CompSci 275, CONSTRAINT Networks
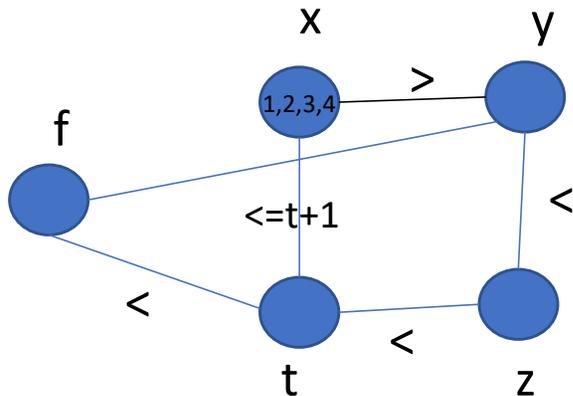
Rina Dechter, Winter 2026

Consistency algorithms, part b
Chapter 3

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

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
  - What is the domain for X in an arc-consistent network?

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)…

$$O(nek^3) \qquad O(nekt)$$
$$O(ek^3) \qquad O(ekt)$$
$$O(ek^2) \qquad O(et)$$

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
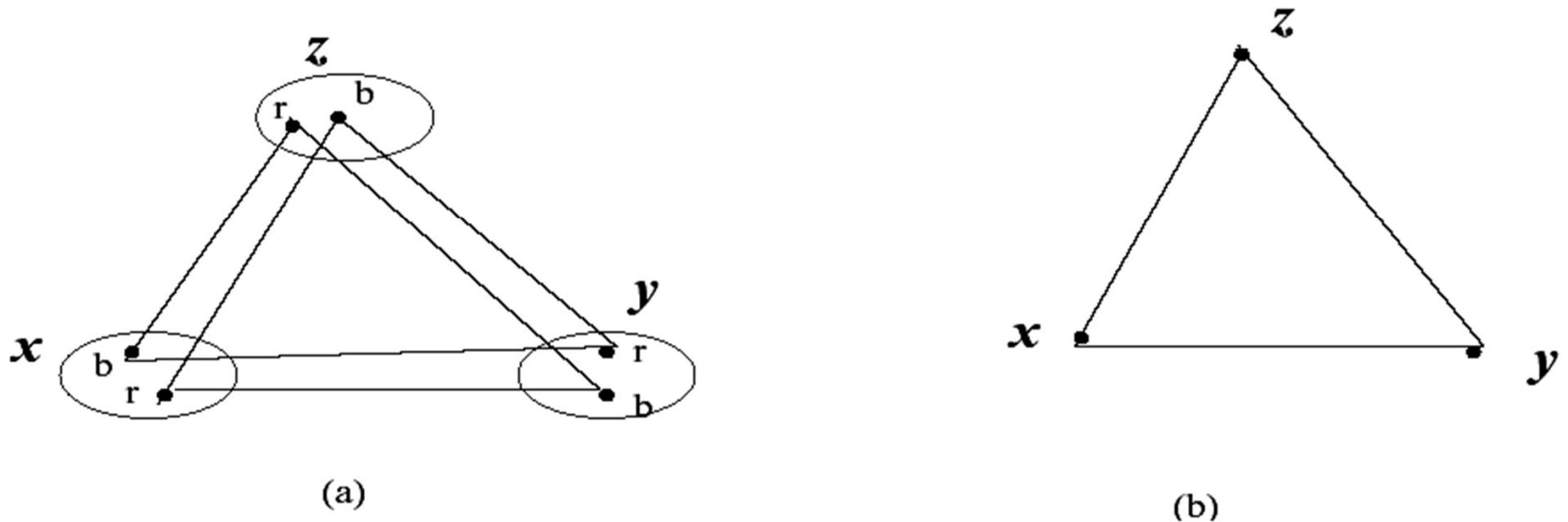
Path-consistency

Figure 3.8: (a) The matching diagram of a 2-value graph coloring problem. (b) Graphical picture of path-consistency using the matching diagram.

Path-consistency (3-consistency)

Definition 3.3.2 (Path-consistency) *Given a constraint network $\mathcal{R} = (X, D, C)$, a two variable set $\{x_i, x_j\}$ is path-consistent relative to variable x_k if and only if for every consistent assignment $(< x_i, a_i >, < x_j, a_j >)$ there is a value $a_k \in D_k$ s.t. the assignment $(< x_i, a_i >, < x_k, a_k >)$ is consistent and $(< x_k, a_k >, < x_j, a_j >)$ is consistent. Alternatively, a binary constraint R_{ij} is path-consistent relative to x_k iff for every pair $(a_i, a_j), \in R_{ij}$, where a_i and a_j are from their respective domains, there is a value $a_k \in D_k$ s.t. $(a_i, a_k) \in R_{ik}$ and $(a_k, a_j) \in R_{kj}$. A subnetwork over three variables $\{x_i, x_j, x_k\}$ is path-consistent iff for any permutation of (i, j, k), R_{ij} is path consistent relative to x_k. A network is path-consistent iff for every R_{ij} (including universal binary relations) and for every $k \neq i, j$ R_{ij} is path-consistent relative to x_k.*

Revise-3

REVISE-3$((x, y), z)$

input: a three-variable subnetwork over (x, y, z), R_{xy}, R_{yz}, R_{xz}.
output: revised R_{xy} path-consistent with z.
1. **for** each pair $(a, b) \in R_{xy}$
2.   **if** no value $c \in D_z$ exists such that $(a, c) \in R_{xz}$ and $(b, c) \in R_{yz}$
3.   **then** delete (a, b) from R_{xy}.
4.   **endif**
5. **endfor**

Figure 3.9: Revise-3

$$R_{xy} \leftarrow R_{xy} \cap \pi_{xy}(R_{xz} \bowtie D_z \bowtie R_{zy})$$

- Complexity: O(k^3)
- Best-case: O(t)
- Worst-case O(tk)

Revise3 = join followed by project

- Join :

x_1	x_2
a	a
b	b

$\bowtie$

x_2	x_3
a	a
a	b
b	a

$=$

x_1	x_2	x_3
a	a	a
a	a	b
b	b	a
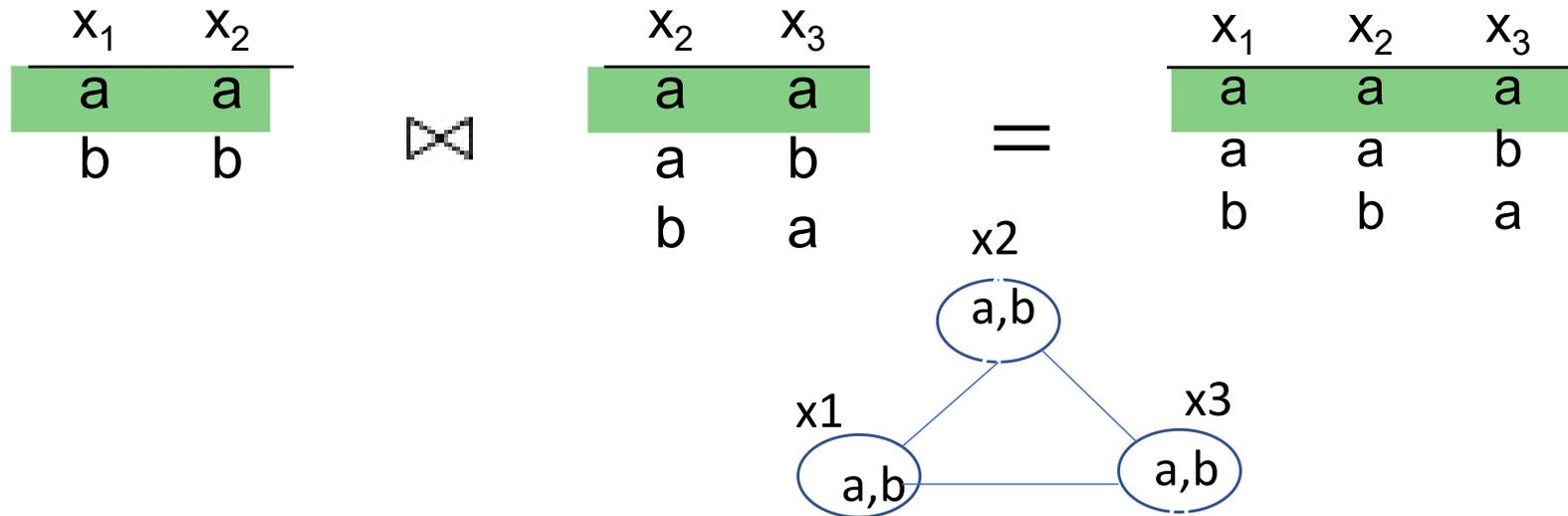
$$R_{ij} \leftarrow R_{ij} \cap \pi_{ij}(R_{ik} \bowtie D_k \bowtie R_{kj})$$

$R_{X_1 X_3} = \{(a,a),(a,b),(b,a)\}$

PC-1

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

input: a network $\mathcal{R} = (X, D, C)$.
output: a path consistent network equivalent to $\mathcal{R}$.
1. **repeat**
2.   **for** $k \leftarrow 1$ to n
3.   **for** $i, j \leftarrow 1$ to n
4. $R_{ij} \leftarrow R_{ij} \cap \pi_{ij}(R_{ik} \bowtie D_k \bowtie R_{kj})$ /* $(Revise - 3((i,j), k))$
5.   **endfor**
6.   **endfor**
7. **until** no constraint is changed.

Figure 3.10: Path-consistency-1 (PC-1)

- Complexity: $O(n^5 k^5)$
- O(n^3) triplets, each take O(k^3) steps → O($n^3 k^3$)
- Max number of loops: O($n^2\, k^2$) .

PC-2

PC-**2**$(\mathcal{R})$

input: a network $\mathcal{R} = (X, D, C)$.
output: $\mathcal{R}'$ a path consistent network equivalent to $\mathcal{R}$.
1. $Q \leftarrow \{(i, k, j) \mid 1 \le i < j \le n, 1 \le k \le n, k \ne i, k \ne j\}$
2. **while** Q is not empty
3.   select and delete a 3-tuple (i, k, j) from Q
4. $R_{ij} \leftarrow R_{ij} \cap \pi_{ij}(R_{ik} \bowtie D_k \bowtie R_{kj})$ /* (Revise-3$((i,j), k)$)
5.   **if** R_{ij} changed **then**
6. $Q \leftarrow Q \cup \{(l, i, j)(l, j, i) \mid 1 \le l \le n, l \ne i, l \ne j\}$
7. **endwhile**

- Complexity: $O(n^3 k^5)$
- (each pair of values deleted may add: 2n-1 triplets, and this can occur as the number of pairs: O($n^2\, k^2$) → size of Q is O($n^3\, k^2$), processing is O(k^3) yielding the result)
- Optimal PC-4: $O(n^3 k^3)$

Example: before and after path-consistency
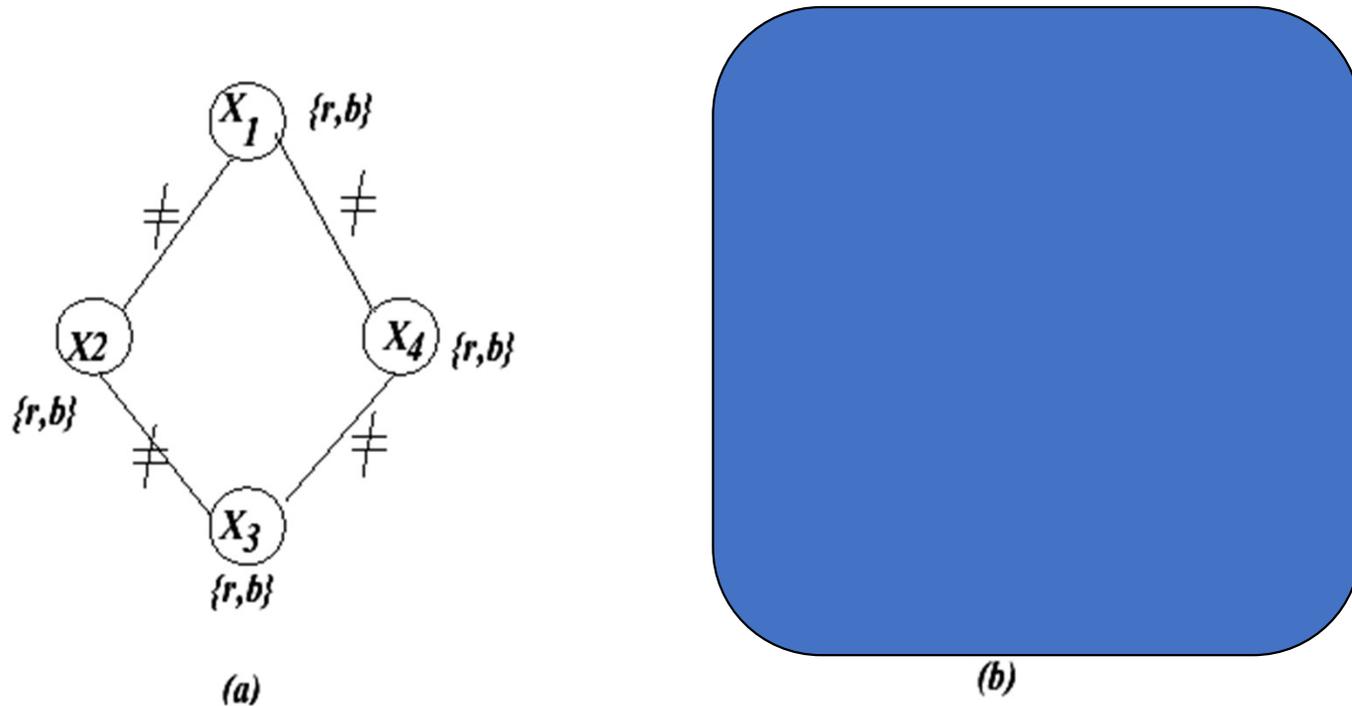
Figure 3.12: A graph-coloring graph (a) before path-consistency (b) after path-consistency

- PC-1 requires 2 processing of each arc while PC-2 may not
- Can we do path-consistency distributedly?

Example: before and after path-consistency
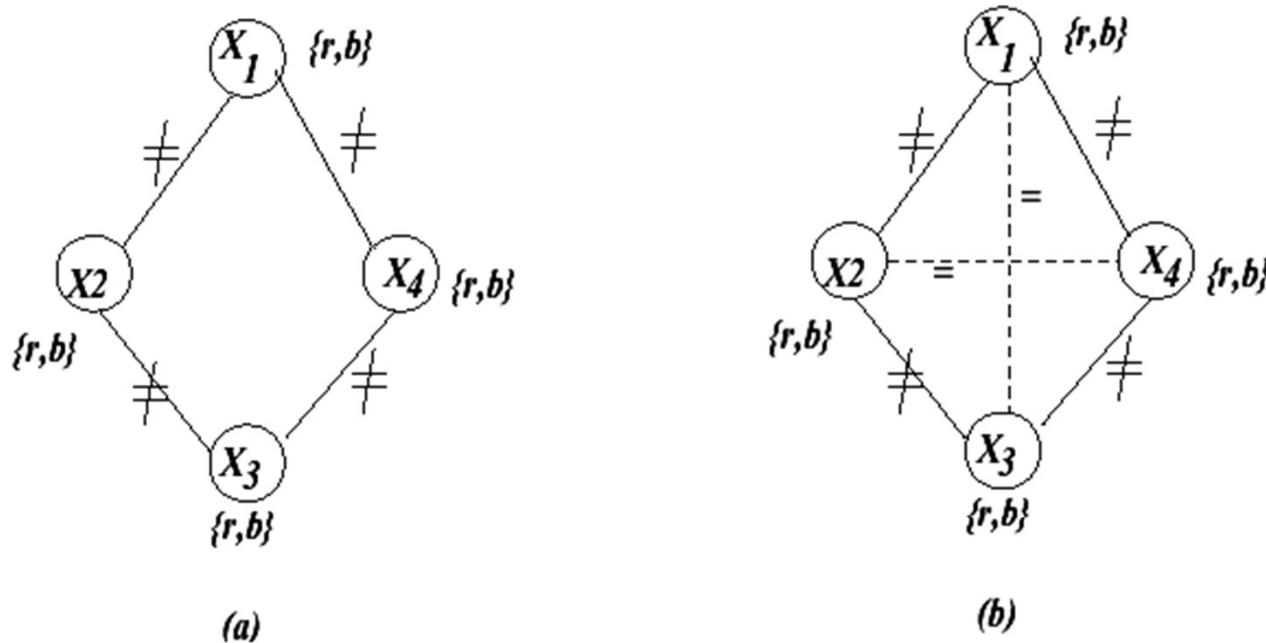
Figure 3.12: A graph-coloring graph (a) before path-consistency (b) after path-consistency

- PC-1 requires 2 processings of each arc while PC-2 may not
- Can we do path-consistency distributedly?

Path-consistency Algorithms

- Apply Revise-3 $O(k^3)$ until no change

$$R_{ij} \leftarrow R_{ij} \cap \pi_{ij}(R_{ik} \bowtie D_k \bowtie R_{kj})$$

- Path-consistency (3-consistency) adds binary constraints.

- PC-1: $O(n^5 k^5)$

- PC-2: $O(n^3 k^5)$

- PC-4 optimal: $O(n^3 k^3)$

I-consistency
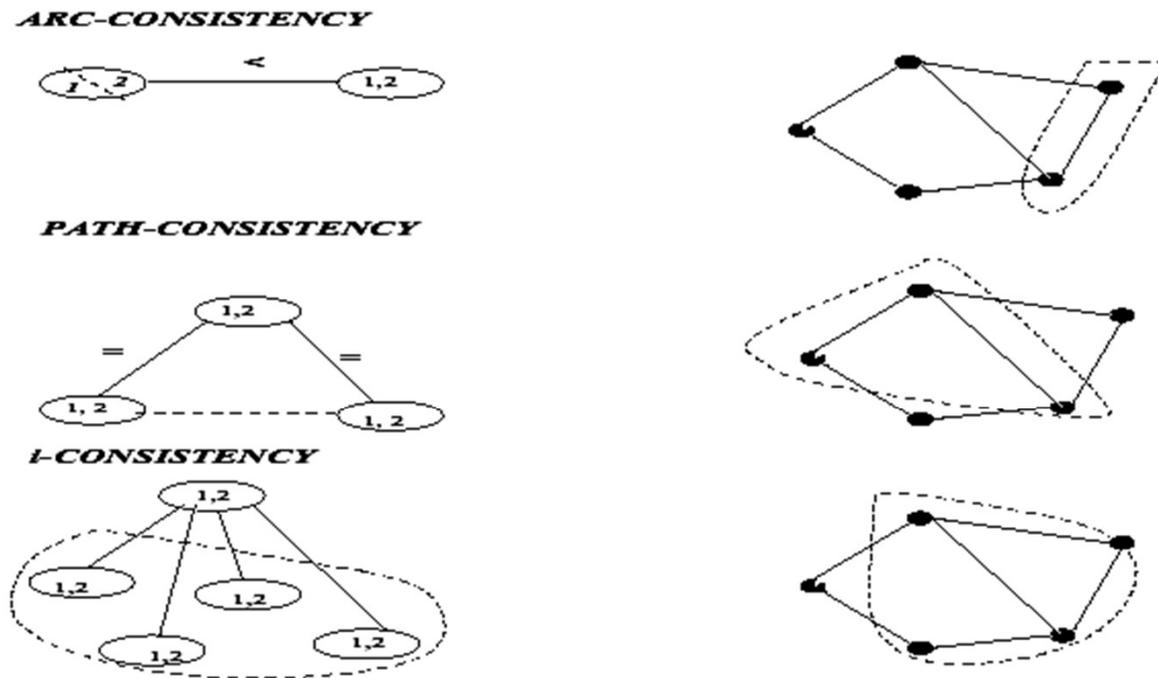
ARC-CONSISTENCY

PATH-CONSISTENCY

i-CONSISTENCY

Figure 3.17: The scope of consistency enforcing: (a) arc-consistency, (b) path-consistency, (c) i-consistency

Higher levels of consistency, global-consistency

Definition:

t. *A network is i-consistent iff given any consistent instantiation of any i − 1 distinct variables, there exists an instantiation of any ith variable such that the i values taken together satisfy all of the constraints among the i variables. A network is strongly i-consistent iff it is j-consistent for all j ≤ i. A strongly n-consistent network, where n is the number of variables in the network, is called globally consistent.*

A Globally consistent network is backtrack-free

Revise-i

REVISE-$i(\{x_1, x_2, ..., x_{i-1}\}, x_i)$

input: a network $\mathcal{R} = (X, D, C)$
output: a constraint R_S, $S = \{x_1, ..., x_{i-1}\}$ i-consistent relative to x_i.
1. **for** each instantiation $\bar{a}_{i-1} = (< x_1, a_1 >, < x_2, a_2 >, ..., < x_{i-1}, a_{i-1} >)$ do,
2. **if** no value of $a_i \in D_i$ exists s.t. $(\bar{a}_{i-1}, a_i)$ is consistent
   then delete $\bar{a}_{i-1}$ from R_S
   (Alternatively, let $\mathcal{S}$ be the set of all subsets of $\{x_1, ..., x_i\}$ that contain x_i
   and appear as scopes of constraints of $\mathcal{R}$, then
   $R_S \leftarrow R_S \cap \pi_S(\bowtie_{S' \subseteq \mathcal{S}} R_{S'}))$
3. **endfor**

Figure 3.14: Revise-i

- Complexity: for binary constraints $\quad O(k^i)$
- For arbitrary constraints: $\quad O((2k)^i)$
- (because there may be O(2^i) constraints to test per tuple)

4-queen example
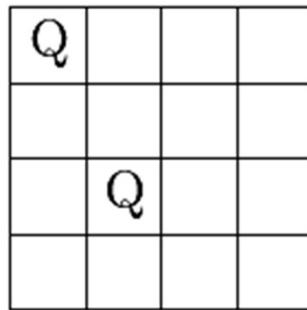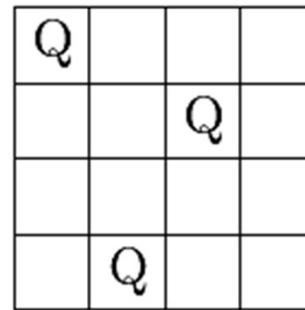
Figure 3.13: (a) Not 3-consistent; (b) Not 4-consistent

i-consistency

I-CONSISTENCY$(\mathcal{R})$

input: a network $\mathcal{R}$.

output: an i-consistent network equivalent to $\mathcal{R}$.

1. **repeat**
2.   **for** every subset $S \subseteq X$ of size $i - 1$, and for every x_i, do
3.   let $\mathcal{S}$ be the set of all subsets in of $\{x_1, ..., x_i\}$ $scheme(\mathcal{R})$ that contain x_i
4. $R_S \leftarrow R_S \cap \pi_S(\bowtie_{S' \in \mathcal{S}} R_{S'})$ (this is Revise-i(S, x_i))
6.   **endfor**
7. **until** no constraint is changed.

This S is different it is all subsets of size I That includes xi

Figure 3.15: i-consistency-1

Theorem 3.4.3 (complexity of i-consistency) *The time and space complexity of brute-force i-consistency* $O(2^i(nk)^{2i})$ *and* $O(n^i k^i)$, *respectively. A lower bound for enforcing i-consistency is* $\Omega(n^i k^i)$. □

Path-consistency vs 3-consistency

Example 3.4.4 Suppose a constraint network involves three variables x, y, z having domains $\{0, 1\}$ and a single ternary constraint $R_{xyz} = \{(0, 0, 0)\}$. Application of the path-consistency algorithm will produce nothing since there are no binary constraints to test; the network is already path-consistent. However, the network is *not* 3-consistent. While we can assign the values $(\langle x, 1 \rangle, \langle y, 1 \rangle)$ (since there is no constraint), we cannot extend this assignment to z in a way that satisfies the given ternary constraints. Indeed, if we

apply 3-consistency to this network we will add the constraint $R_{xy} = \{(< x, 0 >< y, 0 >)\}$ in addition to the constraint $R_x = \{(< x, 0 >)\}$. □

i-consistency
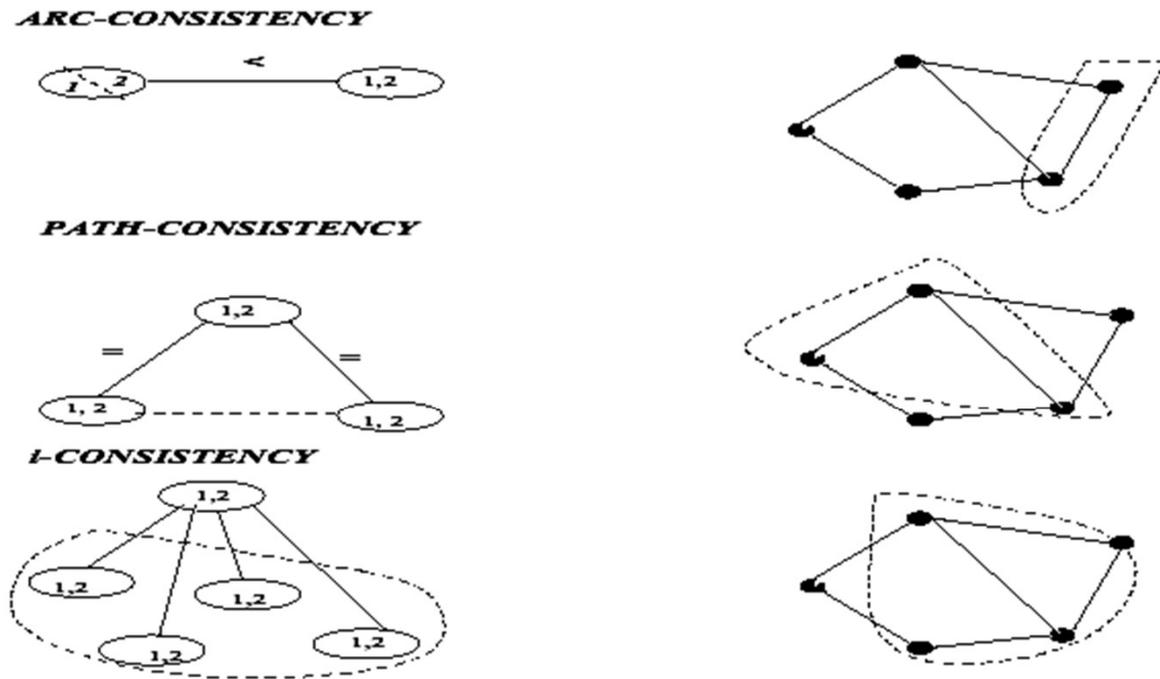
Figure 3.17: The scope of consistency enforcing: (a) arc-consistency, (b) path-consistency, (c) i-consistency

Outline

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

Generalized arc-consistency (GAC) for non-binary constraints

Definition 3.5.1 (generalized arc-consistency) *Given a constraint network $\mathcal{R} = (\mathcal{X}, \mathcal{D}, \mathcal{C})$, with $R_S \in C$, a variable x is* arc-consistent *relative to R_S if and only if for every value $a \in D_x$ there exists a tuple $t \in R_S$ such that $t[x] = a$. t can be called a* support *for a. The constraint R_S is called* arc-consistent *iff it is arc-consistent relative to each of the variables in its scope and a constraint network is* arc-consistent *if all its constraints are arc-consistent.*

$$D_x \leftarrow D_x \cap \pi_x(R_S \bowtie D_{S-x})$$

Complexity: O(t k), t bounds number of tuples.

Relational arc-consistency (different than GAC):

$$R_{S-\{x\}} \leftarrow \pi_{S-\{x\}}(R_S \bowtie D_x).$$

Algorithm 1: AC3 / GAC3

function Revise3(**in** x_i: variable; c: constraint): **Boolean** ;
   begin
1   CHANGE $\leftarrow$ **false**;
2   **foreach** $v_i \in D(x_i)$ **do**
3   **if** $\nexists \tau \in c \cap \pi_{X(c)}(D)$ *with* $\tau[x_i] = v_i$ **then**
4   remove v_i from $D(x_i)$;
5   CHANGE $\leftarrow$ **true**;
6   return CHANGE ;
   end

function AC3/GAC3(**in** X: set): **Boolean** ;
   begin
      /* initalisation */;
7      $Q \leftarrow \{(x_i, c) \mid c \in C, x_i \in X(c)\}$;
      /* propagation */;
8   **while** $Q \neq \emptyset$ **do**
9   select and remove (x_i, c) from Q;
10   **if** *Revise*(x_i, c) **then**
11   **if** $D(x_i) = \emptyset$ **then return false** ;
12   **else** $Q \leftarrow Q \cup \{(x_j, c') \mid c' \in C \wedge c' \neq c \wedge x_i, x_j \in X(c') \wedge j \neq i\}$;
13   return **true** ;
   end

Generalized arc-consistency

Proposition 27 (GAC3). *GAC3 is a sound and complete algorithm for achieving arc consistency that runs in $O(er^3 d^{r+1})$ time and $O(er)$ space, where r is the greatest arity among constraints.*

Examples of generalized AC and relational AC

- x+y+z $\leq$ 15 and z $\geq$ 13 implies

Examples of generalized AC and relational AC

- x+y+z $\leq$ 15 and z $\geq$ 13 implies

 x$\leq$ 1, y$\leq$ 1

- Example of relational arc-consistency

   Here given the 2 top Boolean constraints we infer the 3rd.

$$A \wedge B \rightarrow G,$$

$$\neg G, \Rightarrow$$

Examples of generalized AC and relational AC

- x+y+z $\leq$ 15 and z $\geq$ 13 implies

  x$\leq$1, y$\leq$1

- Example of relational arc-consistency

  Here given the 2 top Boolean constraints we infer the 3rd.

$$A \wedge B \rightarrow G,$$

$$\neg G, \Rightarrow$$

$$\neg A \vee \neg B$$

Sudoku

•Constraint Propagation

•Inference

•Variables: empty slots

•Domains = {1,2,3,4,5,6,7,8,9}

•Constraints: 27 all-different

Each row, column and major block must be alldifferent

"Well posed" if it has unique solution: 27 constraints

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

Global constraints

Constraints of arbitrary scope length defined by expression, a Boolean function

Global constraints are classes of constraints defined by a formula of arbitrary arity (see Section 9.2).

Example 2. The constraint $\texttt{alldifferent}(x_1, x_2, x_3) \equiv (v_i \neq v_j \wedge v_i \neq v_k \wedge v_j \neq v_k)$ allows the infinite set of 3-tuples in $\mathbb{Z}^3$ such that all values are different. The constraint $c(x_1, x_2, x_3) = \{(2, 2, 3), (2, 3, 2), (2, 3, 3), (3, 2, 2), (3, 2, 3), (3, 3, 2)\}$ allows the finite set of 3-tuples containing both values 2 and 3 and only them.

Example for alldiff

- A = {3,4,5,6}
- B = {3,4}
- C= {2,3,4,5}
- D= {2,3,4}
- E = {3,4}
- F= {1,2,3,4,5,6}
- Alldiff (A,B,C,D,E,F)
- Arc-consistency does nothing
- Apply GAC to sol(A,B,C,D,E,F)?
- → A = {6}, F = {1}….
- Alg: bipartite matching kn^1.5
- (Lopez-Ortiz, et. Al, IJCAI-03 pp 245 (A fast and simple algorithm for bounds consistency of alldifferent constraint)

Global constraints

- Alldifferent

- Sum constraint (variable equal the sum of others)

- Global cardinality constraint (a value can be assigned a bounded number of times to a set of variables)

- The cummulative constraint (related to scheduling tasks)

In summary, a global constraint $C = \{C(i)\}$ is a family of scope-parameterized constraints, (normally $i \geq 2$), where $C(i)$ is a constraint whose relation is often defined implicitly by either a natural language statement, or as a set of solutions to a subproblem defined by lower arity explicit constraints (e.g., alldifferent). It is associated with one or more specialized propagation algorithms trying to achieve generalized arc-consistency relative to $C(i)$ (or an approximation of it) in a way that is more efficient than a brute-force approach.

Bounds consistency

Definition 3.5.4 (bounds consistency) *Given a constraint C over a scope S and domain constraints, a variable $x \in S$ is bounds-consistent relative to C if the value $\min\{D_x\}$ (respectively, $\max\{D_x\}$) can be extended to a full tuple t of C. We say that t supports $\min\{D_x\}$. A constraint C is* bounds-consistent *if each of its variables is bounds-consistent.*

Bounds consistency

Example 3.5.5 Consider the constraint problem with variables $x_1, \ldots x_6$, each with domains $1, \ldots, 6$, and constraints:

$$C_1 : x_4 \geq x_1 + 3, \quad C_2 : x_4 \geq x_2 + 3, \quad C_3 : x_5 \geq x_3 + 3, \quad C_4 : x_5 \geq x_4 + 1,$$

$$C_5 : alldifferent\{x_1, x_2, x_3, x_4, x_5\}$$

The constraints are not bounds consistent. For example, the minimum value 1 in the domain of x_4 does not have support in constraint C_1 as there is no corresponding value for x_1 that satisfies the constraint. Enforcing bounds consistency using constraints C_1 through C_4 reduces the domains of the variables as follows: $D_1 = \{1, 2\}$, $D_2 = \{1, 2\}$, $D_3 = \{1, 2, 3\}$ $D_4 = \{4, 5\}$ and $D_5 = \{5, 6\}$. Subsequently, enforcing bounds consistency using constraints C_5 further reduces the domain of C to $D_3 = \{3\}$.Now constraint C_3 is no longer bound consistent. Reestablishing bounds consistency causes the domain of x_5 to be reduced to $\{6\}$. Is the resulting problem already arc-consistent? □

Outline

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

Boolean constraint propagation

- (A V ¬B) and (B)
    - B is arc-consistent relative to A but not vice-versa

- Arc-consistency by resolution:
    res((A V ¬B),B) = A

Given also (B V C), path-consistency:

res((A V ¬B),(B V C) = (A V C)

Relational arc-consistency rule = unit-resolution

$$A \wedge B \rightarrow G, \neg G, \Rightarrow \neg A \vee \neg B$$

Boolean constraint propagation

Procedure UNIT-PROPAGATION
Input: A cnf theory, φ, $d = Q_1, ..., Q_n$.
Output: An equivalent theory such that every unit clause does not appear in any non-unit clause.
1. queue = all unit clauses.
2. **while** queue is not empty, do.
3. $T \leftarrow$ next unit clause from Queue.
4.   **for** every clause β containing T or $\neg T$
5.   **if** β contains T delete β (subsumption elimination)
6.   **else**, For each clause $\gamma = resolve(\beta, T)$.
              if γ, the resolvent, is empty, the theory is unsatisfiable.
7.   **else**, add the resolvent γ to the theory and delete β.
              if γ is a unit clause, add to Queue.
8.   endfor.
9. endwhile.

Theorem 3.6.1 *Algorithm* UNIT-PROPAGATION *has a linear time complexity.*

Consistency for numeric constraints (Gausian elimination)

$$x \in [1,10], \, y \in [5,15],$$

$$x + y = 10$$

$$arc - consistency \Rightarrow x \in [1,5], \, y \in [5,9]$$

Gausian elimination of $\quad x + y = 10, -y \leq -5$

$$z \in [-10,10],$$

$$y + z \leq 3$$

$$path - consistency \Rightarrow x - z \geq 7$$

Gausian Elinination of: $\quad x + y = 10, -y - z \geq -3$

Impact on graphs of i-consistency
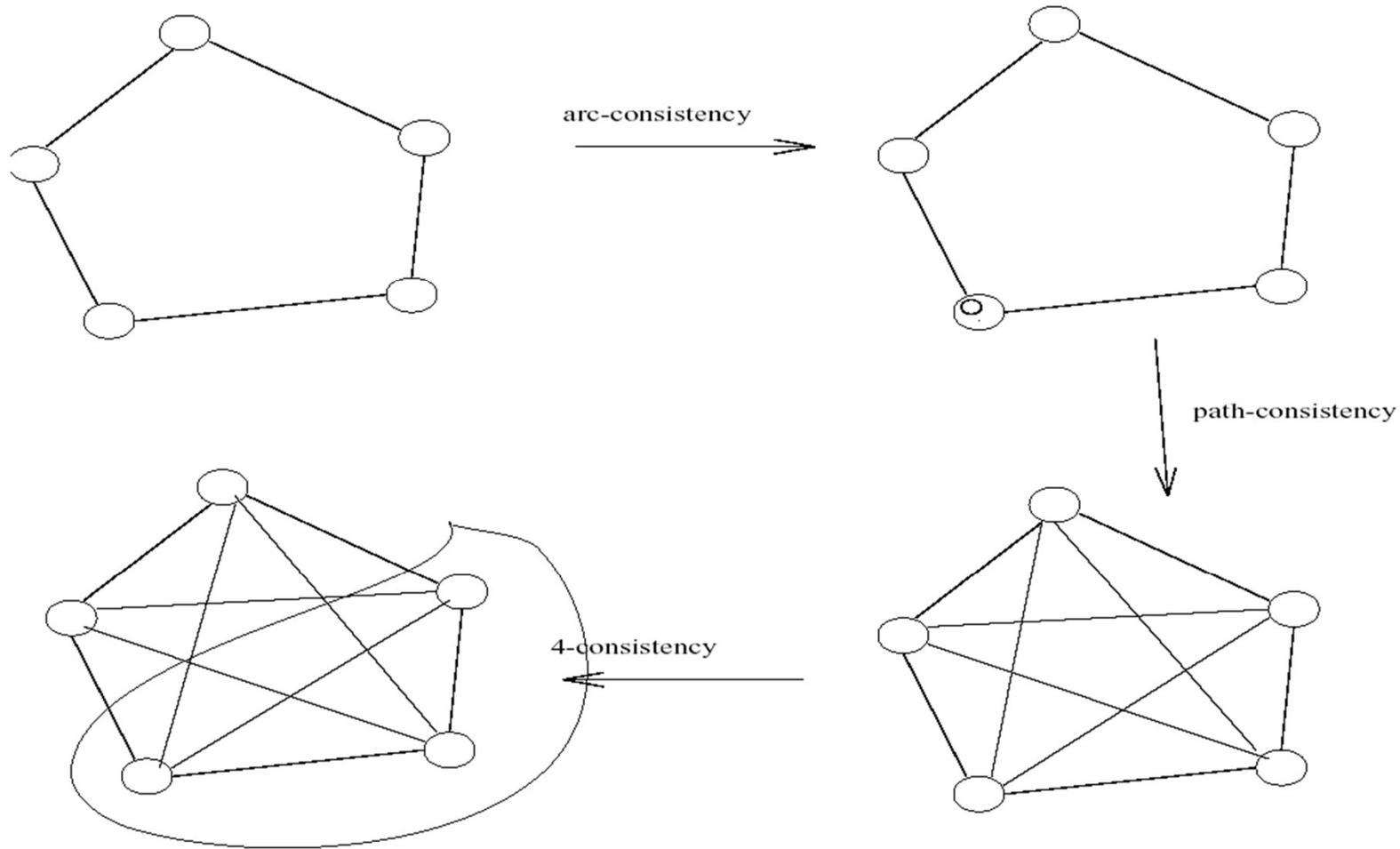
arc-consistency

path-consistency

4-consistency

Outline

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

Distributed arc-consistency (Constraint propagation)

- Implement AC-1 distributedly.

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

- Node x_j sends the message to node x_i

$$h_i^j \leftarrow \pi_i(R_{ij} \otimes D_j)$$

- Node x_i updates its domain:

$$D_i \leftarrow D_i \cap h_i^j$$

- Relational and generalized arc-consistency can be implemented distributedly: sending messages between constraints over the dual graph

$$R_{S-\{x\}} \leftarrow \pi_{S-\{x\}}(R_S \otimes D_x)$$

Relational Arc-consistency

The message that R2 sends to R1 is

$$h_i^j \leftarrow \pi_{l_{ij}}(R_i \bowtie (\bowtie_{k \in ne(i)} h_k^i))$$

R1 updates its relation and domains and sends messages to neighbors

$$D_i \leftarrow D_i \cap (\bowtie_{k \in ne(i)} D_k^i)$$

R_2

A	B
1	2
1	3
2	1
2	3
3	1
3	2

R_1

A
1
2
3

R_3

A	C
1	2
3	2

R_5

B	C	F
1	2	3
3	2	1

R_4

A	B	D
1	2	3
1	3	2
2	1	3
2	3	1
3	1	2
3	2	1

R_6

D	F	G
1	2	3
2	1	3
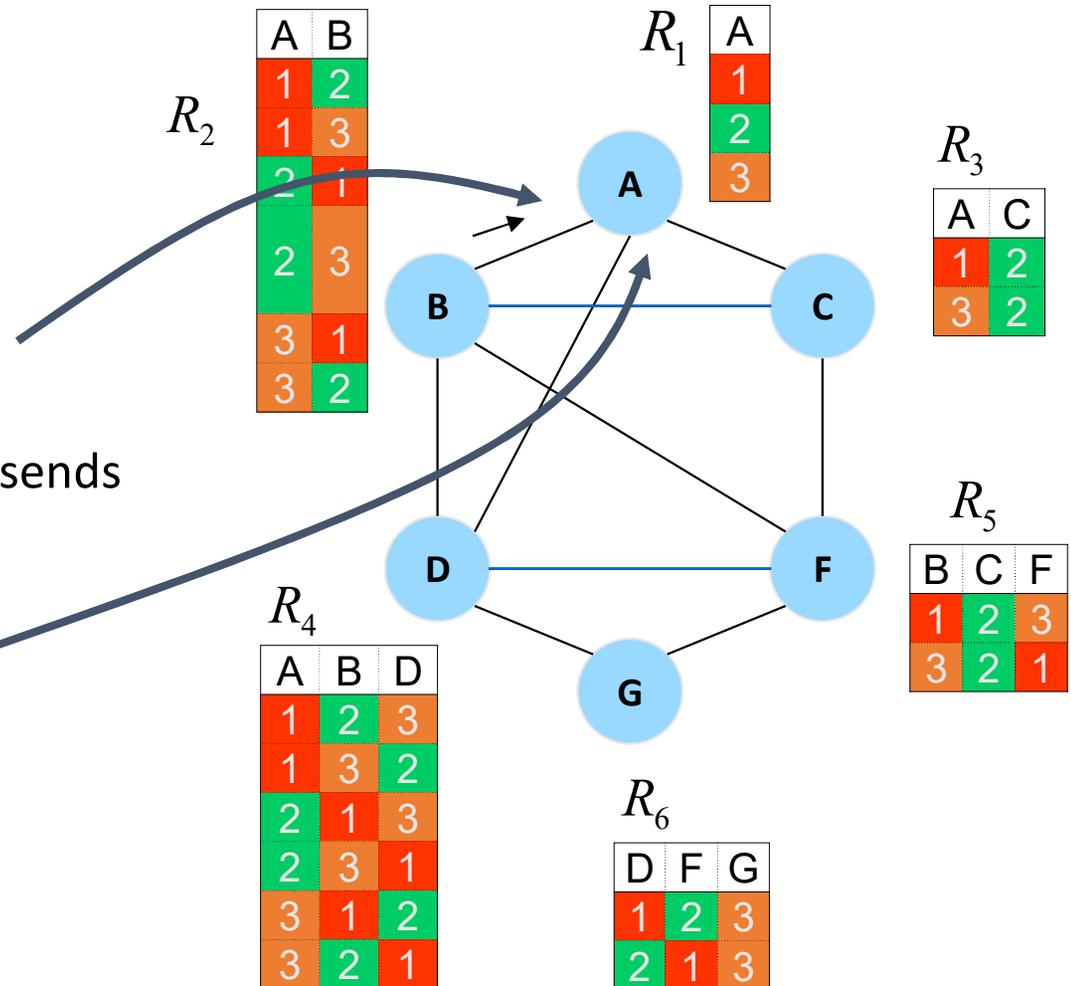
Distributed Relational Arc-Consistency

- DRAC can be applied to the dual problem of any constraint network:

$$h_i^j \leftarrow \pi_{l_{ij}}(R_i \bowtie (\bowtie_{k \in ne(i)} h_k^i)) \qquad (1)$$

$$R_i \leftarrow R_i \cap (\bowtie_{k \in ne(i)} h_k^i) \qquad (2)$$

DRAC on the dual join-graph

R_1

A
1
2
3

R_2

A	B
1	2
1	3
2	1
2	3
3	1
3	2

R_3

A	C
1	2
3	2

R_4

A	B	D
1	2	3
1	3	2
2	1	3
2	3	1
3	1	2
3	2	1

R_5

B	C	F
1	2	3
3	2	1

R_6

D	F	G
1	2	3
2	1	3

Iteration 1

R_1

A
1
2
3

h_2^1

A
1
2
3

h_3^1

A
1
3

h_4^1

A
1
2
3

h_5^2

B
1
3

h_4^2

A	B
1	2
1	3
2	1
2	3
3	1
3	2

h_3^2

A
1
3

h_1^2

A
1
2
3

R_2

A	B
1	2
1	3
2	1
2	3
3	1
3	2

R_3

A	C
1	2
3	2

h_1^3

A
1
2
3

h_2^3

A
1
2
3

h_4^3

A
1
2
3

h_5^3

C
2

h_6^4

D
1
2

h_5^4

B
1
3

h_3^4

A
1
3

h_2^4

A	B
1	2
1	3
2	1
2	3
3	1
3	2

h_1^4

A
1
2
3

R_4

A	B	D
1	2	3
1	3	2
2	1	3
2	3	1
3	1	2
3	2	1

R_5

B	C	F
1	2	3
3	2	1

h_2^5

B
1
2
3

h_3^5

C
2

h_4^5

B
1
2
3

h_6^5

F
1
3

R_6

D	F	G
1	2	3
2	1	3

h_4^6

D
1
2
3

h_5^6

F
1
3

Graph nodes: 1 (A), 2 (AB), 3 (AC), 4 (ABD), 5 (BCF), 6 (DFG)

Edges labeled: A, A, A, A, AB, B, B, C, D, F

$$R_i \leftarrow R_i \cap \left(\bowtie_{k \in ne(i)} h^i_k \right)$$ (2)

Iteration 1

R_1

A
1
3

R_2

A	B
1	3
2	1
2	3
3	1

R_3

A	C
1	2
3	2

R_4

A	B	D
1	3	2
2	3	1
3	1	2
3	2	1

R_5

B	C	F
1	2	3
3	2	1

R_6

D	F	G
2	1	3

$$h_i^j \leftarrow \pi_{l_{ij}}(R_i \bowtie (\bowtie_{k \in ne(i)} h_k^i)) \qquad (1)$$

Iteration 2

R_1

A
1
3

h_2^1

A
1
2
3

h_3^1

A
1
3

h_4^1

A
1
2
3

h_5^2

B
1
3

h_4^2

B
1
2
3

h_1^2

A
1
3

R_2

A	B
1	3
2	1
2	3
3	1

R_3

A	C
1	2
3	2

h_1^3

A
1
3

h_5^3

C
2

h_6^4

D
2

h_2^4

B
1
3

h_1^4

A
1
3

R_4

A	B	D
1	3	2
2	3	1
3	1	2
3	2	1

R_5

B	C	F
1	2	3
3	2	1

h_2^5

B
1
3

h_3^5

C
2

h_6^5

F
1

R_6

D	F	G
2	1	3

h_4^6

D
1
2

h_5^6

F
1
3

$$R_i \leftarrow R_i \cap \left(\bowtie_{k \in ne(i)} h_k^i \right) \qquad (2)$$

Iteration 2

R_1

A
1
3

R_2

A	B
1	3
3	1

R_3

A	C
1	2
3	2

R_4

A	B	D
1	3	2
3	1	2

R_5

B	C	F
3	2	1

R_6

D	F	G
2	1	3

$$h_i^j \leftarrow \pi_{l_{ij}}(R_i \bowtie (\bowtie_{k \in ne(i)} h_k^i)) \qquad (1)$$

Iteration 3

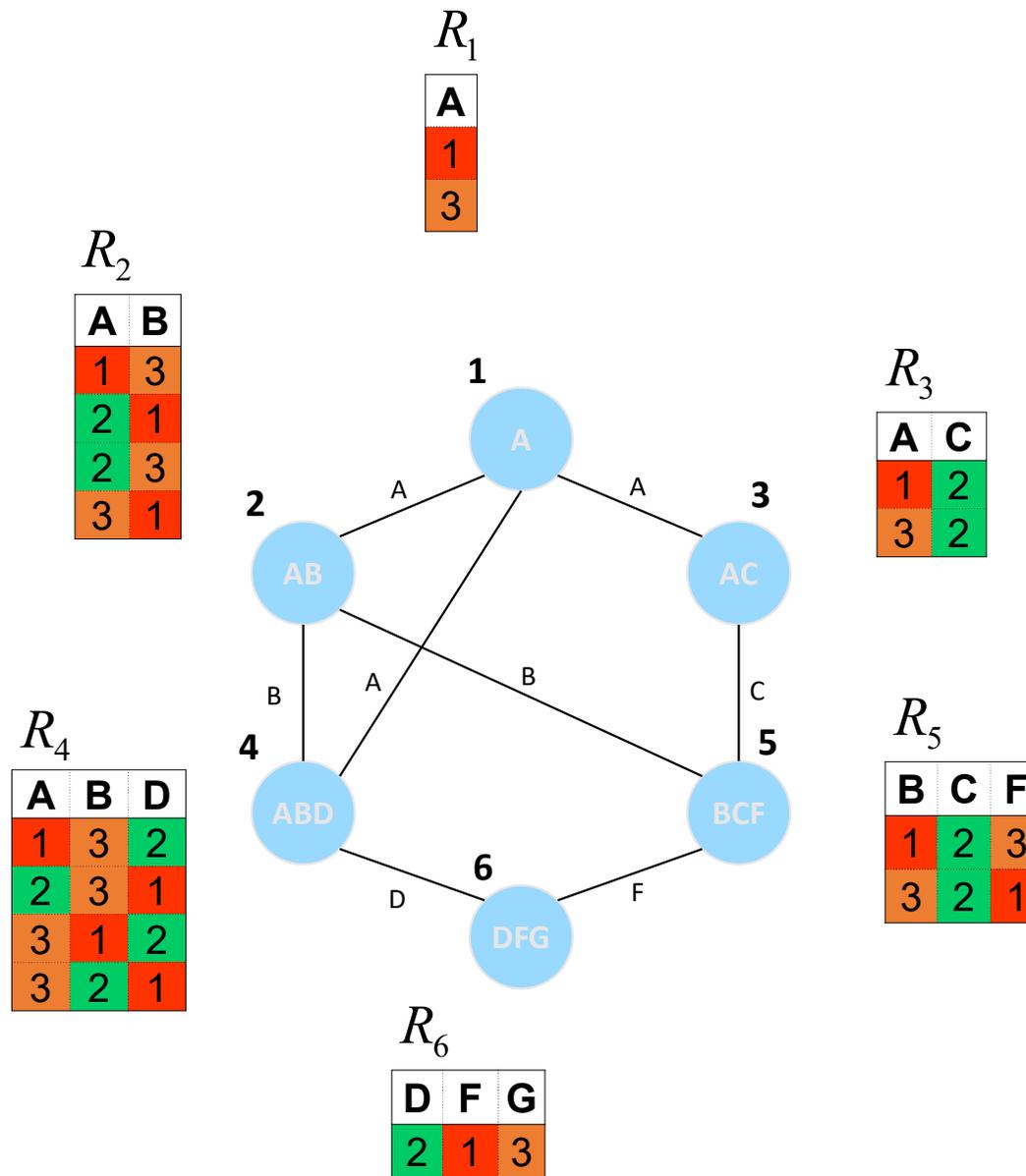$$R_i \leftarrow R_i \cap (\bowtie_{k \in ne(i)} h_k^i) \qquad (2)$$

Iteration 3

R_1

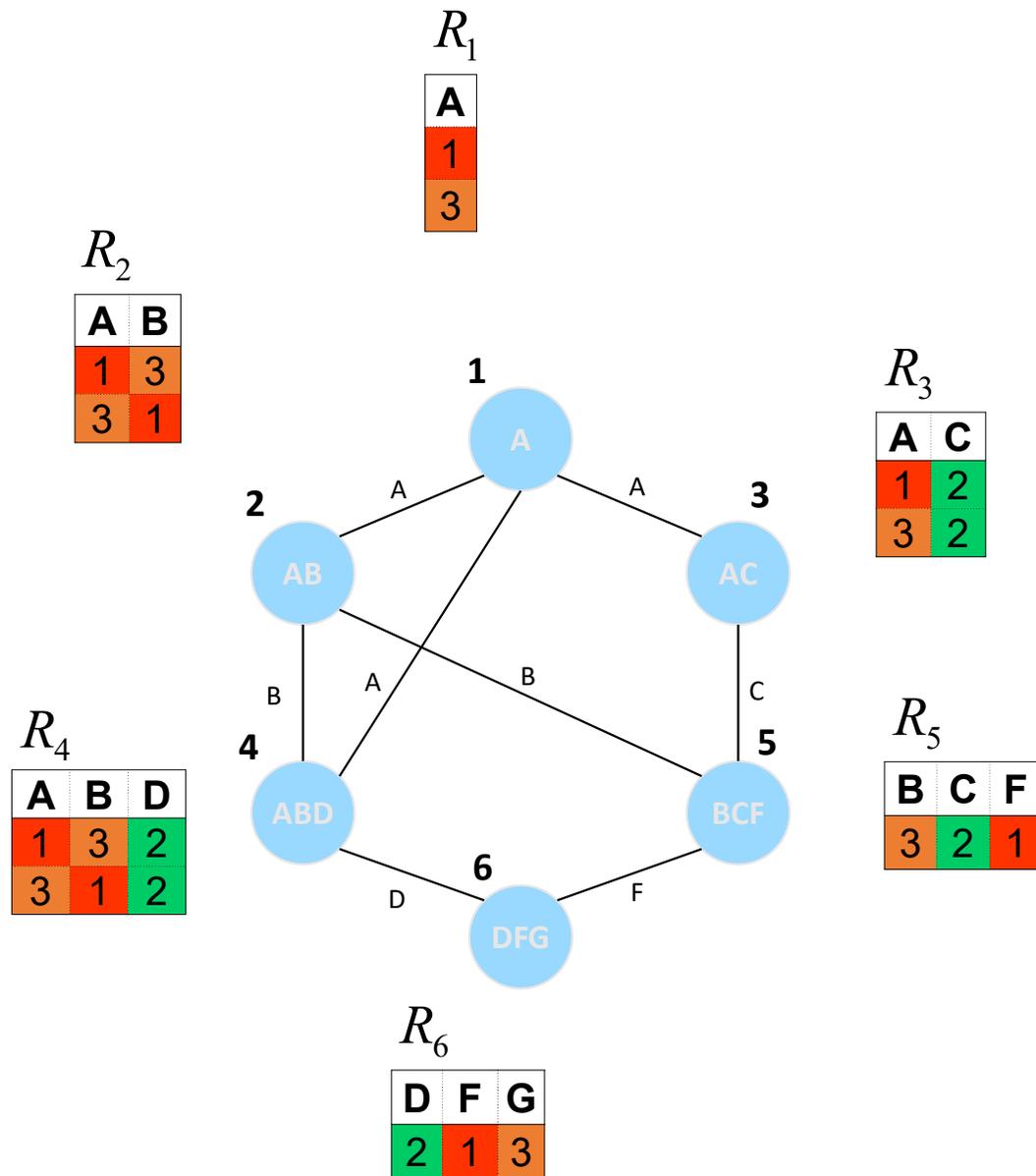| A |
|---|
| 1 |
| 3 |

R_2

A	B
1	3

R_3

A	C
1	2
3	2

R_4

A	B	D
1	3	2
3	1	2

R_5

B	C	F
3	2	1

R_6

D	F	G
2	1	3

$$h_i^j \leftarrow \pi_{l_{ij}}(R_i \bowtie (\bowtie_{k \in ne(i)} h_k^i)) \qquad (1)$$

Iteration 4

R_1

A
1
3

h_2^1

A
1

h_3^1

A
1
3

h_4^1

A
1
3

h_5^2

B
3

h_4^2

B
1
3

h_1^2

A
1
3

R_2

A	B
1	3

R_3

A	C
1	2
3	2

h_1^3

A
1
3

h_5^3

C
2

h_6^4

D
2

h_2^4

B
1
3

h_1^4

A
1
3

R_4

A	B	D
1	3	2
3	1	2

R_5

B	C	F
3	2	1

h_2^5

B
3

h_3^5

C
2

h_6^5

F
1

R_6

D	F	G
2	1	3

h_4^6

D
2

h_5^6

F
1

Graph:
- 1: A
- 2: AB
- 3: AC
- 4: ABD
- 5: BCF
- 6: DFG

Edges: A (1–2), A (1–3), B, A, B, C, D, F
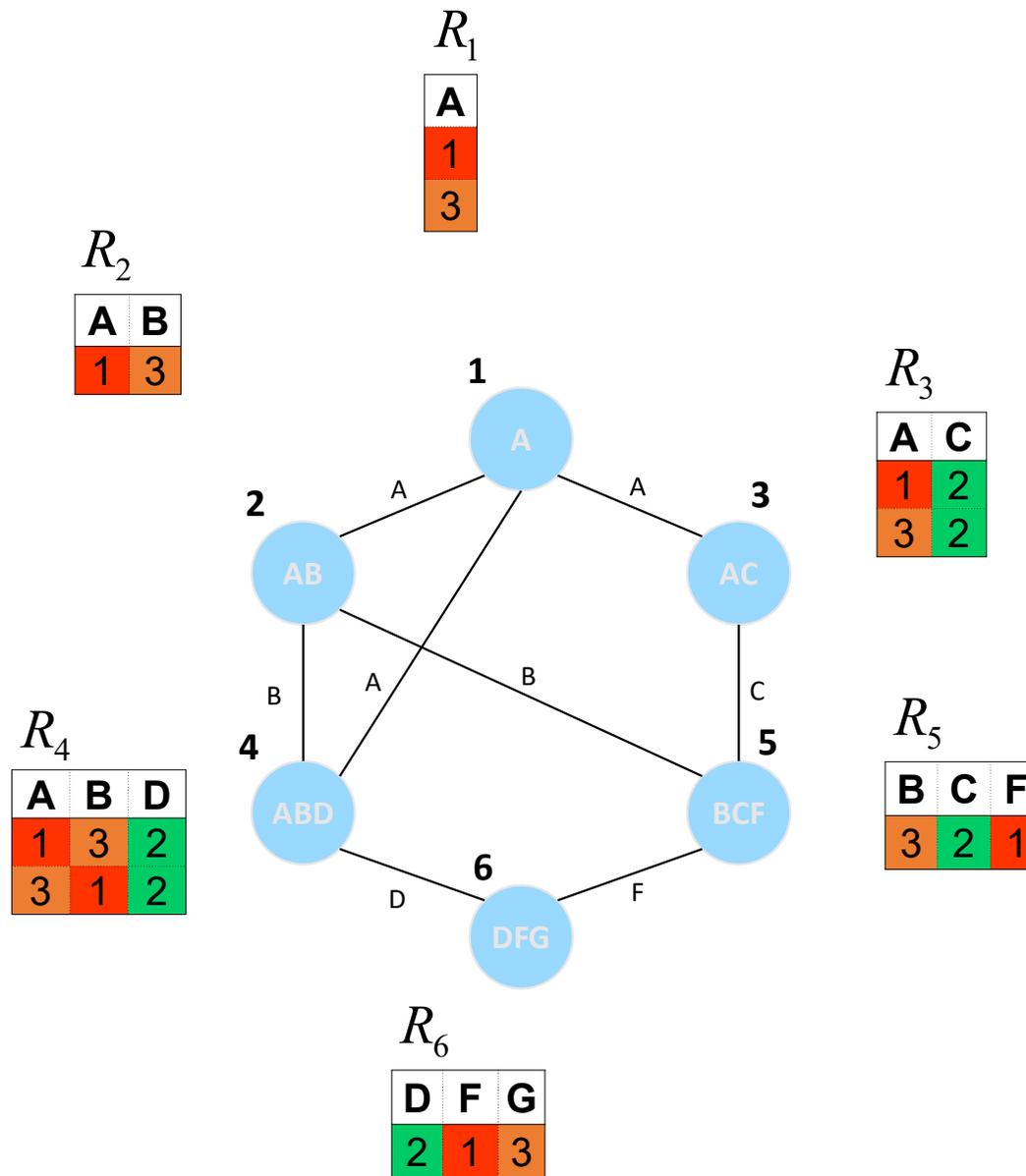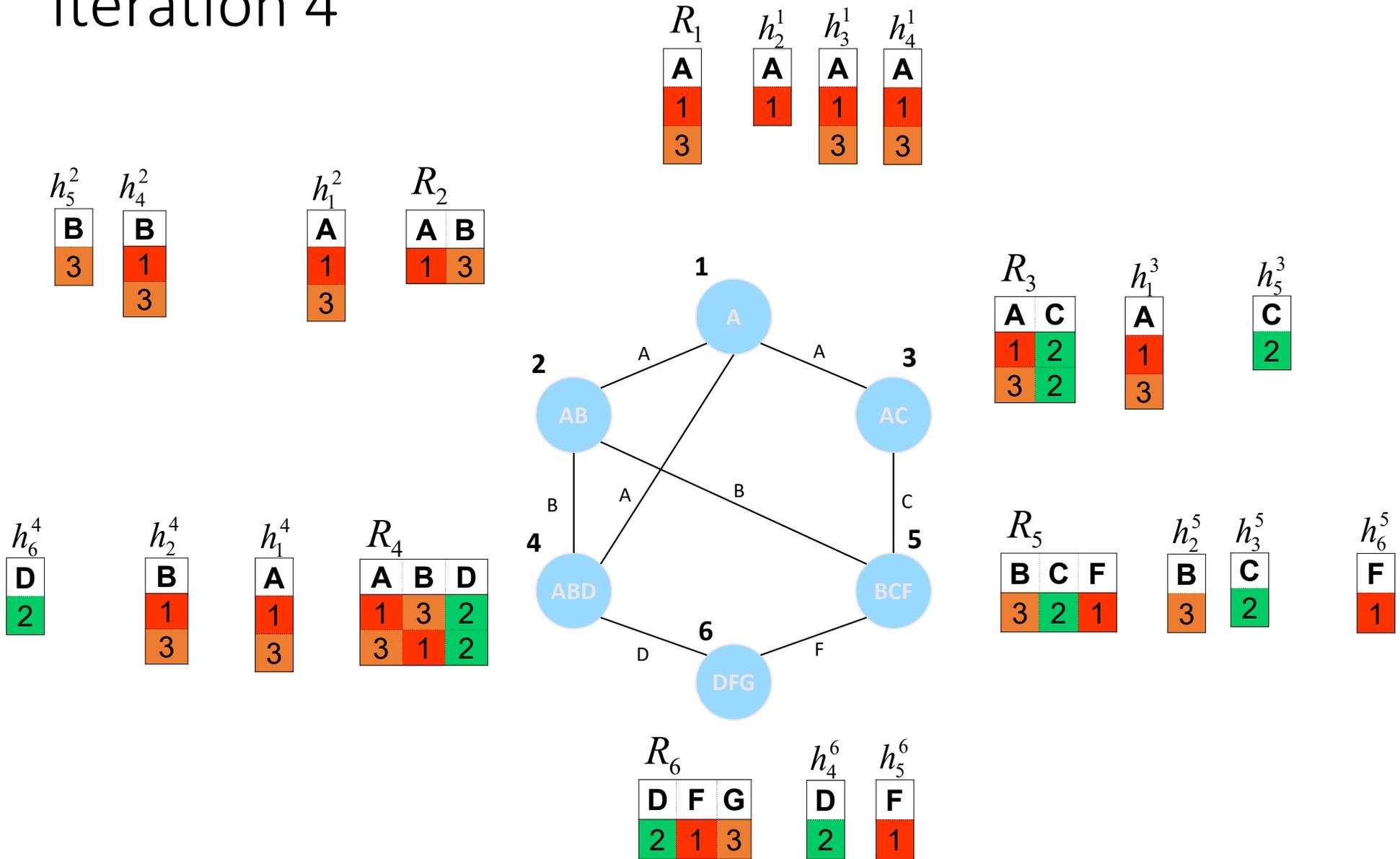
$$R_i \leftarrow R_i \cap (\bowtie_{k \in ne(i)} h_k^i) \qquad (2)$$

Iteration 4

R_1

A
1

R_2

A	B
1	3

R_3

A	C
1	2
3	2

R_4

A	B	D
1	3	2

R_5

B	C	F
3	2	1

R_6

D	F	G
2	1	3

$$h_i^j \leftarrow \pi_{l_{ij}}(R_i \bowtie (\bowtie_{k \in ne(i)} h_k^i)) \qquad (1)$$
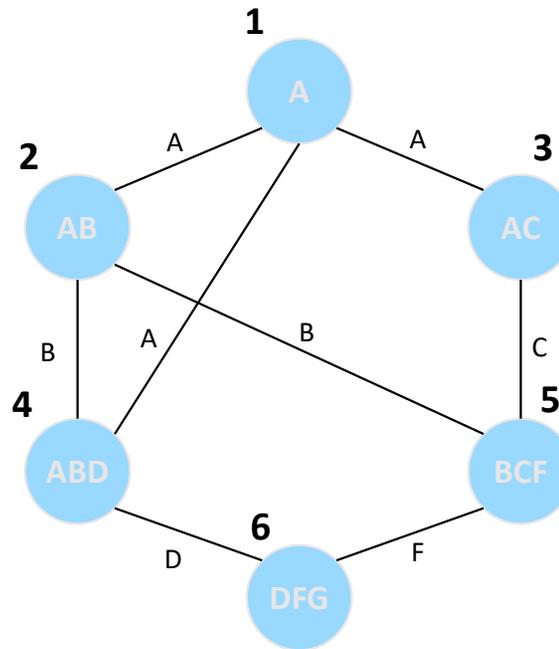
Iteration 5

R_1

A
1

h_2^1

A
1

h_3^1

A
1

h_4^1

A
1

h_5^2

B
3

h_4^2

B
3

h_1^2

A
1

R_2

A	B
1	3

R_3

A	C
1	2
3	2

h_1^3

A
1

h_5^3

C
2

h_6^4

D
2

h_2^4

B
3

h_1^4

A
1

R_4

A	B	D
1	3	2

R_5

B	C	F
3	2	1

h_2^5

B
3

h_3^5

C
2

h_6^5

F
1

R_6

D	F	G
2	1	3

h_4^6

D
2

h_5^6

F
1

$$R_i \leftarrow R_i \cap (\bowtie_{k \in ne(i)} h_k^i) \qquad (2)$$

Iteration 5

R_1

A
1

R_2

A	B
1	3

R_3

A	C
1	2

R_4

A	B	D
1	3	2
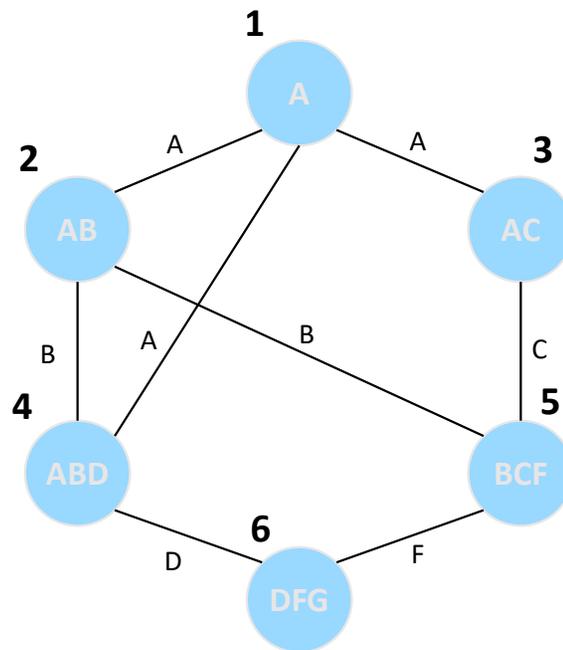
R_5

B	C	F
3	2	1

R_6

D	F	G
2	1	3

Tractable classes

Theorem 3.7.1   1. *The consistency binary constraint networks having no cycles can be decided by arc-consistent*

2. *The consistency of binary constraint networks with bi-valued domains can be decided by path-consistency,*

3. *The consistency of Horn cnf theories can be decided by unit propagation.*

Outline

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