CompSci 275, CONSTRAINT Networks

Rina Dechter, Winter 2026

Consistency algorithms, part a
Chapter 3

The projection network

- The projection network of a relation is obtained by projecting it onto each pair of its variables (yielding a binary network).

- *Relation = {(1,1,2)(1,2,2)(1,2,1)}*
  - *What is the projection network?*

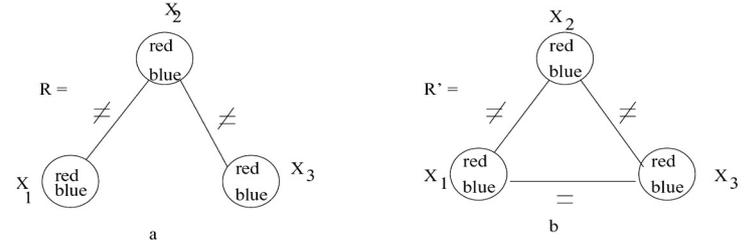- What is the relationship between a relation and its projection network?

x_1	x_2	x_3
1	1	2
1	2	2
2	1	3
2	2	2

- R = {(1,1,2)(1,2,2)(2,1,3)(2,2,2)}

- What are the solutions of its projection network?

x_1	x_2	x_3
1	1	2
1	2	2
2	1	2
2	1	3
2	2	2

Partial order between networks, The minimal network

Definition 2.3.10 *Given two binary networks, $\mathcal{R}'$ and $\mathcal{R}$, on the same set of variables $x_1, ..., x_n$, $\mathcal{R}'$ is at least as tight as $\mathcal{R}$ iff for every i and j, $R'_{ij} \subseteq R_{ij}$.*

- An intersection of two networks is tighter (as tight) than both
- An intersection of two equivalent networks is equivalent to both

Definition: The minimal network is the intersection of all equivalent networks

Theorem: The projection network is identical to the minimal network

The N-queens constraint network

The network has four variables, all with domains $Di = \{1, 2, 3, 4\}$.
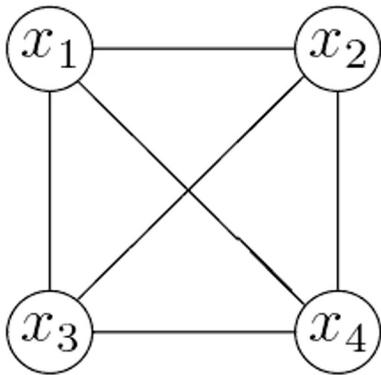(a) The labeled chess board. (b) The constraints between variables.

$$R_{12} = \{(1,3), (1,4), (2,4), (3,1), (4,1), (4,2)\}$$
$$R_{13} = \{(1,2), (1,4), (2,1), (2,3), (3,2), (3,4), (4,1), (4,3)\}$$
$$R_{14} = \{(1,2), (1,3), (2,1), (2,3), (2,4), (3,1), (3,2), (3,4)$$
$$(4,2), (4,3)\}$$
$$R_{23} = \{(1,3), (1,4), (2,4), (3,1), (4,1), (4,2)\}$$
$$R_{24} = \{(1,2), (1,4), (2,1), (2,3), (3,2), (3,4), (4,1), (4,3)\}$$
$$R_{34} = \{(1,3), (1,4), (2,4), (3,1), (4,1), (4,2)\}$$

(a)   (b)

The 4-queens constraint network:
(a) The constraint graph. (b) The minimal binary constraints.
(c) The minimal unary constraints (the domains).

$$M_{12} = \{(2,4),\ (3,1)\}$$
$$M_{13} = \{(2,1),\ (3,4)\}$$
$$M_{14} = \{(2,3),\ (3,2)\}$$
$$M_{23} = \{(1,4),\ (4,1)\}$$
$$M_{24} = \{(1,2),\ (4,3)\}$$
$$M_{34} = \{(1,3),\ (4,2)\}$$

$$D_1 = \{2,3\}$$
$$D_2 = \{1,4\}$$
$$D_3 = \{1,4\}$$
$$D_4 = \{2,3\}$$

(a)   (b)   (c)

Solutions are: (2,4,1,3) (3,1,4,2)

Outline

- Arc-consistency algorithms
- Path-consistency and i-consistency
- Arc-consistency, Generalized arc-consistency, relation arc-consistency
- Global and bound consistency
- Distributed (generalized) arc-consistency
- Consistency operators: join, resolution, Gausian elimination

Consistency methods

- Approximation of inference:
  - Arc, path and i-consistecy
- Methods that transform the original network into tighter and tighter representations

Inference Algorithms
will help search

Definition 3.1.1 (partial solution) *Given a constraint network $\mathcal{R}$, we say that an assignment of values to a subset of the variables $S = \{x_1, ..., x_j\}$ given by $\bar{a} = (< x_1, a_1 >, < x_2, a_2 >, ..., < x_j, a_j >)$ is consistent relative to $\mathcal{R}$ iff it satisfies every constraint R_{S_i} such that $S_i \subseteq S$. The assignment $\bar{a}$ is also called a partial solution of $\mathcal{R}$. The set of all partial solutions of a subset of variables S is denoted by ρ_S or $\rho(S)$.*
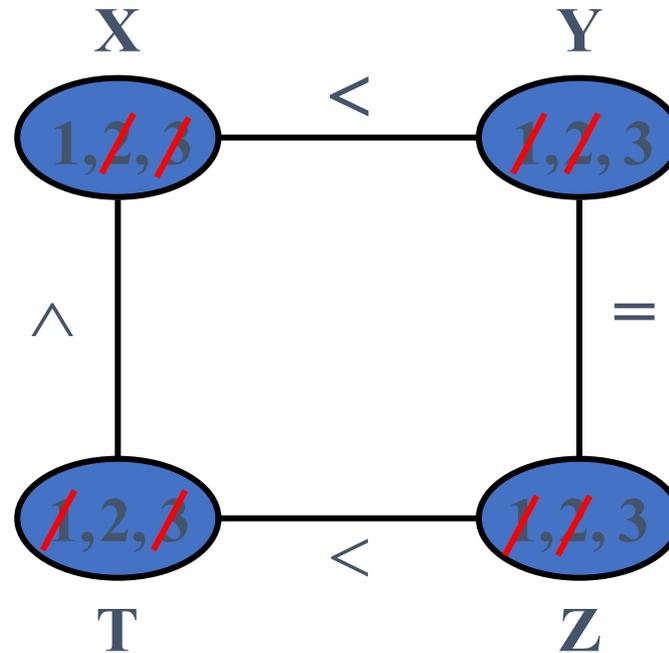
Arc-consistency

$$1 \leq X, Y, Z, T \leq 3$$
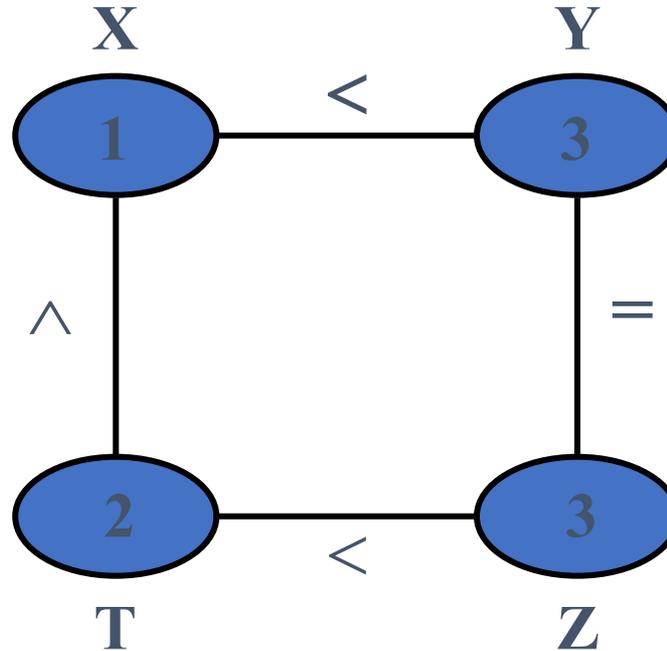$$X < Y$$
$$Y = Z$$
$$T < Z$$
$$X \leq T$$

Arc-consistency

$1 \leq X, Y, Z, T \leq 3$

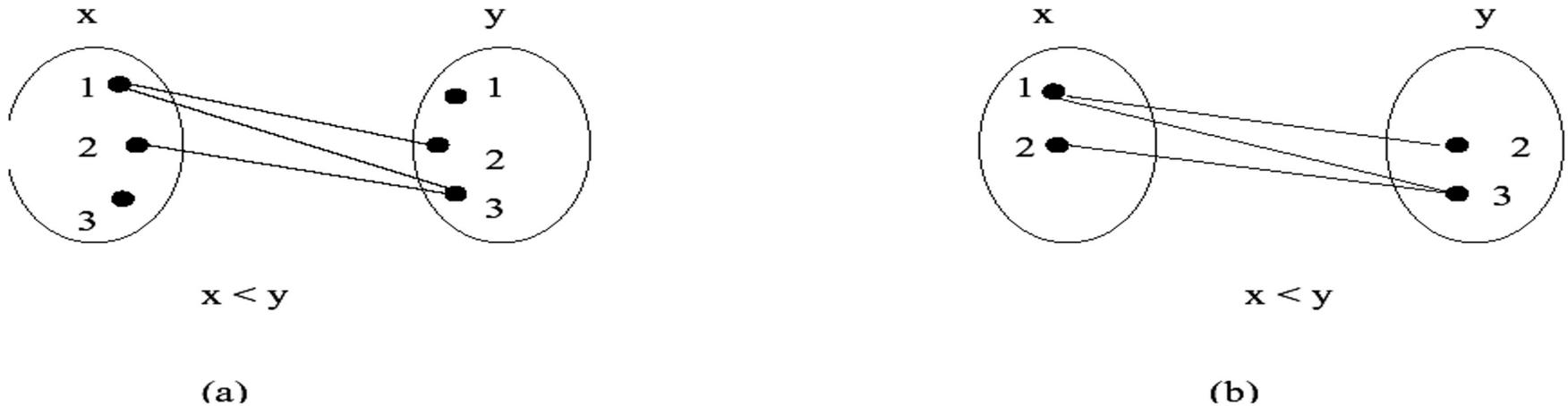$X < Y$

$Y = Z$

$T < Z$

$X \leq T$

Arc-consistency

Figure 3.1: A matching diagram describing the arc-consistency of two variables x and y. In (a) the variables are not arc-consistent. In (b) the domains have been reduced, and the variables are now arc-consistent.

Definition 3.2.2 (arc-consistency) *Given a constraint network* $\mathcal{R} = (\mathcal{X}, \mathcal{D}, \mathcal{C})$, *with* $R_{ij} \in C$, *a variable* x_i *is* arc-consistent *relative to* x_j *if and only if for every value* $a_i \in D_i$ *there exists a value* $a_j \in D_j$ *such that* $(a_i, a_j) \in R_{ij}$. *The subnetwork (alternatively, the arc) defined by* $\{x_i, x_j\}$ *is arc-consistent if and only if* x_i *is arc-consistent relative to* x_j *and* x_j *is arc-consistent relative to* x_i. *A network of constraints is called* arc-consistent *iff all of its arcs (e.g., subnetworks of size 2) are arc-consistent.*

Revise for arc-consistency

$\text{REVISE}((x_i), x_j)$

input: a subnetwork defined by two variables $X = \{x_i, x_j\}$, a distinguished variable x_i, domains: D_i and D_j, and constraint R_{ij}

output: D_i, such that, x_i arc-consistent relative to x_j

1. **for** each $a_i \in D_i$
2.   **if** there is no $a_j \in D_j$ such that $(a_i, a_j) \in R_{ij}$
3.   **then** delete a_i from D_i
4.   **endif**
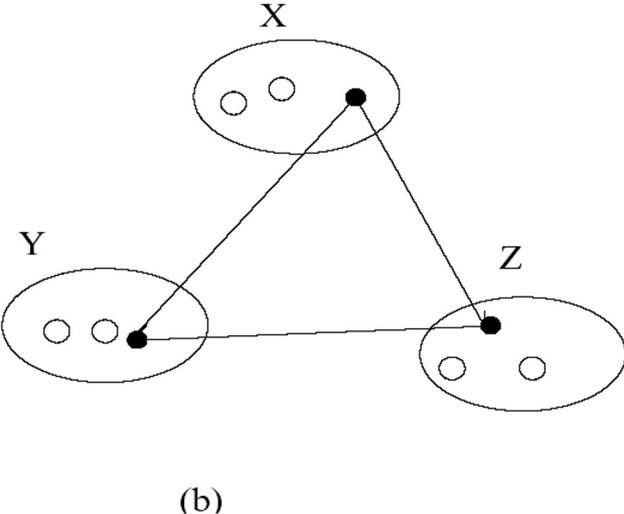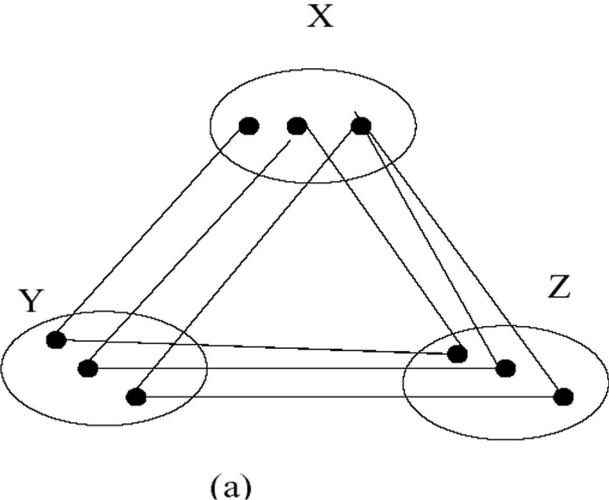5. **endfor**

Figure 3.2: The Revise procedure $\bowtie = \otimes$

$$D_i \leftarrow D_i \cap \pi_i(R_{ij} \bowtie D_j)$$

A matching diagram describing a network of constraints that is not arc-consistent (b) An arc-consistent equivalent network.

(a)

A matching diagram describing a network of constraints that is not arc-consistent (b) An arc-consistent equivalent network.

(a)

(b)
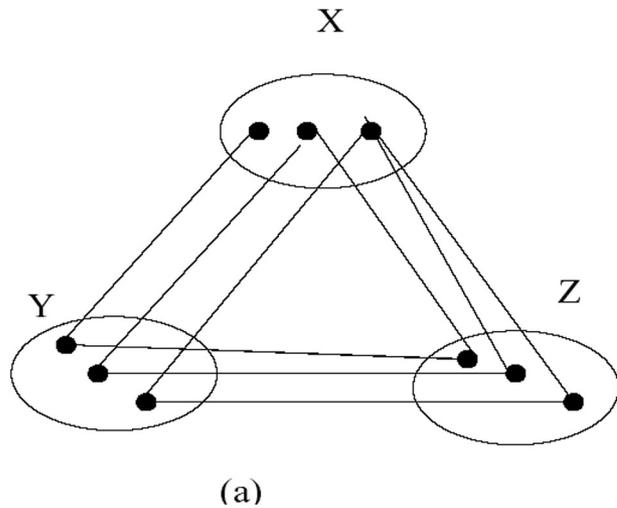
AC-1

AC-1$(\mathcal{R})$

input: a network of constraints $\mathcal{R} = (X, D, C)$
output: $\mathcal{R}'$ which is the loosest arc-consistent network equivalent to $\mathcal{R}$
1. **repeat**
2.   **for** every pair $\{x_i, x_j\}$ that participates in a constraint
3.   Revise$((x_i), x_j)$ (or $D_i \leftarrow D_i \cap \pi_i(R_{ij} \bowtie D_j)$)
4.   Revise$((x_j), x_i)$ (or $D_j \leftarrow D_j \cap \pi_j(R_{ij} \bowtie D_i)$)
5.   **endfor**
6. **until** no domain is changed

Figure 3.4: Arc-consistency-1 (AC-1)

- Complexity (Mackworth and Freuder, 1986): $O(enk^3)$
- e = number of arcs, n variables, k values
- (ek^2, each loop, nk number of loops), best-case = ek,
- Arc-consistency is: $\Omega(ek^2)$

AC-3

AC-3($\mathcal{R}$)

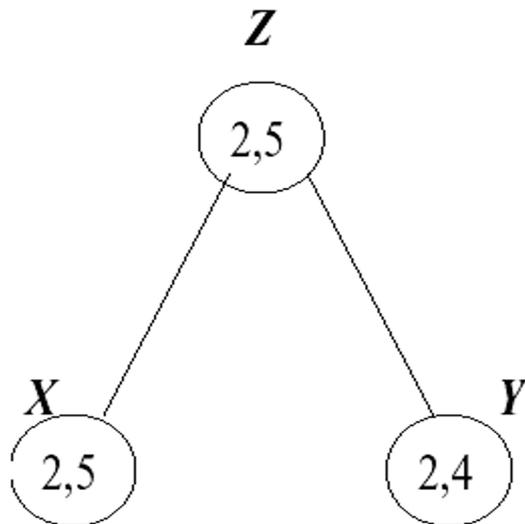**input**: a network of constraints $\mathcal{R} = (X, D, C)$
output: $\mathcal{R}'$ which is the largest arc-consistent network equivalent to $\mathcal{R}$
1. **for** every pair $\{x_i, x_j\}$ that participates in a constraint $R_{ij} \in \mathcal{R}$
2. $queue \leftarrow queue \cup \{(x_i, x_j), (x_j, x_i)\}$
3. **endfor**
4. **while** $queue \neq \{\}$
5.   select and delete (x_i, x_j) from $queue$
6. $Revise((x_i), x_j)$
7.   **if** $Revise((x_i), x_j)$ causes a change in D_i
8.   **then** $queue \leftarrow queue \cup \{(x_k, x_i), i \neq k\}$
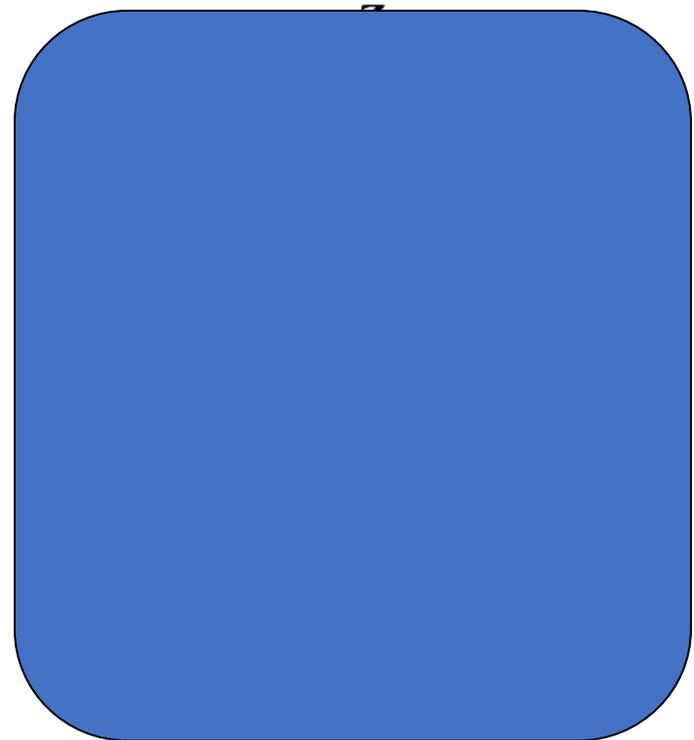9.   **endif**
10. **endwhile**

Figure 3.5: Arc-consistency-3 (AC-3)

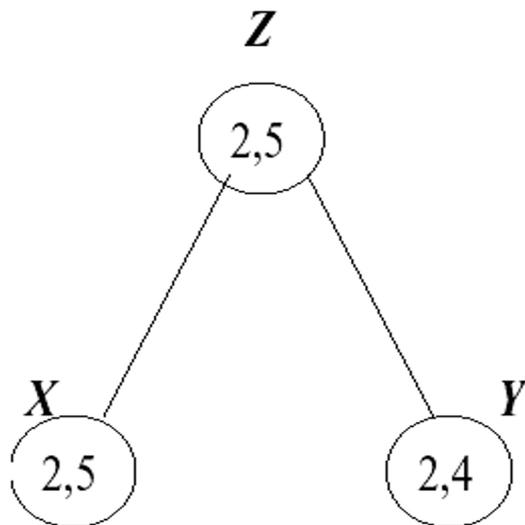- Complexity: $O(ek^3)$ since each arc may be processed in O(2k)
- Best case O(ek),

Example: a 3 variables network with 2 constraints: z divides x and z divides y
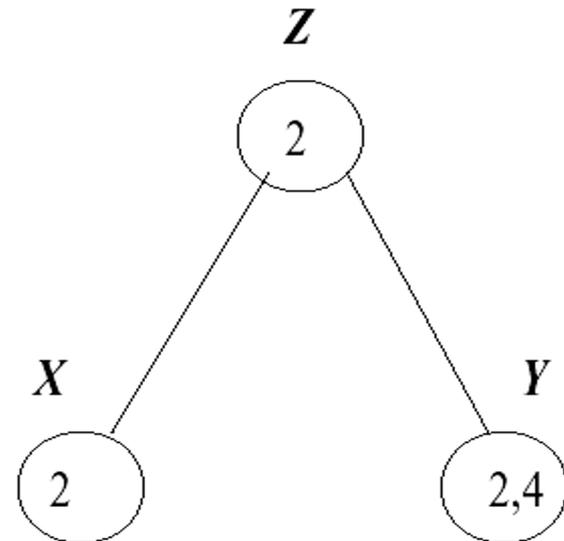## (a) before and (b) after AC-3 is applied.

(a)

Example: A 3 variables network with 2 constraints: z divides x and z divides y
(a) before and (b) after AC-3 is applied.

(a)

(b)

AC-4

$AC\text{-}4(\mathcal{R})$

input: a network of constraints $\mathcal{R}$
output: An arc-consistent network equivalent to $\mathcal{R}$
1. Initialization: $M \leftarrow \emptyset$,
2.   initialize $S_{(x_i,c_i)}$, $counter(i, a_i, j)$ for all R_{ij}
3.   **for** all counters
4.   **if** $counter(x_i, a_i, x_j) = 0$ (if $< x_i, a_i >$ is unsupported by x_j)
5.   **then** add $< x_i, a_i >$ to $LIST$
6.   **endif**
7.   **endfor**
8. **while** $LIST$ is not empty
9.   choose $< x_i, a_i >$ from LIST, remove it, and add it to M
10.   **for** each $< x_j, a_j >$ in $S_{(x_i,a_i)}$
11.   decrement $counter(x_j, a_j, x_i)$
12.   **if** $counter(x_j, a_j, x_i) = 0$
13.   **then** add $< x_j, a_j >$ to $LIST$
14.   **endif**
15.   **endfor**
16. **endwhile**

Figure 3.7: Arc-consistency-4 (AC-4)

• Complexity: $O(ek^2)$

• (Counter is the number of supports to a_i in x_i from x_j. $S_{(xi,ai)}$ is the set of pairs that (x_i, a_i) supports)

Example applying AC-4

Example 3.2.9 Consider the problem in Figure 3.6. Initializing the $S_{(x,a)}$ arrays (indicating all the variable-value pairs that each $<x,a>$ supports), we have :

$S_{(z,2)} = \{<x,2>, <y,2>, <y,4>\}$, $S_{(z,5)} = \{<x,5>\}$, $S_{(x,2)} = \{<z,2>\}$, $S_{(x,5)} = \{<z,5>\}$, $S_{(y,2)} = \{<z,2>\}$, $S_{(y,4)} = \{<z,2>\}$.

For counters we have: $counter(x,2,z) = 1$, $counter(x,5,z) = 1$, $counter(z,2,x) = 1$, $counter(z,5,x) = 1$, $counter(z,2,y) = 2$, $counter(z,5,y) = 0$, $counter(y,2,z) = 1$, $counter(y,4,z) = 1$. (Note that we do not need to add counters between variables that are not directly constrained, such as x and y.) Finally, $List = \{<z,5>\}$, $M = \emptyset$. Once $<z,5>$ is removed from $List$ and placed in M, the counter of $<x,5>$ is updated to $counter(x,5,z) = 0$, and $<x,5>$ is placed in $List$. Then, $<x,5>$ is removed from $List$ and placed in M. Since the only value it supports is $<z,5>$ and since $<z,5>$ is already in M, the $List$ remains empty and the process stops. $\square$

Distributed arc-consistency (Constraint propagation)

- Implement AC-1 distributedly.

- $h_{j \to i}$ node x_j sends the message to node x_i

- Node x_i updates its domain:

- Messages can be sent asynchronously or scheduled in a topological order

$$D_i \leftarrow D_i \cap \pi_i(R_{ij} \otimes D_j)$$

$$\boxed{h_{j \to i}} \leftarrow \pi_i(R_{ij} \otimes D_j)$$

$$D_i \leftarrow D_i \cap \pi_i(R_{ij} \otimes D_j) =$$

$$D_i \leftarrow D_i \cap \boxed{h_{j \to i}}$$

Exercise: make the following network arc-consistent

- Draw the network's primal and dual constraint graph
- Network =
  - Domains {1,2,3,4}
  - Constraints: y < x, z < y, t < z, f<t, x<=t+1, Y<f+2

Arc-consistency Algorithms

- **AC-1**: brute-force, distributed $\qquad\qquad O(nek^3)$
- **AC-3**, queue-based $\qquad\qquad\qquad\quad O(ek^3)$
- **AC-4**, context-based, optimal $\qquad\qquad O(ek^2)$
- **AC-5,6,7**,…. Good in special cases
- **Important:** applied at every node of search
- (n number of variables, e=#constraints, k=domain size)
- Mackworth and Freuder (1977,1983), Mohr and Anderson, (1985)…

Constraint tightness analysis
t = number of tuples bounding a constraint

- **AC-1**: brute-force,

- **AC-3**, queue-based

- **AC-4**, context-based, optimal

- **AC-5,6,7**,…. Good in special cases

- **Important:** applied at every node of search

- (n number of variables, e=#constraints, k=domain size)

- Mackworth and Freuder (1977,1983), Mohr and Anderson, (1985)…

$$O(nek^3) \quad O(nekt)$$

$$O(ek^3) \quad O(ekt)$$

$$O(et)$$

Is arc-consistency enough?

- Example: a triangle graph-coloring with 2 values.
  - Is it arc-consistent?
  - Is it consistent?
- It is not path, or 3-consistent.

Outline

- Arc-consistency algorithms
- **Path-consistency and i-consistency**
- Arc-consistency, Generalized arc-consistency, relation arc-consistency
- Global and bound consistency
- Distributed (generalized) arc-consistency
- Consistency operators: join, resolution, Gausian elimination