

OTTER: Predicados evaluables

Francisco J. Martín Mateos

José A. Alonso Jiménez

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

El problema de las jarras

- Se tienen dos jarras, una de 4 litros de capacidad y otra de 3. Ninguna de ellas tiene marcas de medición. Se tiene una bomba que permite llenar las jarras de agua. Averiguar cómo se puede lograr tener exactamente 2 litros de agua en la jarra de 4 litros de capacidad
- Lenguaje del problema
 - **jarras** (**x**, **y**), haciendo operaciones con las jarras, podemos llegar a tener **x** en la jarra de 4 litros e **y** en la jarra de 3 litros

El problema de las jarras

jarras-a.in

```
set(binary_res) .  
  
formula_list(sos) .  
  % Operadores  
  % Vaciar jarras  
  % vaciar A  
  all x y (jarras(x,y) -> jarras(0,y)) .  
  % vaciar B  
  all x y (jarras(x,y) -> jarras(x,0)) .  
  % Llenar jarras  
  % llenar A  
  all x y (jarras(x,y) -> jarras(4,y)) .  
  % llenar B  
  all x y (jarras(x,y) -> jarras(x,3)) .  
end_of_list.
```

El problema de las jarras

jarras-a.in

```
formula_list(sos).  
  % Vaciar una jarra en otra  
  % vaciar A en B  
  all x y (x+y > 3 & jarras(x,y) -> jarras(x+y-3,3)).  
  % llenar B con A  
  all x y (x+y <= 3 & jarras(x,y) -> jarras(0,x+y)).  
  % vaciar B en A  
  all x y (x+y > 4 & jarras(x,y) -> jarras(4,x+y-4)).  
  % llenar A con B  
  all x y (x+y <= 4 & jarras(x,y) -> jarras(x+y,0)).  
  
  jarras(0,0).  
  all x -jarras(2,x).  
end_of_list.
```

Proceso de búsqueda

```
===== start of search =====
given clause #1: (wt=3) 9 [] jarras(0,0) .
given clause #2: (wt=3) 10 [] -jarras(2,x) .
given clause #3: (wt=6) 1 [] -jarras(x,y) | jarras(0,y) .
given clause #4: (wt=6) 2 [] -jarras(x,y) | jarras(x,0) .
given clause #5: (wt=6) 3 [] -jarras(x,y) | jarras(4,y) .
** KEPT (...): 11 [binary,3.1,9.1] jarras(4,0) .
given clause #6: (wt=3) 11 [binary,3.1,9.1] jarras(4,0) .
given clause #7: (wt=6) 4 [] -jarras(x,y) | jarras(x,3) .
** KEPT (...): 12 [binary,4.1,11.1] jarras(4,3) .
** KEPT (...): 13 [binary,4.1,9.1] jarras(0,3) .
given clause #8: (wt=3) 12 [binary,4.1,11.1] jarras(4,3) .
given clause #9: (wt=3) 13 [binary,4.1,9.1] jarras(0,3) .
given clause #10:
    (wt=13) 6 [] -(x+y<=3) | -jarras(x,y) | jarras(0,x+y) .
** KEPT (...): 14 [binary,6.2,13.1] -(0+3<=3) | jarras(0,0+3) .
** KEPT (...): 15 [binary,6.2,12.1] -(4+3<=3) | jarras(0,4+3) .
...
```

El problema de las jarras

- Evaluación de las operaciones aritméticas
 - Mediante demoduladores

```
list(demodulators) .  
  0+3 = 3.  
  4+3 = 7.  
  ...  
end_of_list.
```

- Mediante la asociación de funciones del problema con operadores evaluables de OTTER

- Aritmética entera
 - Suma: **\$SUM**
 - Resta: **\$DIFF**
 - Producto: **\$PROD**
- Relaciones de orden
 - Igual: **\$EQ**
 - Distinto: **\$NE**
 - Menor: **\$LT**
 - Menor o igual: **\$LE**
 - Mayor: **\$GT**
 - Mayor o igual: **\$GE**
- Operadores booleanos
 - Verdadero: **\$T**
 - Falso: **\$F**
 - Negación: **\$NOT**
 - Conjunción: **\$AND**
 - Disyunción: **\$OR**
- Relaciones de identidad
 - Igual: **\$ID**
 - Distinto: **\$LNE**

El problema de las jarras

- Con operadores evaluables
- `make_evaluable(_, _)`: Asociación de funciones con operadores evaluables

jarras-b.in

```
include("jarras-a.in") .  
  
make_evaluable(_+_ , $SUM(_,_)) .  
make_evaluable(_-_ , $DIFF(_,_)) .  
make_evaluable(_<=_ , $LE(_,_)) .  
make_evaluable(_>_ , $GT(_,_)) .
```

El problema de las jarras

Prueba obtenida

```
2 [] -jarras(x,y) | jarras(x,0).
3 [] -jarras(x,y) | jarras(4,y).
5 [] -(x+y>3) | -jarras(x,y) | jarras(x+y-3,3).
6 [] -(x+y<=3) | -jarras(x,y) | jarras(0,x+y).
9 [] jarras(0,0).
10 [] -jarras(2,x).
11 [binary,3.1,9.1] jarras(4,0).
17 [binary,6.2,2.2] -(x+0<=3) | jarras(0,x+0) | -jarras(x,y).
43 [binary,17.2,3.1] -(x+0<=3) | -jarras(x,y) | jarras(4,x+0).
132 [binary,5.2,11.1,demod,propositional] jarras(1,3).
149 [binary,132.1,43.2,demod,propositional] jarras(4,1).
152 [binary,149.1,5.2,demod,propositional] jarras(2,3).
153 [binary,152.1,10.1] $F.
```

El problema de las jarras

- Formalización usando mínimos

jarras-c.in

```
set (binary_res) .

make_evaluable(_+_, $SUM(_,_)) .
make_evaluable(_-_, $DIFF(_,_)) .

formula_list(sos) .
  all x y (jarras(x,y) -> jarras(0,y)) .
  all x y (jarras(x,y) -> jarras(x,0)) .
  all x y (jarras(x,y) -> jarras(4,y)) .
  all x y (jarras(x,y) -> jarras(x,3)) .
  all x y
    (jarras(x,y) -> jarras(x+y-min(y+x,3),min(y+x,3))) .
  all x y
    (jarras(x,y) -> jarras(min(x+y,4),x+y-min(x+y,4))) .

  jarras(0,0) .
  all x -jarras(2,x) .
end_of_list.
```

El problema de las jarras

- Definición evaluable de `min`
- Demodulación condicional

jarras-d.in

```
include("jarras-c.in").  
  
make_evaluable(_<=_, $LE(_,_)).  
make_evaluable(_>_, $GT(_,_)).  
  
list(demodulators).  
  x<=y -> min(x,y) = x.  
  x>y -> min(x,y) = y.  
end_of_list.
```

El problema de las jarras

- Definición evaluable de `min`
- Condicional evaluable `$IF(_,_,_)`

jarras-e.in

```
include("jarras-c.in").  
  
make_evaluable(_<=_, $LE(_,_)).  
make_evaluable(_>_, $GT(_,_)).  
  
list(demodulators).  
  min(x,y) = $IF(x<=y,x,y).  
end_of_list.
```

Operaciones con listas

- Secuencia de elementos separados por comas y delimitados por corchetes
 - Lista con dos elementos: `[1,2]`
 - Lista vacía: `[]`
- El constructor básico de listas añade un elemento al principio de otra lista.
 - Añadir un elemento a una lista: `[1|x]`

- Pertenencia de un elemento a una lista con demoduladores condicionales

pertenece-a.in

```
list(demodulators).  
pertenece(x, []) = $F.  
$ID(x,y) -> pertenece(x, [y|w]) = $T.  
$NE(x,y) -> pertenece(x, [y|w]) = pertenece(x,w).  
end_of_list.
```

- Pertenencia de un elemento a una lista con condicional evaluable

pertenece-b.in

```
list(demodulators).  
pertenece(x, []) = $F.  
pertenece(x, [y|w]) = $IF($ID(x,y), $T, pertenece(x,w)).  
end_of_list.
```

- Concatenación de listas

concatenacion.in

```
list(demodulators) .  
  append([], w)      = w.  
  append([x|w1], w2) = [x|append(w1, w2)] .  
end_of_list.
```

- Inversión de listas

inversa.in

```
list(demodulators) .  
  inversa(w)          = inversa_aux(w, []).  
  inversa_aux([], w)  = w.  
  inversa_aux([x|w1], w2) = inversa_aux(w1, [x|w2]) .  
end_of_list.
```

conjuntos.in

```
list(demodulators).
pertenece(x, []) = $F.
pertenece(x, [y|w]) = $IF($ID(x,y), $T, pertenece(x,w)).

subconjunto([], w) = $T.
subconjunto([x|w1], w2) = (pertenece(x, w2) &
                           subconjunto(w1, w2)).

interseccion([], w) = [].
interseccion([x|w1], w2) = $IF(pertenece(x, w2),
                                [x|interseccion(w1, w2)],
                                interseccion(w1, w2)).

union([], w) = w.
union([x|w1], w2) = $IF(pertenece(x, w2),
                        union(w1, w2),
                        [x|union(w1, w2)]).

end_of_list.
```

El problema de las jarras

- Generación explícita de la solución

jarras-f.in

```
make_evaluable(_+_ , $SUM(_,_)) .  
make_evaluable(_-_ , $DIFF(_,_)) .  
make_evaluable(_<=_ , $LE(_,_)) .  
make_evaluable(_>_ , $GT(_,_)) .
```

```
set(hyper_res) .
```

```
list(demodulators) .  
  min(x,y) = $IF(x<=y,x,y) .  
end_of_list .
```

El problema de las jarras

- Generación explícita de la solución

jarras-f.in

```
formula_list(sos).
  all x y w (jarras(x,y,w) -> jarras(0,y,["vaciar A"|w])).
  all x y w (jarras(x,y,w) -> jarras(x,0,["vaciar B"|w])).
  all x y w (jarras(x,y,w) -> jarras(4,y,["llenar A"|w])).
  all x y w (jarras(x,y,w) -> jarras(x,3,["llenar B"|w])).
  all x y w (jarras(x,y,w) ->
    jarras(x+y-min(y+x,3),min(y+x,3),["verter A en B"|w])).
  all x y w (jarras(x,y,w) ->
    jarras(min(x+y,4),x+y-min(x+y,4),["verter B en A"|w])).

  jarras(0,0,[]).
end_of_list.

formula_list(passive).
  all x w (jarras(2,x,w) -> $ANS(w)).
end_of_list.
```

El problema de las jarras

- Generación de estados redundantes

Búsqueda fallida

```
given clause #1: (wt=4) 8 [] jarras(0,0,[]).
given clause #2: (wt=10)
  2 [] -jarras(x,y,w) | jarras(0,y,["vaciar A"|w]).
** KEPT (pick-wt=6): 10 [hyper,2,8] jarras(0,0,["vaciar A"]).
given clause #3: (wt=6) 10 [hyper,2,8] jarras(0,0,["vaciar A"]).
** KEPT (pick-wt=8):
  11 [hyper,10,2] jarras(0,0,["vaciar A","vaciar A"]).
given clause #4: (wt=8)
  11 [hyper,10,2] jarras(0,0,["vaciar A","vaciar A"]).
** KEPT (pick-wt=10):
  12 [hyper,11,2] jarras(0,0,["vaciar A","vaciar A","vaciar A"]).
given clause #5: (wt=10)
  3 [] -jarras(x,y,w) | jarras(x,0,["vaciar B"|w]).
** KEPT (pick-wt=10):
  13 [hyper,3,11] jarras(0,0,["vaciar B","vaciar A","vaciar A"]).
** KEPT (pick-wt=8):
  14 [hyper,3,10] jarras(0,0,["vaciar B","vaciar A"]).
** KEPT (pick-wt=6): 15 [hyper,3,8] jarras(0,0,["vaciar B"]).
...
```

El problema de las jarras

- Estados redundantes
 - `jarras(4,0,["llenar A"])`
 - `jarras(4,0,["vaciar B","llenar A"])`
- La función `$IGNORE(...)` indica al sistema que el argumento no debe ser tenido en cuenta en el proceso de subsumción
 - `jarras(4,0,$IGNORE(["llenar A"]))`
 - `jarras(4,0,$IGNORE(["vaciar B","llenar A"]))`

El problema de las jarras

- Generación explícita de la solución

jarras-g.in

```
make_evaluable(_+_, $SUM(_,_)).  
make_evaluable(_-_, $DIFF(_,_)).  
make_evaluable(_<=_, $LE(_,_)).  
make_evaluable(_>_, $GT(_,_)).  
  
set(hyper_res).  
  
list(demodulators).  
  min(x,y) = $IF(x<=y,x,y).  
end_of_list.
```

El problema de las jarras

- Generación explícita de la solución

jarras-g.in

```
formula_list(sos).
  all x y w (jarras(x,y,$IGNORE(w)) ->
              jarras(0,y,$IGNORE(["vaciar A"|w]))).
  all x y w (jarras(x,y,$IGNORE(w)) ->
              jarras(x,0,$IGNORE(["vaciar B"|w]))).
  all x y w (jarras(x,y,$IGNORE(w)) ->
              jarras(4,y,$IGNORE(["llenar A"|w]))).
  all x y w (jarras(x,y,$IGNORE(w)) ->
              jarras(x,3,$IGNORE(["llenar B"|w]))).
  all x y w (jarras(x,y,$IGNORE(w)) ->
              jarras(x+y-min(y+x,3),min(y+x,3),
              $IGNORE(["verter A en B"|w]))).
  all x y w (jarras(x,y,$IGNORE(w)) ->
              jarras(min(x+y,4),x+y-min(x+y,4),
              $IGNORE(["verter B en A"|w]))).

  jarras(0,0,$IGNORE([])).
end_of_list.

formula_list(passive).
  all x w (jarras(2,x,$IGNORE(w)) -> $ANS(w)).
end_of_list.
```

El problema de las jarras

- Subsumción de estados redundantes

Proceso de búsqueda

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

```
given clause #1: (wt=5) 8 [] jarras(0,0,$IGNORE([])).
given clause #2: (wt=12) 2 [] -jarras(x,y,$IGNORE(w)) |
                                jarras(0,y,$IGNORE(["vaciar A"|w])).
    0 [hyper,2,8] jarras(0,0,$IGNORE(["vaciar A"])).
    Subsumed by 8.
given clause #3: (wt=12) 3 [] -jarras(x,y,$IGNORE(w)) |
                                jarras(x,0,$IGNORE(["vaciar B"|w])).
    0 [hyper,3,8] jarras(0,0,$IGNORE(["vaciar B"])).
    Subsumed by 8.
given clause #4: (wt=12) 4 [] -jarras(x,y,$IGNORE(w)) |
                                jarras(4,y,$IGNORE(["llenar A"|w])).
    0 [hyper,4,8] jarras(4,0,$IGNORE(["llenar A"])).
** KEPT (pick-wt=7): 10 [hyper,4,8] jarras(4,0,$IGNORE(["llenar A"])).
...
```

El problema de las jarras

Prueba obtenida

```
1 [] min(x,y) = $IF(x<=y,x,y) .
3 [] -jarras(x,y,$IGNORE(w)) | jarras(x,0,$IGNORE(["vaciar B"|w])).
4 [] -jarras(x,y,$IGNORE(w)) | jarras(4,y,$IGNORE(["llenar A"|w])).
6 [] -jarras(x,y,$IGNORE(w)) |
    jarras(x+y-min(y+x,3),min(y+x,3),$IGNORE(["verter A en B"|w])).
8 [] jarras(0,0,$IGNORE([])).
9 [] -jarras(2,x,$IGNORE(w)) | $ANS(w) .
10 [hyper,4,8] jarras(4,0,$IGNORE(["llenar A"])).
13 [hyper,6,10,demod,1,1]
    jarras(1,3,$IGNORE(["verter A en B","llenar A"])).
14 [hyper,13,3]
    jarras(1,0,$IGNORE(["vaciar B","verter A en B","llenar A"])).
15 [hyper,14,6,demod,1,1]
    jarras(0,1,$IGNORE(["verter A en B","vaciar B",
        "verter A en B","llenar A"])).
16 [hyper,15,4]
    jarras(4,1,$IGNORE(["llenar A","verter A en B","vaciar B",
        "verter A en B","llenar A"])).
17 [hyper,16,6,demod,1,1]
    jarras(2,3,$IGNORE(["verter A en B","llenar A","verter A en B",
        "vaciar B","verter A en B","llenar A"])).
18 [binary,17.1,9.1]
    $ANS(["verter A en B","llenar A","verter A en B","vaciar B",
        "verter A en B","llenar A"]).
```

- McCune, W. *Otter 3.3 Reference Manual* (Argonne National Laboratory, 2003)
 - Cap. 9: “Evaluable Functions and Predicates”