PART II: Constraint Propagation & Global Constraints

Constraint Solver

- Enumerates all possible variable-value combinations via a systematic backtracking tree search.
  - Guesses a value for each variable.
- During search, examines the constraints to remove incompatible (inconsistent) values from the domains of the future (unexplored) variables, via propagation.
  - Shrinks the domains of the future variables.

Search and Propagation

- Search decisions and propagation are interleaved.

Propagation

$X_i \leftarrow v_j$

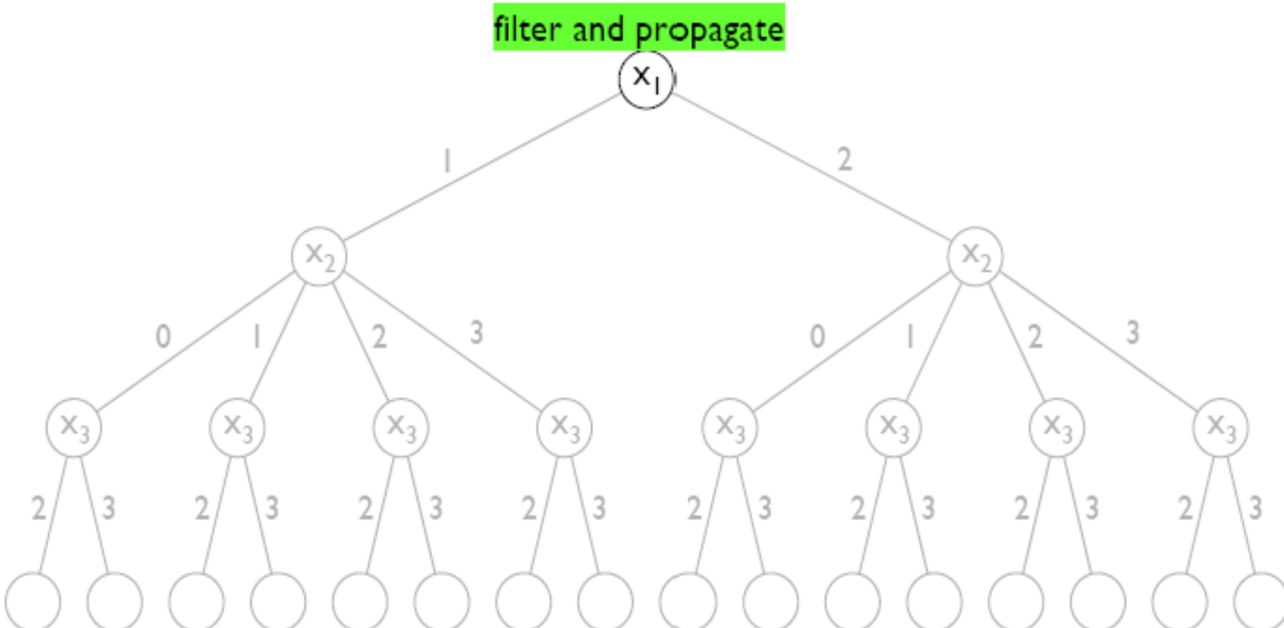
Propagation

$X_{i'} \leftarrow v_{j'}$

Propagation

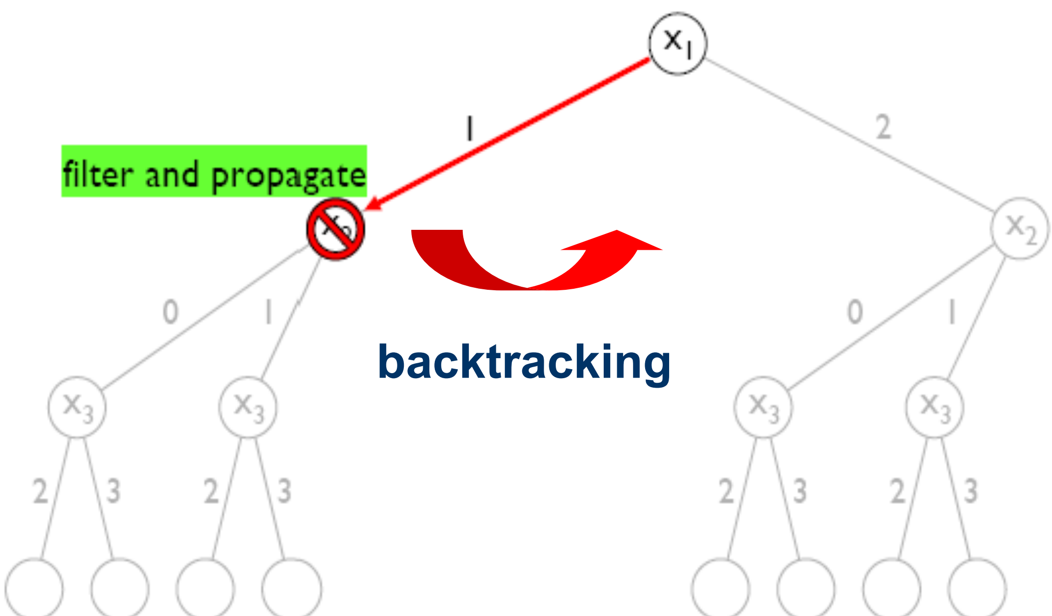
Backtracking Tree Search & Propagation

- $X_1 \in \{1,2\}$ $X_2 \in$ {0,1,~~2~~,~~3~~} $X_3 \in \{2,3\}$
- $X_1 > X_2$ and $X_1 + X_2 = X_3$ and alldifferent($[X_1, X_2, X_3]$)



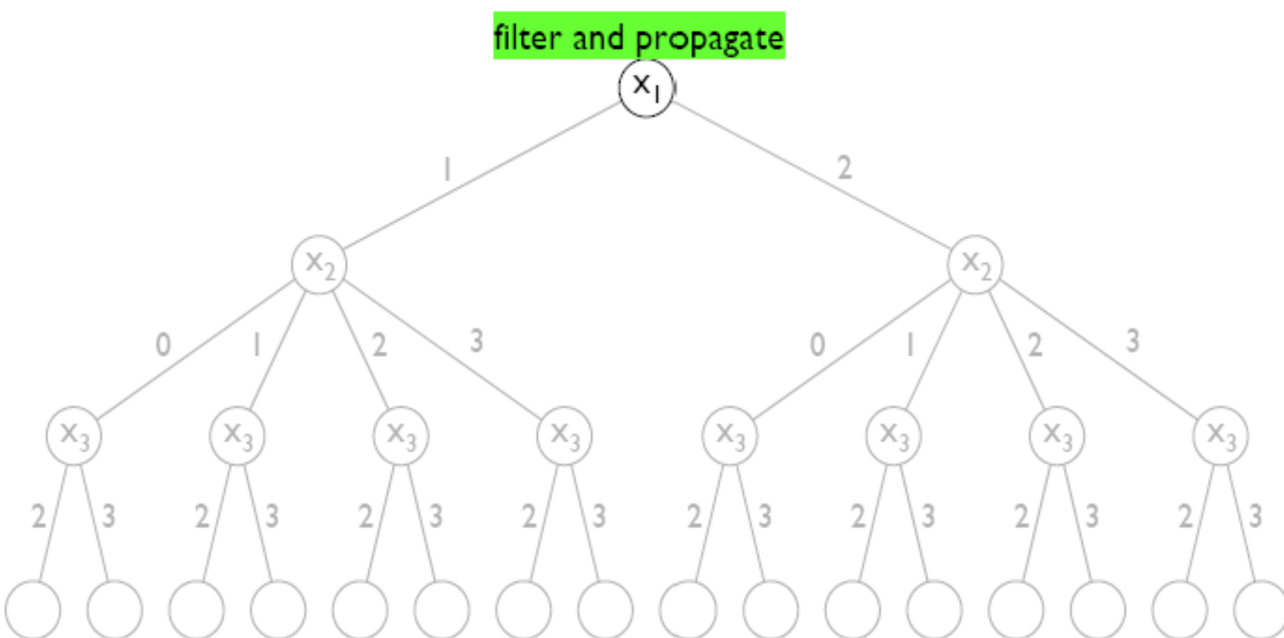
Backtracking Tree Search & Propagation

- $X_1 = 1$, $X_2 \in$ {0,~~1~~} $X_3 \in$ {~~2~~,~~3~~}
- $X_1 > X_2$ and $X_1 + X_2 = X_3$ and alldifferent($[X_1, X_2, X_3]$)



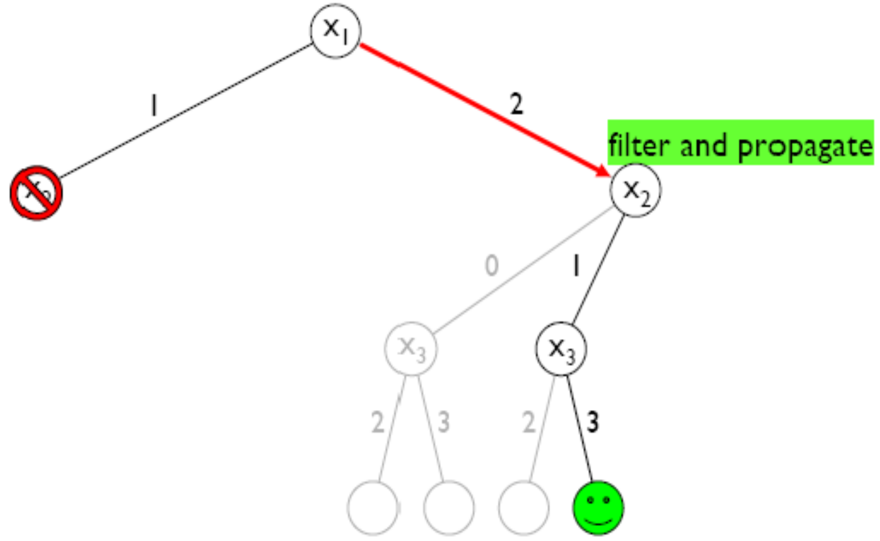
Backtracking Tree Search & Propagation

- $X_1 \in \{1,2\}$ $X_2 \in$ {0,1,~~2~~,~~3~~} $X_3 \in \{2,3\}$
- $X_1 > X_2$ and $X_1 + X_2 = X_3$ and alldifferent($[X_1, X_2, X_3]$)



Backtracking Tree Search & Propagation

- $X_1 = 2$ $X_2 \in \{0,1\}$ $X_3 \in \{2,3\}$
- $X_1 > X_2$ and $X_1 + X_2 = X_3$ and alldifferent($[X_1, X_2, X_3]$)



Outline

- Local Consistency
  - Generalized Arc Consistency (GAC)
  - Bounds Consistency (BC)
- Constraint Propagation
  - Propagation Algorithms
- Specialized Propagation
  - Global Constraints
- Global Constraints for Generic Purposes

Local Consistency

- A form of inference which detects inconsistent partial assignments.
  - Local, because we examine individual constraints.
- Popular local consistencies are domain-based:
  - Generalized Arc Consistency (GAC).
    - Also referred to as Hyper-arc or Domain Consistency;
  - Bounds Consistency (BC).
  - They detect inconsistent partial assignments of the form $X_i = j$, hence:
    - j can be removed from $D(X_i)$ via propagation;
    - propagation can be implemented easily.

Generalized Arc Consistency (GAC)

- A constraint C defined on k variables $C(X_1,\ldots, X_k)$ gives the set of allowed combinations of values (i.e. allowed tuples).
  - $C \subseteq D(X_1) \times \ldots \times D(X_k)$
  - E.g., $D(X_1) = \{0,1\}$, $D(X_2) = \{1,2\}$, $D(X_3) = \{2,3\}$ C: $X_1 + X_2 = X_3$

$$C(X_1,X_2,X_3) = \{(0,2,2), (1,1,2), (1,2,3)\}$$

↓

Each allowed tuple $(d_1,\ldots,d_k) \in C$ where $d_i \in X_i$ is a support for C.

GAC

- $C(X_1,\ldots, X_k)$ is GAC iff:
  - for all X_i in $\{X_1,\ldots, X_k\}$, for all $v \in D(X_i)$, v belongs to a support for C.
- Called Arc Consistency (AC) when k = 2.
- A CSP is GAC iff all its constraints are GAC.

Examples

- $D(X_1) = \{1,2,3\}$, $D(X_2) = \{2,3,4\}$, C: $X_1 = X_2$
  - AC(C)?
    - $1 \in D(X_1)$ and $4 \in D(X_2)$ do not have a support.
    - $X_1 = 1$ and $X_2 = 4$ are inconsistent partial assignments.

- $D(X_1) = \{1,2,3\}$, $D(X_2) = \{1,2\}$, $D(X_3) = \{1,2\}$, C: alldifferent($[X_1, X_2, X_3]$)
  - GAC(C)?
    - $1 \in D(X_1)$ and $2 \in D(X_1)$ do not have support.
    - $X_1 = 1$ and $X_1 = 2$ are inconsistent partial assignments.

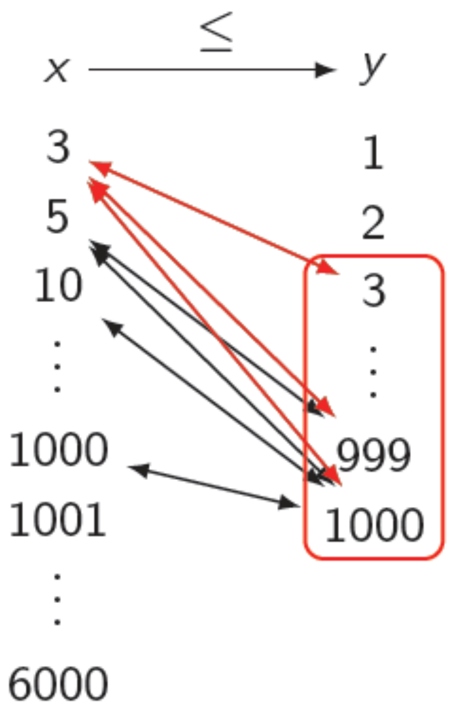
Bounds Consistency (BC)

- Defined for totally ordered (e.g. integer) domains.
- Relaxes the domain of X_i from $D(X_i)$ to $[min(X_i)..max(X_i)]$.
  - E.g., $D(X_i) = \{1,3,5\} \rightarrow [1..5]$
- A bound support is a tuple $(d_1,\ldots,d_k) \in C$ where $d_i \in [min(X_i)..max(Xi)]$.
- $C(X_1,\ldots, X_k)$ is BC iff:
  - For all X_i in $\{X_1,\ldots, X_k\}$, $min(X_i)$ and $max(X_i)$ belong to a bound support.

BC

- Disadvantage
  - BC might not detect all GAC inconsistencies in general.
    - We need to search more.
- Advantages
  - Might be easier to look for a support in a range than in a domain.
  - Achieving BC is often cheaper than achieving GAC.
    - Of interest in arithmetic constraints defined on integer variables with large domains.
  - Achieving BC is enough to achieve GAC for monotonic constraints.

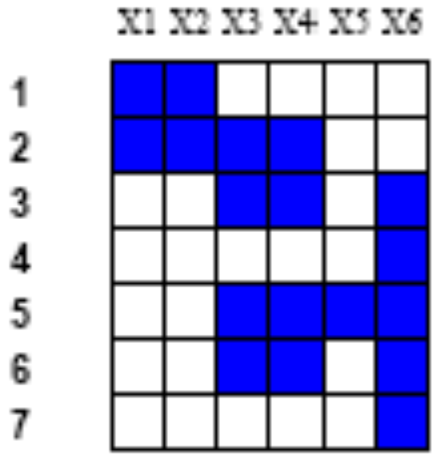
GAC = BC



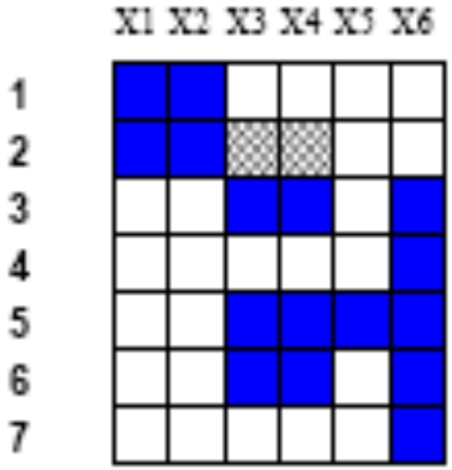
- All values of D(X) ≤ max(Y) are GAC.
- All values of D(Y) ≥ min(X) are GAC.
- Enough to adjust max(X) and min(Y).
  - max(X) ≤ max(Y)
  - min(X) ≤ min(Y)

GAC > BC

- $D(X_1) = D(X_2) = \{1,2\}$, $D(X_3) = D(X_4) = \{2,3,5,6\}$, $X_5 = 5$, $D(X_6) = \{3,4,5,6,7\}$, alldifferent($[X_1, X_2, X_3, X_4, X_5, X_6]$)
- Only $2 \in D(X_3)$ and $2 \in D(X_4)$ have no BC support.



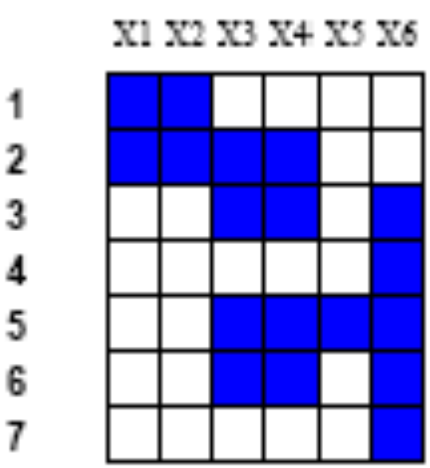
Original



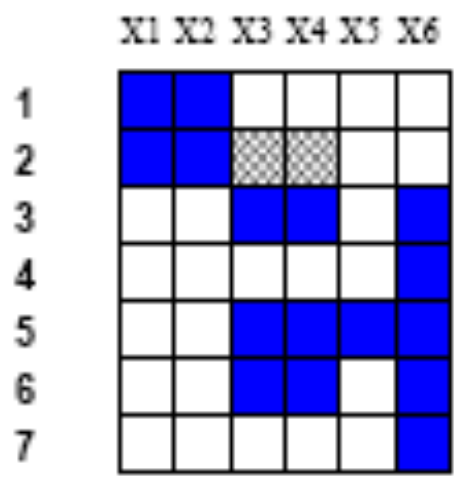
BC

GAC > BC

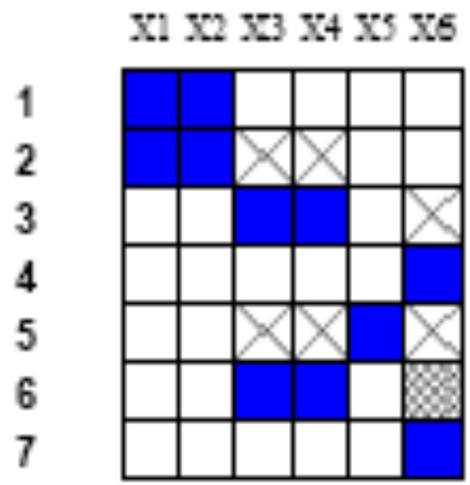
- $D(X_1) = D(X_2) = \{1,2\}$, $D(X_3) = D(X_4) = \{2,3,5,6\}$, $X_5 = 5$, $D(X_6) = \{3,4,5,6,7\}$, alldifferent($[X_1, X_2, X_3, X_4, X_5, X_6]$)
- $\{2,5\} \in D(X_3)$, $\{2,5\} \in D(X_4)$, $\{3,5,6\} \in D(X_6)$ have no GAC support.



Original



BC



GAC

Outline

- Local Consistency
  - Generalized Arc Consistency (GAC)
  - Bounds Consistency (BC)
- Constraint Propagation
  - Propagation Algorithms
- Specialized Propagation
  - Global Constraints
- Global Constraints for Generic Purposes

Constraint Propagation

- Can appear under different names:
  - constraint relaxation
  - filtering
  - local consistency enforcing, …
- A local consistency notion defines properties that a constraint C must satisfy after constraint propagation.
  - The operational behaviour is completely left open.
  - We can develop different algorithms with different complexities to achieve the same effect.
  - The only requirement is to achieve the required property on C.

Propagation Algorithms

- A propagation algorithm achieves a certain level of consistency on a constraint C by removing the inconsistent values from the domains of the variables in C.
- The level of consistency depends on C.
  - GAC if an efficient propagation algorithm can be developed.
  - Otherwise BC or a lower level of consistency.

Propagation Algorithms

- When solving a CSP with multiple constraints:
  - propagation algorithms interact;
  - a propagation algorithm can wake up an already propagated constraint to be propagated again!
  - in the end, propagation reaches a fixed-point and all constraints reach a level of consistency;
  - the whole process is referred as constraint propagation.

Example

- $D(X_1) = D(X_2) = D(X_3) = \{1,2,3\}$
  C_1: alldifferent($[X_1, X_2, X_3]$) C_2: $X_2 < 3$ C_3: $X_3 < 3$
- Let's assume:
  - the order of propagation is C_1, C_2, C_3;
  - propagation algorithms maintain (G)AC.
- Propagation of C_1:
  - nothing happens, C_1 is GAC.
- Propagation of C_2:
  - 3 is removed from $D(X_2)$, C_2 is now AC.
- Propagation of C_3:
  - 3 is removed from $D(X_3)$, C_3 is now AC.
- C_1 is not GAC anymore, because the supports of $\{1,2\} \in D(X_1)$ in $D(X_2)$ and $D(X_3)$ are removed by the propagation of C_2 and C_3.
- Re-propagation of C_1:
  - 1 and 2 are removed from $D(X_1)$, C_1 is now AC.

Properties of Propagation Algorithms

- It may not be enough to remove inconsistent values from domains once.
- A propagation algorithm must wake up again when necessary, otherwise may not achieve the desired local consistency property.
- Events that can trigger a constraint propagation:
  - when the domain of a variable changes (for GAC);
  - when the domain bounds of a variable changes (for BC);
  - when a variable is assigned a value;
  - …

Complexity of Propagation Algorithms

- Assume $|D(X_i)| = d$.
- Following the definition of the local consistency property:
  - one time AC propagation on a C(X_1,X_2) takes $O(d^2)$ time.
- We can do better!

Examples

- C: $X_1 = X_2$
  - $D(X_1) = D(X_2) = D(X_1) \cap D(X_2)$
  - Complexity: the cost of the set intersection operation
- C: $X_1 \neq X_2$
  - When $D(X_i) = \{v\}$, remove v from $D(X_j)$.
  - Complexity: O(1)
- C: $X_1 \leq X_2$
  - $\max(X_1) \leq \max(X_2)$, $\min(X_1) \leq \min(X_2)$
  - Complexity: O(1)

Outline

- Local Consistency
  - Generalized Arc Consistency (GAC)
  - Bounds Consistency (BC)
- Constraint Propagation
  - Propagation Algorithms
- Specialized Propagation
  - Global Constraints
    - Decompositions
    - Ad-hoc Algorithms
- Global Constraints for Generic Purposes

Specialized Propagation

- Propagation specific to a given constraint.
- Advantages
  - Exploits the constraint semantics.
  - Potentially much more efficient than a general propagation approach.

Specialized BC Propagation

- C: $X_1 = X_2 + X_3$
- Observation
  - $\min(X_1)$ cannot be smaller than $\min(X_2) + \min(X_3)$.
  - $\max(X_1)$ cannot be larger than $\max(X_2) + \max(X_3)$.
  - $\min(X_2)$ cannot be smaller than $\min(X_1) - \max(X_3)$.
  - $\max(X_2)$ cannot be larger than $\max(X_1) - \min(X_3)$.
  - X_3 analogous to X_2.
- BC propagation rules
  - $\max(X_1) \leq \max(X_2) + \max(X_3)$, $\min(X_1) \geq \min(X_2) + \min(X_3)$
  - $\max(X_2) \leq \max(X_1) - \min(X_3)$, $\min(X_2) \geq \min(X_1) - \max(X_3)$
  - Similarly for X_3

Example

- $D(X_1) = [4,9]$, $D(X_2) = [3,5]$, $D(X_3) = [2,3]$
  C: $X_1 = X_2 + X_3$

Example

- $D(X_1) = [5,8]$, $D(X_2) = [3,5]$, $D(X_3) = [2,3]$
  C: $X_1 = X_2 + X_3$
- Propagation
  - $\max(X_1) \leq \max(X_2) + \max(X_3)$, $\min(X_1) \geq \min(X_2) + \min(X_3)$

Example

- $D(X_1) = [5,8]$, $D(X_2) = [3,5]$, $D(X_3) = [2,3]$
  C: $X_1 = X_2 + X_3$
- Propagation
  - $\max(X_1) \le \max(X_2) + \max(X_3)$, $\min(X_1) \ge \min(X_2) + \min(X_3)$
  - $\max(X_2) \le \max(X_1) - \min(X_3)$, $\min(X_2) \ge \min(X_1) - \max(X_3)$
  - Similarly for X_3

Example

- $X_1 = 5$, $D(X_2) = [3,5]$, $D(X_3) = [2,3]$

  C: $X_1 = X_2 + X_3$
- Propagation
  - $\max(X_1) \leq \max(X_2) + \max(X_3)$, $\min(X_1) \geq \min(X_2) + \min(X_3)$
  - $\max(X_2) \leq \max(X_1) - \min(X_3)$, $\min(X_2) \geq \min(X_1) - \max(X_3)$
  - Similarly for X_3

Example

- $X_1 = 5$, $D(X_2) = [3]$, $D(X_3) = [2]$
  C: $X_1 = X_2 + X_3$
- Propagation
  - $\max(X_1) \le \max(X_2) + \max(X_3)$, $\min(X_1) \ge \min(X_2) + \min(X_3)$
  - $\max(X_2) \le \max(X_1) - \min(X_3)$, $\min(X_2) \ge \min(X_1) - \max(X_3)$
  - Similarly for X_3

Specialized Propagation

- Propagation specific to a given constraint.
- Advantages
  - Exploits the constraint semantics.
  - Potentially much more efficient than a general propagation approach.
- Disadvantages
  - Limited use.
  - Not always easy to develop one.
- Worth developing for recurring constraints.

Global Constraints

- Capture complex, non-binary and recurring combinatorial substructures arising in a variety of applications.
- Embed specialized propagation which exploits the substructure.

Benefits of Global Constraints

- Modelling benefits
  - Reduce the gap between the problem statement and the model.
  - May allow the expression of constraints that are otherwise not possible to state using primitive constraints (semantic).
- Solving benefits
  - Strong inference in propagation (operational).
  - Efficient propagation (algorithmic).