5

# *Programming with Logic and Constraints*


**Roman Barták**
**Charles University, Prague (CZ)**

roman.bartak@mff.cuni.cz
http://ktiml.mff.cuni.cz/~bartak


## Disjunctions

Let us start with a simple example

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

a(X):- X#<5.
a(X):- X#>7.
```

```
?- a(X).

X in inf..4 ? ;
X in 8..sup ? ;

no
```

**What is the problem?**

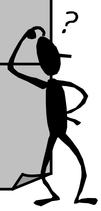
The constraint model is disjunctive, i.e., we need to backtrack to get the model where X>7!

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

a(X):- X#<5 #\/ X#>7.
```

```
?- a(X).

X in inf..sup ? ;

no
```

```
?- a(X), X#>5.

X in 8..sup ? ;

no
```

**The propagator waits until all but one component of the disjunction are proved to fail and then it propagates through the remaining component.**



## Constructive Disjunction

```
:-use_module(library(clpfd)).
a(X):- X in (inf..4) \/ (8..sup).
```

```
?- a(X).
X in (inf..4)\/(8..sup) ? ;
no
```

**Constructive disjunction**

How does it work in general?

$a_1(X) \vee a_2(X) \vee \ldots\ a_n(X)$

- **propagate** each constraint $a_i(X)$ **separately**
- **union** all the restricted **domains** for X

This could be an expensive process!

Actually, it is close to **singleton consistency**:

- X in 1..5 $\Rightarrow$ X=1 $\vee$ X=2 $\vee$ X=3 $\vee$ X=4 $\vee$ X=5

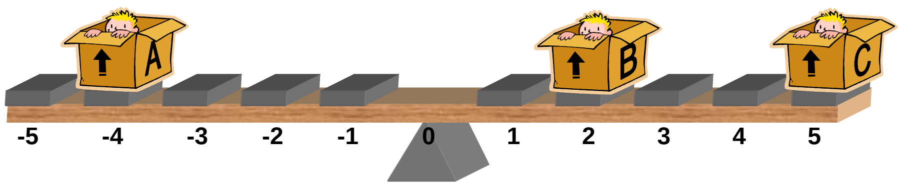
**We can still write special propagators for particular disjunctive constraints!**



## Seesaw problem

**The problem:**

Adam (36 kg), Boris (32 kg) and Cecil (16 kg) want to sit on a seesaw with the length 10 foots such that the minimal distances between them are more than 2 foots and the seesaw is balanced.




**A CSP model:**

| | |
|---|---|
| ■ A,B,C in -5..5 | position |
| ■ 36*A+32*B+16*C = 0 | equilibrium state |
| ■ \|A-B\|>2, \|A-C\|>2, \|B-C\|>2 | minimal distances |

## Seesaw problem
### implementation

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

seesaw(Sol):-
    Sol = [A,B,C],

    domain(Sol,-5,5),
    36*A+32*B+16*C #= 0,
    abs(A-B)#>2, abs(A-C)#>2, abs(B-C)#>2,

    labeling([ff],Sol).
```

```
?- seesaw(X).

X = [-4,2,5] ? ;
X = [-4,4,1] ? ;
X = [-4,5,-1] ? ;
X = [4,-5,1] ? ;
X = [4,-4,-1] ? ;
X = [4,-2,-5] ? ;

no
```

**Symmetry breaking**

- important to reduce search space

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

seesaw(Sol):-
    Sol = [A,B,C],

    domain(Sol,-5,5),
    A #< 0,
    36*A+32*B+16*C #= 0,
    abs(A-B)#>2, abs(A-C)#>2, abs(B-C)#>2,

    labeling([ff],Sol).
```

```
?- seesaw(X).

X = [-4,2,5] ? ;
X = [-4,4,1] ? ;
X = [-4,5,-1] ? ;

no
```

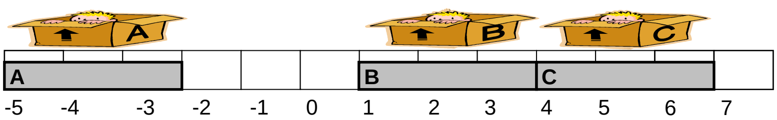


## Seesaw problem
### a different perspective

```
domain([A,B,C],-5,5),
A #< 0,
36*A+32*B+16*C #= 0,
abs(A-B)#>2,
abs(A-C)#>2,
abs(B-C)#>2
```

```
A in -4..0
B in -1..5
C in -5..5
```

- A set of similar constraints typically indicates a structured sub-problem that can be represented using a **global constraint.**




- We can use a global constraint describing **allocation of activities to exclusive resource**.

```
domain([A,B,C],-5,5),
A #< 0,
36*A+32*B+16*C #= 0,
serialized([A,B,C],[3,3,3],[])
```

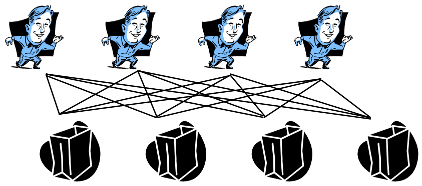
start times | durations | precedences

```
A in -4..0
B in -1..5
C in (-5..-3) \/ (-1..5)
```

## Assignment problem

### The problem:

There are 4 workers and 4 products and a table describing the efficiency of producing the product by a given worker. The task is assign workers to products (one to one) in such a way that the total efficiency is at least 19.



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

### A CSP model:

- W1,W2,W3,W4 in 1..4 — a product per worker
- all_different([W1,W2,W3,W4]) — different products
- $T_{1,W1}+T_{2,W2}+T_{3,W3}+T_{4,W4} \geq 19$ — total efficiency



## Assignment problem
### implementation

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

assignment_p(Sol):-
    Sol = [W1,W2,W3,W4],

    domain(Sol,1,4),
    all_different(Sol),
    element(W1,[7,1,3,4],EW1),
    element(W2,[8,2,5,1],EW2),
    element(W3,[4,3,7,2],EW3),
    element(W4,[3,1,6,3],EW4),
    EW1+EW2+EW3+EW4 #>= 19,

    labeling([ff],Sol).
```

```
?- assignment_p(X).

X = [1,2,3,4] ? ;   19
X = [2,1,3,4] ? ;   19
X = [4,1,2,3] ? ;   21
X = [4,1,3,2] ? ;   20

no
```

### Optimization using B&B

```
    EW1+EW2+EW3+EW4 #= E,

    maximize(labeling([ff],Sol),E).
```

```
?- assignment_p(X).

X = [4,1,2,3] ? ;

no
```

### How does it work?

- **find** first **feasible instantiation** of variables
- **find better instantiation** of variables
- **repeat** until some instantiation of variables exists

## Assignment problem
### a dual model

Why do we **assign products to workers?**

Cannot we do it in an opposite way, that is, to **assign a worker to the product?**

Of course, we can **swap the role of values and variables!**

- This new model is called a **dual model.**

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

assignment_dual(Products):-
    Products = [P1,P2,P3,P4],

    domain(Products,1,4),
    all_different(Products),
    element(P1,[7,8,4,3],EP1),
    element(P2,[1,2,3,1],EP2),
    element(P3,[3,5,7,6],EP3),
    element(P4,[4,1,2,3],EP4),
    EP1+EP2+EP3+EP4 #>= 19,

    labeling([ff],Products).
```

| Number of choice points | |
|---|---|
| Primal model | 15 |
| Dual model | 11 |

```
P1 in 1..2
P2 in 1..4
P3 in 2..4
P4 in 1..4
```

**Which model is better?**

- In this particular case, the dual model propagates earlier (thus it is assumed to be better).



## Assignment problem
### composing models

**We can combine both primal and dual model in a single model to get better domain pruning.**

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

assignment_combined(Workers):-
    Workers= [W1,W2,W3,W4],
    domain(Workers,1,4),
    all_different(Workers),
    element(W1,[7,1,3,4],EW1),
    element(W2,[8,2,5,1],EW2),
    element(W3,[4,3,7,2],EW3),
    element(W4,[3,1,6,3],EW4),
    EW1+EW2+EW3+EW4 #>= 19,

    Products = [P1,P2,P3,P4],
    domain(Products,1,4),
    all_different(Products),
    element(P1,[7,8,4,3],EP1),
    element(P2,[1,2,3,1],EP2),
    element(P3,[3,5,7,6],EP3),
    element(P4,[4,1,2,3],EP4),
    EP1+EP2+EP3+EP4 #>= 19,

    assignment(Workers,Products),

    labeling([ff],Workers).
```

- **a primal model**

```
W1 in (1..2)\/{4}
W2 in 1..4
W3 in 2..4
W4 in 2..4
```

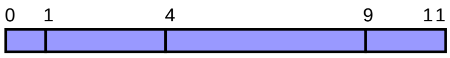
- **a dual model** (redundant)

```
P1 in 1..2
P2 in 1..4
P3 in 2..4
P4 in 1..4
```

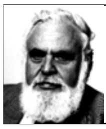
- **a channelling constraint**
- labelling one model is enough

## Golomb ruler

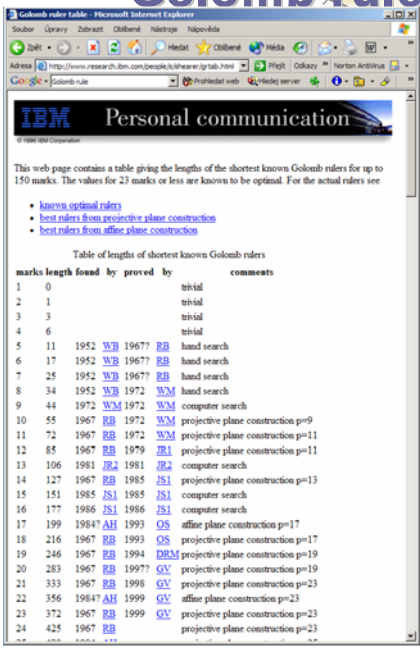
- A **ruler with M marks** such that **distances** between any two marks are **different**.
- The **shortest ruler** is the optimal ruler.




- **Hard** for $M \geq 16$, no exact algorithm for $M \geq 24$!
- Applied in **radioastronomy**.



Solomon W. Golomb
*Professor*
*University of Southern California*
`http://csi.usc.edu/faculty/golomb.html`



Golomb ruler table - Microsoft Internet Explorer

IBM Personal communication

© 1996 IBM Corporation

This web page contains a table giving the lengths of the shortest known Golomb rulers for up to 150 marks. The values for 23 marks or less are known to be optimal. For the actual rulers see

- known optimal rulers
- best rulers from projective plane construction
- best rulers from affine plane construction

Table of lengths of shortest known Golomb rulers

| marks | length | found | by | proved | by | comments |
|---|---|---|---|---|---|---|
| 1 | 0 | | | | | trivial |
| 2 | 1 | | | | | trivial |
| 3 | 3 | | | | | trivial |
| 4 | 6 | | | | | trivial |
| 5 | 11 | 1952 | WB | 1967? | RB | hand search |
| 6 | 17 | 1952 | WB | 1967? | RB | hand search |
| 7 | 25 | 1952 | WB | 1967? | RB | hand search |
| 8 | 34 | 1952 | WB | 1972 | WM | hand search |
| 9 | 44 | 1972 | WM | 1972 | WM | computer search |
| 10 | 55 | 1967 | RB | 1972 | WM | projective plane construction p=9 |
| 11 | 72 | 1967 | RB | 1972 | WM | projective plane construction p=11 |
| 12 | 85 | 1967 | RB | 1979 | JR1 | projective plane construction p=11 |
| 13 | 106 | 1981 | JR2 | 1981 | JR2 | computer search |
| 14 | 127 | 1967 | RB | 1985 | JS1 | projective plane construction p=13 |
| 15 | 151 | 1985 | JS1 | 1985 | JS1 | computer search |
| 16 | 177 | 1986 | JS1 | 1986 | JS1 | computer search |
| 17 | 199 | 1984? | AH | 1993 | OS | affine plane construction p=17 |
| 18 | 216 | 1967 | RB | 1993 | OS | projective plane construction p=17 |
| 19 | 246 | 1967 | RB | 1994 | DRM | projective plane construction p=19 |
| 20 | 283 | 1967 | RB | 1997? | GV | projective plane construction p=19 |
| 21 | 333 | 1967 | RB | 1998 | GV | projective plane construction p=23 |
| 22 | 356 | 1984? | AH | 1999 | GV | affine plane construction p=23 |
| 23 | 372 | 1967 | RB | 1999 | GV | projective plane construction p=23 |
| 24 | 425 | 1967 | RB | | | projective plane construction p=23 |



## Golomb ruler

### CSP model

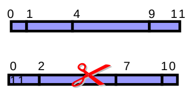
**A base model:**

Variables $X_1, \ldots, X_M$ with the domain 0..M*M

| | |
|---|---|
| $\mathbf{X_1 = 0}$ | *ruler start* |
| $\mathbf{X_1 < X_2 < \ldots < X_M}$ | *no permutations of variables* |
| $\mathbf{\forall i<j \; D_{i,j} = X_j - X_i}$ | *difference variables* |
| $\mathbf{all\_different(\{D_{1,2}, D_{1,3}, \ldots D_{1,M}, D_{2,3}, \ldots D_{M,M-1}\})}$ | |

**Model extensions:**

$\mathbf{D_{1,2} < D_{M-1,M}}$ *symmetry breaking*




better bounds (**implied constraints**) for $D_{i,j}$

$$D_{i,j} = D_{i,i+1} + D_{i+1,i+2} + \ldots + D_{j-1,j}$$

so $\mathbf{D_{i,j} \geq \Sigma_{j-i} = (j-i)*(j-i+1)/2}$ *lower bound*

$$X_M = X_M - X_1 = D_{1,M} = D_{1,2} + D_{2,3} + \ldots D_{i-1,i} + D_{i,j} + D_{j,j+1} + \ldots + D_{M-1,M}$$

$$D_{i,j} = X_M - (D_{1,2} + \ldots D_{i-1,i} + D_{j,j+1} + \ldots + D_{M-1,M})$$

so $\mathbf{D_{i,j} \leq X_M - (M-1-j+i)*(M-j+i)/2}$ *upper bound*

## Golomb ruler
### some results

- **What is the effect of different constraint models?**

| size | base model | base model + symmetry | base model + symmetry + implied constraints |
|---|---|---|---|
| 7 | 220 | 80 | 30 |
| 8 | 1 462 | 611 | 190 |
| 9 | 13 690 | 5 438 | 1 001 |
| 10 | 120 363 | 49 971 | 7 011 |
| 11 | 2 480 216 | 985 237 | 170 495 |

time in milliseconds on Mobile Pentium 4-M 1.70 GHz, 768 MB RAM

- **What is the effect of different search strategies?**

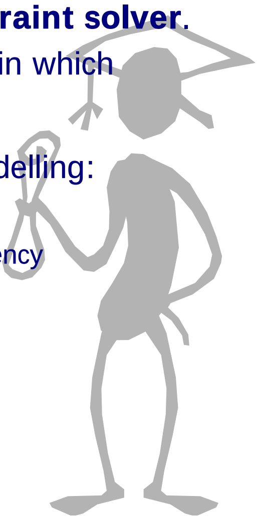
| size | fail first | | | leftmost first | | |
|---|---|---|---|---|---|---|
| | *enum* | *step* | *bisect* | *enum* | *step* | *bisect* |
| 7 | 40 | 60 | 40 | 30 | 30 | 30 |
| 8 | 390 | 370 | 350 | 220 | 190 | 200 |
| 9 | 2 664 | 2 384 | 2 113 | 1 182 | 1 001 | 921 |
| 10 | 20 870 | 17 545 | 14 982 | 8 782 | 7 011 | 6 430 |
| 11 | 1 004 515 | 906 323 | 779 851 | 209 251 | 170 495 | 159 559 |

time in milliseconds on Mobile Pentium 4-M 1.70 GHz, 768 MB RAM



## Modeling rules

- Determining the **efficiency of different models** is a difficult problem and one which **relies upon** an **understanding** of the underlying **constraint solver**.
- Usually, the **best model** will be the one in which information is **propagated first.**
- Some **rules of thumb** for constraint modelling:
  - **global constraints**
    - (+) strengthen propagation with good efficiency
  - **symmetry breaking**
    - (+) reduce search space
  - **implied constraints**
    - (+) strengthen propagation
    - (–) but add overhead



# Conclusions

## Systems: SICStus Prolog

**www.sics.se/sicstus**

- a strong Prolog system with libraries for solving constraints (FD, Boolean, Real)
- arithmetical, logical, and some global constraints
  - an interface for defining new filtering algorithms
- depth-first search with customizable value and variable selection (also optimization)
  - it is possible to use Prolog backtracking

## Systems: ECLiPSe

**www.icparc.ic.ac.uk/eclipse**

- a Prolog system with libraries for solving constraints (FD, Real, Sets)
- integration with OR packages (CPLEX, XPRESS-MP)
- arithmetical, logical, and some global constraints
  - an interface for defining new filtering algorithms
- Prolog depth-first search (also optimization)
- a repair library for implementing local search techniques



## Systems: CHIP

**www.cosytec.com**

- a constraint solver in C with Prolog as a host language, also available as C and C++ libraries
- popularized the concept of global constraints
  - different, order, resource, tour, dependency
- it is hard to go beyond the existing constraints

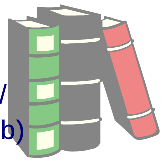
## Resources: Prolog

### Printed

- **Programming in Prolog**
  W.F. Clocksin & C.S. Mellish, Springer-Verlag, Berlin, 1986
- **The Art of Prolog**
  L. Sterling & E. Shapiro, The MIT Press, Cambridge, Massachusetts, 1986
- **The Craft of Prolog**
  O'Keefe R.A., MIT Press, 1990
- **PROLOG Programming for Artificial Intelligence**
  Bratko I., Addison-Wesley, Reading, MA, 2001 (third edition)

### On-line

- **On-line Guide to Prolog Programming** (tutorial)
  http://kti.mff.cuni.cz/~bartak/prolog/
- **Learn Prolog Now!** (tutorial)
  http://www.coli.uni-saarland.de/~kris/learn-prolog-now/
- **Association for Logic Programming** (community web)
  http://www.cwi.nl/projects/alp/





## Resources: Constraints

### Printed

- **Constraint Satisfaction in Logic Programming**
  P. Van Hentenryck:, MIT Press, 1989
- **Constraint Processing**
  R. Dechter, Morgan Kaufmann, 2003
- **Programming with Constraints: An Introduction**
  K. Marriott and P.J. Stuckey, The MIT Press, 1998
- **Foundations of Constraint Satisfaction**
  E. Tsang, Academic Press, 1993

### On-line

- **On-line Guide to Constraint Programming** (tutorial)
  http://kti.mff.cuni.cz/~bartak/constraints/
- **Constraints Archive** (archive and links)
  http://4c.ucc.ie/web/archive/index.jsp
- **Constraint Programming online** (community web)
  http://www.cp-online.org/

ESSLLI 2005 – Programming with Logic and Constraints