

OTTER: Introducción

Francisco J. Martín Mateos
José A. Alonso Jiménez

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

¿Qué es OTTER?

OTTER es un demostrador automático para la Lógica de Primer Orden con Igualdad.

- El lenguaje es el de la lógica de primer orden con una sintaxis similar a PROLOG.
- Es un demostrador por refutación: busca una contradicción a partir de un conjunto de fórmulas.
- El razonamiento se realiza principalmente por resolución (resolución binaria, resolución UR, hiperresolución), con reglas de paramodulación y demodulación para tratar la igualdad.
- Las pruebas son secuencias de fórmulas en las que se indica la forma en que cada una de ellas se obtiene a partir de las anteriores. Estas secuencias terminan en la fórmula falsa.

¿Qué es OTTER?

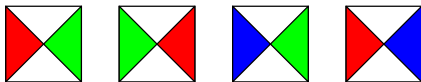
- Desarrollado por William McCune en la sección de Matemáticas y Ciencias de la Computación del “Argonne National Laboratory” (Chicago - EEUU).
- Aplicaciones: Algebra abstracta y lógica formal.
- Éxito más importante: Solución al problema de Robbins.

Instalación y ejecución

- Windows: `otter33-Win32.zip`
- Fuentes: `otter-3.3f.tgz`
- Incluir la formalización en un fichero de entrada.
- Ejecución en línea de comandos:
`otter < entrada.in > salida.out`

Ejemplos de problema

- Rompecabezas: Un puzzle tiene cuatro tipos distintos de fichas. Estas fichas son las siguientes:



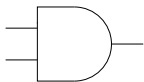
Encontrar todas las maneras posibles de colocar en línea cuatro de estas fichas, no necesariamente distintas, de forma que los colores de los lados adyacentes coincidan.

Ejemplos de problema

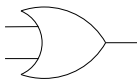
- Matemáticas: Sea $\mathbf{G}$ un grupo y $\mathbf{e}$ su elemento neutro.
Demostrar que si para todo $\mathbf{x}$ de $\mathbf{G}$, $\mathbf{x} \times \mathbf{x} = \mathbf{e}$, entonces $\mathbf{G}$ es conmutativo.
- Axiomas de grupo:
 - $\forall \mathbf{x} (\mathbf{e} \times \mathbf{x} = \mathbf{x})$
 - $\forall \mathbf{x} (\mathbf{x} \times \mathbf{e} = \mathbf{x})$
 - $\forall \mathbf{x} (\mathbf{x}^{-1} \times \mathbf{x} = \mathbf{e})$
 - $\forall \mathbf{x} (\mathbf{x} \times \mathbf{x}^{-1} = \mathbf{e})$
 - $\forall \mathbf{x} \forall \mathbf{y} \forall \mathbf{z} ((\mathbf{x} \times \mathbf{y}) \times \mathbf{z} = \mathbf{x} \times (\mathbf{y} \times \mathbf{z}))$
- Hipótesis:
 - $\forall \mathbf{x} (\mathbf{x} \times \mathbf{x} = \mathbf{e})$
- Conclusión:
 - $\forall \mathbf{x} \forall \mathbf{y} (\mathbf{x} \times \mathbf{y} = \mathbf{y} \times \mathbf{x})$

Ejemplos de problema

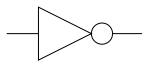
- Diseño de circuitos lógicos: Diseñar un circuito lógico que actúe como una componente OR construido exclusivamente con componentes AND y NOT.



i_1	i_2	o_1
0	0	0
0	1	0
1	0	0
1	1	1



i_1	i_2	o_1
0	0	0
0	1	1
1	0	1
1	1	1



i_1	o_1
0	1
1	0

Alfabeto de la lógica de primer orden

- Variables.
- Funciones: $\mathbf{f(x_1, \dots, x_n)}$
- Predicados: $\mathbf{P(x_1, \dots, x_m)}$
- Conectivas lógicas:
 - $\neg$ (negación),
 - $\wedge$ (conjunción),
 - $\vee$ (disyunción),
 - $\rightarrow$ (implicación),
 - $\leftrightarrow$ (equivalencia).
- Cuantificadores:
 - $\forall$ (universal),
 - $\exists$ (existencial).
- Símbolos auxiliares: “(” y “)”.

Elementos del lenguaje de OTTER

- Variables, funciones y predicados:
Cadenas alfanuméricas, \$ y _.
- No se consideran variables libres, todas las variables deben aparecer cuantificadas.
- Conectivas lógicas
 - (negación),
 - & (conjunción),
 - | (disyunción),
 - > (implicación),
 - <-> (equivalencia).
- Cuantificadores
 - all** (universal),
 - exists** (existencial).
- Símbolos auxiliares: (y).

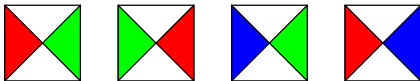
Fórmulas de la lógica de primer orden

- Cualquier predicado es una fórmula:
 $P(x, y)$ (fórmulas atómicas).
- Si F y G son fórmulas: $(\neg F)$ (negación de F),
 $(F \wedge G)$ (conjunción de F y G),
 $(F \vee G)$ (disyunción de F y G),
 $(F \rightarrow G)$ (F implica G),
 $(F \leftrightarrow G)$ (equivalencia entre F y G).
- Si F es una fórmula y x una variable
 $(\forall x F)$ (para todo x se tiene F),
 $(\exists x F)$ (existe un x para el que se tiene F).
- Eliminación de paréntesis
 - Paréntesis externos.
 - Precedencia: $\forall, \exists, \neg, \wedge, \vee, \rightarrow, \leftrightarrow$.
 - Asociatividad: $\wedge$ y $\vee$ asocian por la derecha.
 - Agrupación de cuantificadores consecutivos del mismo tipo.

- $(\forall x (\neg P(x)))$
 $\Rightarrow_{\text{Simp}} \forall x \neg P(x)$
 $\Rightarrow_{\text{Otter}} \text{all } x \neg P(x)$
- $(\forall x (\exists y P(x, y)))$
 $\Rightarrow_{\text{Simp}} \forall x \exists y P(x, y)$
 $\Rightarrow_{\text{Otter}} \text{all } x \text{ exists } y P(x, y)$
- $(\forall x (\forall y (\forall z P(x, y, z))))$
 $\Rightarrow_{\text{Simp}} \forall x y z P(x, y, z)$
 $\Rightarrow_{\text{Otter}} \text{all } x y z P(x, y, z)$
- $(\forall x ((P(x) \wedge (Q(x) \wedge R(x))) \rightarrow S(x)))$
 $\Rightarrow_{\text{Simp}} \forall x (P(x) \wedge Q(x) \wedge R(x) \rightarrow S(x))$
 $\Rightarrow_{\text{Otter}} \text{all } x (P(x) \ \& \ Q(x) \ \& \ R(x) \rightarrow S(x))$

Ejemplos: Rompecabezas

- Rompecabezas: Un puzzle tiene cuatro tipos distintos de fichas. Estas fichas son las siguientes:



Encontrar todas las maneras posibles de colocar en línea cuatro de estas fichas, no necesariamente distintas, de forma que los colores de los lados adyacentes coincidan.

Ejemplos: Rompecabezas

- Lenguaje del problema
 - `ficha(x,y,z)`: La ficha `x` tiene a la izquierda el color `y`, y a la derecha el color `z`.
 - `posicion(x1,x2,x3,x4)`: Las fichas `x1`, `x2`, `x3`, `x4` forman una posible combinación para resolver el problema.
- Formalización en OTTER.
 - Las fórmulas se agrupan en listas. `formula_list(sos)` y `end_of_list` delimitan el conjunto soporte.
 - Las opciones de búsqueda se indican mediante banderas y parámetros. `set(auto2)` activa el modo automático.

- Formalización en OTTER
 - Si las fichas **x1**, **x2**, **x3**, **x4** se pueden colocar en línea de forma que los colores de los lados adyacentes coinciden, entonces forman una combinación válida para resolver el problema.

```
all x1 x2 x3 x4 y1 y2 y3 y4 y5
  ((ficha(x1,y1,y2) & ficha(x2,y2,y3) &
    ficha(x3,y3,y4) & ficha(x4,y4,y5)) ->
    posicion(x1,x2,x3,x4)) .
```

Ejemplos: Rompecabezas

- Formalización en OTTER
 - Las fichas que se pueden colocar se identifican con los números 1, 2, 3 y 4.

```
ficha(1,rojo,verde) .  
ficha(2,verde,rojo) .  
ficha(3,azul,verde) .  
ficha(4,rojo,azul) .
```

- ¿Se puede obtener alguna combinación válida para resolver el problema?

```
all x1 x2 x3 x4 -posicion(x1,x2,x3,x4) .
```

- Formalización en OTTER

fichas.in

```
set(auto2).
```

```
formula_list(sos).
```

```
all x1 x2 x3 x4 y1 y2 y3 y4 y5  
  ((ficha(x1,y1,y2) & ficha(x2,y2,y3) &  
    ficha(x3,y3,y4) & ficha(x4,y4,y5)) ->  
    posicion(x1,x2,x3,x4)).
```

```
ficha(1,rojo,verde).  
ficha(2,verde,rojo).  
ficha(3,azul,verde).  
ficha(4,rojo,azul).
```

```
all x1 x2 x3 x4 -posicion(x1,x2,x3,x4).  
end_of_list.
```


- Ejecución: `otter < fichas.in > fichas.out`

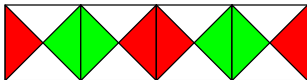
Prueba obtenida

```
1 [] -ficha(x,y,z) | -ficha(u,z,v) | -ficha(w,v,v6) |  
   -ficha(v7,v6,v8) | posicion(x,u,w,v7) .  
2 [] ficha(1,rojo,verde) .  
3 [] ficha(2,verde,rojo) .  
6 [] -posicion(x,y,z,u) .  
7 [hyper,3,1,2,3,2] posicion(1,2,1,2) .  
8 [binary,7.1,6.1] $F.
```

Ejemplos: Rompecabezas

- Interpretación de la solución

7 [hyper,3,1,2,3,2] posicion(1,2,1,2)



- Otras cuestiones

- ¿Se pueden encontrar todas las soluciones?
- ¿Y si las fichas no se pueden repetir?
- ¿Y si las fichas se pueden girar?

Ejemplos: Matemáticas

- Matemáticas: Sea $\mathbf{G}$ un grupo y $\mathbf{e}$ su elemento neutro.
Demostrar que si para todo $\mathbf{x}$ de $\mathbf{G}$, $\mathbf{x} \times \mathbf{x} = \mathbf{e}$, entonces $\mathbf{G}$ es conmutativo.
- Axiomas de grupo:
 - $\forall \mathbf{x} (\mathbf{e} \times \mathbf{x} = \mathbf{x})$
 - $\forall \mathbf{x} (\mathbf{x} \times \mathbf{e} = \mathbf{x})$
 - $\forall \mathbf{x} (\mathbf{x}^{-1} \times \mathbf{x} = \mathbf{e})$
 - $\forall \mathbf{x} (\mathbf{x} \times \mathbf{x}^{-1} = \mathbf{e})$
 - $\forall \mathbf{x} \forall \mathbf{y} \forall \mathbf{z} ((\mathbf{x} \times \mathbf{y}) \times \mathbf{z} = \mathbf{x} \times (\mathbf{y} \times \mathbf{z}))$
- Hipótesis:
 - $\forall \mathbf{x} (\mathbf{x} \times \mathbf{x} = \mathbf{e})$
- Conclusión:
 - $\forall \mathbf{x} \forall \mathbf{y} (\mathbf{x} \times \mathbf{y} = \mathbf{y} \times \mathbf{x})$

- Lenguaje del problema (usando funciones):
 - `mult(x,y)`: Devuelve el resultado de la operación $\mathbf{x} \times \mathbf{y}$.
 - `inv(x)`: Devuelve el resultado de la operación $\mathbf{x}^{-1}$.
- Lenguaje del problema (usando operadores):
 - `op(400,xfy,*)`: El símbolo $*$ representa el operador binario $\times$ del grupo $\mathbf{G}$. La expresión $\mathbf{x} \times \mathbf{y}$ se representa como $\mathbf{x} * \mathbf{y}$.
 - `op(300,yf,^)`: El símbolo $^$ representa el operador inversa $^{-1}$ del grupo $\mathbf{G}$. La expresión $\mathbf{x}^{-1}$ se representa como $\mathbf{x}^{\wedge}$.

- Formalización en OTTER (usando funciones)

- Axiomas de grupo:

```
all x (mult(e,x) = x) .  
all x (mult(x,e) = x) .  
all x (mult(inv(x),x) = e) .  
all x (mult(x,inv(x)) = e) .  
all x y z (mult(mult(x,y),z) = mult(x,mult(y,z))) .  
all x (x = x) .
```

- Hipótesis:

```
all x (mult(x,x) = e) .
```

- Conclusión:

```
mult(b,a) != mult(a,b) .
```

- Formalización en OTTER (usando operadores)

- Axiomas de grupo:

```
all x (e * x = x) .  
all x (x * e = x) .  
all x (x^ * x = e) .  
all x (x * x^ = e) .  
all x y z ((x * y) * z = x * (y * z)) .  
all x (x = x) .
```

- Hipótesis:

```
all x (x * x = e) .
```

- Conclusión:

```
b * a != a * b.
```

- Formalización en OTTER

grupos.in

```
op(400, xfy, *).
op(300, yf, ^).

set(auto2).

formula_list(sos).
  all x (e * x = x).
  all x (x * e = x).
  all x (x^ * x = e).
  all x (x * x^ = e).
  all x y z ((x * y) * z = x * (y * z)).
  all x (x = x).

  all x (x * x = e).

  b * a != a * b.
end_of_list.
```

- Ejecución: `otter < grupos.in > grupos.out`

Prueba obtenida

```
2,1 [] e*x=x.
4,3 [] x*e=x.
9 [] (x*y)*z=x*y*z.
12 [] x*x=e.
14 [] b*a!=a*b.
17 [para_into,9.1.1.1,12.1.1,demod,2,flip.1] x*x*y=y.
23 [para_into,9.1.1,12.1.1,flip.1] x*y*x*y=e.
33 [para_from,23.1.1,17.1.1.2,demod,4,flip.1] x*y*x=y.
42 [para_from,33.1.1,17.1.1.2] x*y=y*x.
43 [binary,42.1,14.1] $F.
```


- Interpretación de la solución

17 [para_into, 9.1.1.1, 12.1.1, demod, 2, flip.1] $x * x * y = y$

9 $(X \times Y) \times Z = X \times (Y \times Z)$

$\Downarrow$

$$\left. \begin{array}{l} (x \times x) \times y = x \times (x \times y) \\ x \times x = e \end{array} \right\} \Rightarrow$$

$\Uparrow$

12 $X \times X = e$

$$\Rightarrow \left. \begin{array}{l} e \times y = x \times (x \times y) \\ e \times y = y \end{array} \right\} \Rightarrow y = x \times (x \times y)$$

$\Uparrow$

2 $e \times Y = Y$

- Interpretación de la solución

23 [para_into, 9.1.1, 12.1.1, flip.1] $x*y*x*y=e$

9 $(X \times Y) \times Z = X \times (Y \times Z)$

$\Downarrow$

$$\left. \begin{array}{l} (x \times y) \times (x \times y) = x \times (y \times (x \times y)) \\ (x \times y) \times (x \times y) = e \end{array} \right\} \Rightarrow$$

$\Uparrow$

12

$$X \times X = e$$

$$\Rightarrow e = x \times (y \times (x \times y))$$

- Interpretación de la solución

33 [para_from,23.1.1,17.1.1.2,demod,4,flip.1] $x*y*x=y$

17
$$Y = X \times (X \times Y)$$

$\Downarrow$

$$\left. \begin{aligned} x \times (y \times x) &= y \times (y \times (x \times (y \times x))) \\ e &= y \times (x \times (y \times x)) \end{aligned} \right\} \Rightarrow$$

$\Uparrow$

23
$$e = X \times (Y \times (X \times Y))$$

$$\Rightarrow \left. \begin{aligned} x \times (y \times x) &= y \times e \\ y \times e &= y \end{aligned} \right\} \Rightarrow x \times (y \times x) = y$$

$\Uparrow$

2
$$Y \times e = Y$$

- Interpretación de la solución

42 [para_from, 33.1.1, 17.1.1.2] $x*y=y*x$

17 $Y = X \times (X \times Y)$

$\Downarrow$

$$\left. \begin{array}{l} y \times x = x \times (x \times (y \times x)) \\ x \times (y \times x) = y \end{array} \right\} \Rightarrow$$

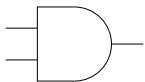
$\Uparrow$

33 $X \times (Y \times X) = Y$

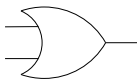
$$\Rightarrow y \times x = x \times y$$

Ejemplos: Diseño de circuitos lógicos

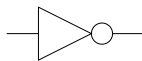
- Diseño de circuitos lógicos: Diseñar un circuito lógico que actúe como una componente OR construido exclusivamente con componentes AND y NOT.



i_1	i_2	o_1
0	0	0
0	1	0
1	0	0
1	1	1



i_1	i_2	o_1
0	0	0
0	1	1
1	0	1
1	1	1



i_1	o_1
0	1
1	0

Ejemplos: Diseño de circuitos lógicos

- Lenguaje del problema
 - **salida**(**x1**, **x2**, **x3**, **x4**): Se puede construir un circuito lógico cuyo comportamiento se puede describir con la siguiente tabla:

i₁	i₂	o₁
0	0	x1
0	1	x2
1	0	x3
1	1	x4

- Formalización en OTTER
 - Si se pueden construir los circuitos representados por `salida(x1,x2,x3,x4)` y `salida(y1,y2,y3,y4)`, entonces usando una componente AND se puede construir el circuito representado por `salida(and(x1,y1), and(x2,y2), and(x3,y3), and(x4,y4))`.

```
all x1 x2 x3 x4 y1 y2 y3 y4
  (salida(x1,x2,x3,x4) & salida(y1,y2,y3,y4) ->
   salida(and(x1,y1), and(x2,y2),
           and(x3,y3), and(x4,y4))) .
```

- Comportamiento de la conectiva lógica `and`:

```
all x (and(x,0) = 0) .
all x (and(x,1) = x) .
```

- Formalización en OTTER
 - Si se puede construir el circuito representados por `salida(x1,x2,x3,x4)`, entonces usando una componente NOT se puede construir el circuito representado por `salida(not(x1),not(x2),not(x3),not(x4))`.

```
all x1 x2 x3 x4
(salida(x1,x2,x3,x4) ->
 salida(not(x1),not(x2),not(x3),not(x4))) .
```

- Comportamiento de la conectiva lógica `not`:

```
not(0) = 1.
not(1) = 0.
```


- Formalización en OTTER

- Se pueden construir los circuitos representados por las entradas:
`salida(0,0,1,1)` y `salida(0,1,0,1)`.

```
salida(0,0,1,1).  
salida(0,1,0,1).
```

- ¿Se puede construir el circuito representado por `salida(0,1,1,1)`?

```
-salida(0,1,1,1).
```

- Formalización en OTTER

circuitos.in

```
set(auto2).

formula_list(sos).
  all x1 x2 x3 x4 y1 y2 y3 y4
    (salida(x1,x2,x3,x4) & salida(y1,y2,y3,y4) ->
      salida(and(x1,y1),and(x2,y2),
        and(x3,y3),and(x4,y4))) .

  all x1 x2 x3 x4
    (salida(x1,x2,x3,x4) ->
      salida(not(x1),not(x2),not(x3),not(x4))) .

salida(0,0,1,1).
salida(0,1,0,1).
-salida(0,1,1,1).

all x (and(x,0) = 0) .
all x (and(x,1) = x) .
not(0) = 1.
not(1) = 0.
end_of_list.
```

- Ejecución: `otter < circuitos.in > circuitos.out`

Prueba obtenida

```
1 [] -salida(x,y,z,u) | -salida(v,w,v6,v7) |
    salida(and(x,v),and(y,w),and(z,v6),and(u,v7)).
2 [] -salida(x,y,z,u) |
    salida(not(x),not(y),not(z),not(u)).
3 [] salida(0,0,1,1).
4 [] salida(0,1,0,1).
5 [] -salida(0,1,1,1).
7,6 [] and(x,0)=0.
9,8 [] and(x,1)=x.
11,10 [] not(0)=1.
13,12 [] not(1)=0.
16 [hyper,3,2,demod,11,11,13,13] salida(1,1,0,0).
17 [hyper,4,2,demod,11,13,11,13] salida(1,0,1,0).
22 [hyper,17,1,16,demod,9,7,9,7] salida(1,0,0,0).
27 [hyper,22,2,demod,13,11,11,11] salida(0,1,1,1).
28 [binary,27.1,5.1] $F.
```

Ejemplos: Diseño de circuitos lógicos

- Interpretación de la solución

3 [] salida(0,0,1,1)

4 [] salida(0,1,0,1)

3

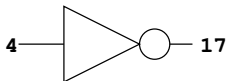
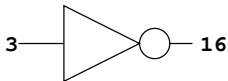
4

Ejemplos: Diseño de circuitos lógicos

- Interpretación de la solución

16 [hyper, 3, 2, demod, 11, 11, 13, 13] salida(1, 1, 0, 0)

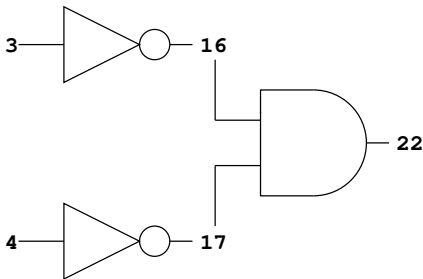
17 [hyper, 4, 2, demod, 11, 13, 11, 13] salida(1, 0, 1, 0)



Ejemplos: Diseño de circuitos lógicos

- Interpretación de la solución

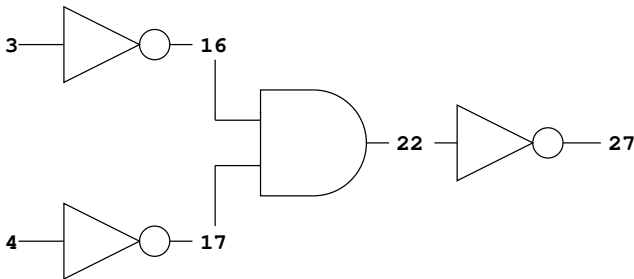
22 [hyper,17,1,16,demod,9,7,9,7] salida(1,0,0,0)



Ejemplos: Diseño de circuitos lógicos

- Interpretación de la solución

27 [hyper, 22, 2, demod, 13, 11, 11, 11] salida (0, 1, 1, 1)



- G. Kolata. *Computer Math Proof Shows Reasoning Power* (The New York Times, 10 de Diciembre de 1996)
<http://www.nytimes.com/library/cyber/week/1210math.html>
- McCune, W. *Otter 3.3 Reference Manual* (Argonne National Laboratory, 2003)