Tema 9: Resolución de problemas con razonamiento automático

José A. Alonso Jiménez
Miguel A. Gutiérrez Naranjo

Dpto. de Ciencias de la Computación e Inteligencia Artificial

Universidad de Sevilla

Rompecabeza: Problema del baile

- **Problema: En un baile hay 25 personas. La primera mujer ha bailado con los 10 primeros hombres, la segunda con los 11 primeros, la tercera con los 12 primeros y así sucesivamente. Los 10 primeros hombres han bailado con todas las mujeres, el undécimo con todas menos con la primera y así sucesivamente. ¿Cuántas mujeres y hombres hay en el baile?.**
- **Entrada** `baile.in`

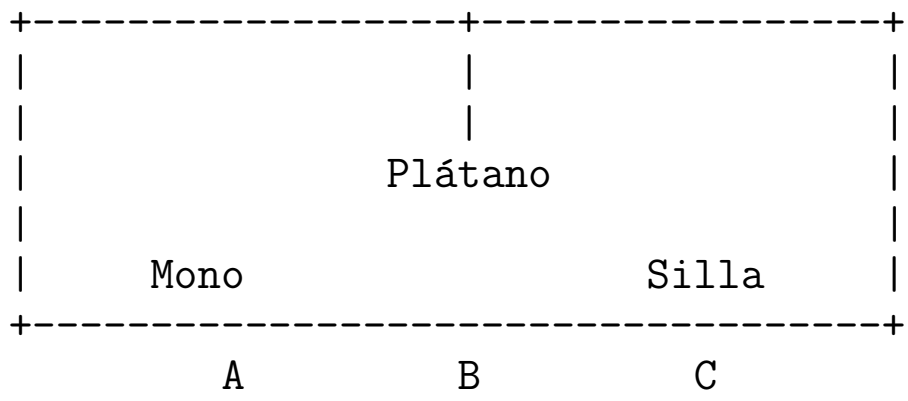
```
make_evaluable(_+_, $SUM(_,_)).
make_evaluable(_<_, $LT(_,_)).
make_evaluable(_==_, $EQ(_,_)).
set(binary_res).
set(hyper_res).

list(sos).
total(25).
bailado(mujer(1),10).
-bailado(mujer(x),y) | -total(z) | - (x + y < z)
 | bailado(mujer(x+1),y+1).
-bailado(mujer(x),y) | -total(x+y) | $ans(x,y).
end_of_list.
```

Rompecabeza: Problema del baile

- **Solución**

```
 1 [] total(25).
 2 [] bailado(mujer(1),10).
 3 [] -bailado(mujer(x),y)| -total(z)| -(x+y<z)
      | bailado(mujer(x+1),y+1).
 4 [] -bailado(mujer(x),y)| -total(x+y)|$ans(x,y).
12 [hyper,3,2,1,eval,demod] bailado(mujer(2),11).
15 [hyper,12,3,1,eval,demod] bailado(mujer(3),12).
18 [hyper,15,3,1,eval,demod] bailado(mujer(4),13).
21 [hyper,18,3,1,eval,demod] bailado(mujer(5),14).
24 [hyper,21,3,1,eval,demod] bailado(mujer(6),15).
27 [hyper,24,3,1,eval,demod] bailado(mujer(7),16).
30 [hyper,27,3,1,eval,demod] bailado(mujer(8),17).
31 [binary,30.1,4.1,demod] -total(25)|$ans(8,17).
32 [binary,31.1,1.1] $ans(8,17).
```

Planificación: Problema del mono



- **Representación:**
  `vale(pos_mono(X),pos_platano(Y),pos_silla(Z),Plan)`
  significa que en el estado obtenido aplicando el `Plan` **(inverso) al estado inicial se verifica que la posición del mono es** `X`**, la del plátano es** `Y` **y la de la silla es** `Z`

- **Entrada** `mono.in`

```
set(prolog_style_variables).
set(input_sequent).
set(output_sequent).
set(ur_res).

list(usable).
posicion(X), posicion(Y),
vale(pos_mono(X),pos_platano(Pp),pos_silla(Ps),Plan)
->
vale(pos_mono(Y),pos_platano(Pp),pos_silla(Ps),
     [andar(X,Y)|Plan]).

posicion(Y),
vale(pos_mono(X),pos_platano(Pp),pos_silla(X),Plan)
->
vale(pos_mono(Y),pos_platano(Pp),pos_silla(Y),
     [empujar(X,Y)|Plan]).
```

Planificación: Problema del mono

```
vale(pos_mono(P),pos_platano(P),pos_silla(P),Plan)
->
coge_platano([subir|Plan]).
end_of_list.

list(sos).
-> posicion(a).
-> posicion(b).
-> posicion(c).

-> vale(pos_mono(a),pos_platano(b),pos_silla(c),[]).

coge_platano(Plan) -> resp(inversa(Plan,[])).
end_of_list.

list(passive).
resp(Plan) -> $ans(Plan).
end_of_list.

list(demodulators).
-> inversa([X|L1],L2) = inversa(L1,[X|L2]).
-> inversa([],L) = L.
end_of_list.
```

Planificación: Problema del mono

```
1 [] posicion(X), posicion(Y),
     vale(pos_mono(X),pos_platano(Pp),pos_silla(Ps),Plan)
     -> vale(pos_mono(Y),pos_platano(Pp),pos_silla(Ps),
             [andar(X,Y)|Plan]).
2 [] posicion(Y),
     vale(pos_mono(X),pos_platano(Pp),pos_silla(X),Plan)
     -> vale(pos_mono(Y),pos_platano(Pp),pos_silla(Y),
             [empujar(X,Y)|Plan]).
3 [] vale(pos_mono(P),pos_platano(P),pos_silla(P),Plan)
     -> coge_platano([subir|Plan]).
4 [] -> posicion(a).
5 [] -> posicion(b).
6 [] -> posicion(c).
7 [] -> vale(pos_mono(a),pos_platano(b),pos_silla(c),[]).
8 [] coge_platano(Plan) -> resp(inversa(Plan,[])).
9 [] resp(Plan) -> $ans(Plan).
10 []  -> inversa([X|L1],L2)=inversa(L1,[X|L2]).
11 []  -> inversa([],L)=L.
12 [hyper,7,1,4,6]
   -> vale(pos_mono(c),pos_platano(b),pos_silla(c),
           [andar(a,c)]).
16 [hyper,12,2,5]
   -> vale(pos_mono(b),pos_platano(b),pos_silla(b),
           [empujar(c,b),andar(a,c)]).
33 [hyper,16,3]
   -> coge_platano([subir,empujar(c,b),andar(a,c)]).
40 [hyper,33,8,demod,10,10,10,11]
   -> resp([andar(a,c),empujar(c,b),subir]).
41 [binary,40.1,9.1]
   -> $ans([andar(a,c),empujar(c,b),subir]).
```

Problema de las jarras

- **Enunciado:**
  - **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.**
- **Entrada** `jarras.in`

```
set(prolog_style_variables).
set(input_sequent).
set(output_sequent).
make_evaluable(_+_, $SUM(_,_)).
make_evaluable(_-_, $DIFF(_,_)).
make_evaluable(_<=_, $LE(_,_)).
make_evaluable(_>_, $GT(_,_)).
set(hyper_res).
```

Problema de las jarras

```
list(usable).
e(X,Y)            -> e(3,Y).
e(X,Y)            -> e(0,Y).
e(X,Y)            -> e(X,4).
e(X,Y)            -> e(X,0).
e(X,Y), X+Y <= 4 -> e(0,Y+X).
e(X,Y), X+Y >  4 -> e(X - (4-Y), 4).
e(X,Y), X+Y <= 3 -> e(X+Y, 0).
e(X,Y), X+Y >  3 -> e(3, Y - (3-X)).
end_of_list.

list(sos).
-> e(0,0).  % Estado inicial
e(X,2) ->.  % Estado final
end_of_list.
```

- **Prueba**

```
 2 [] e(X,Y) -> e(0,Y).
 3 [] e(X,Y) -> e(X,4).
 7 [] e(X,Y), X+Y<=3 -> e(X+Y,0).
 8 [] e(X,Y), X+Y>3 -> e(3,Y- (3-X)).
 9 []  -> e(0,0).
10 [] e(X,2) -> .
11 [hyper,9,3]  -> e(0,4).
13 [hyper,11,8,eval,demod]  -> e(3,1).
16 [hyper,13,2]  -> e(0,1).
18 [hyper,16,7,eval,demod]  -> e(1,0).
20 [hyper,18,3]  -> e(1,4).
22 [hyper,20,8,eval,demod]  -> e(3,2).
23 [binary,22.1,10.1]  -> .
```