Tema DA–3: Procedimiento general de demostración por resolución proposicional en OTTER

José A. Alonso Jiménez
Miguel A. Gutiérrez Naranjo

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

Razonamiento proposicional con OTTER

- Base de conocimiento
  - Base de reglas:
    * R1: Si el animal tiene pelos es mamífero.
    * R2: Si el animal da leche es mamífero.
    * R3: Si el animal es un mamífero y tiene pezuñas es ungulado.
    * R4: Si el animal es un mamífero y rumia es ungulado.
    * R5: Si el animal es un ungulado y tiene cuello largo es una jirafa.
    * R6: Si el animal es un ungulado y tiene rayas negras es una cebra.
  - Base de hechos:
    * H1: El animal tiene pelos.
    * H2: El animal tiene pezuñas.
    * H3: El animal tiene rayas negras.
  - Consecuencia
    * El animal es una cebra.

Razonamiento proposicional con OTTER

- **Representación en OTTER (`animales.in`)**

```
formula_list(sos).
tiene_pelos | da_leche -> es_mamifero.                  % Reglas 1 y 2
es_mamifero & (tiene_pezuñas | rumia) -> es_ungulado.   % Reglas 3 y 4
es_ungulado & tiene_cuello_largo -> es_jirafa.          % Regla 5
es_ungulado & tiene_rayas_negras -> es_cebra.           % Regla 6

tiene_pelos & tiene_pezuñas & tiene_rayas_negras.       % Hechos 1, 2 y 3

-es_cebra.                                              % -Conclusión
end_of_list.

set(binary_res).          % Regla de inferencia: Resolución binaria
set(very_verbose).        % Da mucha información sobre generación de cláusulas.
```

- **Procesamiento con OTTER**

```
otter <animales.in >animales.out
```

Razonamiento proposicional con OTTER

- **Transformación a cláusulas**

```
-------> sos clausifies to:
list(sos).
1 [] -tiene_pelos | es_mamifero.
2 [] -da_leche | es_mamifero.
3 [] -es_mamifero | -tiene_pezuñas | es_ungulado.
4 [] -es_mamifero | -rumia | es_ungulado.
5 [] -es_ungulado | -tiene_cuello_largo | es_jirafa.
6 [] -es_ungulado | -tiene_rayas_negras | es_cebra.
7 [] tiene_pelos.
8 [] tiene_pezuñas.
9 [] tiene_rayas_negras.
10 [] -es_cebra.
end_of_list.
```

- **Cláusulas**
  - **Definiciones:**
    - **Una cláusula es una disyunción de literales.**
    - **Un literal es un átomo o la negación de un átomo.**

Razonamiento proposicional con OTTER

- **Procedimiento de transformación a cláusulas:**

  1. **Eliminar las equivalencias usando la relación**
     $$A \leftrightarrow B \equiv (A \rightarrow B) \wedge (B \rightarrow A) \quad (1)$$

  2. **Eliminar las implicaciones usando la equivalencia**
     $$A \rightarrow B \equiv \neg A \vee B \quad (2)$$

  3. **Interiorizar las negaciones usando las equivalencias**
     $$\neg(A \wedge B) \equiv \neg A \vee \neg B \quad (3)$$
     $$\neg(A \vee B) \equiv \neg A \wedge \neg B \quad (4)$$
     $$\neg\neg A \equiv A \quad (5)$$

  4. **Interiorizar las disyunciones usando la propiedad distributiva de la disyunción sobre la conjunción**
     $$A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C) \quad (6)$$
     $$(A \wedge B) \vee C \equiv (A \vee C) \wedge (B \vee C) \quad (7)$$

Razonamiento proposicional con OTTER

- **Reglas dependientes**

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

- **Búsqueda de la prueba**

```
========== start of search ==========

given clause #1: (wt=1) 7 [] tiene_pelos.

given clause #2: (wt=1) 8 [] tiene_pezugnas.

given clause #3: (wt=1) 9 [] tiene_rayas_negras.

given clause #4: (wt=1) 10 [] -es_cebra.
```

Razonamiento proposicional con OTTER

```
given clause #5: (wt=2) 1 [] -tiene_pelos|es_mamifero.

  0 [binary,1.1,7.1] es_mamifero.
** KEPT (pick-wt=1): 11 [binary,1.1,7.1] es_mamifero.
11 back subsumes 2.
11 back subsumes 1.

given clause #6: (wt=1) 11 [binary,1.1,7.1] es_mamifero.

given clause #7: (wt=3) 3 [] -es_mamifero| -tiene_pezugnas|es_ungulado.

  0 [binary,3.1,11.1] -tiene_pezugnas|es_ungulado.
** KEPT (pick-wt=1): 12 [binary,3.1,11.1,unit_del,8] es_ungulado.

  0 [binary,3.2,8.1] -es_mamifero|es_ungulado.
  Subsumed by 12.
12 back subsumes 4.
12 back subsumes 3.
```

Razonamiento proposicional con OTTER

```
given clause #8: (wt=1) 12 [binary,3.1,11.1,unit_del,8] es_ungulado.

given clause #9: (wt=3) 5 [] -es_ungulado| -tiene_cuello_largo|es_jirafa.

  0 [binary,5.1,12.1] -tiene_cuello_largo|es_jirafa.
** KEPT (pick-wt=2): 13 [binary,5.1,12.1] -tiene_cuello_largo|es_jirafa.
13 back subsumes 5.

given clause #10: (wt=2) 13 [binary,5.1,12.1] -tiene_cuello_largo|es_jirafa.

given clause #11: (wt=3) 6 [] -es_ungulado| -tiene_rayas_negras|es_cebra.

  0 [binary,6.1,12.1] -tiene_rayas_negras|es_cebra.
** KEPT (pick-wt=0): 14 [binary,6.1,12.1,unit_del,9,10] $F.

-----> EMPTY CLAUSE at   0.00 sec ----> 14 [binary,6.1,12.1,unit_del,9,10] $F.
```

Razonamiento proposicional con OTTER

- **Demostración**

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

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

1 [] -tiene_pelos|es_mamifero.
3 [] -es_mamifero| -tiene_pezugnas|es_ungulado.
6 [] -es_ungulado| -tiene_rayas_negras|es_cebra.
7 [] tiene_pelos.
8 [] tiene_pezugnas.
9 [] tiene_rayas_negras.
10 [] -es_cebra.
11 [binary,1.1,7.1] es_mamifero.
12 [binary,3.1,11.1,unit_del,8] es_ungulado.
14 [binary,6.1,12.1,unit_del,9,10] $F.

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


Search stopped by max_proofs option.
```

Razonamiento proposicional con OTTER

- **Estadísticas**

```
------------- statistics -------------
clauses given                  11
clauses generated               5
  binary_res generated          5
  factors generated             0
clauses forward subsumed        1
  (subsumed by sos)             1
unit deletions                  4
clauses kept                    3
empty clauses                   1
clauses back subsumed           5
usable size                     8

----------- times (seconds) -----------
user CPU time          0.00          (0 hr, 0 min, 0 sec)
system CPU time        0.01          (0 hr, 0 min, 0 sec)
```

Razonamiento proposicional con OTTER

- **Cláusulas usadas**

```
 1 [] -tiene_pelos | es_mamifero.
 2 [] -da_leche | es_mamifero.
 3 [] -es_mamifero | -tiene_pezugnas |es_ungulado.
 4 [] -es_mamifero | -rumia | es_ungulado.
 5 [] -es_ungulado | -tiene_cuello_largo | es_jirafa.
 6 [] -es_ungulado | -tiene_rayas_negras |es_cebra.
 7 [] tiene_pelos.
 8 [] tiene_pezugnas.
 9 [] tiene_rayas_negras.
10 [] -es_cebra.
11 [binary,1.1,7.1] es_mamifero.
12 [binary,3.1,11.1,unit_del,8] es_ungulado.
13 [binary,5.1,12.1] -tiene_cuello_largo|es_jirafa.
14 [binary,6.1,12.1,unit_del,9,10] $F.
```

Razonamiento proposicional con OTTER

- **Evolución del soporte y usable**

N	Soporte	Usable
0	1 2 3 4 5 6 7 8 9 10	
1	1 2 3 4 5 6 8 9 10	7
2	1 2 3 4 5 6 9 10	8 7
3	1 2 3 4 5 6 10	9 8 7
4	1 2 3 4 5 6	10 9 8 7
5	3 4 5 6 11	10 9 8 7
6	3 4 5 6	11 10 9 8 7
7	5 6 12	11 10 9 8 7
8	5 6	12 11 10 9 8 7
9	6 13	12 11 10 9 8 7
10	6	13 12 11 10 9 8 7
11	14	6 13 12 11 10 9 8 7

Razonamiento proposicional con OTTER

- **Procedimiento de búsqueda de pruebas**

```
Mientras el soporte es no vacío y no se ha encontrado una refutación
 1. Seleccionar como cláusula actual la cláusula menos pesada del soporte;
 2. Mover la cláusula actual del soporte a usable;
 3. Calcular las resolventes de la cláusula actual con las cláusulas usables.
 4. Procesar cada una de las resolventes calculadas anteriormente.
 5. Añadir al soporte cada una de las cláusulas procesadas que supere
    el procesamiento.
```

Razonamiento proposicional con OTTER

El procesamiento de cada una de resolventes consta de los siguientes pasos (los indicados con * son opcionales):

```
 *  1. Escribir la resolvente.
 *  2. Aplicar a la resolvente eliminación unitaria (i.e. elimina los literales
       de la resolvente tales que hay una cláusula unitaria  complementaria en
       usable o en soporte).
    3. Descartar la resolvente y salir si la resolvente es una tautología.
    4. Descartar la resolvente y salir si la resolvente es subsumida por alguna
       cláusula de usable o del soporte (subsunción hacia adelante).
    5. Añadir la resolvente al soporte.
 *  6. Escribir la resolvente retenida.
    7. Si la resolvente tiene 0 literales, se ha encontrado una refutación.
    8. Si la resolvente tiene 1 literal, entonces buscar su complementaria
       (refutación) en usable y soporte.
 *  9. Escribir la demostración si se ha encontrado una refutación.
 ------------
 * 10. Descartar cada cláusula de usable o del soporte subsumida por la
       resolvente (subsunción hacia atrás).
```

El paso 10 no se da hasta que los pasos 1-9 se han aplicado a todas las resolventes.

Razonamiento proposicional con OTTER (II)

- **Redistribución del soporte** (`animales_2.in`)

```
formula_list(usable).
tiene_pelos | da_leche -> es_mamifero.                  % Reglas 1 y 2
es_mamifero & (tiene_pezuñas | rumia) -> es_ungulado.   % Reglas 3 y 4
es_ungulado & tiene_cuello_largo -> es_jirafa.          % Regla 5
es_ungulado & tiene_rayas_negras -> es_cebra.           % Regla 6

tiene_pelos & tiene_pezuñas & tiene_rayas_negras.       % Hechos 1, 2 y 3
end_of_list.

formula_list(sos).
-es_cebra.                                              % -Conclusión
end_of_list.

set(binary_res).        % Regla de inferencia: resolución binaria.
set(very_verbose).      % Da mucha información sobre generación de cláusulas.
```

- **Procesamiento con OTTER**

```
otter <animales_2.in >animales_2.out
```

Razonamiento proposicional con OTTER (II)

- **Transformación a cláusulas**

```
-------> usable clausifies to:
list(usable).
1 [] -tiene_pelos|es_mamifero.
2 [] -da_leche|es_mamifero.
3 [] -es_mamifero| -tiene_pezugnas|es_ungulado.
4 [] -es_mamifero| -rumia|es_ungulado.
5 [] -es_ungulado| -tiene_cuello_largo|es_jirafa.
6 [] -es_ungulado| -tiene_rayas_negras|es_cebra.
7 [] tiene_pelos.
8 [] tiene_pezugnas.
9 [] tiene_rayas_negras.
end_of_list.

-------> sos clausifies to:
list(sos).
10 [] -es_cebra.
end_of_list.
```

Razonamiento proposicional con OTTER (II)

- **Búsqueda de la prueba**

```
given clause #1: (wt=1) 10 [] -es_cebra.

  0 [binary,10.1,6.3] -es_ungulado| -tiene_rayas_negras.
** KEPT (pick-wt=1): 11 [binary,10.1,6.3,unit_del,9] -es_ungulado.
11 back subsumes 6.
11 back subsumes 5.

given clause #2: (wt=1) 11 [binary,10.1,6.3,unit_del,9] -es_ungulado.

  0 [binary,11.1,4.3] -es_mamifero| -rumia.
** KEPT (pick-wt=2): 12 [binary,11.1,4.3] -es_mamifero| -rumia.

  0 [binary,11.1,3.3] -es_mamifero| -tiene_pezugnas.
** KEPT (pick-wt=1): 13 [binary,11.1,3.3,unit_del,8] -es_mamifero.
12 back subsumes 4.
13 back subsumes 12.
13 back subsumes 3.
```

Razonamiento proposicional con OTTER (II)

```
given clause #3: (wt=1) 13 [binary,11.1,3.3,unit_del,8] -es_mamifero.

  0 [binary,13.1,2.2] -da_leche.
** KEPT (pick-wt=1): 14 [binary,13.1,2.2] -da_leche.

  0 [binary,13.1,1.2] -tiene_pelos.
** KEPT (pick-wt=1): 15 [binary,13.1,1.2] -tiene_pelos.

----> UNIT CONFLICT at   0.00 sec ----> 16 [binary,15.1,7.1] $F.
```

Razonamiento proposicional con OTTER (II)

- **Demostración**

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

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

1 [] -tiene_pelos|es_mamifero.
3 [] -es_mamifero| -tiene_pezugnas|es_ungulado.
6 [] -es_ungulado| -tiene_rayas_negras|es_cebra.
7 [] tiene_pelos.
8 [] tiene_pezugnas.
9 [] tiene_rayas_negras.
10 [] -es_cebra.
11 [binary,10.1,6.3,unit_del,9] -es_ungulado.
13 [binary,11.1,3.3,unit_del,8] -es_mamifero.
15 [binary,13.1,1.2] -tiene_pelos.
16 [binary,15.1,7.1] $F.
```

Razonamiento proposicional con OTTER (II)

- **Estadísticas**

```
------------- statistics ------------
clauses given                  3
clauses generated              5
  binary_res generated         5
  factors generated            0
unit deletions                 2
clauses kept                   5
empty clauses                  1
clauses back subsumed          5
usable size                    8
sos size                       2

----------- times (seconds) -----------
user CPU time          0.00          (0 hr, 0 min, 0 sec)
system CPU time        0.00          (0 hr, 0 min, 0 sec)
```

Eliminación de tautologías y factorización

- **Entrada**

```
list(sos).
-p | -q.
p | q.
p | -q.
-p | q.
end_of_list.

set(binary_res).
set(very_verbose).
```

- **Búsqueda de la prueba**

```
given clause #1: (wt=2) 1 [] -p| -q.

given clause #2: (wt=2) 2 [] p|q.

  0 [binary,2.1,1.1] q| -q.

  0 [binary,2.2,1.2] p| -p.
```

Eliminación de tautologías y factorización

```
given clause #3: (wt=2) 3 [] p| -q.

  0 [binary,3.1,1.1] -q| -q.
** KEPT (pick-wt=1): 5 [binary,3.1,1.1,factor_simp] -q.

  0 [binary,3.2,2.2] p|p.
** KEPT (pick-wt=1): 6 [binary,3.2,2.2,factor_simp] p.
5 back subsumes 3.
5 back subsumes 1.
6 back subsumes 2.

given clause #4: (wt=1) 5 [binary,3.1,1.1,factor_simp] -q.

given clause #5: (wt=1) 6 [binary,3.2,2.2,factor_simp] p.

given clause #6: (wt=2) 4 [] -p|q.

  0 [binary,4.1,6.1] q.
** KEPT (pick-wt=1): 7 [binary,4.1,6.1] q.

----> UNIT CONFLICT at   0.00 sec ----> 8 [binary,7.1,5.1] $F.
```

Eliminación de tautologías y factorización

- **Prueba**

```
---------------- PROOF ----------------

1 [] -p| -q.
2 [] p|q.
3 [] p| -q.
4 [] -p|q.
5 [binary,3.1,1.1,factor_simp] -q.
6 [binary,3.2,2.2,factor_simp] p.
7 [binary,4.1,6.1] q.
8 [binary,7.1,5.1] $F.

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