Razonamiento automático

Tema 3: Resolución proposicional en OTTER

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

Dpto. 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:

    C El animal es una cebra

Razonamiento proposicional con OTTER

- Formalización en OTTER

"animales_1.in"

```
formula_list(sos).
 tiene_pelos | da_leche -> es_mamifero.
 es_mamifero & (tiene_pezuñas | rumia) -> es_ungulado.
 es_ungulado & tiene_cuello_largo -> es_jirafa.
 es_ungulado & tiene_rayas_negras -> es_cebra.
 tiene_pelos & tiene_pezuñas & tiene_rayas_negras.
 -es_cebra.
end_of_list.

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

- Ejecución en OTTER :

```
otter < animales_1.in > animales_1.out
```

Preprocesamiento

"animales_1.out"

```
-------> sos clausifies to:

list(sos).
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.
end_of_list.
```

Transformación a cláusulas

- Una cláusula es una disyunción de literales.
- Un literal es un átomo o la negación de un átomo.
- 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 las propiedades distributivas
     $$A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C) \quad (6)$$
     $$(A \wedge B) \vee C \equiv (A \vee C) \wedge (B \vee C) \quad (7)$$

Características activas

- Reglas dependientes:

”animales_1.out”

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

- Resolución binaria: `set(binary_res).`
- Factorización: `set(factor).`
- Eliminación unitaria: `set(unit_deletion).`

Búsqueda de la prueba

"animales_1.out"

```
========== 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.
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 #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 0.01 sec --> 14 [binary,6.1,12.1,unit_del,9,10] $F.
```

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.
  *6. Descartar cada cláusula de usable o del soporte subsumida por alguna resolvente (subsunción hacia atrás).

Procesamiento de las resolventes

*1 Escribir la resolvente.

*2 Aplicar a la resolvente eliminación unitaria (elimina los literales de la resolvente tales que hay una cláusula unitaria complementaria en usable o en soporte.

3 Si la resolvente es una tautología se descarta.

4 Si la resolvente es subsumida por alguna cláusula de usable o del soporte (subsunción hacia delante) se descarta.

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.

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.
```

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

Demostración

"animales_1.out"

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

Estadísticas

"animales_1.out"

```
-------------- 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)
```

Redistribución del soporte

"animales_2.in"

```
formula_list(usable).
 tiene_pelos | da_leche -> es_mamifero.
 es_mamifero & (tiene_pezuñas | rumia) -> es_ungulado.
 es_ungulado & tiene_cuello_largo -> es_jirafa.
 es_ungulado & tiene_rayas_negras -> es_cebra.

 tiene_pelos & tiene_pezuñas & tiene_rayas_negras.
end_of_list.

formula_list(sos).
 -es_cebra.
end_of_list.

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

Nueva demostración

"animales_2.out"

```
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.
```

Eliminación de tautologías y factorización

"tautologias_factorizacion.in"

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

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

Búsqueda de la prueba

"tautologias_factorizacion.out"

```
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.
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.
```

Demostración

"tautologias_factorizacion.out"

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

Bibliografía

- McCune, W. OTTER *3.3 Reference Manual* (2003)