

# Constraint Programming: Propagation

① Representation of CSP

② Arc Consistency

③ k-consistency

④ Non-binary Constraints

⑤ Bounds Consistency

# Graph Representation of CSP

- **Constraint representation**
  - using mathematical/logical formula
  - by extension (list of compatible k-tuples, 0-1 matrix)
- **Graph:** nodes, edges (edge connects two nodes)
- **Hypergraph:** nodes, hyper-edges (hyper-edge connects set of nodes)
- CSP representation through **constraint hypergraph**
  - node = variable, hyper-edge = constraint

- **Example**

- variables  $X_1, \dots, X_6$  with domain 0..1

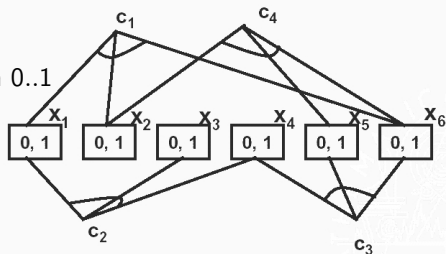
- constraints:

$$c_1 : X_1 + X_2 + X_6 = 1$$

$$c_2 : X_1 - X_3 + X_4 = 1$$

$$c_3 : X_4 + X_5 - X_6 > 0$$

$$c_4 : X_2 + X_5 - X_6 = 0$$



# Binary CSP

## Binary CSP

- CSP with binary constraints only
- unary constraints encoded into the domain

## Constraint graph for binary CSP

- hypergraph not necessary  
graph sufficient (constraint defined on two nodes only)

## Each CSP can be transformed to "corresponding" binary CSP

### Binarization: pros and cons

- getting of unified CSP, many algorithms proposed for binary CSPs
- unfortunately significant increase of the problem size

### Non-binary constraints

- more complex propagation algorithms
- semantics of constraints allows for stronger propagation
  - example: all\_different vs. set of inequality constraints

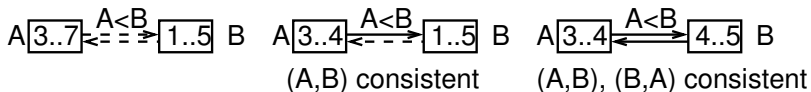
# Node and Arc Consistency

## Node consistency (NC)

- each value from the domain of variable  $V_i$  satisfies all unary constraints over  $V_i$

## Arc consistency (AC) for binary CSP

- arc**  $(V_i, V_j)$  is **arc consistent**, iff there is a value  $y$  for each value  $x$  from the domain of  $V_i$  such that the assignment  $V_i = x, V_j = y$  satisfies all binary constraints over  $V_i, V_j$ .
- arc consistency is **directional**
  - consistency of arc  $(V_i, V_j)$  does not guarantee consistency of  $(V_j, V_i)$



- CSP** is **arc consistent**, iff all arcs (in both directions) are arc consistent

# Arc Revision

- How to make arc  $(V_i, V_j)$  arc consistent?
- Remove such values  $x$  from  $D_i$  that are inconsistent with current domain of  $V_j$  (there is no value  $y$  for  $V_j$  such that assignment  $V_i = x$  and  $V_j = y$  satisfies all binary constraints between  $V_i$  and  $V_j$ )

procedure revise( $(V_i, V_j)$ )

Deleted := false

for  $\forall x$  in  $D_i$  do

    if there is no  $y \in D_j$  such that  $(x, y)$  is consistent

    then  $D_i := D_i - \{x\}$

        Deleted := true

    end if

return Deleted

end revise

- domain( $[V_1, V_2], 2, 4$ ),  $V_1 \# < V_2$   
    revise( $(V_1, V_2)$ ) removes 4 from  $D_1$ ,  $D_2$  is not changed

# Computation of Arc Consistency

How to make CSP arc consistent?

- revisions must be repeated if the domain of some variable was changed
- more effective: revisions can be done with the help of **queue**
  - we add to queue arcs which consistency could have been violated by value removal(s)

What arcs must be exactly revised after domain reduction of  $V_k$ ?

- arcs  $(V_i, V_k)$  leading to variable  $V_k$  with reduced domain



- let  $(V_k, V_m)$  caused domain reduction of  $V_k$ : then arc  $(V_m, V_k)$  leading from  $V_m$  may not be revised (change does not influence it)

# AC-3 Algorithm

procedure AC-3(G)

$Q := \{(V_i, V_j) \mid (V_i, V_j) \in \text{edges}(G), i \neq j\}$

  while Q non empty do

    choose and remove  $(V_k, V_m)$  from Q

    if revise( $(V_k, V_m)$ ) then % additions of arcs

$Q := Q \cup \{(V_i, V_k) \in \text{edges}(G), i \neq k, i \neq m\}$  % still not in queue

  end while

end AC-3

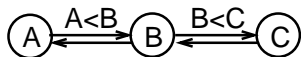
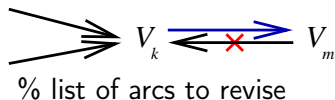
Example:

A < B, B < C:  $(\underline{3}..7, \underline{1}..5, \underline{1}..5) \xrightarrow{AB} (3..4, \underline{1}..5, \underline{1}..5) \xrightarrow{BA} (3..4, \underline{4}..5, \underline{1}..5)$   
 $\xrightarrow{BC} (3..4, 4, \underline{1}..5) \xrightarrow{CB} (3..4, 4, 5) \xrightarrow{AB} (3, 4, 5)$

AC-3 is the most common today but it is still not optimal!

Excercise: What will be domains of A,B,C after AC-3 for:

- domain([A,B,C],1,10), A  $\neq$  B + 1, C  $\neq$  B



# Is it arc consistency sufficient?

AC removes many inconsistent values

- do we get solution of the problem then?
- do we know that solution of the problem exists?
  - $\text{domain}([X,Y,Z],1,2)$ ,  $X \neq Y$ ,  $Y \neq Z$ ,  $Z \neq X$ 
    - arc consistent
    - no solution exists

NO  
NO

Why we use AC then?

- sometimes we obtain solution directly
  - some domain is emptied  $\Rightarrow$  no solution exists
  - all domains have one element only  $\Rightarrow$  we have solution
- general case: size of the solution search is decreased

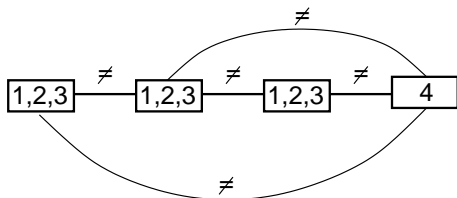
# k-consistency

Have NC and AC anything in common?

- NC: consistency of one variable
- AC: consistency of two variables
- ... and we can continue

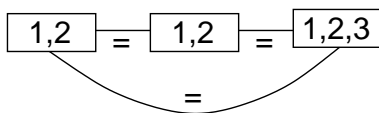
CSP is **k-consistent** iff, any consistent assignment of  $(k-1)$  variables can be extended to any  $k$ -th variable

k-consistency defined for general CSPs including non-binary constraints



4-consistent graph

# Strong k-consistency



3-consistent graph

(1, 1) can be extended to (1, 1, 1)

(2, 2) can be extended to (2, 2, 2)

(1, 3) and (2, 3) are not consistent tuples (we do not extend those)

not 2-consistent

(3) cannot be extended

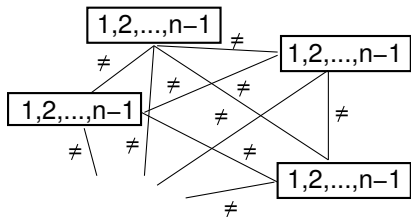
CSP is **strongly k-consistent** iff it is j-consistent for each  $j \leq k$

- strong k-consistency  $\Rightarrow$  k-consistency
- strong k-consistency  $\Rightarrow$  j-consistency  $\forall j \leq k$
- k-consistency  $\not\Rightarrow$  strong k-consistency
  
- NC = strong 1-consistency = 1-consistency
- AC = (strong) 2-consistency

# Consistency for Finding of the Solution

If we have graph with  $n$  nodes, how strong consistency is necessary to obtain solution directly?

- strong  $n$ -consistency is necessary for the graph with  $n$  nodes
  - $n$ -consistency does not suffice (see earlier example)
  - strong  $k$ -consistency for  $k < n$  does not suffice too



graph with  $n$  nodes  
domains  $1..(n-1)$

strong  $k$ -consistent for each  $k < n$   
there is no solution yet

**Strong  $n$ -consistency is necessary for the graph with  $n$  nodes**

- exponential complexity!

# Non-binary Constraints

k-consistency have exponential complexity, it is not used in practice

n-ary constraints are used directly

Constraint is **generally arc consistent (GAC)** iff for each variable  $V_i$  from this constraint and for each its value  $x \in D_i$ , there is an assignment  $y \in D_j$  for each remaining variable  $V_j$  in the constraint such that it is satisfied

- $A + B = C$ ,  $A$  in 1..3,  $B$  in 2..4,  $C$  in 3..7 is GAC

Semantics of constraints is used

- special type of consistency for global constraints
  - e.g. all\_different
- bounds consistency can be used
  - propagation when the smallest or largest domain value changes

One constraint may use different types of consistency

- $A \neq B$ : arc consistency, bounds consistency

# Consistency Algorithm for Non-binary Constraints

Algorithm with **queue of variables** (sometimes also called AC-8)

```
procedure AC-8(Q)
```

```
while Q non empty do
```

```
    choose and remove  $V_j \in Q$ 
```

```
    for  $\forall c$  such that  $V_j \in scope(c)$  do
```

```
         $W := revise(V_j, c)$ 
```

```
        //  $W$  is the set of variables with changed domain
```

```
        if  $\exists V_i \in W$  such that  $D_i = \emptyset$  then return fail
```

```
         $Q := Q \cup \{W\}$ 
```

```
end AC-8
```

- revisions are repeated until there are domain changes
- $scope(c)$ : set of variables on which  $c$  is defined

## Implementation

- set of **constraints to be propagated** maintained for each variable
- user defines REVISE procedures based on the constraint type

# Bounds Consistency

Constraint is **bounds consistent (BC)** iff for each variable  $V_i$  from this constraint and for each its value  $x \in D_i$ , there is an assignment of remaining variables in the constraint such that it is satisfied and  $\min(D_i) \leq y_i \leq \max(D_i)$  holds for selected assignment  $y_i$  of  $V_i$

Bounds consistency: weaker than generalized arc consistency

Propagation when **minimal and maximal value (bounds)** changed only

## Bounds consistency for inequalities

- $A \#> B \Rightarrow \min(A) \geq \min(B)+1, \max(B) \leq \max(A)-1$
- example: A in 4..10, B in 6..18,  $A \#> B$   
 $\min(A) \geq 6+1 \Rightarrow A$  in 7..10  
 $\max(B) \leq 10-1 \Rightarrow B$  in 6..9
- and similar for:  $A \#< B, A \#>= B, A \#=< B$

# Bounds Consistency and Arithmetic Constraints

$$\begin{aligned}A \# = B + C \Rightarrow \min(A) \geq \min(B) + \min(C), \max(A) \leq \max(B) + \max(C) \\ \min(B) \geq \min(A) - \max(C), \max(B) \leq \max(A) - \min(C) \\ \min(C) \geq \min(A) - \max(B), \max(C) \leq \max(A) - \min(B)\end{aligned}$$

- change of  $\min(A)$  causes the change of  $\min(B)$  and  $\min(C)$  only
- change of  $\max(A)$  causes the change of  $\max(B)$  and  $\max(C)$  only, ...

Example:

$$A \text{ in } 1..10, B \text{ in } 1..10, A \# = B + 2, A \# > 5, A \# \setminus = 8$$

$$\begin{aligned}A \# = B + 2 \Rightarrow \min(A) \geq 1 + 2, \max(A) \leq 10 + 2 \Rightarrow A \text{ in } 3..10 \\ \Rightarrow \min(B) \geq 1 - 2, \max(B) \leq 10 - 2 \Rightarrow B \text{ in } 1..8\end{aligned}$$

$$\begin{aligned}A \# > 5 \Rightarrow \min(A) \geq 6 \Rightarrow A \text{ in } 6..10 \\ \Rightarrow \min(B) \geq 6 - 2 \Rightarrow B \text{ in } 4..8 \quad (\text{new propagation for } A \# = B + 2)\end{aligned}$$

$$\begin{aligned}A \# \setminus = 8 \Rightarrow A \text{ in } (6..7) \setminus (9..10) \\ (\text{bounds same, no propagation for } A \# = B + 2)\end{aligned}$$

# Exercises: Bounds and Arc Consistency

- ① Define rules for bounds consistency of the constraints  
 $A \neq B - C, A \neq B + C$
- ② What are rules for bounds consistency of the constraint  $X \neq Y + 5$ ?  
How propagations are processed in the following example?  
 $X \text{ in } 1..20, Y \text{ in } 1..20, X \neq Y + 5, Y \neq 10.$
- ③ What is the difference between bounds and arc consistency? Show it on the example.
- ④ How arc consistency is achieved in the following example?  
 $\text{domain}([X, Y, Z], 1, 5), X \neq Y, Z \neq Y + 1$