  
6  
7  formula_list(sos).  
8  all x (Estado(Luna, x) -> $ANS(x)).  
9  end_of_list.  
10  
11 set(ur_res).
```

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)
- Genesereth, M.R. *Computational Logic*
 - 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)