Constraint Propagation:

The Heart of Constraint Programming

Zeynep KIZILTAN
Department of Computer Science
University of Bologna

Email: zeynep@cs.unibo.it
URL: http://zeynep.web.cs.unibo.it/

What is it about?

- 4-6 hour lectures about constraint programming in general and constraint propagation in specific.
  - Part I: Overview of constraint programming
  - Part II: Constraint propagation
  - Part III: Some useful pointers
- Aim:
  - Teach the basics of constraint programming.
  - Emphasize the importance of constraint propagation.
  - Point out the advanced topics.
  - Inform about the literature.

Warning

- We will see how constraint programming works.
- No programming examples.

PART I: Overview of Constraint Programming

Outline

- Constraint Satisfaction Problems (CSPs)
- Constraint Programming (CP)
  - Modelling
  - Backtracking Tree Search
  - Local Consistency and Constraint Propagation

Constraints are everywhere!

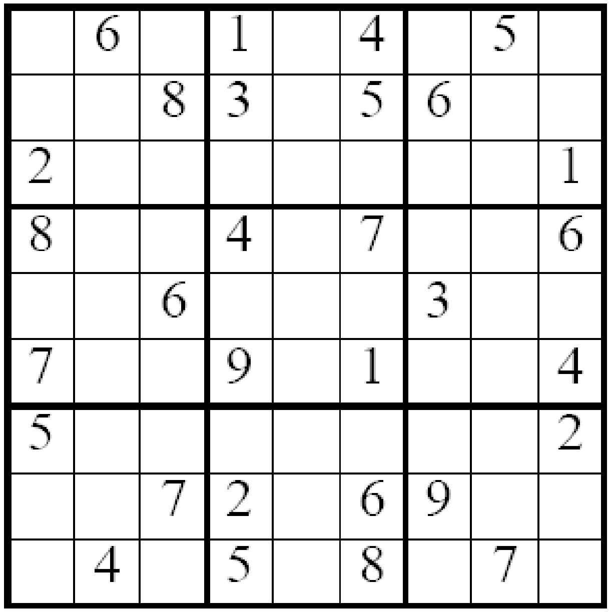
- No meetings before 9am.
- No registration of marks before April 2.
- The lecture rooms have a capacity.
- Two lectures of a student cannot overlap.
- No two trains on the same track at the same time.
- Salary > 45k Euros ☺

…

Constraint Satisfaction Problems

- A constraint is a restriction.
- There are many real-life problems that require to give a decision in the presence of constraints:
  - flight / train scheduling;
  - scheduling of events in an operating system;
  - staff rostering at a company;
  - course time tabling at a university …
- Such problems are called Constraint Satisfaction Problems (CSPs).

Sudoku: An everyday-life example



	6		1		4		5	
		8	3		5	6		
2								1
8			4		7			6
		6				3		
7			9		1			4
5								2
		7	2		6	9		
	4		5		8		7	

CSPs: More formally

- A CSP is a triple **<X,D,C>** where:
  - **X** is a set of decision variables **$\{X_1,\ldots,X_n\}$**.
  - **D** is a set of domains **$\{D_1,\ldots,D_n\}$** for X:
    - D_i is a set of possible values for X_i.
    - usually assume finite domain.
  - **C** is a set of constraints **$\{C_1,\ldots,C_m\}$**:
    - C_i is a relation over $X_j,\ldots,X_k$, giving the set of combination of allowed values.
    - $C_i \subseteq D(X_j)\ \mathrm{x}\ \ldots \mathrm{x}\ D(X_k)$

- A solution to a CSP is an assignment of values to the variables which satisfies all the constraints simultaneously.

CSPs: A simple example

- Variables

  $X = \{X_1, X_2, X_3\}$

- Domains

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

- Constraints

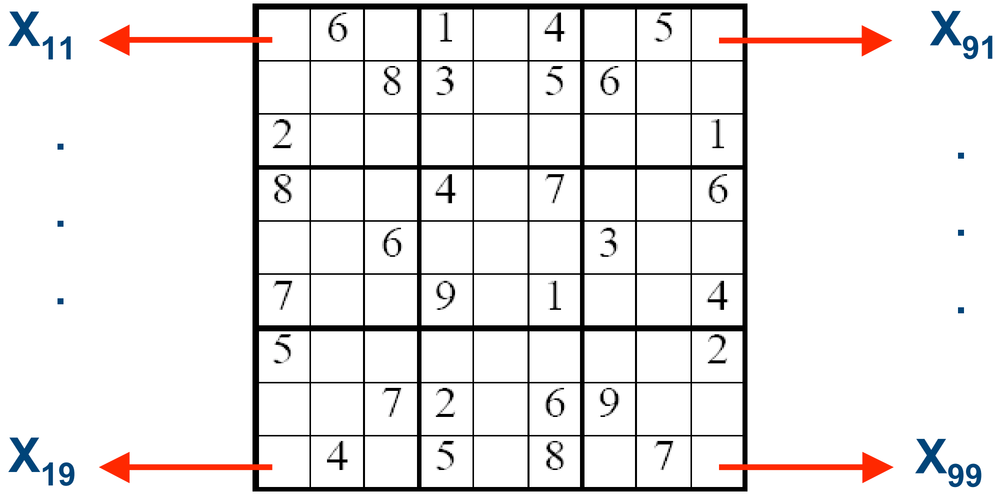
  $X_1 > X_2$ and $X_1 + X_2 = X_3$ and $X_1 \neq X_2 \neq X_3 \neq X_1$

- Solution

  $X_1 = 2$, $X_2 = 1$, $X_3 = 3$

  alldifferent($[X_1, X_2, X_3]$)

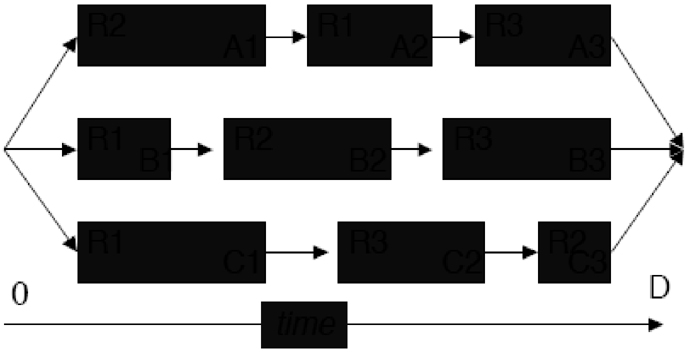
Sudoku: An everyday-life example



	6		1		4		5	
		8	3		5	6		
2								1
8			4		7			6
		6				3		
7			9		1			4
5								2
		7	2		6	9		
	4		5		8		7	

- A simple CSP
  - 9x9 variables (X_{ij}) with domains {1,...,9}
  - Not-equals constraints on the rows, columns, and 3x3 boxes. E.g.,
    alldifferent([X_{11}, X_{21}, X_{31}, …, X_{91}])
    alldifferent([X_{11}, X_{12}, X_{13}, …, X_{19}])
    alldifferent([X_{11}, X_{21}, X_{31}, X_{12}, X_{22}, X_{32}, X_{13}, X_{23}, X_{33}])

Job-Shop Scheduling: A real-life example



- Schedule jobs, each using a resource for a period, in time D by obeying the precedence and capacity constraints
- A very common industrial problem.
- CSP:
  - variables represent the operations;
  - domains represent the start times;
  - constraints specify precedence and exclusivity.

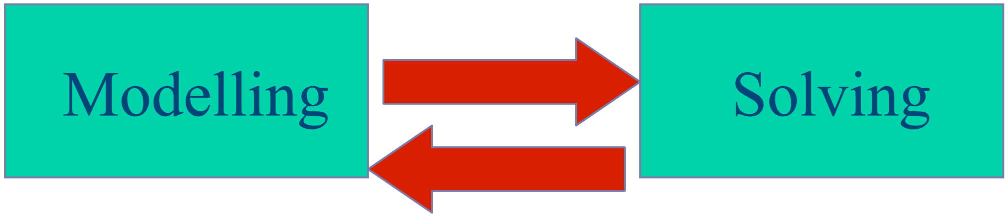
CSPs

- Search space: $D(X_1)$ x $D(X_2)$x … x $D(X_n)$
  - very large!
- Constraint satisfaction is NP-complete:
  - no polynomial time algorithm is known to exist!
  - I can get no satisfaction ☹
- We need general and efficient methods to solve CSPs:
  - Integer and Linear Programming (satisfying linear constraints on 0/1 variables and optimising a criterion)
  - SAT (satisfying CNF formulas on 0/1 variables)
  - …
  - Constraint Programming

How does it exactly work?

Core of CP

- CP is composed of two parts that are strongly interconnected:



Core of CP-Modelling

The CP user models the problem as a CSP:

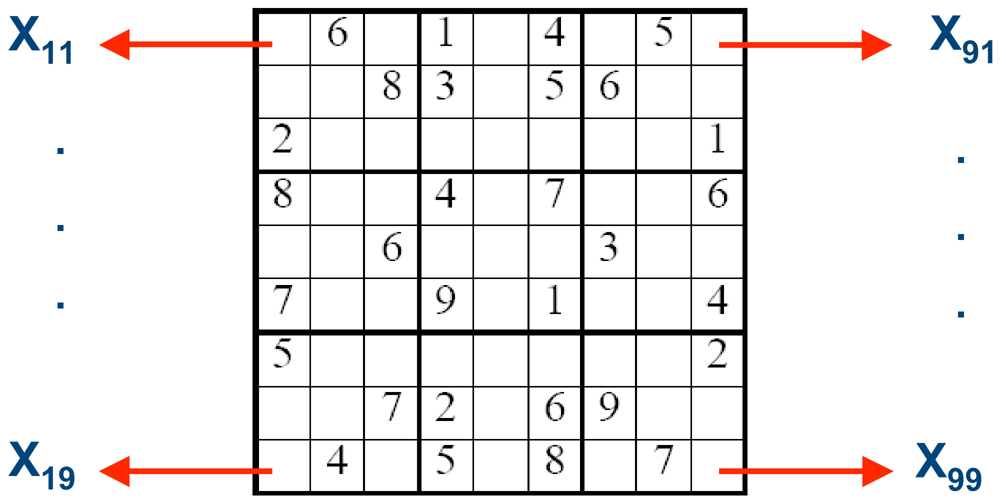
- define the variables and their domains;
- specify solutions by posting constraints on the variables:
  - off-the-shelf constraints or user-defined constraints.
- a constraint can be thought of a reusable component with a propagation algorithm.

WAIT TO UNDERSTAND WHAT I MEAN ☺

Modelling

- Modelling is a critical aspect.
- Given the human understanding of a problem, we need to answer questions like:
  - which variables shall I choose?
  - which constraints shall I enforce?
  - shall I use off-the-self constraints or define and integrate my own?
  - are some constraints redundant, therefore can be avoided?
  - are there any implied constraints?
  - among alternative models, which one shall I prefer?

A problem with a simple model



	6		1		4		5	
		8	3		5	6		
2								1
8			4		7			6
		6				3		
7			9		1			4
5								2
		7	2		6	9		
	4		5		8		7	

- A simple CSP
  - 9x9 variables (X_{ij}) with domains {1,...,9}
  - Not-equals constraints on the rows, columns, and 3x3 boxes, eg.,

    alldifferent([X_{11}, X_{21}, X_{31}, …, X_{91}])

    alldifferent([X_{11}, X_{12}, X_{13}, …, X_{19}])

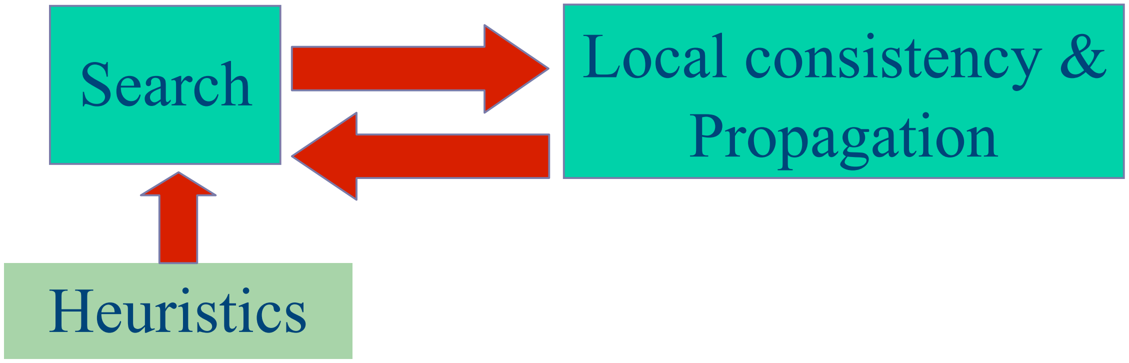
    alldifferent([X_{11}, X_{21}, X_{31}, X_{12}, X_{22}, X_{32}, X_{13}, X_{23}, X_{33}])

A problem with a complex model

- Consider a permutation problem:
  - find a permutation of the numbers {1,...,n} s.t. some constraints are satisfied.
- One model:
  - variables (X_i) for positions, domains for numbers {1,...,n}.
- Dual model:
  - variables (Y_i) for numbers {1,…,n}, domains for positions.
- Often different views allow different expression of the constraints and different implied constraints:
  - can be hard to decide which is better!
- We can use multiple models and combine them via channelling constraints to keep consistency between the variables:
  - $X_i = j \leftrightarrow Y_j = i$

Core of CP-Solving

- The user lets the CP technology solve the CSP:
  - choose a search algorithm:
    - usually backtracking tree search.
  - integrate local consistency and propagation.
  - choose heuristics for branching:
    - which variable to branch on?
    - which value to branch on?



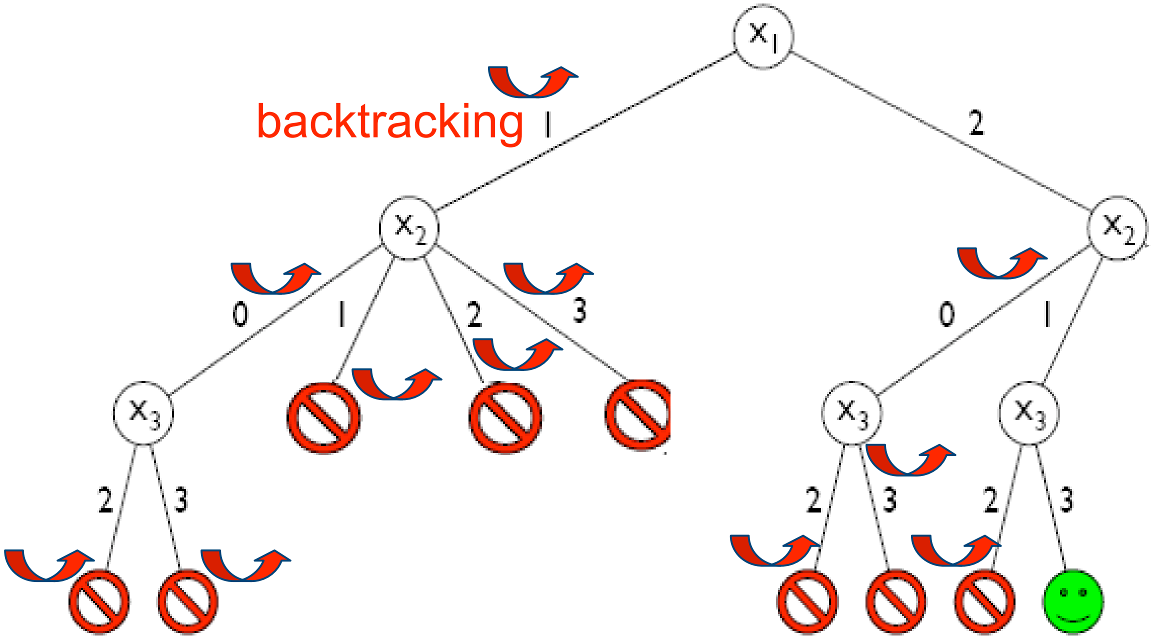
Backtracking Tree Search

- A possible efficient and simple method.
- Variables are instantiated sequentially.
- Whenever all the variables of a constraint is instantiated, the validity of the constraint is checked.
- If a partial instantiation violates a constraint, backtracking is performed to the most recently instantiated variable that still has alternative values.
- Backtracking eliminates a subspace from the cartesian product of all variable domains.
- Essentially performs a depth-first search.

Backtracking Tree Search

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

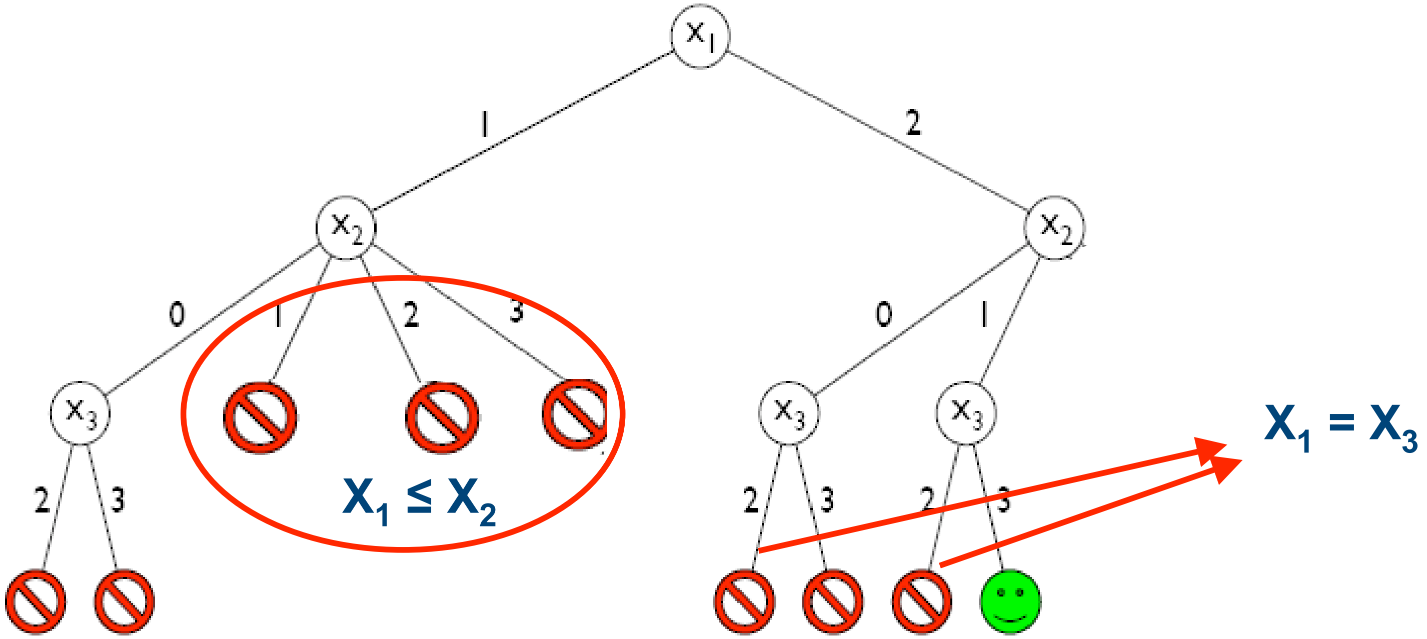
Backtracking tree search



Fails 8 times!

Backtracking Tree Search

- Backtracking suffers from thrashing ☹ :
  - performs checks only with the current and past variables;
  - search keeps failing for the same reasons.



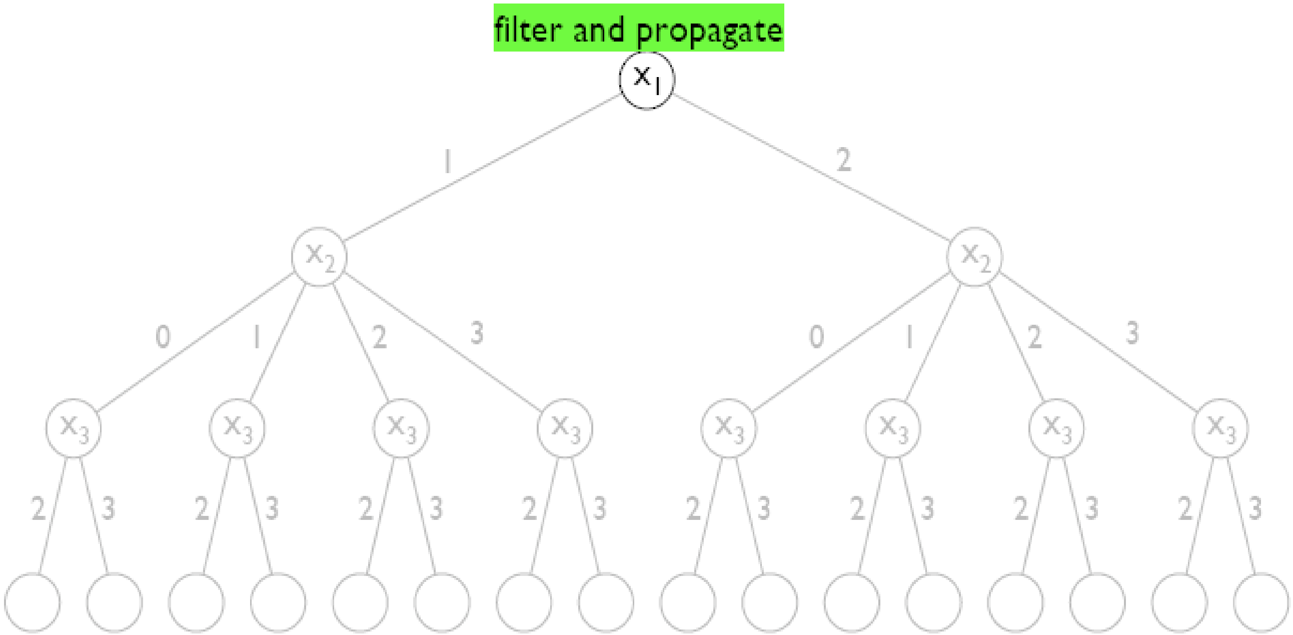
Constraint Programming

- Integrates local consistency and constraint propagation into the backtracking search. Consequently:
  - we can reason about the properties of constraints and their effect on their variables;
  - some values can be filtered from some domains, reducing the backtracking search space significantly!

Constraint Programming

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

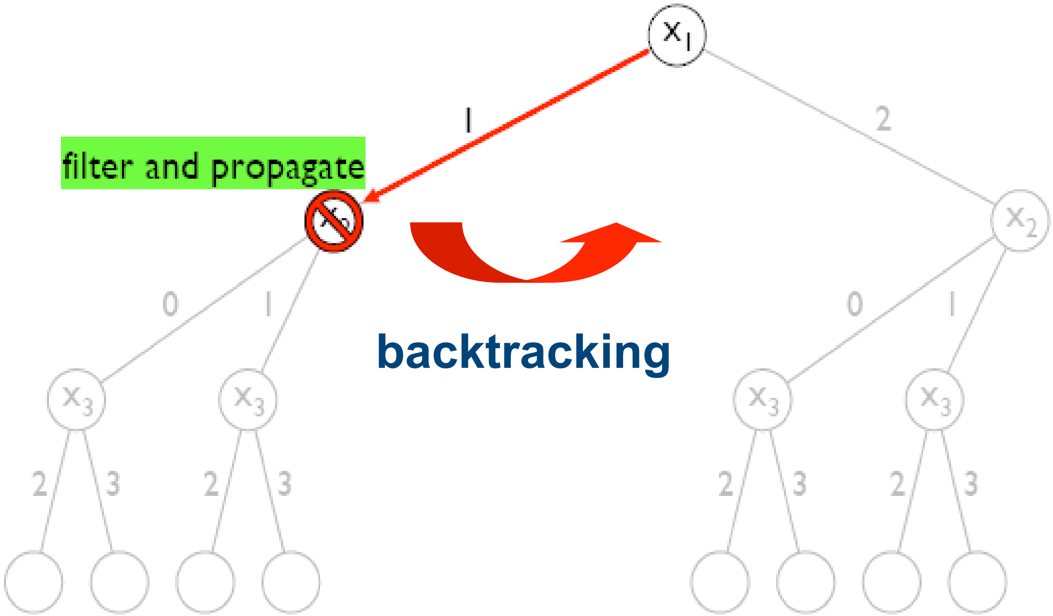
Backtracking tree search + local consistency/propagation



Constraint Programming

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

  Backtracking tree search + local consistency/propagation



filter and propagate

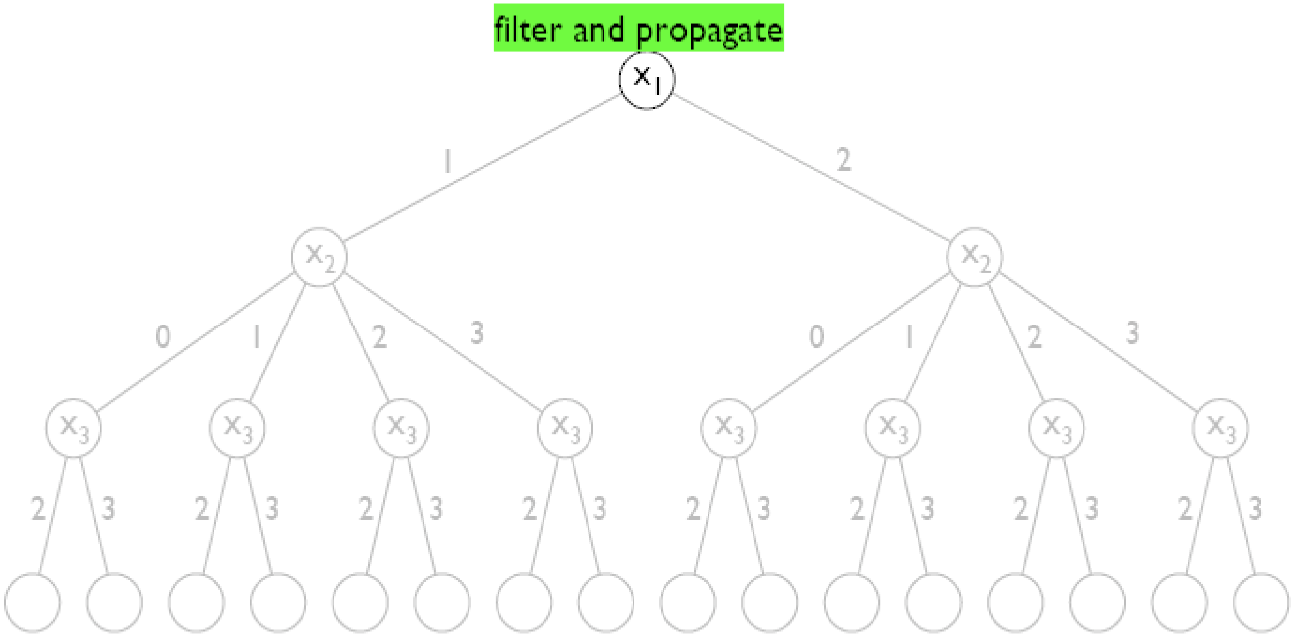
backtracking

Constraint Programming

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

Backtracking tree search + local consistency/propagation

filter and propagate

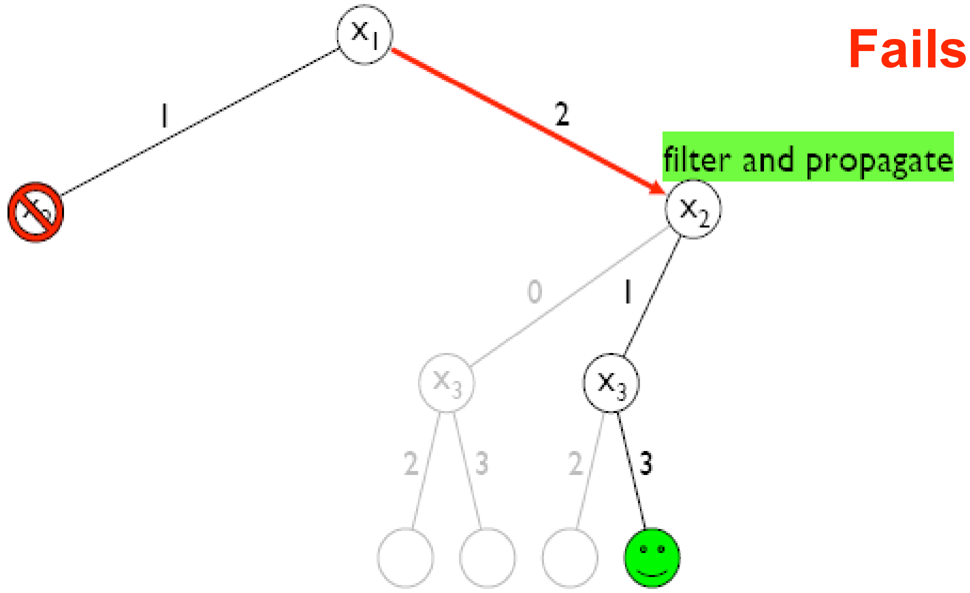


Constraint Programming

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

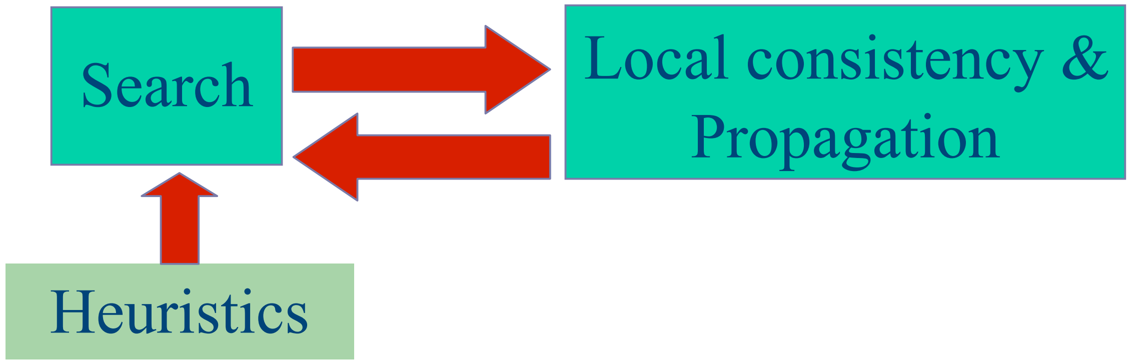
Backtracking tree search + local consistency/propagation

Fails only once!



Local consistency & Propagation & Heuristics

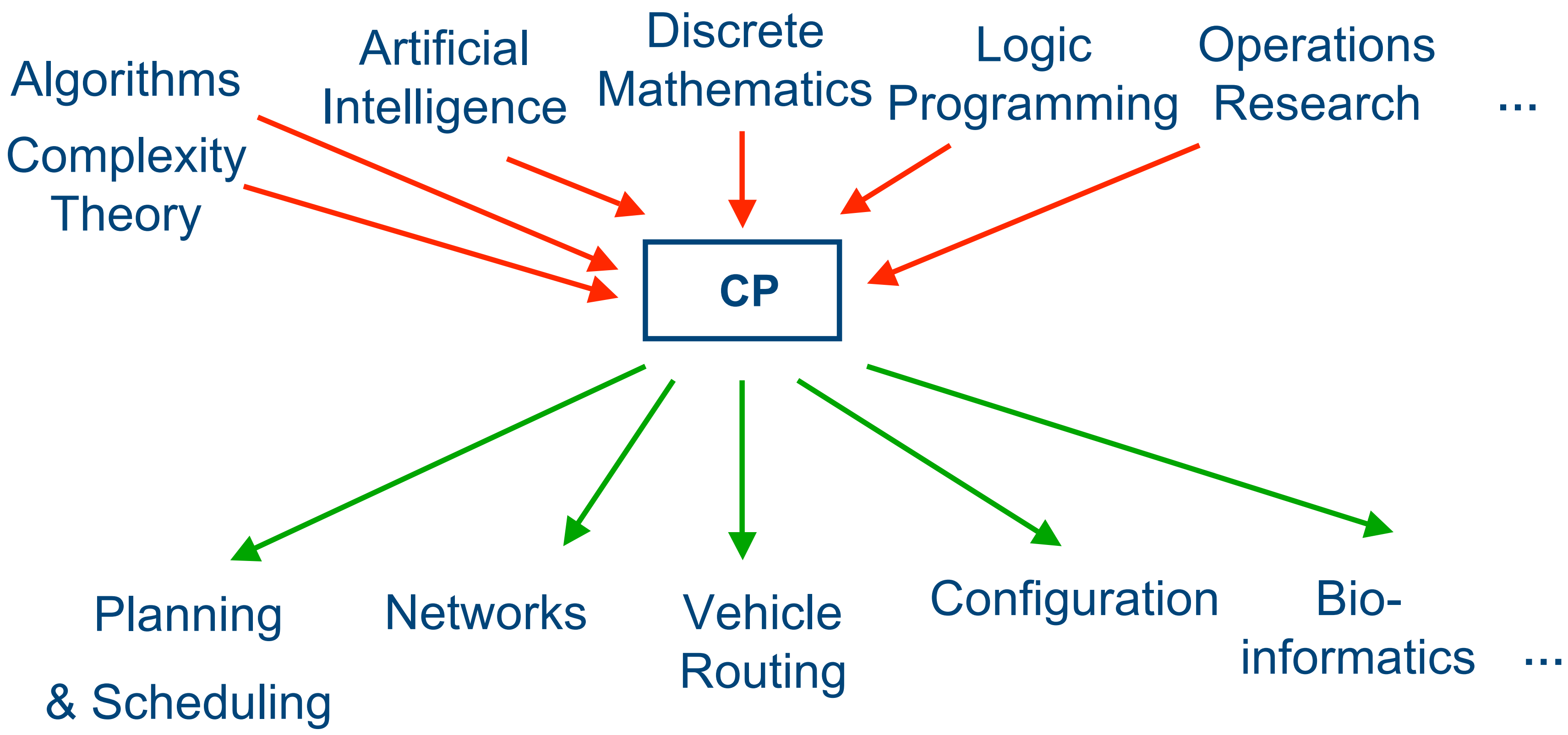
- Central to the process of solving CSPs which are inherently intractable.



CP

- Programming, in the sense of mathematical programming:
  - the user states declaratively the constraints on a set of decision variables.
  - an underlying solver solves the constraints and returns a solution.
- Programming, in the sense of computer programming:
  - the user needs to program a strategy to search for a solution.
  - otherwise, solving process can be inefficient.

CP


Algorithms

Artificial Intelligence

Discrete Mathematics

Logic Programming

Operations Research

…

Complexity Theory

CP

Planning & Scheduling

Networks

Vehicle Routing

Configuration

Bio-informatics

…

CP

- Solve SUDOKU using CP!

  http://www.cs.cornell.edu/gomes/SUDOKU/Sudoku.html

  - very easy, not worth spending minutes ☺
  - you can decide which newspaper provides the toughest Sudoku instances ☺

CP

- Constraints can be embedded into:
  - logic programming (constraint logic programming)
    - Prolog III, CLP(R), SICStus Prolog, ECLiPSe, CHIP, …
  - functional programming
    - Oz
  - imperative programming
    - often via a separate library
    - ILOG Solver, Gecode, Choco, Minion, …

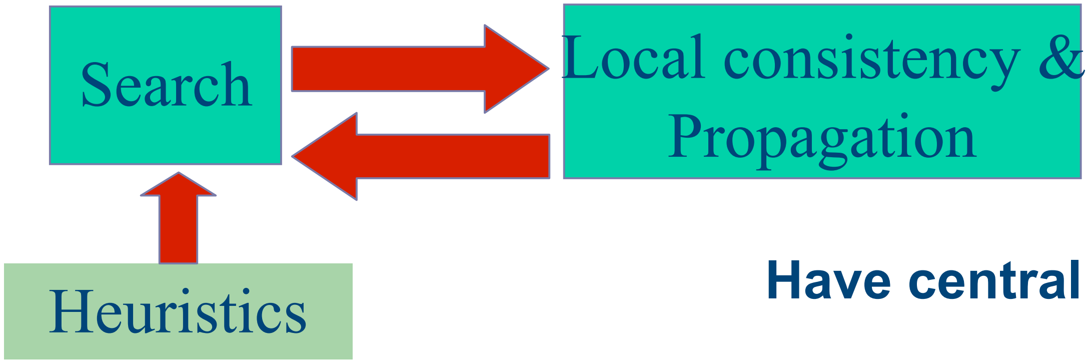
NOTE: We will not commit to any CP language/library, rather use a mathematical and/or natural notation.

PART II: Constraint Propagation

Local Consistency & Constraint Propagation

PART I: The user lets the CP technology solve the CSP:

- choose a search algorithm (usually backtracking tree search);
- design heuristics for branching;
- integrate local consistency and propagation.



Have central affect

What exactly are they?
How do they work?

Outline

- Local Consistency
  - Arc Consistency (AC)
  - Generalised Arc Consistency (GAC)
  - Bounds Consistency (BC)
  - Higher Levels of Consistency
- Constraint Propagation
  - Propagation Algorithms
- Specialised Propagation Algorithms
  - Global Constraints
  - Alldifferent Constraint
  - Other Examples of Global Constraints
- Generalised Algorithms
  - GAC Schema

Local Consistency

- Backtrack tree search aims to extend a partial instantiation of variables to a complete and consistent one.
  - The search space is too large!
- Some inconsistent partial assignments obviously cannot be completed.
- Local consistency is a form of inference which detects inconsistent partial assignments.
  - Consequently, the backtrack search commits into less inconsistent instantiations.
- Local, because we examine individual constraints.
  - Remember that global consistency is NP-complete!

Local Consistency: An example

- $D(X_1) = \{1,2\}$, $D(X_2) = \{3,4\}$, C_1: $X_1 = X_2$, C_2: $X_1 + X_2 \geq 1$
- $X_1 = 1$
- $X_1 = 2$
- $X_2 = 3$
- $X_4 = 4$
  - no need to check the individual assignments.
  - no need to check the other constraint.
  - unsatisfiability of the CSP can be inferred without having to search!

all inconsistent partial assignments
wrt the constraint $X_1 = X_2$

Several Local Consistencies

- Most popular local consistencies:
  - Arc Consistency (AC)
  - Generalised Arc Consistency (GAC)
  - 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.

Arc Consistency (AC)

- Defined for binary constraints.
- A binary constraint **C** is a relation on two variables **X_i** and **X_j**, giving the set of allowed combinations of values (i.e. tuples):
  - **$C \subseteq D(X_i) \times D(X_j)$**
- C is AC iff:
  - forall $v \in D(X_i)$, exists $w \in D(X_j)$ s.t. $(v,w) \in C$.
    - $v \in D(X_i)$ is said to have a support wrt the constraint C.
  - forall $w \in D(X_j)$, exists $v \in D(X_i)$ s.t. $(v,w) \in C$.
    - $w \in D(X_j)$ is said to have a support wrt the constraint C.
- A CSP is AC iff all its binary constraints are AC.

AC: An example

- $D(X_1) = \{1,2,3\}$, $D(X_2) = \{2,3,4\}$, C: $X_1 = X_2$
- AC(C)?
  - $1 \in D(X_1)$ does not have a support.
  - $2 \in D(X_1)$ has $2 \in D(X_2)$ as support.
  - $3 \in D(X_1)$ has $3 \in D(X_2)$ as support.
  - $2 \in D(X_2)$ has $2 \in D(X_1)$ as support.
  - $3 \in D(X_2)$ has $3 \in D(X_1)$ as support.
  - $4 \in D(X_2)$ does not have a support.
- $X_1 = 1$ and $X_2 = 4$ are inconsistent partial assignments.
- $1 \in D(X_1)$ and $4 \in D(X_2)$ must be removed to achieve AC.
- $D(X_1) = \{2,3\}$, $D(X_2) = \{2,3\}$, C: $X_1 = X_2$.
  - AC(C)

Propagation!

Generalised Arc Consistency

- Generalisation of AC to n-ary constraints.
- A constraint **C** is a relation on k variables **$X_1,\dots, X_k$:**
  - **$C \subseteq D(X_1) \times \dots \times D(X_k)$**
- A support is a tuple $<d_1,\dots,d_k> \in C$ where $d_i \in D(X_i)$.
- C is GAC iff:
  - forall X_i in {**$X_1,\dots, X_k$**}, forall $v \in D(X_i)$, v belongs to a support.
- AC is a special case of GAC.
- A CSP is GAC iff all its constraints are GAC.

GAC: An example

- $D(X_1) = \{1,2,3\}$, $D(X_2) = \{1,2\}$, $D(X_3) = \{1,2\}$
  C: alldifferent($[X_1, X_2, X_3]$)

- GAC(C)?
  - $X_1 = 1$ and $X_1 = 2$ are not supported!
- $D(X_1) = \{3\}$, $D(X_2) = \{1,2\}$, $D(X_3) = \{1,2\}$
  C: $X_1 \neq X_2 \neq X_3$
  - GAC(C)

Bounds Consistency (BC)

- Defined for totally ordered (e.g. integer) domains.
- Relaxes the domain of X_i from $D(X_i)$ to $[\min(X_i)..\max(X_i)]$.
- Advantage:
  - it might be easier to look for a support in a range than in a domain;
  - achieving BC is often cheaper than achieving GAC;
  - achieving BC is enough to achieve GAC for monotonic constraints.
- Disadvantage:
  - BC might not detect all GAC inconsistencies in general.

Bounds Consistency (BC)

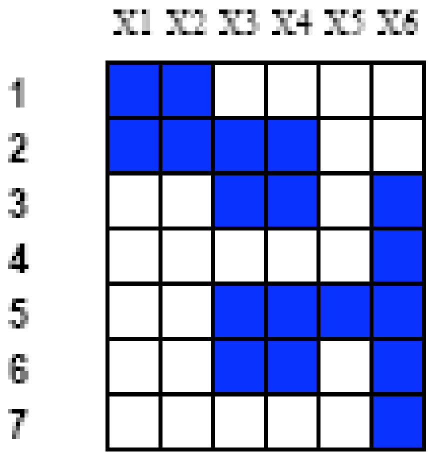
- A constraint **C** is a relation on k variables **$X_1,\ldots, X_k$:**
  - **$C \subseteq D(X_1) \times \ldots \times D(X_k)$**
- A bound support is a tuple $<d_1,\ldots,d_k> \in C$ where $d_i \in [\min(X_i)..\max(Xi)]$.
- C is BC iff:
  - forall X_i in {$\mathbf{X_1,\ldots, X_k}$}, $\min(X_i)$ and $\max(X_i)$ belong to a bound support.

GAC > BC: An example

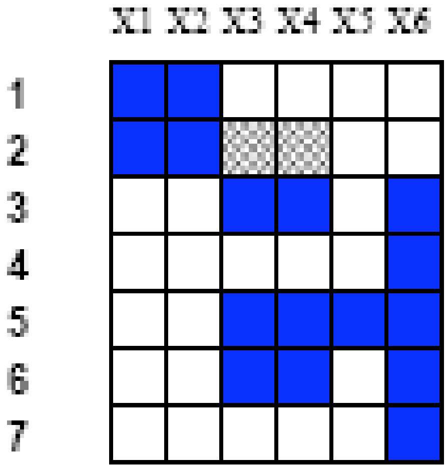
- $D(X_1) = D(X_2) = \{1,2\}$, $D(X_3) = D(X_4) = \{2,3,5,6\}$, $D(X_5) = \{5\}$, $D(X_6) = \{3,4,5,6,7\}$

  C: alldifferent($[X_1, X_2, X_3, X_4, X_5, X_6]$)

- BC(C): $2 \in D(X_3)$ and $2 \in D(X_4)$ have no support.



Original



BC

GAC > BC: An example

- $D(X_1) = D(X_2) = \{1,2\}$, $D(X_3) = D(X_4) = \{2,3,5,6\}$, $D(X_5) = \{5\}$, $D(X_6) = \{3,4,5,6,7\}$

  C: alldifferent($[X_1, X_2, X_3, X_4, X_5, X_6]$)

- GAC(C): $\{2,5\} \in D(X_3)$, $\{2,5\} \in D(X_4)$, $\{3,5,6\} \in D(X_6)$ have no support.

	X1	X2	X3	X4	X5	X6
1	■	■				
2	■	■	■	■		
3			■	■		■
4						■
5			■	■	■	■
6			■	■		■
7						■

Original

	X1	X2	X3	X4	X5	X6
1	■	■				
2	■	■	▨	▨		
3			■	■		■
4						■
5			■	■	■	■
6			■	■		■
7						■

BC

	X1	X2	X3	X4	X5	X6
1	■	■				
2	■	■	☒	☒		
3			■	■		☒
4						■
5			☒	☒	■	☒
6			■	■		▨
7						■

GAC

GAC = BC: An example

- $D(X_1) = \{1,2,3\}$, $D(X_2) = \{1,2,3\}$, C: $X_1 < X_2$
- BC(C):
  - $D(X_1) = \{1,2\}$, $D(X_2) = \{2,3\}$
- BC(C) = GAC(C):
  - a support for $\min(X_2)$ supports all the values in $D(X_2)$.
  - a support for max(X1) supports all the values in D(X1).

Higher Levels of Consistencies

- Path consistency, k-consistencies, (i,j) consistencies, …
- Not much used in practice:
  - detect inconsistent partial assignments with more than one <variable,value> pair.
  - cannot be enforced by removing single values from domains.
- Domain based consistencies stronger than (G)AC.
  - Singleton consistencies, triangle-based consistencies, …
  - Becoming popular:
    - shaving in scheduling.

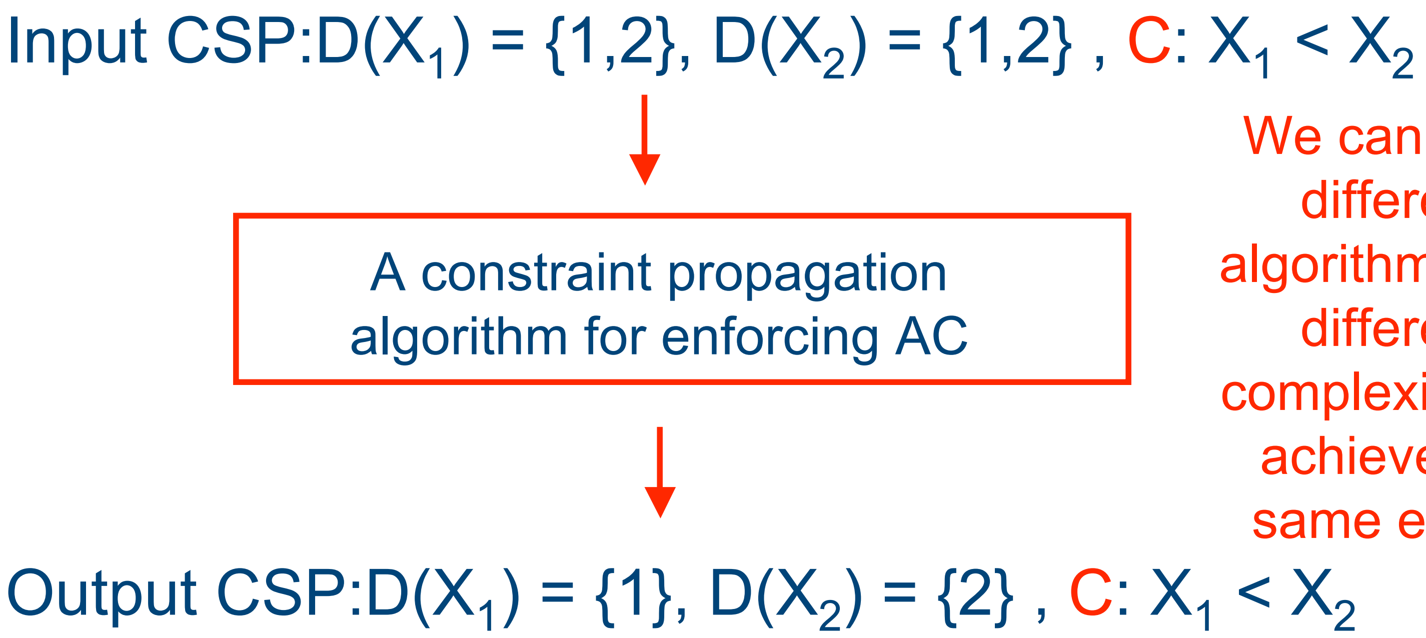
Outline

- Local Consistency
  - Arc Consistency (AC)
  - Generalised Arc Consistency (GAC)
  - Bounds Consistency (BC)
  - Higher Levels of Consistency
- Constraint Propagation
  - Constraint Propagation Algorithms
- Specialised Propagation Algorithms
  - Global Constraints
  - Alldifferent Constraint
  - Other Examples of Global Constraints
- Generalised Algorithms
  - GAC Schema, AC Algorithms

Constraint Propagation

- Can appear under different names:
  - constraint relaxation
  - filtering algorithm
  - local consistency enforcing, …
- Similar concepts in other fields:
  - unit propagation in SAT.
- Local consistencies define properties that a CSP must satisfy after constraint propagation:
  - the operational behaviour is completely left open;
  - the only requirement is to achieve the required property on the CSP.

Constraint Propagation: A simple example



Input CSP:$D(X_1) = \{1,2\}$, $D(X_2) = \{1,2\}$, C: $X_1 < X_2$

A constraint propagation algorithm for enforcing AC

Output CSP:$D(X_1) = \{1\}$, $D(X_2) = \{2\}$, C: $X_1 < X_2$

We can write different algorithms with different complexities to achieve the same effect.

Constraint Propagation Algorithms

- A constraint propagation algorithm propagates a constraint C.
    - It removes the inconsistent values from the domains of the variables of C.
    - It makes C locally consistent.
    - The level of consistency depends on C:
        - GAC might be NP-complete, BC might not be possible, …

Constraint 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.

Constraint Propagation: An 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;
  - each algorithm maintains (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 Constraint Propagation Algorithms

- It is not enough to remove inconsistent values from domains.
- A constraint propagation algorithm must wake up when necessary, otherwise may not achieve the desired local consistency property.
- Events that trigger a constraint propagation:
  - when the domain of a variable changes;
  - when one variable is assigned a value;
  - when the minimum or the maximum values of a domain changes.

Outline

- Local Consistency
  - Arc Consistency (AC)
  - Generalised Arc Consistency (GAC)
  - Bounds Consistency (BC)
  - Higher Levels of Consistency
- Constraint Propagation
  - Propagation Algorithms
- Specialised Propagation Algorithms
  - Global Constraints
  - Alldifferent Constraint
  - Other Examples of Global Constraints
- Generalised Propagation Algorithms
  - GAC Schema, AC Algorithms

Specialised Propagation Algorithms

- A constraint propagation algorithm can be general or specialised:
  - general, if it is applicable to any constraint;
  - specialised, if it is specific to a constraint, exploiting the constraint semantics.
- Many real-life constraints are complex and non-binary.
- A global constraint is a complex and non-binary constraint which encapsulates a specialised propagation algorithm.

Benefits of Global Constraints

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

Alldifferent Constraint

- Alldifferent constraint
  - useful in a variety of assignment problems
    - e.g. permutation, timetabling, production problems, …
  - alldifferent ($[X_1, X_2, \ldots, X_n]$) holds iff

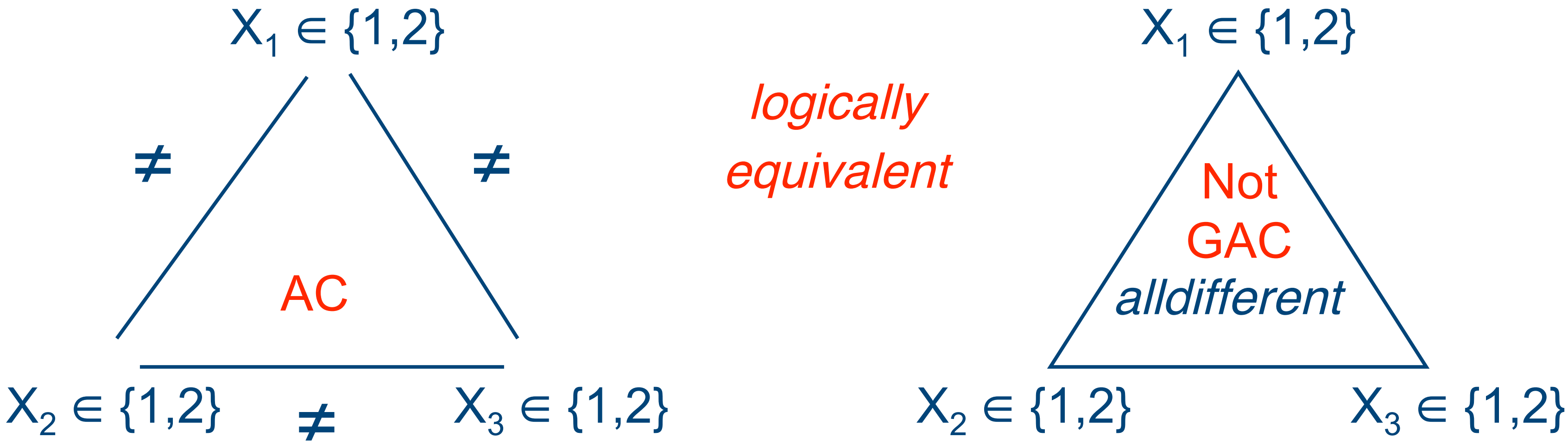
    $X_i \neq X_j$ forall $i < j \in \{1,\ldots,n\}$

Alldifferent Constraint

- Modelling Benefits
  - One constraint instead of $X_i \neq X_j$ forall $i < j \in \{1,\ldots,n\}$
- Solving Benefits
  - Efficient algorithms to maintain GAC, BC, … (algorithmic)

Alldifferent Constraint

- Solving Benefits (operational)
  - GAC > AC on the decomposition



Alldifferent Constraint

- GAC algorithm based on matching theory.
  - Establishes a relation between the solutions of the constraint and the properties of a graph.
  - Runs in time $O(dn^{1.5})$.
- Value graph: bipartite graph between variables and their possible values.
- Matching: set of edges with no two edges having a node in common.
- Maximal matching: largest possible matching.

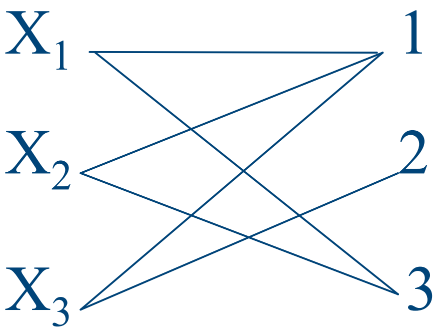
Alldifferent Constraint

- An assignment of values to the variables X_1, X_2, …, X_n is a solution iff it corresponds to a maximal matching.
  - Edges that do not belong to a maximal matching can be deleted.
- The challenge is to compute such edges efficiently.
  - Exploit concepts like strongly connected components, alternating paths, …

Alldifferent Constraint

- $D(X_1) = \{1,3\}$, $D(X_2) = \{1,3\}$, $D(X_3)= \{1,2\}$

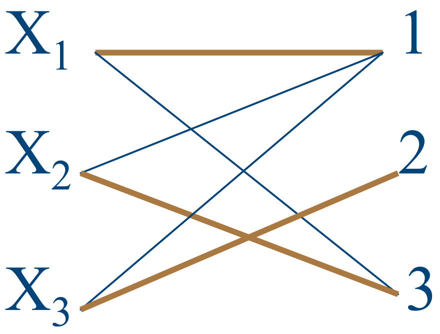
Variable-value graph



Alldifferent Constraint

- $D(X_1) = \{1,3\}$, $D(X_2) = \{1,3\}$, $D(X_3)= \{1,2\}$

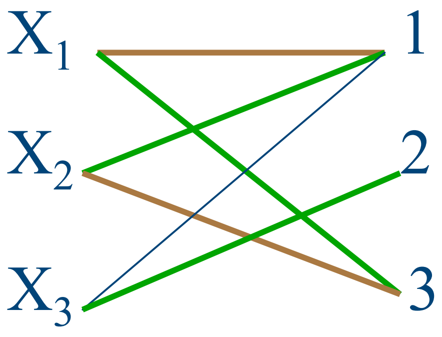
A maximal matching



Alldifferent Constraint

- $D(X_1) = \{1,3\}$, $D(X_2) = \{1,3\}$, $D(X_3)= \{\cancel{1},2\}$

Another maximal matching



Does not belong to any maximal matching

Other Examples of Global Constraints

- NValue constraint:
  - useful in counting problems
  - NValue ($[X_1, X_2, \ldots, X_n]$, N) holds iff $N = |\{X_i \mid 1 \leq i \leq n\}|$
  - NValue ([1, 2, 2, 1, 3], 3)
- Element constraint:
  - useful in variable subscripts
  - Element (V, N, $[X_1, X_2, \ldots, X_n]$) holds iff $X_N = V$
  - Element (3, 2, [1, 3, 4])
- Global cardinality constraint:
  - useful in occurrence problems
  - GCC ($[X_1, X_2, \ldots, X_n]$, $[v_1, \ldots, v_m]$, $[O_1, \ldots, O_m]$) iff
    forall $j \in \{1,\ldots,m\}$ $O_j = |\{X_i \mid X_i = v_j, 1 \leq i \leq n\}|$
  - GCC ([1, 1, 2], [1, 2], [2, 1])

Other Examples of Global Constraints

- Lex ($[X_1, X_2, \ldots, X_n], [Y_1, Y_2, \ldots, Y_n]$)
  - useful in symmetry breaking
  - Lex ($[X_1, X_2, \ldots, X_n], [Y_1, Y_2, \ldots, Y_n]$) holds iff:

    $X_1 < Y_1$ OR

    ($X_1 = Y_1$ AND $X_2 < Y_2$) OR

    …

    ($X_1 = Y_1$ AND $X_2 = Y_2$ AND …. AND $X_n < Y_n$) OR

    ($X_1 = Y_1$ AND $X_2 = Y_2$ AND …. AND $X_n = Y_n$
  - Lex ([1, 2, 3],[1, 3, 4])

Outline

- Local Consistency
  - Arc Consistency (AC)
  - Generalised Arc Consistency (GAC)
  - Bounds Consistency (BC)
  - Higher Levels of Consistency
- Constraint Propagation
  - Propagation Algorithms
- Specialised Propagation Algorithms
  - Global Constraints
  - Alldifferent Constraint
  - Other Examples of Global Constraints
- Generalised Propagation Algorithms
  - GAC Schema, AC Algorithms

Generalised Propagation Algorithms

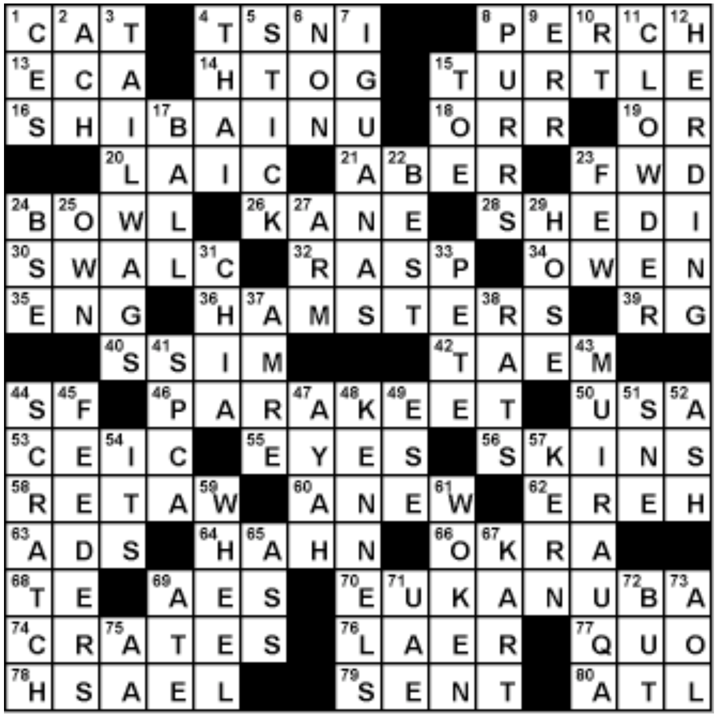
- Not all constraints have nice semantics we can exploit to devise an efficient specialised propagation algorithm.
- Consider a product configuration problem:
  - compatibility constraints on hardware components:
    - only certain combinations of components work together.
  - compatibility may not be a simple pairwise relationship:
    - video cards supported function of motherboard, CPU, clock speed, O/S, ...

Production Configuration Problem

- 5-ary constraint:
  - Compatible (motherboard345, intelCPU, 2GHz, 1GBRam, 80GBdrive).
  - Compatible (motherboard346, intelCPU, 3GHz, 2GBRam, 100GBdrive).
  - Compatible (motherboard346, amdCPU, 2GHz, 2GBRam, 100GBdrive).
  - …

Crossword Puzzle

- Constraints with different arity:
  - $\text{Word}_1 ([X_1,X_2,X_3])$
  - $\text{Word}_2 ([X_1,X_{13},X_{16}])$
  - …
- No simple way to decide acceptable words other than to put them in a table.


C A T ■ T S N I ■ ■ P E R C H
E C A ■ H T O G ■ T U R T L E
S H I B A I N U ■ O R R ■ O R
■ ■ L A I C ■ A B E R ■ F W D
B O W L ■ K A N E ■ S H E D I
S W A L C ■ R A S P ■ O W E N
E N G ■ H A M S T E R S ■ R G
■ ■ S S I M ■ ■ ■ T A E M ■ ■
S F ■ P A R A K E E T ■ U S A
C E I C ■ E Y E S ■ S K I N S
R E T A W A N E W ■ E R E H
A D S ■ H A H N ■ O K R A ■ ■
T E ■ A E S ■ E U K A N U B A
C R A T E S ■ L A E R ■ Q U O
H S A E L ■ ■ S E N T ■ A T L

GAC Schema

- A generic propagation algorithm.
  - Enforces GAC on an n-ary constraint given by:
    - a set of allowed tuples;
    - a set of disallowed tuples;
    - a predicate answering if a constraint is satisfied or not.
  - Sometimes called the “table” constraint:
    - user supplies table of acceptable values.
- Complexity: O(d^k) time
- Hence, k cannot be too large!
  - ILOG Solver limits it to 3 or so.

Arc Consistency Algorithms

- Generic AC algorithms with different complexities and advantages:
  - AC3
  - AC4
  - AC6
  - AC2001
  - …

PART III: Some Useful Pointers about CP

(Incomplete) List of Advanced Topics

- Modelling
- Global constraints, propagation algorithms
- Search algorithms
- Heuristics
- Symmetry breaking
- Optimisation
- Local search
- Soft constraints, preferences
- Temporal constraints
- Quantified constraints
- Continuous constraints
- Planning and scheduling
- SAT
- Complexity and tractability
- Uncertainty
- Robustness
- Structured domains
- Randomisation
- Hybrid systems
- Applications
- Constraint systems
- No good learning
- Explanations
- Visualisation

Literature

- Books
  - My PhD dissertation ☺
  - Handbook of Constraint Programming

    F. Rossi, P. van Beek, T. Walsh (eds), Elsevier Science, 2006.

    Some online chapters:

    Chapter 1 - Introduction
    Chapter 3 - Constraint Propagation
    Chapter 6 - Global Constraints
    Chapter 10 - Symmetry in CP
    Chapter 11 - Modelling

Literature

- Books
  - Constraint Logic Programming Using Eclipse
    K. Apt and M. Wallace, Cambridge University Press, 2006.
  - Principles of Constraint Programming
    K. Apt, Cambridge University Press, 2003.
  - Constraint Processing
    Rina Dechter, Morgan Kaufmann, 2003.
  - Constraint-based Local Search
    Pascal van Hentenryck and Laurent Michel, MIT Presss, 2005.
  - The OPL Optimization Programming Languages
    Pascal Van Hentenryck, MIT Press, 1999.

Literature

- People
  - Barbara Smith
    - Modelling, symmetry breaking, search heuristics
    - Tutorials and book chapter
  - Christian Bessiere
    - Constraint propagation
    - Global constraints
      - Nvalue constraint
    - Book chapter
  - Jean-Charles Regin
    - Global constraints
      - Alldifferent, global cardinality, cardinality matrix
  - Toby Walsh
    - Modelling, symmetry breaking, global constraints
    - Various tutorials

Literature

- **Journals**
  - Constraints
  - Artificial Intelligence
  - Journal of Artificial Intelligence Research
  - Journal of Heuristics
  - Intelligenza Artificiale (AI*IA)
  - Informs Journal on Computing
  - Annals of Mathematics and Artificial Intelligence

Literature

- **Conferences**
  - Principles and Practice of Constraint Programming
    http://www.cs.ualberta.ca/~ai/cp/
  - Integration of AI and OR Techniques in CP
    http://www.cs.cornell.edu/~vanhoeve/cpaior/
  - National Conference on AI (AAAI)
    http://www.aaai.org
  - International Joint Conference on Artificial Intelligence (IJCAI)
    http://www.ijcai.org
  - European Conference on Artificial Intelligence (ECAI)
    http://www.eccai.org
  - International Symposium on Practical Aspects of Declarative Languages (PADL)
    http://www.informatik.uni-trier.de/~ley/db/conf/padl/index.html

Literature

- **Schools and Tutorials**
  - ACP summer schools:

    2005: http://www.math.unipd.it/~frossi/cp-school/

    2006: http://www.cse.unsw.edu.au/~tw/school.html

    2007: http://www.iiia.csic.es/summerschools/sscp2007/

    2008: http://www-circa.mcs.st-and.ac.uk/cpss2008/
  - AI conference tutorials (IJCAI’07, IJCAI’05, ECAI’04 …).
  - CP conference tutorials.
  - CP-AI-OR master classes.

Literature

- **Solvers & Languages**
  - Choco (http://choco.sourceforge.net/)
  - Comet (http://www.comet-online.org/)
  - Eclipse (http://eclipse.crosscoreop.com/)
  - FaCiLe (http://www.recherche.enac.fr/opti/facile/)
  - Gecode (http://www.gecode.org/)
  - ILOG Solver (http://www.ilog.com)
  - Koalog Constraint Solver (http://www.gecode.org/)
  - Minion (http://minion.sourceforge.net/)
  - OPL (http://www.ilog.com/products/oplstudio/)
  - Sicstus Prolog (http://www.sics.se/isl/sicstuswww/site/index.html)