St. Francis Xavier University

CSCI-564
CONSTRAINT PROCESSING AND HEURISTIC SEARCH

LECTURE 22 – CONSISTENCY

Dr. Jean-Alexis Delamer

Recap

- Constraint inference:
  - New constraints can be inferred from an initial set of constraints.
- The new constraints might:
  - Create constraints between variables that were not initially constrained.
  - Tighten existing constraints.
- We can approximate a relation with a binary projection network.

Arc-Consistency

- Minimal network has the following local consistency property:
  - Any value in the domain of a single variable can be extended consistently by any other variable.

- This property is called arc-consistency.

Basic Consistency

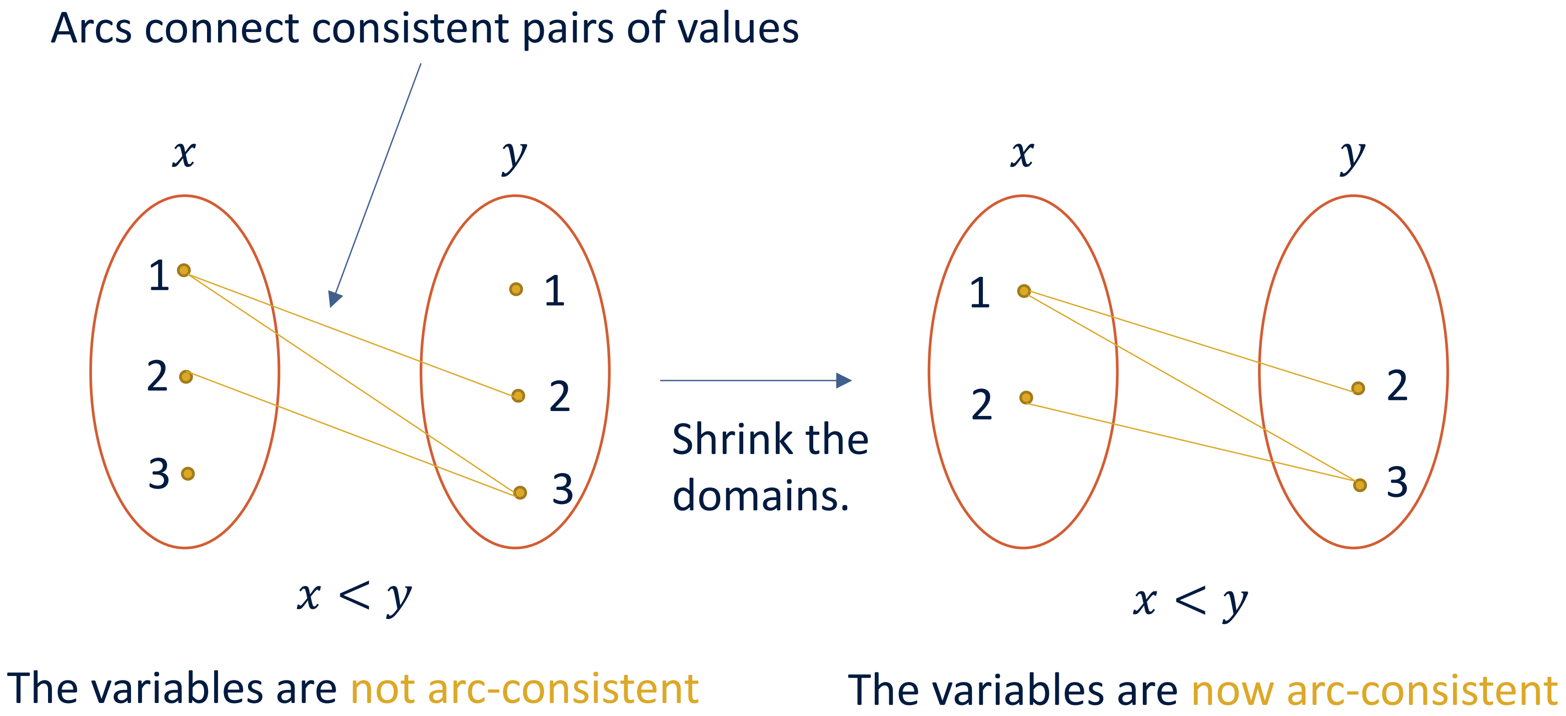
- We generally consider two types of consistency.
- Given variables i, j, k with values x, y, z, the predicate $P_{ijk}(x, y, z)$ is true iff the 3-tuple $(x, y, z) \in R_{ijk}$

- Node consistency: checking $P_i(x)$
- Arc consistency: checking $P_{ij}(x, y)$

Basic consistency

Consider:

- Two variables x and y
- whose domains are $D_x = D_y = \{1,2,3\}$.
- And a constraint $R_{xy} = x < y$



This type of diagram is not a constraint graph network!

Arc-Consistency

- Definition Arc-consistency:
  - Given a constraint network $\rho = (X, D, C)$, with $R_{ij} \in C$, a variable x_i is arc-consistent relative to x_j iff 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 constraint R_{ij} is arc consistent iff 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 are arc-consistent

Revise

- A first algorithm for arc-consistency.

```
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 is arc-consistent relative to x_j.

For each a_i ∈ D_i
    If there is no a_j ∈ D_j such that (a_i, a_j) ∈ R_ij
        Delete a_i from D_i
    EndIf
EndFor
```

Revise

- What is the complexity of the revise procedure?
  - Each value in D_i is compared, in the worst case, with each value in D_j.
  - The complexity of Revise is $O(n^2)$, where n bounds the domain size.

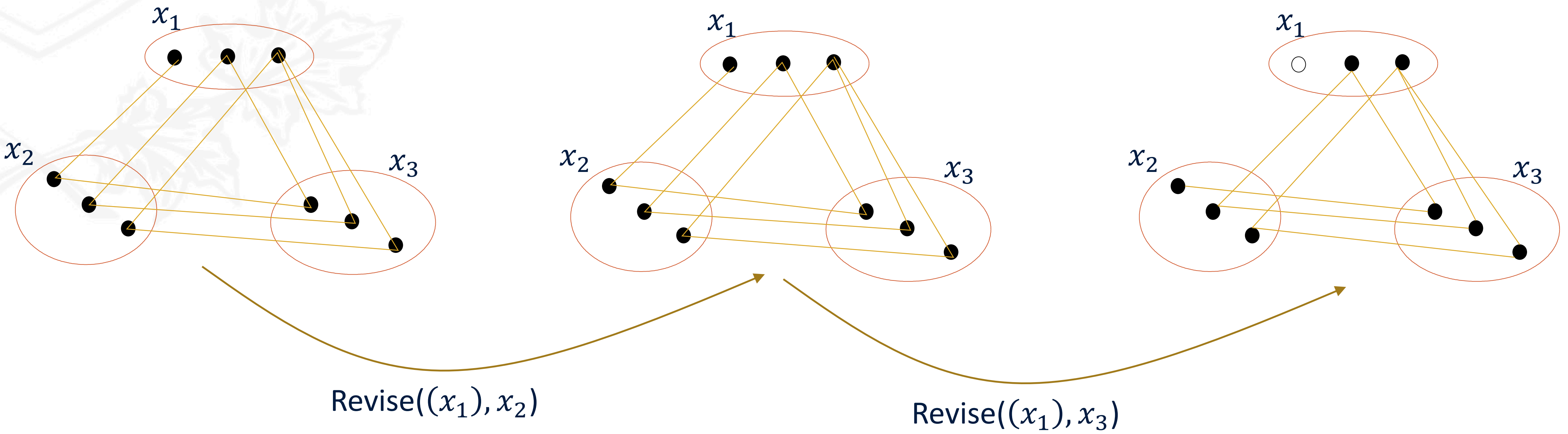
- Revise can also be described using composition:
  - $D_i \leftarrow D_i \cap \pi_i(R_{ij} \bowtie D_j)$

- Arc-consistency of a whole network is accomplished by applying Revise to all pairs of variables.

Revise

- Is applying Revise to all pair of variables enough?
  - No, Revise may need to be applied more than once.

Then x_1 is not arc-consistent with x_2 anymore.



AC-1

- First algorithm for AC-1:

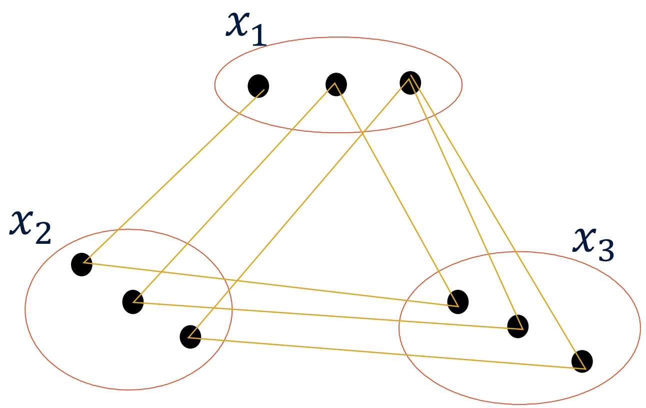
```
AC-1(ρ)
Input: a network of constraints ρ = (X,D,C).
Output: ρ′ which s the loosest arc-consistent network
equivalent to ρ

Repeat
    For each {x_i, x_j} ∈ ρ
        REVISE((x_i), x_j)
        REVISE((x_j), x_i)
    EndFor
Until no domain is changed
```

What is the complexity?

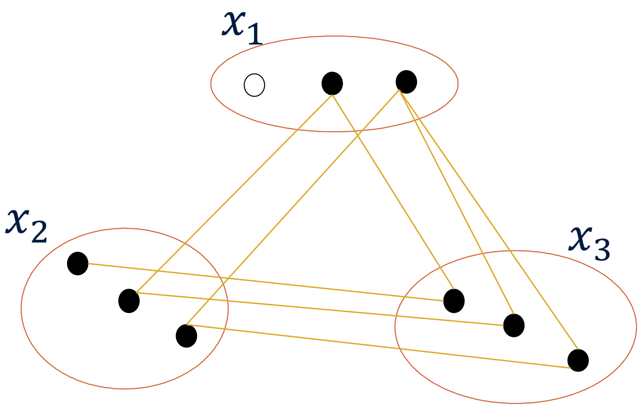
- e = number of arcs, k variables and n values (domain size).
- Arc consistency is $O(ekn^3)$

Example

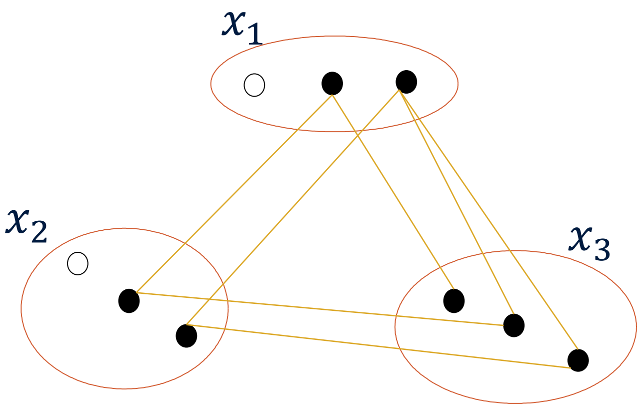


StFX

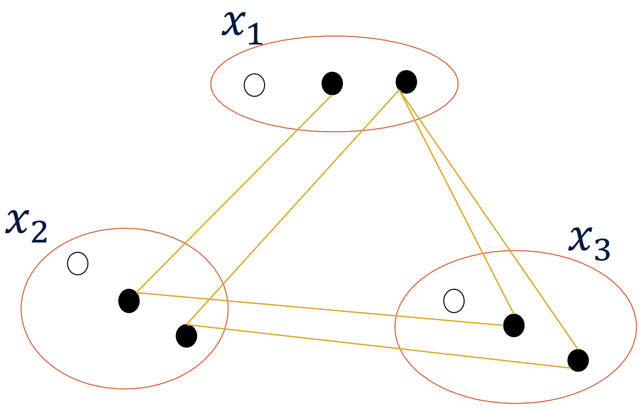
Example



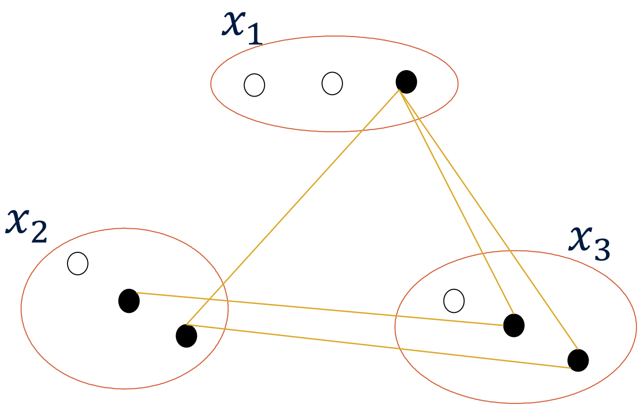
Example



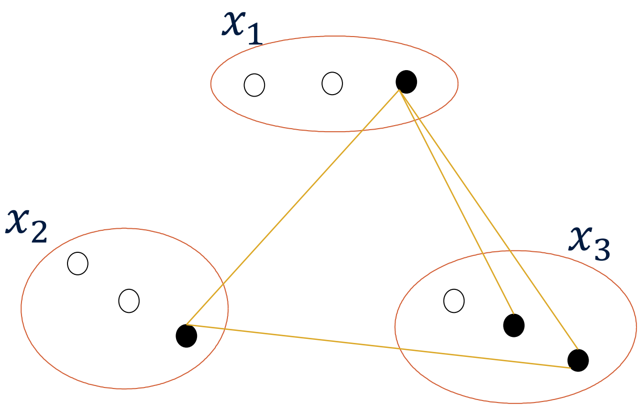
Example



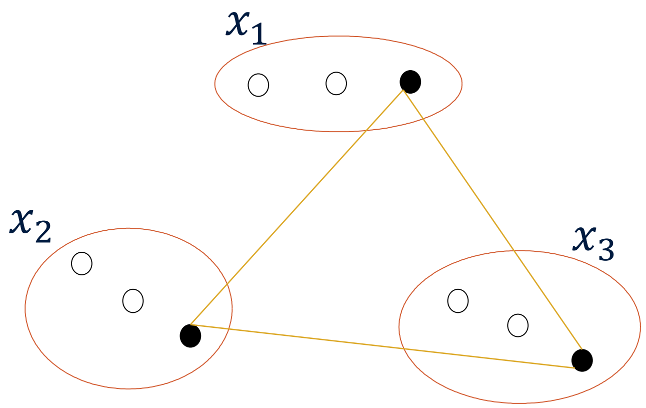
Example



Example

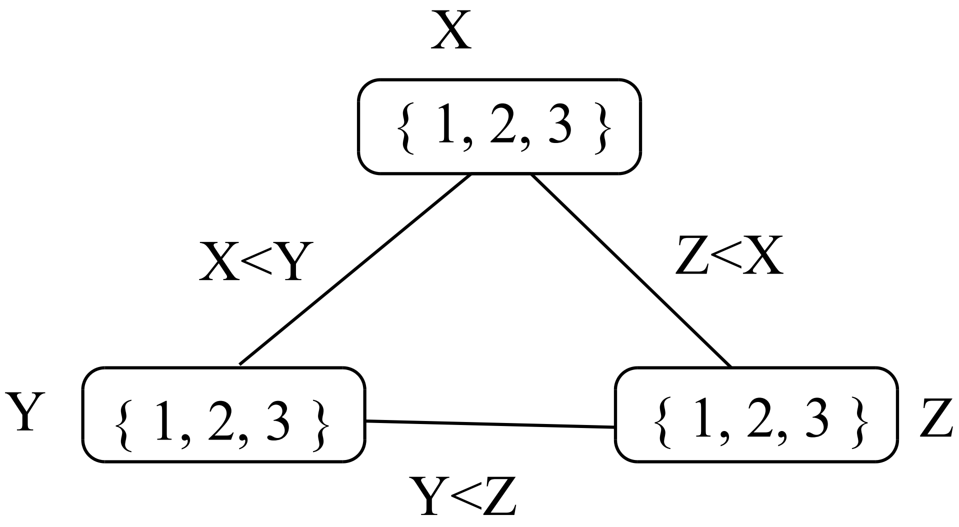


Example



AC-1

- AC-1 may discover inconsistency.



AC-3

- Considering the complexity of AC-1 it is necessary to improve the performance.
  - No need to process all the constraints if only a few domains were reduced in the previous loop.

- AC-3:
  - Creates a queue of constraints to be processed.
  - Once a constraint has been processed remove the constraint from the queue.
  - Place back the constraint only if the domain of the second variable is modified.

AC-3

```
AC-3(ρ)
Input: a network of constraints ρ = (X,D,C).
Output: ρ′ which s the loosest arc-consistent network
equivalent to ρ

For each pair {x_i, x_j} that participates in a constraint R_ij ∈ ρ
    queue←queue∪ {(x_i, x_j), (x_j, x_i)}
EndFor
While queue≠ ∅
    select and delete (x_i, x_j) from queue
    REVISE((x_i), x_j)
    If REVISE((x_i), x_j) causes a change in D_i
        queue←queue∪ {(x_k, x_i), k ≠ i, k ≠ j}
    EndIf
EndWhile
```

Example

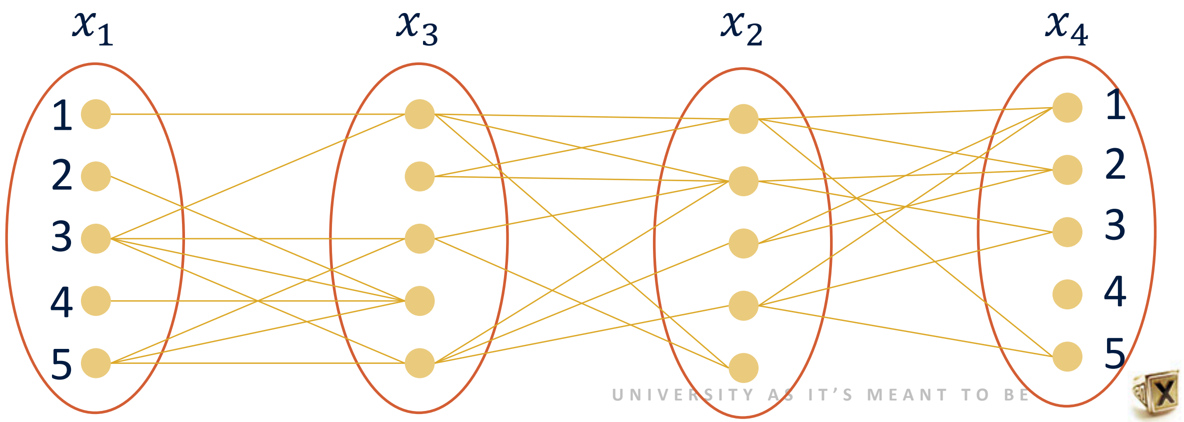
$$D_{x_1} = \{1, 2, 3, 4, 5\}$$
$$D_{x_2} = \{1, 2, 3, 4, 5\}$$
$$D_{x_3} = \{1, 2, 3, 4, 5\}$$
$$D_{x_4} = \{1, 2, 3, 4, 5\}$$

R_{23}= {(2, 2), (4, 5), (2, 5), (3, 5), (2, 3), (5, 1), (1, 2), (5, 3), (2, 1), (1, 1)}

R_{13}= {(5, 5), (2, 4), (3, 5), (3, 3), (5, 3), (4, 4), (5, 4), (3, 4), (1, 1), (3, 1)}

R_{24}= {(1, 2), (3, 2), (3, 1), (4, 5), (2, 3), (4, 1), (1, 1), (4, 3), (2, 2), (1, 5)}

Queue = $\{R_{24}, R_{42}, R_{13}, R_{23}, R_{31}, R_{32}\}$



Example

$$\begin{aligned} D_{x_1} &= \{1,2,3,4,5\} \\ D_{x_2} &= \{1,2,3,4,5\} \\ D_{x_3} &= \{1,2,3,4,5\} \\ D_{x_4} &= \{1,2,3,4,5\} \end{aligned}$$

We modified x_2, so we need to add the other constraint with x_2 in the queue.

R_{23}= {(2, 2), (4, 5), (2, 5), (3, 5), (2, 3), (5, 1), (1, 2), (5, 3), (2, 1), (1, 1)}

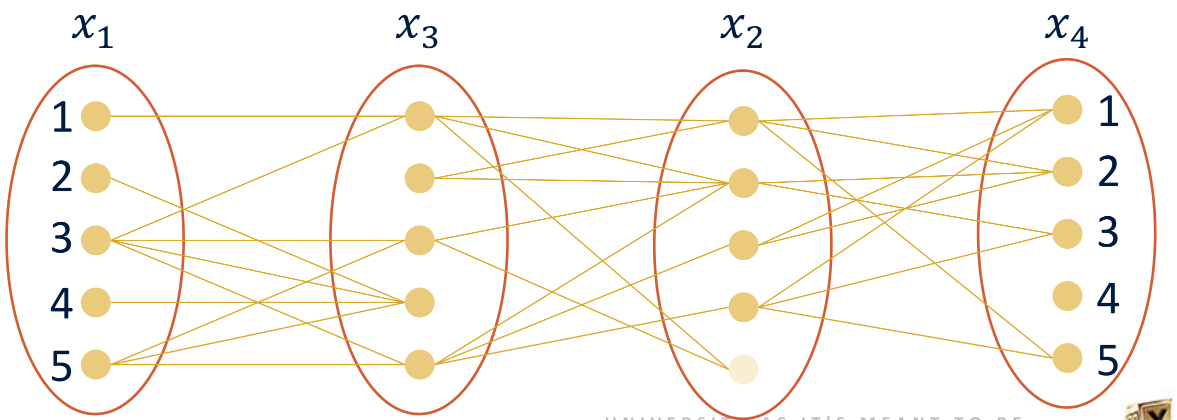
R_{13}= {(5, 5), (2, 4), (3, 5), (3, 3), (5, 3), (4, 4), (5, 4), (3, 4), (1, 1), (3, 1)}

R_{24}= {(1, 2), (3, 2), (3, 1), (4, 5), (2, 3), (4, 1), (1, 1), (4, 3), (2, 2), (1, 5)}

Queue = $\{R_{42}, R_{13}, R_{23}, R_{31}, R_{32}\}$

Revise(x_2, x_4) $D_{x_2} \leftarrow D_{x_2} \backslash \{5\} = \{1,2,3,4\}$

In this case, R_{32} is already inside.



Example

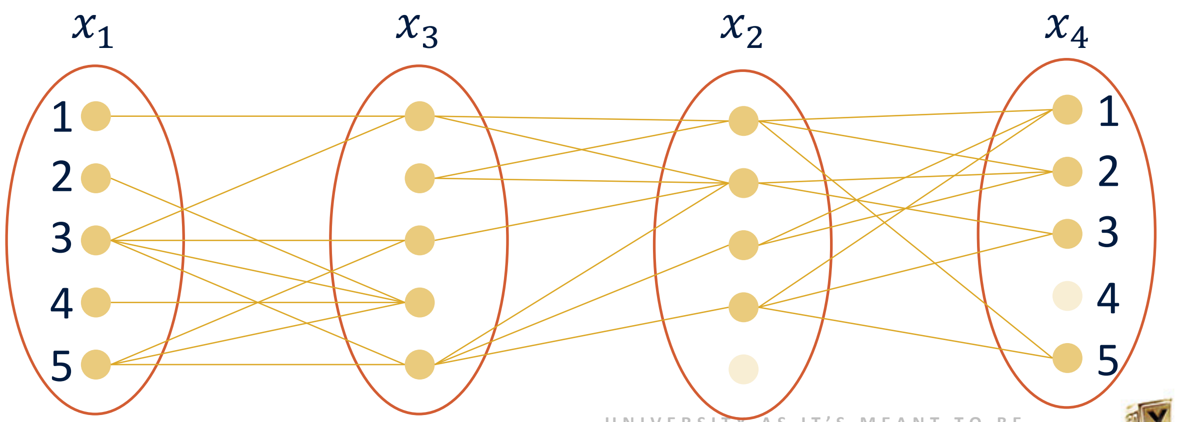
$$
\begin{aligned}
D_{x_1} &= \{1,2,3,4,5\}\\
D_{x_2} &= \{1,2,3,4\}\\
D_{x_3} &= \{1,2,3,4,5\}\\
D_{x_4} &= \{1,2,3,\cancel{4},5\}
\end{aligned}
$$

We modified x_4, so we need to add the other constraint with x_4 in the queue.

R_{23}= {(2, 2), (4, 5), (2, 5), (3, 5), (2, 3), (5, 1), (1, 2), (5, 3), (2, 1), (1, 1)}

R_{13}= {(5, 5), (2, 4), (3, 5), (3, 3), (5, 3), (4, 4), (5, 4), (3, 4), (1, 1), (3, 1)}

R_{24}= {(1, 2), (3, 2), (3, 1), (4, 5), (2, 3), (4, 1), (1, 1), (4, 3), (2, 2), (1, 5)}



Queue = $\{R_{13}, R_{23}, R_{31}, R_{32}\}$

Revise(x_4, x_2) $D_{x_4} \leftarrow D_{x_4} \backslash \{4\} = \{1,2,3,5\}$

In this case, it was the only constraint with x_4

Example

$$
\begin{aligned}
D_{x_1} &= \{1,2,3,4,5\}\\
D_{x_2} &= \{1,2,3,4\}\\
D_{x_3} &= \{1,2,3,4,5\}\\
D_{x_4} &= \{1,2,3,5\}
\end{aligned}
$$

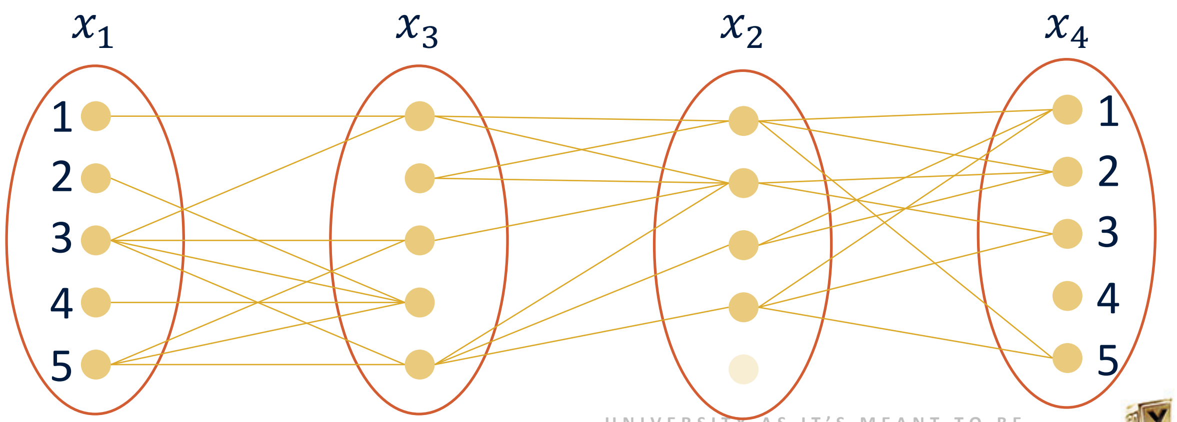
R_{23}= {(2, 2), (4, 5), (2, 5), (3, 5), (2, 3), (5, 1), (1, 2), (5, 3), (2, 1), (1, 1)}

R_{13}= {(5, 5), (2, 4), (3, 5), (3, 3), (5, 3), (4, 4), (5, 4), (3, 4), (1, 1), (3, 1)}

R_{24}= {(1, 2), (3, 2), (3, 1), (4, 5), (2, 3), (4, 1), (1, 1), (4, 3), (2, 2), (1, 5)}

Queue = $\{R_{23}, R_{31}, R_{32}\}$

Revise(x_1, x_3) $D_{x_1} \leftarrow D_{x_1} = \{1,2,3,4,5\}$



Example

$$
\begin{aligned}
D_{x_1} &= \{1, 2, 3, 4, 5\}\\
D_{x_2} &= \{1, 2, 3, 4\}\\
D_{x_3} &= \{1, 2, 3, 4, 5\}\\
D_{x_4} &= \{1, 2, 3, 5\}
\end{aligned}
$$

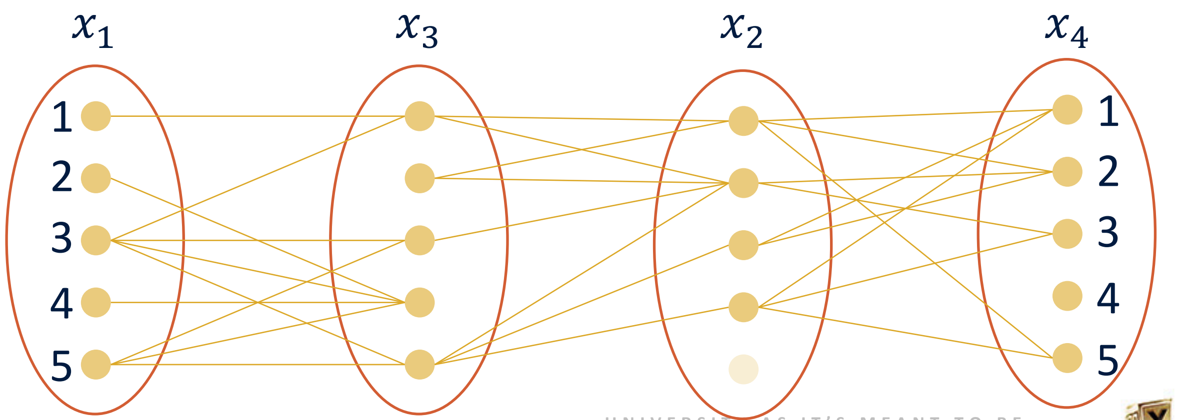
R_{23}= {(2, 2), (4, 5), (2, 5), (3, 5), (2, 3), (5, 1), (1, 2), (5, 3), (2, 1), (1, 1)}

R_{13}= {(5, 5), (2, 4), (3, 5), (3, 3), (5, 3), (4, 4), (5, 4), (3, 4), (1, 1), (3, 1)}

R_{24}= {(1, 2), (3, 2), (3, 1), (4, 5), (2, 3), (4, 1), (1, 1), (4, 3), (2, 2), (1, 5)}

Queue = {R_{31}, R_{32}}

Revise(x_2, x_3) $D_{x_2} \leftarrow D_{x_2} = \{1,2,3,4\}$



Example

$$
\begin{aligned}
D_{x_1} &= \{1, 2, 3, 4, 5\}\\
D_{x_2} &= \{1, 2, 3, 4\}\\
D_{x_3} &= \{1, \cancel{2}, 3, 4, 5\}\\
D_{x_4} &= \{1, 2, 3, 5\}
\end{aligned}
$$

We modified x_3, so we need to add the other constraint with x_3 in the queue.

R_{23}= {(2, 2), (4, 5), (2, 5), (3, 5), (2, 3), (5, 1), (1, 2), (5, 3), (2, 1), (1, 1)}

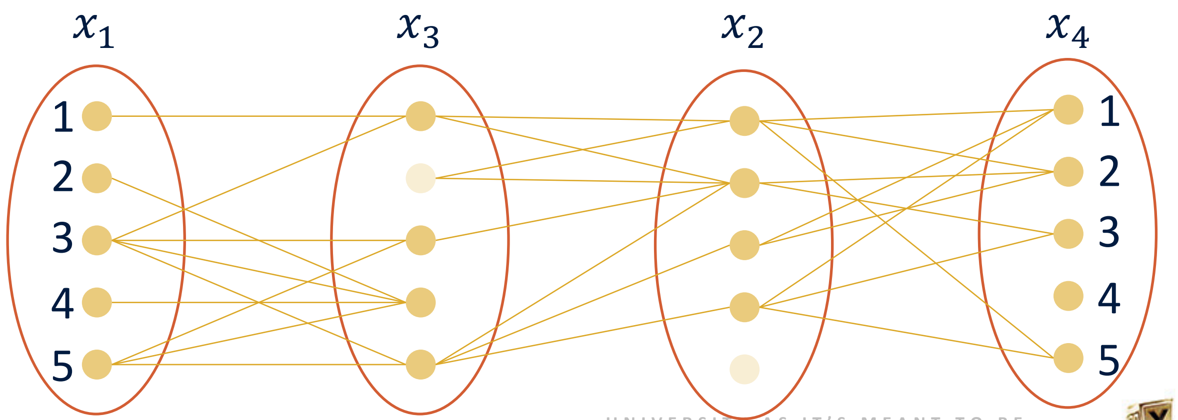
R_{13}= {(5, 5), (2, 4), (3, 5), (3, 3), (5, 3), (4, 4), (5, 4), (3, 4), (1, 1), (3, 1)}

R_{24}= {(1, 2), (3, 2), (3, 1), (4, 5), (2, 3), (4, 1), (1, 1), (4, 3), (2, 2), (1, 5)}

R_{23}

Queue = $\{R_{32}\}$

Revise(x_3, x_1) $D_{x_3} \leftarrow D_{x_3} \backslash \{2\} = \{1,3,4,5\}$



Example

$$
\begin{aligned}
D_{x_1} &= \{1, 2, 3, 4, 5\} \\
D_{x_2} &= \{1, 2, 3, 4\} \\
D_{x_3} &= \{1, 3, 4, 5\} \\
D_{x_4} &= \{1, 2, 3, 5\}
\end{aligned}
$$

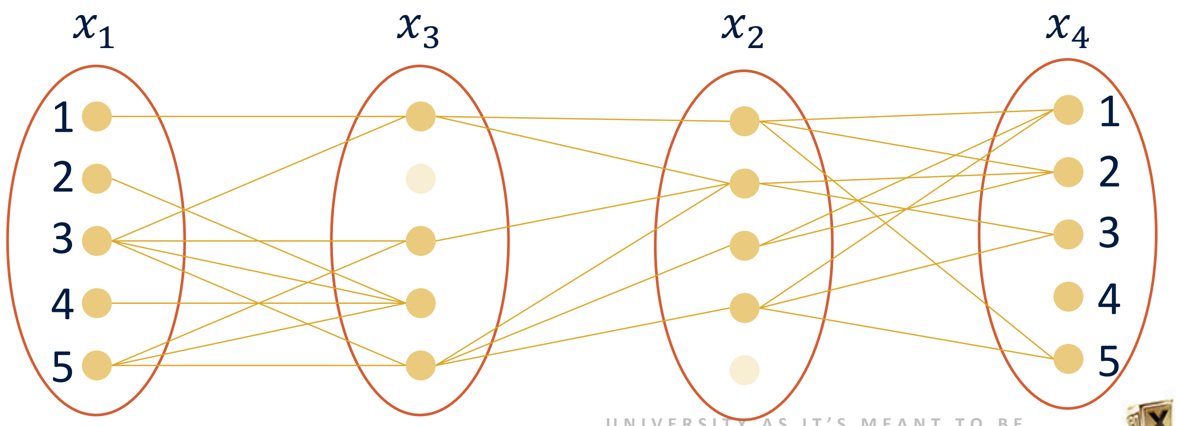
R_{23}= {(2, 2), (4, 5), (2, 5), (3, 5), (2, 3), (5, 1), (1, 2), (5, 3), (2, 1), (1, 1)}

R_{13}= {(5, 5), (2, 4), (3, 5), (3, 3), (5, 3), (4, 4), (5, 4), (3, 4), (1, 1), (3, 1)}

R_{24}= {(1, 2), (3, 2), (3, 1), (4, 5), (2, 3), (4, 1), (1, 1), (4, 3), (2, 2), (1, 5)}

Queue = $\{R_{32}\}$

Revise(x_2, x_3) $D_{x_2} \leftarrow D_{x_2}$



Example

$$
\begin{aligned}
D_{x_1} &= \{1, 2, 3, 4, 5\} \\
D_{x_2} &= \{1, 2, 3, 4\} \\
D_{x_3} &= \{1, 3, \cancel{4}, 5\} \\
D_{x_4} &= \{1, 2, 3, 5\}
\end{aligned}
$$

We modified x_3, so we need to add the other constraint with 3 in the queue.

R_{23}= {(2, 2), (4, 5), (2, 5), (3, 5), (2, 3), (5, 1), (1, 2), (5, 3), (2, 1), (1, 1)}

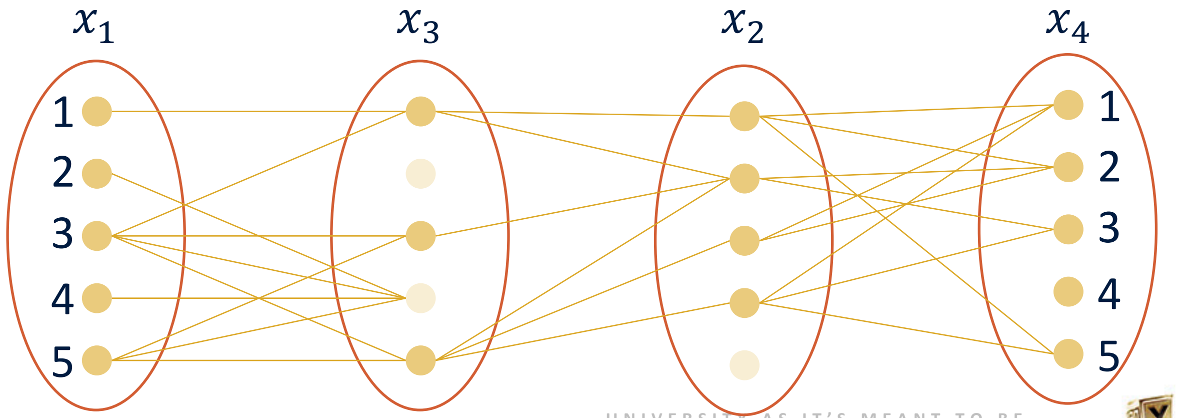
R_{13}= {(5, 5), (2, 4), (3, 5), (3, 3), (5, 3), (4, 4), (5, 4), (3, 4), (1, 1), (3, 1)}

R_{24}= {(1, 2), (3, 2), (3, 1), (4, 5), (2, 3), (4, 1), (1, 1), (4, 3), (2, 2), (1, 5)}

R_{13}

Queue = {}

Revise(x_3, x_2) $D_{x_3} \leftarrow D_{x_3} \backslash \{4\} = \{1,3,5\}$



Example

$$
\begin{aligned}
D_{x_1} &= \{1, \cancel{2}, 3, \cancel{4}, 5\}\\
D_{x_2} &= \{1, 2, 3, 4\}\\
D_{x_3} &= \{1, 3, 5\}\\
D_{x_4} &= \{1, 2, 3, 5\}
\end{aligned}
$$

We modified x_1, so we need to add the other constraint with x_1 in the queue.

R_{23}= {(2, 2), (4, 5), (2, 5), (3, 5), (2, 3), (5, 1), (1, 2), (5, 3), (2, 1), (1, 1)}

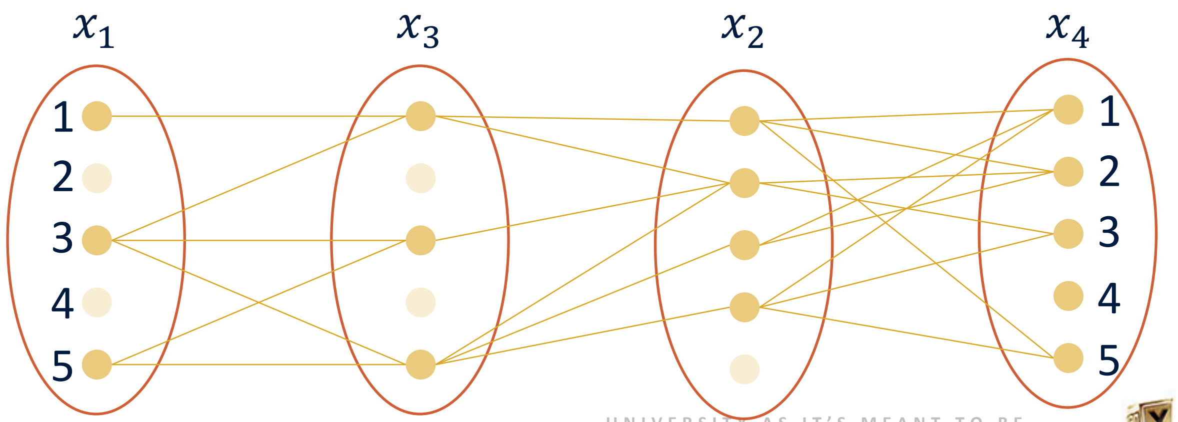
R_{13}= {(5, 5), (2, 4), (3, 5), (3, 3), (5, 3), (4, 4), (5, 4), (3, 4), (1, 1), (3, 1)}

R_{24}= {(1, 2), (3, 2), (3, 1), (4, 5), (2, 3), (4, 1), (1, 1), (4, 3), (2, 2), (1, 5)}

Queue = {}

Revise(x_1, x_3) $\quad D_{x_1} \leftarrow D_{x_1} \backslash \{2, 4\} = \{1,3,5\}$

In this case, it was the only constraint with x_1



Example

$$
\begin{aligned}
D_{x_1} &= \{1,3,5\} \\
D_{x_2} &= \{1,2,3,4\} \\
D_{x_3} &= \{1,3,5\} \\
D_{x_4} &= \{1,2,3,5\}
\end{aligned}
$$

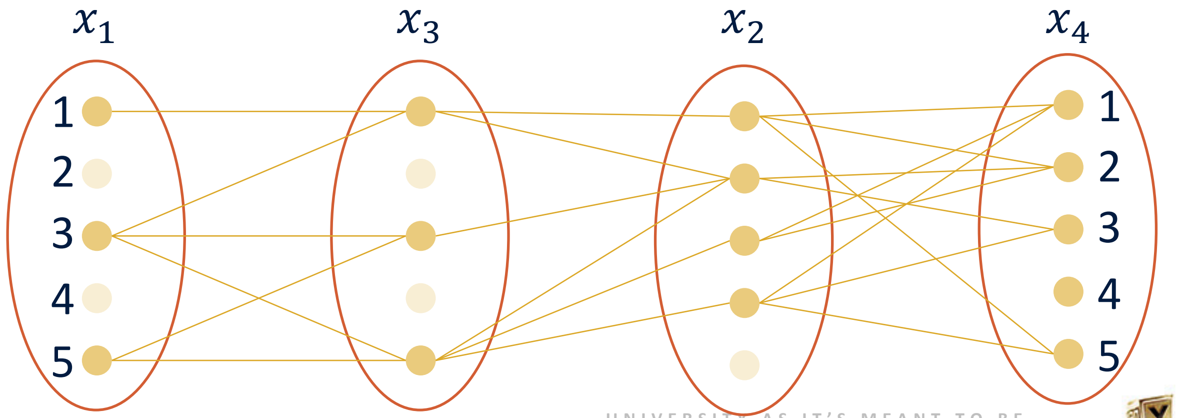
R_{23}= {(2, 2), (4, 5), (2, 5), (3, 5), (2, 3), (5, 1), (1, 2), (5, 3), (2, 1), (1, 1)}

R_{13}= {(5, 5), (2, 4), (3, 5), (3, 3), (5, 3), (4, 4), (5, 4), (3, 4), (1, 1), (3, 1)}

R_{24}= {(1, 2), (3, 2), (3, 1), (4, 5), (2, 3), (4, 1), (1, 1), (4, 3), (2, 2), (1, 5)}

Queue = {}

Queue empty, it’s the end.



AC-3

- The complexity of AC-3 is $O(en^3)$.
  - Each constraint are processed at most $2n$ times where n bounds the domain size.
    - Why?
    - Each time it is reintroduced into the queue, the domain of a variable has been reduced by at least 1.
    - There are at most $2n$ values.
  - There are e binary constraints.
    - Processing each one takes $O(n^2)$.

AC-3

- Is the AC-3 optimal?
  - It seems hat no algorithm can do better than $O(en^2)$.
    - The worst-case of just verifying arc-consistency requires en^2 operations.
  - No, AC-3 is not optimal.

- An update as been proposed: AC-4
  - AC-4 is optimal.
  - It does not use Revise or the composition operator.

AC-4

- What are the differences?
  - It associates each value a_i in the domain of x_i with the number of support from variable x_j.
    - Number of values in x_j that are consistent with a_i.
  - A value a_i is removed from D_i if it has no support.
  - It maintains:
    - A list of unsupported variable-value pairs.
    - A counter array of supports for a_i from x_j.
    - An array $S_{(x_j, a_j)}$ that points to all values in other variables supported by (x_j, a_j)

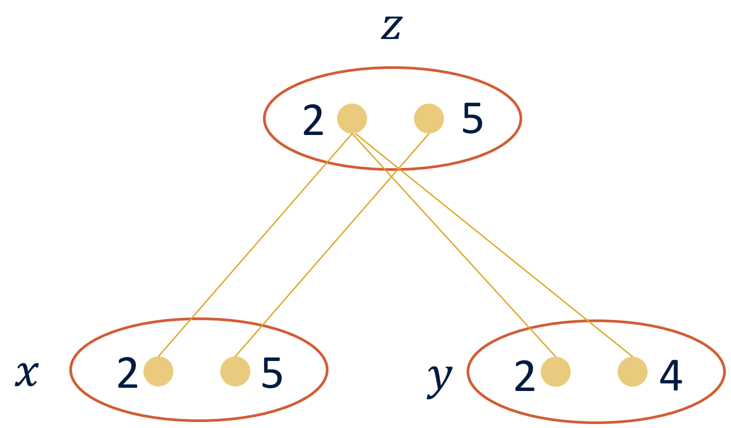
AC-4

```
AC-4(ρ)
Input: a network of constraints ρ = (X,D,C).
Output: ρ′ which s the loosest arc-consistent network equivalent to ρ

M ← ∅
List← ∅
Initialize S(x_i,c_i), counter(i,a_i,j) for all R_ij
ForEach counters
    If counter(x_i,a_i,x_j) = 0
        List← (x_i,a_i)
    EndIf
EndFor
While List is not empty
    choose (x_i,a_i) from List, remove it and add it to M
    ForEach (x_j,a_j) in S(x_i,a_i)
        decrement counter(x_j,a_j,x_i)
        If counter(x_j,a_j,x_i) = 0
            List← (x_j,a_j)
        EndIf
    EndFor
EndWhile
```

Example

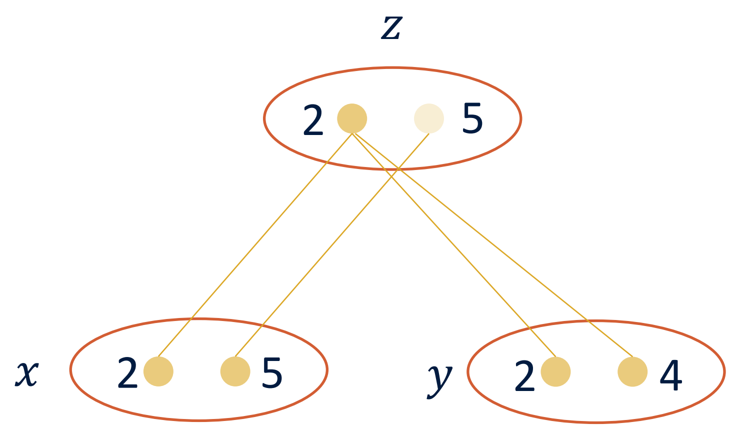
- $S_{(z,2)} = \{(x,2),(y,2),(y,4)\}, S_{(z,5)} = \{(x,5)\}, S_{(x,2)} = \{(z,2)\},\ \ S_{(x,5)} = \{(z,5)\}, S_{(y,2)} = \{(z,2)\}, S_{(y,4)} = \{(z,2)\}$
- $counter(x,2,z) = 1, counter(x,5,z) = 1, counter(z,2,x) = 1,\ counter(z,5,x) = 1, counter(z,2,y) = 1, counter(z,5,y) = 0,\ counter(y,2,z) = 1, counter(y,4,z) = 1$
- $List = \{(z,5)\}$
- $M = \emptyset$



Example

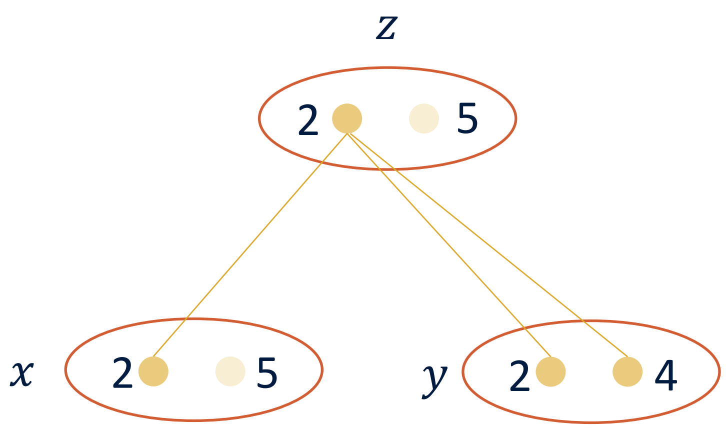
- $S_{(z,2)} = \{(x,2),(y,2),(y,4)\}, S_{(z,5)} = \{(x,5)\}, S_{(x,2)} = \{(z,2)\},\ \ S_{(x,5)} = \{(z,5)\}, S_{(y,2)} = \{(z,2)\}, S_{(y,4)} = \{(z,2)\}$
- $counter(x,2,z) = 1, counter(x,5,z) = 0, counter(z,2,x) = 1,\ counter(z,5,x) = 1, counter(z,2,y) = 1, counter(z,5,y) = 0,\ counter(y,2,z) = 1, counter(y,4,z) = 1$
- $List = \{\}$
- $M = \{(z,5)\}$

$(x,5)$



Example

- $S_{(z,2)} = \{(x,2),(y,2),(y,4)\}, S_{(z,5)} = \{(x,5)\}, S_{(x,2)} = \{(z,2)\},\ \ S_{(x,5)} = \{(z,5)\}, S_{(y,2)} = \{(z,2)\}, S_{(y,4)} = \{(z,2)\}$
- $counter(x,2,z) = 1, counter(x,5,z) = 0, counter(z,2,x) = 1,\ counter(z,5,x) = 1, counter(z,2,y) = 1, counter(z,5,y) = 0,\ counter(y,2,z) = 1, counter(y,4,z) = 1$
- $List = \{\}$
- $M = \{(z,5),(x,5)\}$



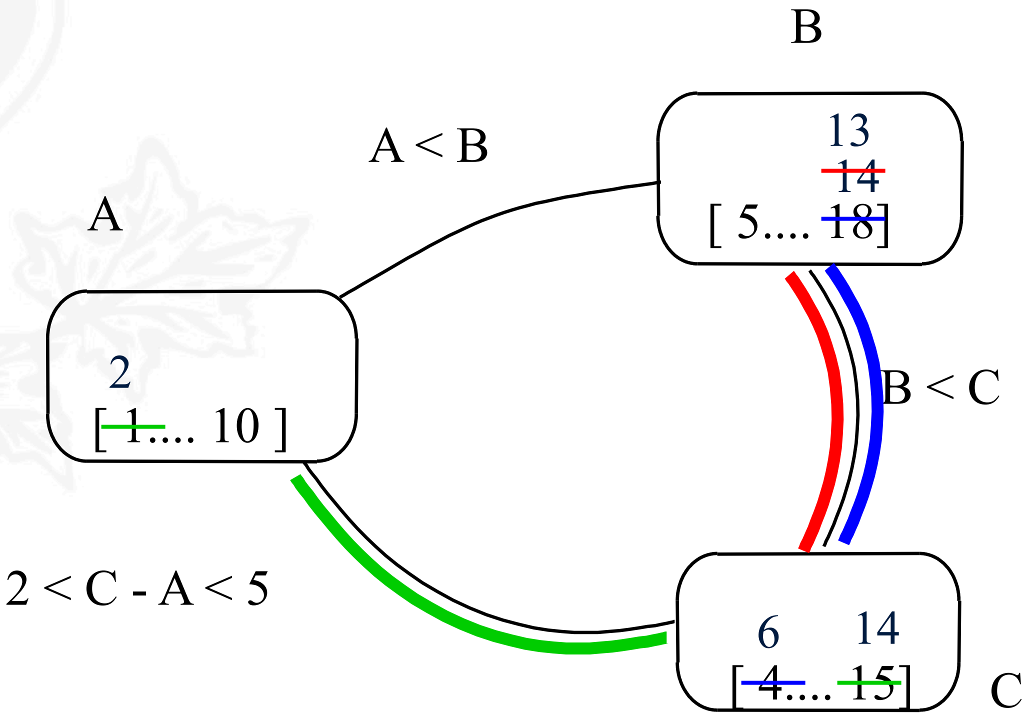
Nothing is added to List, the algorithm stops.

Arc-consistency Algorithms

- AC-1: brute-force $O(ken^3)$
- AC-3: queue-based $O(en^3)$
- AC-4: context-based, optimal $O(en^2)$
- AC-5,6,7,… Good in special cases

Constraint checking

- Can be used to propagate the constraints.



1- B: [5 .. 14]

C: [6 .. 15]

2- A: [2 .. 10]

C: [6 .. 14]

3- B: [5 .. 13]

Is Arc-consistency enough?

- Example: a triangle graph-coloring with 2 values.
  - Is it arc-consistent?
  - Is it consistent?

Yes

- It is not path, or 3-consistent.

No