Let us start with a simple example

```prolog
:-use_module(library(clpfd)).
a(X) :- X<5.
a(X) :- X>7.
```

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

```prolog
?- a(X).
```

no

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

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

How does it work in general?

\[ a_1(X) \lor a_2(X) \lor \ldots a_n(X) \]

- **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**:

\[ X \text{ in } 1..5 \Rightarrow X=1 \lor X=2 \lor X=3 \lor X=4 \lor X=5 \]

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 feet such that the minimal distances between them are more than 2 feet and the seesaw is balanced.

A CSP model:

- \( A,B,C \text{ in } -5..5 \) position
- \( 36\cdot A+32\cdot B+16\cdot C = 0 \) equilibrium state
- \( |A-B|>2, |A-C|>2, |B-C|>2 \) minimal distances
Symmetry breaking

important to reduce search space

A set of similar constraints typically indicates a structured subproblem that can be represented using a global constraint.

We can use a global constraint describing allocation of exclusive resource.
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.

<table>
<thead>
<tr>
<th></th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
</tr>
</thead>
<tbody>
<tr>
<td>W1</td>
<td>7</td>
<td>1</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>W2</td>
<td>8</td>
<td>2</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>W3</td>
<td>4</td>
<td>3</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>W4</td>
<td>3</td>
<td>1</td>
<td>6</td>
<td>3</td>
</tr>
</tbody>
</table>

A CSP model:

- W1,W2,W3,W4 in 1..4  
- all_different([W1,W2,W3,W4])  
- T1,W1+T2,W2+T3,W3+T4,W4 $\geq$ 19  

Implementation:

```prolog
:-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).
```

Optimization using B&B

```prolog
?- 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
```

How does it work?

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

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

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

A base model:

Variables \( X_1, \ldots, X_M \) with the domain \( 0 \ldots M \times M \)

\( X_1 = 0 \) \hspace{1cm} \text{ruler start}

\( X_1 < X_2 < \ldots < X_M \) \hspace{1cm} \text{no permutations of variables}

\( \forall i < j \hspace{3pt} D_{i,j} = X_j - X_i \) \hspace{1cm} \text{difference variables}

all_different\( \{ D_{1,2}, D_{1,3}, \ldots, D_{1,M}, D_{2,3}, \ldots, D_{M,M-1} \} \)
What is the effect of different constraint models?

<table>
<thead>
<tr>
<th>size</th>
<th>base model</th>
<th>base model + symmetry</th>
<th>base model + symmetry + implied constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>220</td>
<td>80</td>
<td>30</td>
</tr>
<tr>
<td>8</td>
<td>1462</td>
<td>611</td>
<td>190</td>
</tr>
<tr>
<td>9</td>
<td>13690</td>
<td>5438</td>
<td>1001</td>
</tr>
<tr>
<td>10</td>
<td>120363</td>
<td>49971</td>
<td>7011</td>
</tr>
<tr>
<td>11</td>
<td>2480216</td>
<td>985237</td>
<td>170495</td>
</tr>
</tbody>
</table>

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

What is the effect of different search strategies?

<table>
<thead>
<tr>
<th>size</th>
<th>fail first</th>
<th>leftmost first</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>enum</td>
<td>step</td>
</tr>
<tr>
<td>7</td>
<td>40</td>
<td>60</td>
</tr>
<tr>
<td>8</td>
<td>390</td>
<td>370</td>
</tr>
<tr>
<td>9</td>
<td>2664</td>
<td>2384</td>
</tr>
<tr>
<td>10</td>
<td>20870</td>
<td>17545</td>
</tr>
<tr>
<td>11</td>
<td>1004515</td>
<td>906323</td>
</tr>
</tbody>
</table>

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
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

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**
- **The Art of Prolog**
- **The Craft of Prolog**
- **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**
- **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/