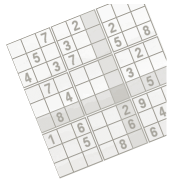


Constraint Programming

Practical Exercises

Roman Barták
Department of Theoretical Computer Science and Mathematical Logic

Constraint Modelling

Homework

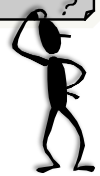
- Write a program to solve the letter puzzle DONALD + GERARD = ROBERT. Use the constraint model with carry bits.

The Letter Puzzle

- Solve the letter puzzle DONALD + GERARD = ROBERT using the constraint model with carry bits.

```
:-use_module(library(clpfd)).
solve(Sol):-
  Sol=[D,O,N,A,L,G,E,R,B,T],
  Carry = [P1,P2,P3,P4,P5],
  domain(Sol,0,9), domain(Carry,0,1),
  D #\= 0, G #\= 0, R #\= 0,
  D+D    #= T + 10*P1,
  L+R+P1 #= R + 10*P2,
  A+A+P2 #= E + 10*P3,
  N+R+P3 #= B + 10*P4,
  O+E+P4 #= O + 10*P5,
  D+G+P5 #= R,
  all_different(Sol),
  append(Sol,Carry,Vars),
  labeling([],Vars).
```

```
?- solve(X).
no
```



```
?- solve(X).
X = [5,2,6,4,8,1,9,7,3,0] ;
no
```

Tabular constraints

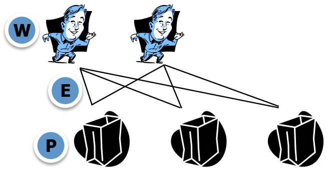
- What if we need to describe a general relation?
- How to describe a constraint specified by a set of compatible values?
- **table(Vars,Extension)**

```
?-table([[X,Y,Z]],[[1,2,3],[1,3,4]]).
X = 1,
Y in 2..3,
Z in 3..4
```

It is possible to specify identical constraint over more variable tuples.
`table([[X,Y,Z],[P,Q,R]], ...)`

Modelling – the table constraint

Assume two workers producing three possible products with different efficiencies. The efficiency of each worker for each product is given by a table.



	P1	P2	P3
W1	7	1	3
W2	8	2	5

Propose a constraint connecting variables W(worker), P(product), and E(efficiency).

```
W in {1,2}, P in 1..3,
table([[W,P,E]], [[1,1,7],[1,2,1],[1,3,3],
[2,1,8],[2,2,2],[2,3,5]])
```

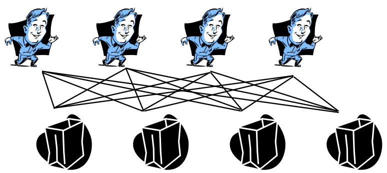
Binary tabular relations

- N-ary tabular constraints can be transformed to binary constraints using the **hidden variable encoding** via the constraint encoding.
- **`element(X,List,Y)`**
  - Y is X-th element in the List: $Y = List_X$
- **Approach**
  - each tuple is identified by a number
  - for each variable we introduce one element constraint, where the list consists of values of the variable in tuples
- Example: [[1,2,3], [1,3,4], [2,4,4]]

```
?-element(I, [1,1,2], X), element(I, [2,3,4], Y),
  element(I, [3,4,4], Z)
I in 1..3,
X in 1..2,
Y in 2..4,
Z in 3..4
```

Modelling – the element constraints

Assume four workers producing four possible products with different efficiencies. The efficiency of each worker for each product is given by a table.



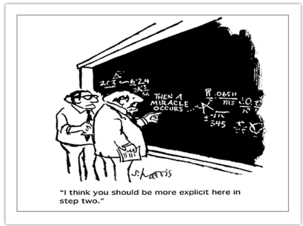
	P1	P2	P3	P4
W1	7	1	3	4
W2	8	2	5	1
W3	4	3	7	2
W4	3	1	6	3

Let Wi be a variable denoting the product being produced by worker i. Describe constraints connecting Wi with efficiency Ei.

```
element(W1,[7,1,3,4],E1),
element(W2,[8,2,5,1],E2),
element(W3,[4,3,7,2],E3),
element(W4,[3,1,6,3],E4).
```

Problem modelling

- Which **decision variables** are needed?
  - variables denoting the problem solution
  - they also define the search space
- Which **values** can be assigned to variables?
  - the definition of domains influences the constraints used
- How to formalise **constraints**?
  - Which constraints are available?
  - auxiliary variables may be necessary



4-queens

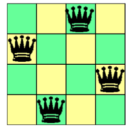
- Propose a constraint model for solving the 4-queens problem (place four queens to a chessboard of size 4x4 such that there is no conflict).

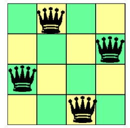
```
:-use_module(library(clpfd)).

queens([(X1,Y1),(X2,Y2),(X3,Y3),(X4,Y4)]):-
   Rows = [X1,X2,X3,X4], Columns = [Y1,Y2,Y3,Y4],
   domain(Rows,1,4),
   domain(Columns,1,4),
   all_different(Rows), all_different(Columns),
   abs(X1-X2) #\= abs(Y1-Y2),
   abs(X1-X3) #\= abs(Y1-Y3), abs(X1-X4) #\= abs(Y1-Y4),
   abs(X2-X3) #\= abs(Y2-Y3), abs(X2-X4) #\= abs(Y2-Y4),
   abs(X3-X4) #\= abs(Y3-Y4),
   append(Rows,Columns, Variables),
   labeling([], Variables).
```

4-queens: analysis

```
?- queens(L).
L = [(1,2),(2,4),(3,1),(4,3)] ? ;
L = [(1,3),(2,1),(3,4),(4,2)] ? ;
L = [(1,2),(2,4),(4,3),(3,1)] ? ;
L = [(1,3),(2,1),(4,2),(3,4)] ? ;
L = [(1,2),(3,1),(2,4),(4,3)] ? ;
L = [(1,3),(3,4),(2,1),(4,2)] ? ;
L = [(1,2),(3,1),(4,3),(2,4)] ? ;
L = [(1,3),(3,4),(4,2),(2,1)] ? ;
…
```





Where is the problem?

- Different assignments describe the same solution!
- There are only two different solutions (very „similar" solutions).
- The search space is non-necessarily large.

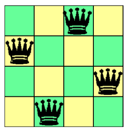
Solution

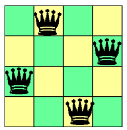
- pre-assign queens to rows (or to columns)

4-queens: a better model

```
:-use_module(library(clpfd)).
queens4(Queens):-
   Queens = [X1,X2,X3,X4],
   domain(Queens,1,4),
   all_different(Queens),
   abs(X1-X2) #\= 1, abs(X1-X3) #\= 2, abs(X1-X4) #\= 3,
   abs(X2-X3) #\= 1, abs(X2-X4) #\= 2,
   abs(X3-X4) #\= 1,
   labeling([], Queens).

?- queens4(Q).
Q = [2,4,1,3] ? ;
Q = [3,1,4,2] ? ;
no
```





Model properties:

- less variables (= smaller state space)
- less constraints (= faster propagation)

Homework:

- Write a constraint model for arbitrary number of queens (given as input)
- think about further improvements

Knapsack problem

- The smuggler has a knapsack with capacity **9 units**. It is possible to fill it in with **whisky bottles** (each consumes **4 units**), **perfume bottles** (each consumes **3 units**), and **cigarette boxes** (each consumes **2 units**). Any mix od products can be used. The profit from whisky is **15 dollars**, from perfumes **10 dollars**, and from cigarettes **7 dollars**. What can be placed to the knapsack if the required total profit is at least 30 dollars?
- Propose a constraint model to solve the problem.

Homework