Topic 14: Propagation

(Version of 6th November 2020)

Pierre Flener

Optimisation Group
Department of Information Technology
Uppsala University
Sweden

Course 1DL441:
Combinatorial Optimisation and Constraint Programming,
whose part 1 is Course 1DL451:
Modelling for Combinatorial Optimisation

\(^1\) Based partly on material by Christian Schulte
Outline

1. Intuition
   - Example 1
   - Example 2
   - Example 3

2. Algorithms
   - Reminders of Discrete Mathematics
   - Solving: Overview
   - Propagator for a Constraint
   - Fixpoint of Multiple Propagators
Outline

1. Intuition
   Example 1
   Example 2
   Example 3

2. Algorithms
   Reminders of Discrete Mathematics
   Solving: Overview
   Propagator for a Constraint
   Fixpoint of Multiple Propagators
Outline

1. Intuition
   Example 1
   Example 2
   Example 3

2. Algorithms
   Reminders of Discrete Mathematics
   Solving: Overview
   Propagator for a Constraint
   Fixpoint of Multiple Propagators
### Example (Agricultural experiment design, AED)

<table>
<thead>
<tr>
<th></th>
<th>plot1</th>
<th>plot2</th>
<th>plot3</th>
<th>plot4</th>
<th>plot5</th>
<th>plot6</th>
<th>plot7</th>
</tr>
</thead>
<tbody>
<tr>
<td>barley</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>corn</td>
<td>✓</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
</tr>
<tr>
<td>millet</td>
<td>✓</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>oats</td>
<td>-</td>
<td>✓</td>
<td>-</td>
<td>✓</td>
<td>-</td>
<td>✓</td>
<td>-</td>
</tr>
<tr>
<td>rye</td>
<td>-</td>
<td>✓</td>
<td>-</td>
<td>-</td>
<td>✓</td>
<td>-</td>
<td>✓</td>
</tr>
<tr>
<td>spelt</td>
<td>-</td>
<td>-</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
<td>-</td>
<td>✓</td>
</tr>
<tr>
<td>wheat</td>
<td>-</td>
<td>-</td>
<td>✓</td>
<td>-</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
</tr>
</tbody>
</table>

**Constraints to be satisfied:**

1. Equal growth load: Every plot grows 3 grains.
2. Equal sample size: Every grain is grown in 3 plots.
3. Balance: Every grain pair is grown in 1 common plot.

**Instance:** 7 plots, 7 grains, 3 grains/plot, 3 plots/grain, balance 1.

General term: balanced incomplete block design (BIBD).
### Example (Agricultural experiment design, AED)

<table>
<thead>
<tr>
<th></th>
<th>plot1</th>
<th>plot2</th>
<th>plot3</th>
<th>plot4</th>
<th>plot5</th>
<th>plot6</th>
<th>plot7</th>
</tr>
</thead>
<tbody>
<tr>
<td>barley</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>corn</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>millet</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>oats</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rye</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>spelt</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>wheat</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**Constraints to be satisfied:**

1. **Equal growth load:** Every plot grows 3 grains.
2. **Equal sample size:** Every grain is grown in 3 plots.
3. **Balance:** Every grain pair is grown in 1 common plot.

**Instance:** 7 plots, 7 grains, 3 grains/plot, 3 plots/grain, balance 1.

**General term:** balanced incomplete block design (BIBD).
In a BIBD, the plots are blocks and the grains are varieties:

Example (BIBD integer model: ✓ ~→ 1 and − ~~ 0)

```plaintext
enum Varieties; enum Blocks;
int: blockSize; int: sampleSize; int: balance;
array[Varieties,Blocks] of var 0..1: BIBD;
solve satisfy;
constraint forall(b in Blocks)
  (blockSize = count(BIBD[..,b], 1));
constraint forall(v in Varieties)
  (sampleSize = count(BIBD[v,..], 1));
constraint forall(v, w in Varieties where v < w)
  (balance = count([BIBD[v,b]*BIBD[w,b] | b in Blocks], 1));
```

Example (Instance data for our AED)

```plaintext
Varieties = {barley,...,wheat}; Blocks = {plot1,...,plot7};
blockSize = 3; sampleSize = 3; balance = 1;
```
Store after filling the first four rows

Example (BIBD *integer* model)

<table>
<thead>
<tr>
<th></th>
<th>plot1</th>
<th>plot2</th>
<th>plot3</th>
<th>plot4</th>
<th>plot5</th>
<th>plot6</th>
<th>plot7</th>
</tr>
</thead>
<tbody>
<tr>
<td>barley</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>corn</td>
<td>✓</td>
<td>-</td>
<td>-</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>millet</td>
<td>✓</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>oats</td>
<td>-</td>
<td>✓</td>
<td>-</td>
<td>✓</td>
<td>-</td>
<td>✓</td>
<td>-</td>
</tr>
<tr>
<td>rye</td>
<td>?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>spelt</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>wheat</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Store after filling the first four rows

<table>
<thead>
<tr>
<th></th>
<th>plot1</th>
<th>plot2</th>
<th>plot3</th>
<th>plot4</th>
<th>plot5</th>
<th>plot6</th>
<th>plot7</th>
</tr>
</thead>
<tbody>
<tr>
<td>barley</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>corn</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>✓</td>
<td>✓</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>millet</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>oats</td>
<td>–</td>
<td>✓</td>
<td>–</td>
<td>✓</td>
<td>–</td>
<td>✓</td>
<td>–</td>
</tr>
<tr>
<td>rye</td>
<td>?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>spelt</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>wheat</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

But plot1 **cannot** grow rye as that would violate the first constraint (every plot grows 3 grains).
# Example (BIBD *integer* model)

<table>
<thead>
<tr>
<th></th>
<th>plot1</th>
<th>plot2</th>
<th>plot3</th>
<th>plot4</th>
<th>plot5</th>
<th>plot6</th>
<th>plot7</th>
</tr>
</thead>
<tbody>
<tr>
<td>barley</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>corn</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>✓</td>
<td>✓</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>millet</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>oats</td>
<td>–</td>
<td>✓</td>
<td>–</td>
<td>✓</td>
<td>–</td>
<td>✓</td>
<td>–</td>
</tr>
<tr>
<td>rye</td>
<td>–</td>
<td>✓</td>
<td>–</td>
<td>✓</td>
<td>–</td>
<td>✓</td>
<td>–</td>
</tr>
<tr>
<td>spelt</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>wheat</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

But plot1 **cannot** grow rye as that would violate the first constraint (every plot grows 3 grains).
Store after filling the first four rows

Example (BIBD integer model)

<table>
<thead>
<tr>
<th></th>
<th>plot1</th>
<th>plot2</th>
<th>plot3</th>
<th>plot4</th>
<th>plot5</th>
<th>plot6</th>
<th>plot7</th>
</tr>
</thead>
<tbody>
<tr>
<td>barley</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>corn</td>
<td>✔</td>
<td>-</td>
<td>-</td>
<td>✔</td>
<td>✔</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>millet</td>
<td>✔</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>oats</td>
<td>-</td>
<td>✔</td>
<td>-</td>
<td>✔</td>
<td>-</td>
<td>✔</td>
<td>-</td>
</tr>
<tr>
<td>rye</td>
<td>-</td>
<td>✔</td>
<td>-</td>
<td>✔</td>
<td>-</td>
<td>✔</td>
<td>-</td>
</tr>
<tr>
<td>spelt</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>wheat</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

But plot1 cannot grow rye as that would violate the first constraint (every plot grows 3 grains). Actually, plot1 cannot grow oats, spelt, or wheat either, for the same reason, and this was already propagated when trying the search guess that plot1 grow millet!
Store after filling the first four rows

Example (BIBD integer model)

<table>
<thead>
<tr>
<th></th>
<th>plot1</th>
<th>plot2</th>
<th>plot3</th>
<th>plot4</th>
<th>plot5</th>
<th>plot6</th>
<th>plot7</th>
</tr>
</thead>
<tbody>
<tr>
<td>barley</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>corn</td>
<td>✓</td>
<td>-</td>
<td>-</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>millet</td>
<td>✓</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>oats</td>
<td>-</td>
<td>✓</td>
<td>-</td>
<td>✓</td>
<td>-</td>
<td>✓</td>
<td>-</td>
</tr>
<tr>
<td>rye</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
</tr>
<tr>
<td>spelt</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>wheat</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

But plot1 cannot grow rye as that would violate the first constraint (every plot grows 3 grains). Actually, plot1 cannot grow oats, spelt, or wheat either, for the same reason, and this was already propagated when trying the search guess that plot1 grow millet!
Continuing . . .

### Example (BIBD integer model)

<table>
<thead>
<tr>
<th></th>
<th>plot1</th>
<th>plot2</th>
<th>plot3</th>
<th>plot4</th>
<th>plot5</th>
<th>plot6</th>
<th>plot7</th>
</tr>
</thead>
<tbody>
<tr>
<td>barley</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>corn</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>✓</td>
<td>✓</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>millet</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>oats</td>
<td>–</td>
<td>✓</td>
<td>–</td>
<td>✓</td>
<td>–</td>
<td>✓</td>
<td>–</td>
</tr>
<tr>
<td>rye</td>
<td>–</td>
<td>?</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>spelt</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>wheat</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>
Continuing ... 

Example (BIBD *integer* model)

<table>
<thead>
<tr>
<th></th>
<th>plot1</th>
<th>plot2</th>
<th>plot3</th>
<th>plot4</th>
<th>plot5</th>
<th>plot6</th>
<th>plot7</th>
</tr>
</thead>
<tbody>
<tr>
<td>barley</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>corn</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>✓</td>
<td>✓</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>millet</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>oats</td>
<td>–</td>
<td>✓</td>
<td>–</td>
<td>✓</td>
<td>–</td>
<td>✓</td>
<td>–</td>
</tr>
<tr>
<td>rye</td>
<td>–</td>
<td>?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>spelt</td>
<td>–</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>wheat</td>
<td>–</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Guess:** Let plot2 grow rye. **Strategy:** ✓ guesses first.
Continuing . . .

Example (BIBD integer model)

<table>
<thead>
<tr>
<th></th>
<th>plot1</th>
<th>plot2</th>
<th>plot3</th>
<th>plot4</th>
<th>plot5</th>
<th>plot6</th>
<th>plot7</th>
</tr>
</thead>
<tbody>
<tr>
<td>barley</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>corn</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>millet</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>oats</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>rye</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>spelt</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>wheat</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

**Guess:** Let plot2 grow rye. **Strategy:** ✓ guesses first.
### Example (BIBD integer model)

<table>
<thead>
<tr>
<th></th>
<th>plot1</th>
<th>plot2</th>
<th>plot3</th>
<th>plot4</th>
<th>plot5</th>
<th>plot6</th>
<th>plot7</th>
</tr>
</thead>
<tbody>
<tr>
<td>barley</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>corn</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>✓</td>
<td>✓</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>millet</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>oats</td>
<td>–</td>
<td>✓</td>
<td>–</td>
<td>✓</td>
<td>–</td>
<td>✓</td>
<td>–</td>
</tr>
<tr>
<td>rye</td>
<td>–</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>spelt</td>
<td>–</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>wheat</td>
<td>–</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Propagation: plot2 cannot grow spelt and wheat as otherwise the first constraint (every plot grows 3 grains) would be violated for plot2.
Continuing ...

**Example (BIBD integer model)**

<table>
<thead>
<tr>
<th></th>
<th>plot1</th>
<th>plot2</th>
<th>plot3</th>
<th>plot4</th>
<th>plot5</th>
<th>plot6</th>
<th>plot7</th>
</tr>
</thead>
<tbody>
<tr>
<td>barley</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>corn</td>
<td>✓</td>
<td>-</td>
<td>-</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>millet</td>
<td>✓</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>oats</td>
<td>-</td>
<td>✓</td>
<td>-</td>
<td>✓</td>
<td>-</td>
<td>✓</td>
<td>-</td>
</tr>
<tr>
<td>rye</td>
<td>-</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>spelt</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>wheat</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Propagation:** plot2 cannot grow spelt and wheat as otherwise the first constraint (every plot grows 3 grains) would be violated for plot2.
Continuing . . .

### Example (BIBD integer model)

<table>
<thead>
<tr>
<th></th>
<th>plot1</th>
<th>plot2</th>
<th>plot3</th>
<th>plot4</th>
<th>plot5</th>
<th>plot6</th>
<th>plot7</th>
</tr>
</thead>
<tbody>
<tr>
<td>barley</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>corn</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>✓</td>
<td>✓</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>millet</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>oats</td>
<td>–</td>
<td>✓</td>
<td>–</td>
<td>✓</td>
<td>–</td>
<td>✓</td>
<td>–</td>
</tr>
<tr>
<td>rye</td>
<td>–</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>spelt</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>wheat</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

**Propagation:** plot3, plot4, and plot6 *cannot* grow rye as otherwise the third constraint (every grain pair is grown in 1 common plot) would be violated.
Continuing ... 

Example (BIBD *integer* model)

<table>
<thead>
<tr>
<th></th>
<th>plot1</th>
<th>plot2</th>
<th>plot3</th>
<th>plot4</th>
<th>plot5</th>
<th>plot6</th>
<th>plot7</th>
</tr>
</thead>
<tbody>
<tr>
<td>barley</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>corn</td>
<td>✓</td>
<td>—</td>
<td>—</td>
<td>✓</td>
<td>✓</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>millet</td>
<td>✓</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>oats</td>
<td>—</td>
<td>✓</td>
<td>—</td>
<td>✓</td>
<td>—</td>
<td>✓</td>
<td>—</td>
</tr>
<tr>
<td>rye</td>
<td>—</td>
<td>✓</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>spelt</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>wheat</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

Propagation: plot3, plot4, and plot6 cannot grow rye as otherwise the third constraint (every grain pair is grown in 1 common plot) would be violated.
Example (BIBD *integer* model)

<table>
<thead>
<tr>
<th></th>
<th>plot1</th>
<th>plot2</th>
<th>plot3</th>
<th>plot4</th>
<th>plot5</th>
<th>plot6</th>
<th>plot7</th>
</tr>
</thead>
<tbody>
<tr>
<td>barley</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>corn</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>✓</td>
<td>✓</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>millet</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>oats</td>
<td>–</td>
<td>✓</td>
<td>–</td>
<td>✓</td>
<td>–</td>
<td>✓</td>
<td>–</td>
</tr>
<tr>
<td>rye</td>
<td>–</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>spelt</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>wheat</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

**Propagation:** plot5 and plot7 must grow rye as otherwise the second constraint (every grain is grown in 3 plots) would be violated for rye.
Continuing …

Example (BIBD *integer* model)

<table>
<thead>
<tr>
<th></th>
<th>plot1</th>
<th>plot2</th>
<th>plot3</th>
<th>plot4</th>
<th>plot5</th>
<th>plot6</th>
<th>plot7</th>
</tr>
</thead>
<tbody>
<tr>
<td>barley</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>corn</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>✓</td>
<td>✓</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>millet</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>oats</td>
<td>–</td>
<td>✓</td>
<td>–</td>
<td>✓</td>
<td>–</td>
<td>✓</td>
<td>–</td>
</tr>
<tr>
<td>rye</td>
<td>–</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>✓</td>
<td>–</td>
<td>✓</td>
</tr>
<tr>
<td>spelt</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>✓</td>
</tr>
<tr>
<td>wheat</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

Propagation: plot5 and plot7 *must* grow rye as otherwise the second constraint (every grain is grown in 3 plots) would be violated for rye.
Example (BIBD integer model)

<table>
<thead>
<tr>
<th></th>
<th>plot1</th>
<th>plot2</th>
<th>plot3</th>
<th>plot4</th>
<th>plot5</th>
<th>plot6</th>
<th>plot7</th>
</tr>
</thead>
<tbody>
<tr>
<td>barley</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>corn</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>millet</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>oats</td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>rye</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>spelt</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>wheat</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>

Propagation: plot3 must grow spelt and wheat as otherwise the first constraint (every plot grows 3 grains) would be violated for plot3.
Continuing . . .

### Example (BIBD *integer* model)

<table>
<thead>
<tr>
<th></th>
<th>plot1</th>
<th>plot2</th>
<th>plot3</th>
<th>plot4</th>
<th>plot5</th>
<th>plot6</th>
<th>plot7</th>
</tr>
</thead>
<tbody>
<tr>
<td>barley</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>corn</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>✓</td>
<td>✓</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>millet</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>oats</td>
<td>–</td>
<td>✓</td>
<td>–</td>
<td>✓</td>
<td>–</td>
<td>✓</td>
<td>–</td>
</tr>
<tr>
<td>rye</td>
<td>–</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>✓</td>
<td>–</td>
<td>✓</td>
</tr>
<tr>
<td>spelt</td>
<td>–</td>
<td>–</td>
<td>✓</td>
<td>–</td>
<td>✓</td>
<td>–</td>
<td>✓</td>
</tr>
<tr>
<td>wheat</td>
<td>–</td>
<td>–</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Propagation:** plot3 must grow spelt and wheat as otherwise the first constraint (every plot grows 3 grains) would be violated for plot3.
Continuing ... 

Example (BIBD *integer* model)

<table>
<thead>
<tr>
<th></th>
<th>plot1</th>
<th>plot2</th>
<th>plot3</th>
<th>plot4</th>
<th>plot5</th>
<th>plot6</th>
<th>plot7</th>
</tr>
</thead>
<tbody>
<tr>
<td>barley</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>corn</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>millet</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>oats</td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>rye</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>spelt</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>wheat</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>

**Propagation:** No more propagation possible.
Continuing …

Example (BIBD integer model)

<table>
<thead>
<tr>
<th></th>
<th>plot1</th>
<th>plot2</th>
<th>plot3</th>
<th>plot4</th>
<th>plot5</th>
<th>plot6</th>
<th>plot7</th>
</tr>
</thead>
<tbody>
<tr>
<td>barley</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>corn</td>
<td>✓</td>
<td>-</td>
<td>-</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>millet</td>
<td>✓</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>oats</td>
<td>-</td>
<td>✓</td>
<td>-</td>
<td>✓</td>
<td>-</td>
<td>✓</td>
<td>-</td>
</tr>
<tr>
<td>rye</td>
<td>-</td>
<td>✓</td>
<td>-</td>
<td>-</td>
<td>✓</td>
<td>-</td>
<td>✓</td>
</tr>
<tr>
<td>spelt</td>
<td>-</td>
<td>-</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
<td>-</td>
<td>✓</td>
</tr>
<tr>
<td>wheat</td>
<td>-</td>
<td>-</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>


Propagation: etc.
Outline

1. Intuition
   Example 1
   Example 2
   Example 3

2. Algorithms
   Reminders of Discrete Mathematics
   Solving: Overview
   Propagator for a Constraint
   Fixpoint of Multiple Propagators
Example (Propagation to *Domain* Consistency)

Find $a \in \{1, 2, \ldots, 9\}$ and $b \in \{0, 1, \ldots, 8\}$ such that

\[
\begin{align*}
2 \cdot a + 4 \cdot b &= 24 \\
a + b &= 9
\end{align*}
\]

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$b$</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
</tbody>
</table>
Example (Propagation to Domain Consistency)

Find $a \in \{1, 2, \ldots, 9\}$ and $b \in \{0, 1, \ldots, 8\}$ such that

\[
2 \cdot a + 4 \cdot b = 24
\]
\[
a + b = 9
\]

State $2 \cdot a + 4 \cdot b = 24$: prune unsupported values of $a$:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$b$</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
</tbody>
</table>
### Example (Propagation to Domain Consistency)

Find $a \in \{1, 2, \ldots, 9\}$ and $b \in \{0, 1, \ldots, 8\}$ such that

\[
2 \cdot a + 4 \cdot b = 24 \\
a + b = 9
\]

State $2 \cdot a + 4 \cdot b = 24$: prune unsupported values of $a$:

<table>
<thead>
<tr>
<th>$a$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b$</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>
Example (Propagation to *Domain* Consistency)

Find $a \in \{1, 2, \ldots, 9\}$ and $b \in \{0, 1, \ldots, 8\}$ such that

$$2 \cdot a + 4 \cdot b = 24$$

$$a + b = 9$$

State $2 \cdot a + 4 \cdot b = 24$: prune unsupported values of $b$:

<table>
<thead>
<tr>
<th>a</th>
<th>2</th>
<th>4</th>
<th>6</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

- 10 -
Example (Propagation to *Domain* Consistency)

Find $a \in \{1, 2, \ldots, 9\}$ and $b \in \{0, 1, \ldots, 8\}$ such that

\[
2 \cdot a + 4 \cdot b = 24
\]

\[
a + b = 9
\]

State $2 \cdot a + 4 \cdot b = 24$: prune unsupported values of $b$:

<table>
<thead>
<tr>
<th>$a$</th>
<th>2</th>
<th>4</th>
<th>6</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b$</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>
Example (Propagation to *Domain* Consistency)

Find $a \in \{1, 2, \ldots, 9\}$ and $b \in \{0, 1, \ldots, 8\}$ such that

\[
2 \cdot a + 4 \cdot b = 24 \\
a + b = 9
\]

<table>
<thead>
<tr>
<th></th>
<th>2</th>
<th>4</th>
<th>6</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>a</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>b</strong></td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

Keep propagator for $2 \cdot a + 4 \cdot b = 24$, as not subsumed: its constraint is not definitely true under the current store.
Example (Propagation to Domain Consistency)

Find \( a \in \{1, 2, \ldots, 9\} \) and \( b \in \{0, 1, \ldots, 8\} \) such that

\[
2 \cdot a + 4 \cdot b = 24
\]
\[
a + b = 9
\]

State \( a + b = 9 \): prune unsupported values of \( a \):

\[
\begin{array}{cccccc}
  & 2 & 4 & 6 & 8 \\
\hline
a & | & | & | & |
\hline
b & 2 & 3 & 4 & 5
\end{array}
\]
Example (Propagation to Domain Consistency)

Find \( a \in \{1, 2, \ldots, 9\} \) and \( b \in \{0, 1, \ldots, 8\} \) such that

\[
2 \cdot a + 4 \cdot b = 24 \\
\]

\[
a + b = 9
\]

State \( a + b = 9 \): prune unsupported values of \( a \):

\[
\begin{array}{c|cccccc}
   a & \phantom{2}2 & \phantom{2}4 & \phantom{2}6 & \phantom{2}8 \\
----&----&----&----&----
   b & \phantom{2}2 & \phantom{2}3 & \phantom{2}4 & \phantom{2}5 \\
\end{array}
\]
**Example (Propagation to *Domain* Consistency)**

Find $a \in \{1, 2, \ldots, 9\}$ and $b \in \{0, 1, \ldots, 8\}$ such that

\[
2 \cdot a + 4 \cdot b = 24 \\
2 \cdot a + b = 9
\]

State $a + b = 9$: prune unsupported values of $b$:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th>4</th>
<th>6</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>
Example (Propagation to Domain Consistency)

Find $a \in \{1, 2, \ldots, 9\}$ and $b \in \{0, 1, \ldots, 8\}$ such that

\[
2 \cdot a + 4 \cdot b = 24
\]
\[
a + b = 9
\]

State $a + b = 9$: prune unsupported values of $b$:

<table>
<thead>
<tr>
<th>$a$</th>
<th></th>
<th>4</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b$</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>
### Example (Propagation to Domain Consistency)

Find $a \in \{1, 2, \ldots, 9\}$ and $b \in \{0, 1, \ldots, 8\}$ such that

\[
\begin{align*}
2 \cdot a + 4 \cdot b &= 24 \\
a + b &= 9
\end{align*}
\]

<table>
<thead>
<tr>
<th>$a$</th>
<th></th>
<th>4</th>
<th>6</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$b$</td>
<td>3</td>
<td>5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Keep propagator for $a + b = 9$, as not subsumed: its constraint is not definitely true under the current store.
Example (Propagation to *Domain* Consistency)

Find \( a \in \{1, 2, \ldots, 9\} \) and \( b \in \{0, 1, \ldots, 8\} \) such that

\[
\begin{align*}
2 \cdot a + 4 \cdot b &= 24 \\
a + b &= 9
\end{align*}
\]

Run \( 2 \cdot a + 4 \cdot b = 24 \): prune unsupported values of \( a \):

<table>
<thead>
<tr>
<th>( a )</th>
<th>( b )</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
</tr>
</tbody>
</table>
Example (Propagation to Domain Consistency)

Find \( a \in \{1, 2, \ldots, 9\} \) and \( b \in \{0, 1, \ldots, 8\} \) such that

\[
\begin{align*}
2 \cdot a + 4 \cdot b &= 24 \\
        a + b  &=  9
\end{align*}
\]

Run \(2 \cdot a + 4 \cdot b = 24\): prune unsupported values of \( a \):

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>4</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>3</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>
Problem, Model, and Propagation

**Example (Propagation to Domain Consistency)**

Find $a \in \{1, 2, \ldots, 9\}$ and $b \in \{0, 1, \ldots, 8\}$ such that

\[
2 \cdot a + 4 \cdot b = 24 \\
\quad a + b = 9
\]

Run $2 \cdot a + 4 \cdot b = 24$: prune unsupported values of $b$:

<table>
<thead>
<tr>
<th>$a$</th>
<th></th>
<th></th>
<th></th>
<th>6</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$b$</td>
<td></td>
<td></td>
<td>3</td>
<td>5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example (Propagation to Domain Consistency)

Find $a \in \{1, 2, \ldots, 9\}$ and $b \in \{0, 1, \ldots, 8\}$ such that

$$2 \cdot a + 4 \cdot b = 24$$
$$a + b = 9$$

Run $2 \cdot a + 4 \cdot b = 24$: prune unsupported values of $b$:

<table>
<thead>
<tr>
<th>$a$</th>
<th></th>
<th></th>
<th></th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b$</td>
<td>3</td>
<td>5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Find $a \in \{1, 2, \ldots, 9\}$ and $b \in \{0, 1, \ldots, 8\}$ such that

\[
2 \cdot a + 4 \cdot b = 24 \\
a + b = 9
\]

Dispose of propagator for $2 \cdot a + 4 \cdot b = 24$, as subsumed: its constraint is definitely true under the current store.
Example (Propagation to *Domain* Consistency)

Find $a \in \{1, 2, \ldots, 9\}$ and $b \in \{0, 1, \ldots, 8\}$ such that

\[
2 \cdot a + 4 \cdot b = 24
\]
\[
a + b = 9
\]

Run $a + b = 9$: prune unsupported values of $a$:

<table>
<thead>
<tr>
<th>a</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- 10 -
Example (Propagation to *Domain* Consistency)

Find \( a \in \{1, 2, \ldots, 9\} \) and \( b \in \{0, 1, \ldots, 8\} \) such that

\[
2 \cdot a + 4 \cdot b = 24 \\
a + b = 9
\]

Run \( a + b = 9 \): prune unsupported values of \( b \):

\[
\begin{array}{cccccc}
a & & & & & 6 \\
b & & & & 3 & \\
\end{array}
\]
Example (Propagation to *Domain* Consistency)

Find \( a \in \{1, 2, \ldots, 9\} \) and \( b \in \{0, 1, \ldots, 8\} \) such that

\[
2 \cdot a + 4 \cdot b = 24 \\
a + b = 9
\]

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b</td>
<td></td>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Dispose of propagator for \( a + b = 9 \), as subsumed: its constraint is definitely true under the current store.
Example (Propagation to Domain Consistency)

Find $a \in \{1, 2, \ldots, 9\}$ and $b \in \{0, 1, \ldots, 8\}$ such that

\[
2 \cdot a + 4 \cdot b = 24 \\
6 + b = 9
\]

Here is a truth table:

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$b$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

No propagators are left: all solutions are found. No search!
Example (Propagation to *Domain* Consistency)

Find $a \in \{1, 2, \ldots, 9\}$ and $b \in \{0, 1, \ldots, 8\}$ such that

\[
2 \cdot a + 4 \cdot b = 24 \\
a + b = 9
\]

<table>
<thead>
<tr>
<th>$a$</th>
<th></th>
<th></th>
<th></th>
<th>6</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$b$</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

This general propagation method works for all systems of constraints (linear or not, equalities or inequalities, etc), no matter how many constraints and decision variables.
Outline

1. Intuition
   Example 1
   Example 2
   Example 3

2. Algorithms
   Reminders of Discrete Mathematics
   Solving: Overview
   Propagator for a Constraint
   Fixpoint of Multiple Propagators
Example (Propagation to *Bounds(\textasteriskcentered) Consistency*)

Find $a \in \{1, 2, \ldots, 9\}$ and $b \in \{0, 1, \ldots, 8\}$ such that

$$2 \cdot a + 4 \cdot b = 24$$

<table>
<thead>
<tr>
<th>$a$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b$</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
</tbody>
</table>
Problem, Model, and Propagation

Example (Propagation to $Bounds(\ast)$ Consistency)

Find $a \in \{1, 2, \ldots, 9\}$ and $b \in \{0, 1, \ldots, 8\}$ such that

$$2 \cdot a + 4 \cdot b = 24$$

State $2 \cdot a + 4 \cdot b = 24$: prune unsupported bounds of $a$:

<table>
<thead>
<tr>
<th>a</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
</tbody>
</table>
Example (Propagation to \textit{Bounds}(\ast) Consistency)

Find $a \in \{1, 2, \ldots, 9\}$ and $b \in \{0, 1, \ldots, 8\}$ such that

$$2 \cdot a + 4 \cdot b = 24$$

State $2 \cdot a + 4 \cdot b = 24$: prune unsupported bounds of $a$:

<table>
<thead>
<tr>
<th>$a$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b$</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
</tbody>
</table>
Example (Propagation to \textit{Bounds}(\ast) Consistency)

Find $a \in \{1, 2, \ldots, 9\}$ and $b \in \{0, 1, \ldots, 8\}$ such that

$$2 \cdot a + 4 \cdot b = 24$$

State $2 \cdot a + 4 \cdot b = 24$: prune unsupported bounds of $b$:

\[
\begin{array}{cccccccc}
a & & & & & & & \\
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
\end{array}
\]

\[
\begin{array}{cccccccc}
b & & & & & & & \\
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
\end{array}
\]
Problem, Model, and Propagation

Example (Propagation to \textit{Bounds}(\ast) Consistency)

Find $a \in \{1, 2, \ldots, 9\}$ and $b \in \{0, 1, \ldots, 8\}$ such that

$$2 \cdot a + 4 \cdot b = 24$$

State $2 \cdot a + 4 \cdot b = 24$: prune unsupported bounds of $b$:

<table>
<thead>
<tr>
<th></th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td></td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>$b$</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example (Propagation to \textit{Bounds}(\ast) Consistency)

Find \( a \in \{1, 2, \ldots, 9\} \) and \( b \in \{0, 1, \ldots, 8\} \) such that
\[
2 \cdot a + 4 \cdot b = 24
\]

\[
\begin{array}{cccccccc}
 a & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
 b & 2 & 3 & 4 & 5 & & & \\
\end{array}
\]

Keep the propagator for \( 2 \cdot a + 4 \cdot b = 24 \), as \textit{not} subsumed.
Example (Propagation to \textit{Bounds}(\times) Consistency)

Find \( a \in \{1, 2, \ldots, 9\} \) and \( b \in \{0, 1, \ldots, 8\} \) such that

\[
2 \cdot a + 4 \cdot b = 24
\]

\[
\begin{array}{cccccccc}
a & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
b & 2 & 3 & 4 & 5 & & & \\
\end{array}
\]

Keep the propagator for \( 2 \cdot a + 4 \cdot b = 24 \), as not subsumed.

Some propagators are left: no solutions found yet. Search!
Outline

1. Intuition
   Example 1
   Example 2
   Example 3

2. Algorithms
   Reminders of Discrete Mathematics
   Solving: Overview
   Propagator for a Constraint
   Fixpoint of Multiple Propagators
Outline

1. Intuition
   Example 1
   Example 2
   Example 3

2. Algorithms
   Reminders of Discrete Mathematics
   Solving: Overview
   Propagator for a Constraint
   Fixpoint of Multiple Propagators
**Definition (Strict partial order)**

A **strict partial order** is a pair \( \langle X, \sqsubset \rangle \), where \( X \) is a set over which the binary relation \( \sqsubset \) is irreflexive (\( \forall x \in X : x \not\sqsubset x \)) and transitive (\( \forall x, y, z \in X : x \sqsubset y \wedge y \sqsubset z \Rightarrow x \sqsubset z \)).

**Example:** \((\mathbb{Z}, <)\) is a strict partial order.

**Definition (Well-founded order)**

A **well-founded order** is a strict partial order \( \langle X, \sqsubset \rangle \) in which there is no infinite decreasing sequence \( \cdots \sqsubset x_3 \sqsubset x_2 \sqsubset x_1 \).

**Examples:** \((\mathbb{N}, <)\); \((2^S, \subset)\) for a set \( S \); and loop variants.

**Definition (Lexicographic order)**

Given two well-founded orders \( \langle X, \sqsubset_X \rangle \) and \( \langle Y, \sqsubset_Y \rangle \), the **lexicographic order** \( \langle X \times Y, \sqsubset_{\text{lex}} \rangle \) is well-founded, where \( \langle x_1, y_1 \rangle \sqsubset_{\text{lex}} \langle x_2, y_2 \rangle \) iff either \( x_1 \sqsubset_X x_2 \) or \( x_1 = x_2 \wedge y_1 \sqsubset_Y y_2 \). Similarly for composing more than two orders.

**Examples:** \( \text{lex\_less} \) is \((\mathbb{N}^*, <_{\text{lex}})\); loop variant of slide 28.
Intuition

Example 1
Example 2
Example 3

Algorithms

Reminders of Discrete Mathematics
Solving: Overview
Propagator for a Constraint
Fixpoint of Multiple Propagators

Functions

Definition (Fixpoint)

A fixpoint of a function \( f : X \to X \) is an element \( x \in X \) that does not change under \( f \), that is \( f(x) = x \).

Example: A store of the set \( S \) of all possible stores can be a fixpoint of a propagator, which is a total function in \( S \to S \).

Idempotent functions compute fixpoints:

Definition (Idempotency)

A function \( f \) is idempotent iff it is equal to its composition with itself: \( \forall x : f(f(x)) = f(x) \).

Example: A propagator \( p : S \to S \) can be idempotent.
Outline

1. Intuition
   Example 1
   Example 2
   Example 3

2. Algorithms
   Reminders of Discrete Mathematics
   Solving: Overview
   Propagator for a Constraint
   Fixpoint of Multiple Propagators
Solving

Systematic search, for a satisfaction problem:

1: **propagate** all constraints; **backtrack** if empty domain
2: if only fixed variables, then show solution & **backtrack**
3: **while** there is at least one scheduled propagator **do**
4: select non-fixed variable \( v \) of current domain \( \text{dom}(v) \)
5: partition \( \text{dom}(v) \) using guesses (say \( v = d \) & \( v \neq d \), or \( v > d \) & \( v \leq d \), for a selected value \( d \in \text{dom}(v) \))
6: for each guess: **recurse** upon adding it as constraint

For an **optimisation problem**: before backtracking at line 2 add the constraint that any next solution must be better.

**Strategies:**

- Line 4: **variable selection**: smallest domain, . . .
- Line 5: **value selection**: maximum, median, . . .
- Line 5: **guess selection**: equality, bisection, . . .
- Tree **exploration**: depth-first search, . . .
Strength of Stores

**Definition (Store strength comparison, denoted \( s \prec t \))**

Store \( s \) is (strictly) stronger than store \( t \) if and only if \( s(v) \subseteq t(v) \) for every decision variable \( v \), and \( s(v) \subsetneq t(v) \) for at least one decision variable \( v \).

\((\mathcal{S}, \prec)\) is a well-founded (and hence partial) order.

**Example (Store strength comparison)**

Consider these stores for variables \( \{x, y\} \) over \( \{4, 5, 7\} \):

\[
\begin{align*}
s_1 &= \{x \mapsto \{4, 5\}, y \mapsto \{5, 7\}\} \\
s_2 &= \{x \mapsto \{5\}, y \mapsto \{5, 7\}\} \\
s_3 &= \{x \mapsto \{5, 7\}, y \mapsto \{4, 5, 7\}\}
\end{align*}
\]

Note: \( s_2 \prec s_1 \) and \( s_2 \prec s_3 \), but \( s_1 \) and \( s_3 \) are incomparable.
Outline

1. Intuition
   Example 1
   Example 2
   Example 3

2. Algorithms
   Reminders of Discrete Mathematics
   Solving: Overview
   Propagator for a Constraint
   Fixpoint of Multiple Propagators
Definition (Propagator)

A propagator $p_c$ for a constraint $c$ modifies a store so that:

- **Contracting**: The result store is stronger than or equal to ($\preceq$) the input store: $p_c(s) \preceq s$ or $p_c(s) = s$, for any $s$.

- **Correct**: Each solution to $c$ in the store remains there: $d \in s \Rightarrow d \in p_c(s)$, for any store $s$ and solution $d$ to $c$.

- **Checking**: For each solution to $c$, no domain is shrunk: $d \in c \iff p_c(s) = s$, for any fixed store $s$ denoting $d$.

- **Monotonic** (optional): Strength-ordered stores remain ordered: $s_1 \preceq s_2 \Rightarrow p_c(s_1) \preceq p_c(s_2)$, for any $s_1$ and $s_2$.

Example (Domain-consistency propagator for $x \leq y$)

$$p_{x \leq y}(s) = \left\{ \begin{array}{l}
x \mapsto \{ n \in s(x) \mid n \leq \max(s(y)) \}, \\
y \mapsto \{ n \in s(y) \mid n \geq \min(s(x)) \}
\end{array} \right\}$$

$$p_{x \leq y}(\{x \mapsto \{1, 3, 5, 9\}, y \mapsto \{0, 2, 4\}\}) = \{x \mapsto \{1, 3\}, y \mapsto \{2, 4\}\}$$
Motivation for Monotonicity

Counter-example

Consider the non-monotonic propagator for constraint \( c \)

\[
p_c(s) = \begin{cases} 
\text{if } s(x) = \{4, 5, 7\} & \text{then } \{x \mapsto \{4\}\} \\
\text{else } s
\end{cases}
\]

and the stores \( s_1 = \{x \mapsto \{4, 5\}\} \) and \( s_2 = \{x \mapsto \{4, 5, 7\}\} \):

\( s_1 \preceq s_2 \) but \( p_c(s_2) = \{x \mapsto \{4\}\} \preceq \{x \mapsto \{4, 5\}\} = p_c(s_1) \)

The result stores could also be incomparable; note that \( \prec \) and \( \preceq \) are partial ordering relations.

But propagation would be propagator-order-dependent:

\[
p_c(p_{x<7}(s_2)) = \{x \mapsto \{4, 5\}\} \neq \{x \mapsto \{4\}\} = p_{x<7}(p_c(s_2))
\]

This might lead to unexpected solver behaviour.
Consequences of Propagator Definition

- Property of propagation, if only monotonic propagators:
  - **Order independence:** Propagators may be invoked in any order: their weakest common fixpoint is unique. E.g., from \( \{ x, y \mapsto \{3, 4, 5\} \} \), the weakest fixpoint of \( p_{x \geq y} \) and \( p_{y > 3} \) is \( \{ x, y \mapsto \{4, 5\} \} \), whereas a strongest fixpoint is a solution store, such as \( \{ x, y \mapsto \{5\} \} \).

  Only this property depends on monotonicity.

- Properties of a propagator \( p_c \) for a constraint \( c \):
  - **Correctness:** Each monotonic propagator necessarily is correct, so the latter requirement does not have to be proven separately for a propagator proven monotonic.
  - **Non-solution identification:** For a non-solution to \( c \), the domain of some decision variable becomes empty.
**Idempotency of propagators is not required:**
Every DC propagator is idempotent; a BC propagator may be non-idempotent: see Ex. 2.9 on p. 19 of Course Notes.

**Terminology:**
In the literature, the deletion of domain values is also called pruning, filtering, contraction, or narrowing. If a domain loses its last value, then we say that there was a domain wipe-out, and the propagator must fail.

**Definition (Model)**
A model of a CSP $\langle V, U, C \rangle$ is a tuple $\langle V, U, P \rangle$, where $P$ is the set of propagators chosen for the constraints $C$. Similarly for a model of a COP.

For propagator algorithms, see Topic 16: Propagators.
Outline

1. Intuition
   Example 1
   Example 2
   Example 3

2. Algorithms
   Reminders of Discrete Mathematics
   Solving: Overview
   Propagator for a Constraint
   Fixpoint of Multiple Propagators
Naïve Fixpoint Algorithm

Let \( \langle V, U, P[, f] \rangle \) be a model where there is a common domain \( U \) for all variables of \( V \), without loss of generality.

Let \( s_0 = \{ v \mapsto U \mid v \in V \} \) be the initial store, where every decision variable \( v \) of \( V \) is mapped to the universe \( U \).

Call to build the root of the search tree: Propagate\((P, s_0)\).

```plaintext
function Propagate\((R, s)\)
while \( \exists q \in R : q(s) \not\subseteq s \) do
  select \( q \in R : q(s) \not\subseteq s \)
  \( s := q(s) \)
return \( s \) // post: \( s \) is a common fixpoint of \( R \)
```
Toward More Realistic Propagation

Why is the previous algorithm naïve?
For the condition of its `while` loop:

- We may examine a propagator that does not depend in some sense on the propagator that was just run.
- We do not maintain the set of propagators that are known to be at fixpoint.

So we may examine a propagator that cannot prune values.

Variables of a propagator:
Let \( \text{var}(p) \) denote the set of decision variables of the constraint implemented by propagator \( p \):

- Running \( p \) has no effect on \( \text{dom}(v) \), for \( v \in V \setminus \text{var}(p) \).
- Running \( p \) is independent of \( \text{dom}(v) \), for \( v \in V \setminus \text{var}(p) \).
Variable-Directed Fixpoint Algorithm

Call to build the root of the search tree: Propagate\((P, P, s_0)\).

```plaintext
function Propagate\((R, Q, s)\) // \(R = \) all prop.s; \textbf{pre:} \(Q \subseteq R\)
while \(Q \neq \emptyset\) do // \textbf{invariant:} every \(p \in R \setminus Q\) is at fixpt
    // \textbf{variant:} \(\langle s, |Q| \rangle\)
    select \(q \in Q\) // prop.s of \(Q\) are possibly not at fixpt
    \(Q := Q \setminus \{q\}\)
    \(s' := q(s)\) // \(s' \preceq s\)
    \(\text{ModVars} := \{v \in \text{var}(q) \mid s(v) \neq s'(v)\}\)
    \(\text{DepProps} := \{p \in R \mid \exists v \in \text{var}(p) : v \in \text{ModVars}\}\)
    \(Q := Q \cup \text{DepProps} \) // maybe \(q \in Q\): optional idempot.
    \(s := s'\)
return \(s\) // \textbf{post:} \(s\) is a common fixpoint of \(R\)
```
Toward Further Improved Propagation

Propagators signal status to avoid some useless runs:

- Propagator $p$ is failed upon a domain wipe-out.
- Propagator $p$ is subsumed (or entailed) by store $s$ iff all stronger stores are fixpoints: $\forall s' \preceq s : p(s') = s'$. This status is an obligation when $s$ is a solution store. Such a propagator can safely be disposed of in the model.
- Otherwise, if so, ideally signal that $p$ is at fixpoint for $s$.
- It is always safe to signal that a propagator $p$ is possibly not at fixpoint for the result store $s$.

Examples (Subsumption)

$p_{x \leq y}$ is subsumed by $\{x \mapsto \{1, 3\}, y \mapsto \{3, 5\}\}$, but not by $\{x \mapsto \{1, 3, 4\}, y \mapsto \{3, 5\}\}$. A DC propagator of a unary constraint, like $x \in \{1, 3, 5\}$, is subsumed upon its first run.
Propagators with Status Message

Example (Domain-consistency propagator for $x \leq y$)

\[
p_{x\leq y}(s) = \text{let } s' = \begin{cases} 
  x \mapsto \{n \in s(x) \mid n \leq \max(s(y))\}, \\
  y \mapsto \{n \in s(y) \mid n \geq \min(s(x))\}
\end{cases} \text{ in }
\]

\[
\begin{align*}
\text{if } s'(x) &= \emptyset \lor s'(y) = \emptyset \text{ then } \langle \text{Failed}, \emptyset \rangle \\
\text{else if } \max(s'(x)) &\leq \min(s'(y)) \text{ then } \langle \text{Subsumed}, s' \rangle \\
\text{else } \langle \text{AtFixpt}, s' \rangle
\end{align*}
\]

Note that $\min(s(x))$ and $\max(s(y))$ do not change: hence $s'$ is at least a fixpoint for $p_{x\leq y}$ and at best subsumes it!

Responsibility:
The burden of signalling, in reasonable runtime, a proper status message is on the programmer of a propagator.
function Propagate(R, Q, s) // non-subsumed prop.s in R
while Q ≠ ∅ do // invariant: ... ; variant: ...
    select q ∈ Q
    Q := Q \ {q}
    ⟨m, s'⟩ := q(s)
    // s' ≤ s
if m = Failed then return ⟨R, ∅⟩ endif
if m = Subsumed then R := R \ {q} endif
ModVars := {v ∈ var(q) | s(v) ≠ s'(v)}
DepProps := {p ∈ R | ∃v ∈ var(p): v ∈ ModVars}
if m = AtFixpt then DepProps := DepProps \ {q} endif
Q := Q ∪ DepProps
s := s'
return ⟨R, s⟩ // post: s is a common fixpoint of R
Toward Even Further Improved Propagation

Signalling *how* domains were modified:
Mutually exclusive *modification events* for each variable \( v \):

1. **None(\( v \))**: the domain of \( v \) was not changed.
2. **Failed(\( v \))**: the domain of \( v \) was wiped out.
3. **Fixed(\( v \))**: the domain of \( v \) was pruned to a singleton.
4. **Min(\( v \))**: the lower bound of \( \text{dom}(v) \) was increased.
   **Max(\( v \))**: the upper bound of \( \text{dom}(v) \) was decreased.
5. **Any(\( v \))**: the domain of \( v \) was otherwise pruned.

Gecode: **Min(\( v \))** and **Max(\( v \))** are bundled into Bounded(\( v \)).

☞ It is often simple to decide whether a propagator remains at fixpoint depending on *how* another propagator prunes domains of decision variables they share: variable sharing is no longer the sole criterion for adding propagators to \( Q \).
**Propagator Conditions**

**Example (Domain-consistency propagator for \( x \leq y \))**

\[
p_{x \leq y}(s) = \begin{cases} 
  x & \mapsto \left\{ n \in s(x) \mid n \leq \max(s(y)) \right\}, \\
  y & \mapsto \left\{ n \in s(y) \mid n \geq \min(s(x)) \right\}
\end{cases}
\]

\[\text{PropConds}(p_{x \leq y}) = \{ \text{Min}(x), \text{Max}(y) \}\]

Promise: If the propagator is at fixpoint, then it will remain at fixpoint, unless \( \min(\text{dom}(x)) \) or \( \max(\text{dom}(y)) \) changes.

**Example (Domain-consistency propagator for \( x \neq y \))**

\[
p_{x \neq y}(s) = \begin{cases} 
  x & \mapsto s(x) \setminus \text{if } |s(y)| = 1 \text{ then } s(y) \text{ else } \emptyset, \\
  y & \mapsto s(y) \setminus \text{if } |s(x)| = 1 \text{ then } s(x) \text{ else } \emptyset
\end{cases}
\]

\[\text{PropConds}(p_{x \neq y}) = \{ \text{Fixed}(x), \text{Fixed}(y) \}\]

Promise: If the propagator is at fixpoint, then it will remain at fixpoint, unless \( \text{dom}(x) \) or \( \text{dom}(y) \) becomes a singleton.
Assumptions

Responsibilities, under Gecode:

- The programmer of propagator \( p \) states \( \text{PropConds}(p) \).
- The solver computes as follows the set \( \text{Conds}(s, s') \) of propagator conditions raised by applying a propagator \( q \) to a store \( s \), giving \( s' = q(s) \):

<table>
<thead>
<tr>
<th>Modification event</th>
<th>Conditions added to ( \text{Conds}(s, s') )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{Fixed}(v) )</td>
<td>( \text{Fixed}(v), \text{Bounded}(v), \text{Any}(v) )</td>
</tr>
<tr>
<td>( \text{Bounded}(v) )</td>
<td>( \text{Bounded}(v), \text{Any}(v) )</td>
</tr>
<tr>
<td>( \text{Any}(v) )</td>
<td>( \text{Any}(v) )</td>
</tr>
<tr>
<td>( \text{None}(v) )</td>
<td>( \text{None}(v) )</td>
</tr>
</tbody>
</table>

- The solver schedules a propagator \( p \) (adds \( p \) to \( Q \)) if the conditions \( \text{Conds}(s, s') \) raised by propagator \( q \) intersect with the propagator conditions \( \text{PropConds}(p) \).
Status-and-Condition-Directed Fixpt Algo.

\begin{verbatim}
function Propagate(R, Q, s)
while Q \neq \emptyset do // invariant: \ldots ; variant: \ldots
    select q \in Q
    Q := Q \setminus \{q\}
    \langle m, s' \rangle := q(s) // s' \preceq s
    if m = Failed then return \langle R, \emptyset \rangle endif
    if m = Subsumed then R := R \setminus \{q\} endif
    ModVars := \{v \in \text{var}(q) \mid s(v) \neq s'(v)\}
    DepProps := \{p \in R \mid \text{Conds}(s, s') \cap \text{PropConds}(p) \neq \emptyset\}
    if m = AtFixpt then DepProps := DepProps \setminus \{q\} endif
    Q := Q \cup DepProps
    s := s'
return \langle R, s \rangle // post: s is a common fixpoint of R
\end{verbatim}
Yet Further Optimisations

Priorities: The set $Q$ is implemented as a queue:
How to do “select $q \in Q$”?  
- According to cost: cheapest first  
- According to expected impact: highest impact first  
- In general: first-in first-out queue

Propagator rewriting:

Example

When all domain values for $x$ are smaller than those for $y$, then the propagator for $\max(x, y) = z$ can be replaced by the propagator for $y = z$.

Further reading:
For a more formal treatment of all these issues, including proofs, see the Course Notes.