Consistency algorithms

Chapter 3

Arc-consistency

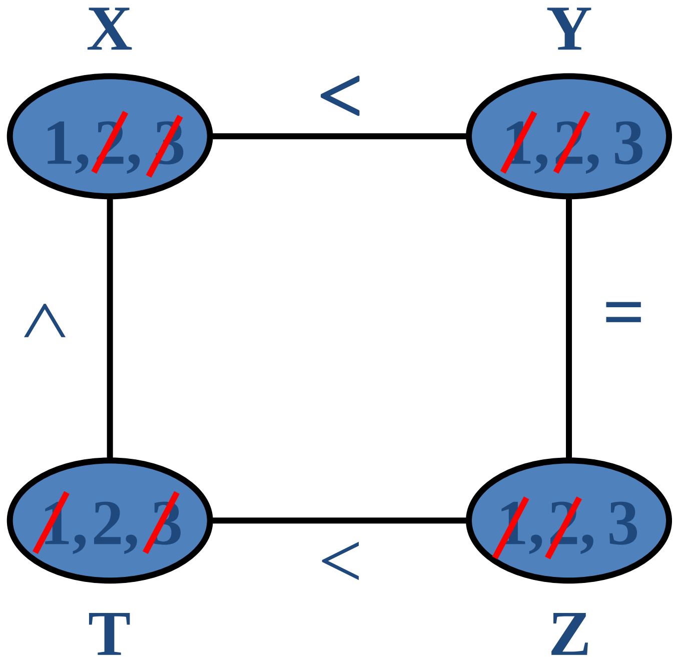
$1 \leq X, Y, Z, T \leq 3$

$X < Y$

$Y = Z$

$T < Z$

$X \leq T$



Arc-consistency

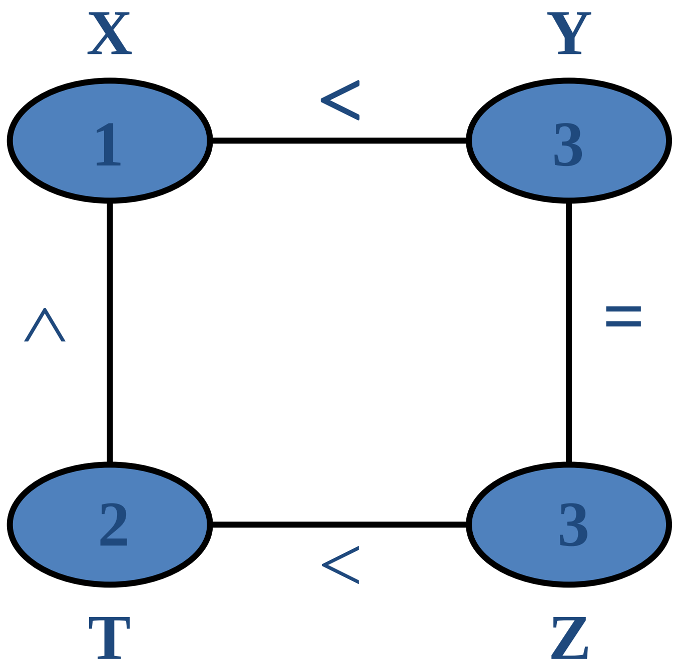
$1 \leq X, Y, Z, T \leq 3$

$X < Y$

$Y = Z$

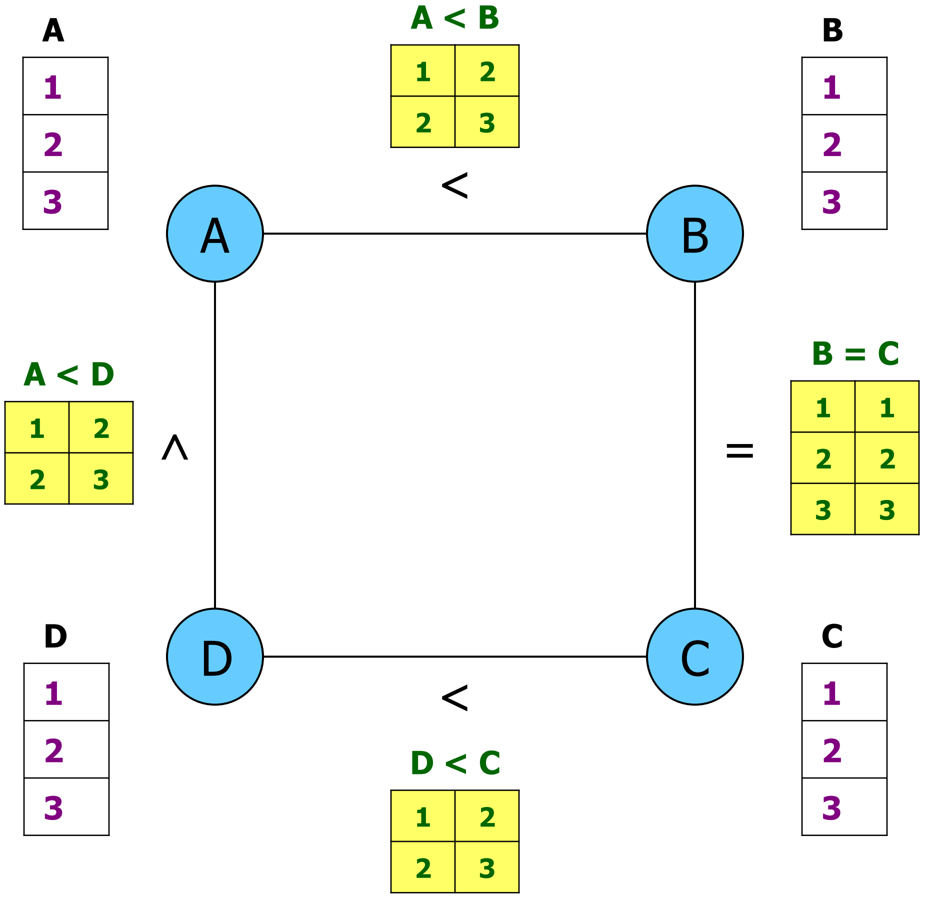
$T < Z$

$X \leq T$



Arc-consistency

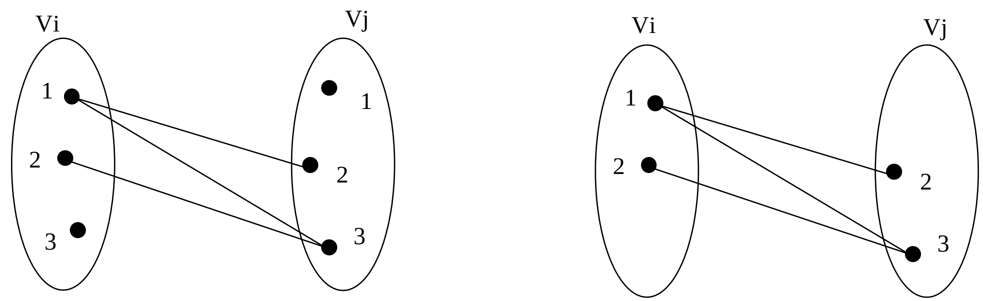
- Sound
- Incomplete
- Always converges (polynomial)



Arc-consistency

Definition: Given a constraint graph **G**,

- A <u>variable V_i is arc-consistent relative to</u> V_j iff for every value $a \in D_{Vi}$, there exists a value $b \in D_{Vj}$ | $(a, b) \in R_{Vi,Vj}$.



- <u>The constraint $R_{Vi,Vj}$ is arc-consistent</u> iff
  - V_i is arc-consistent relative to V_j and
  - V_j is arc-consistent relative to V_i.
- <u>A binary CSP is arc-consistent</u> iff every constraint (or sub-graph of size 2) is arc-consistent

Revise 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}

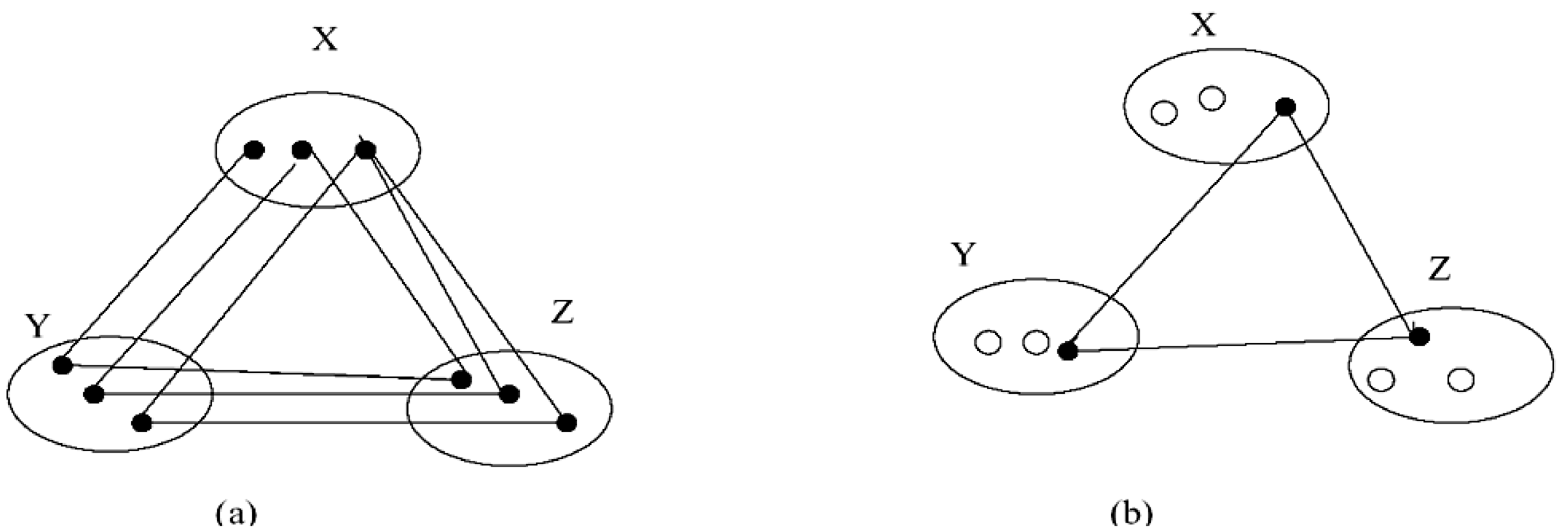
output: D_i, such that, x_i arc-consistent relative to x_j

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

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

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



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 $\mathcal{R}$

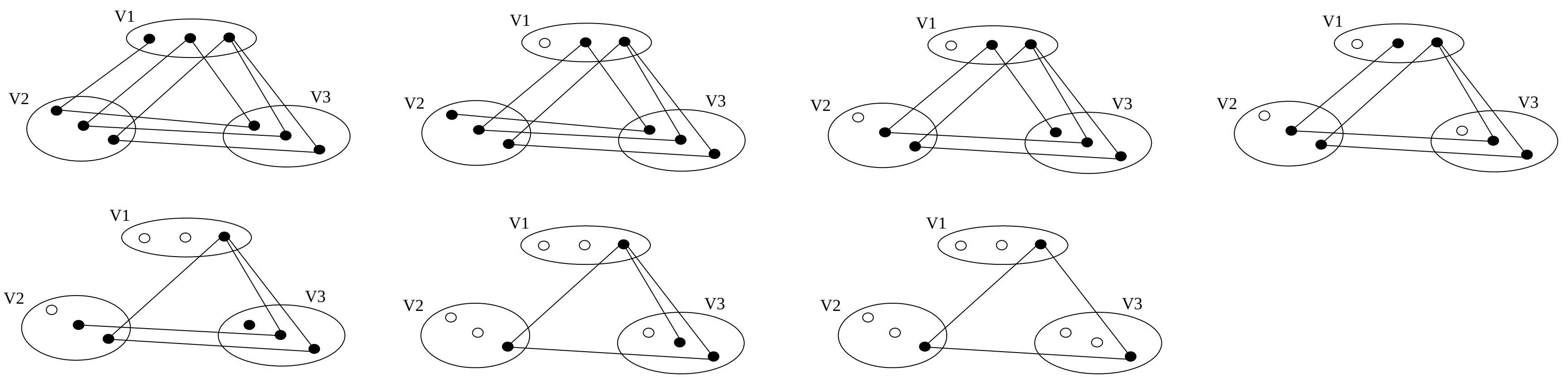
1. **repeat**
2.   **for** every pair $\{x_i, x_j\}$ that participates in a constraint
3.   Revise($(x_i), x_j$) (or $D_i \leftarrow D_i \cap \pi_i(R_{ij} \bowtie D_j)$)
4.   Revise($(x_j), x_i$) (or $D_j \leftarrow D_j \cap \pi_j(R_{ij} \bowtie D_i)$)
5.   **endfor**
6. **until** no domain is changed

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

- Complexity (Mackworth and Freuder, 1986):
- *e* = number of arcs, *n* variables, *k* values
- (*ek^2*, each loop, *nk* number of loops), best-case = *ek*,
- Arc-consistency is: $\Omega(ek^2)$
- Complexity of AC-1: O(enk^3)

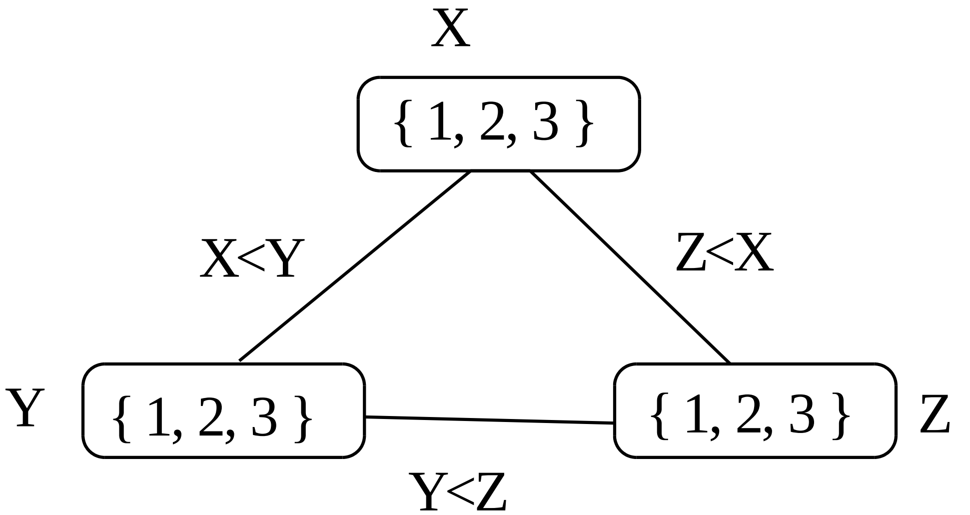
Arc consistency

1. AC may discover the solution



Arc consistency

2. AC may discover inconsistency



AC-3

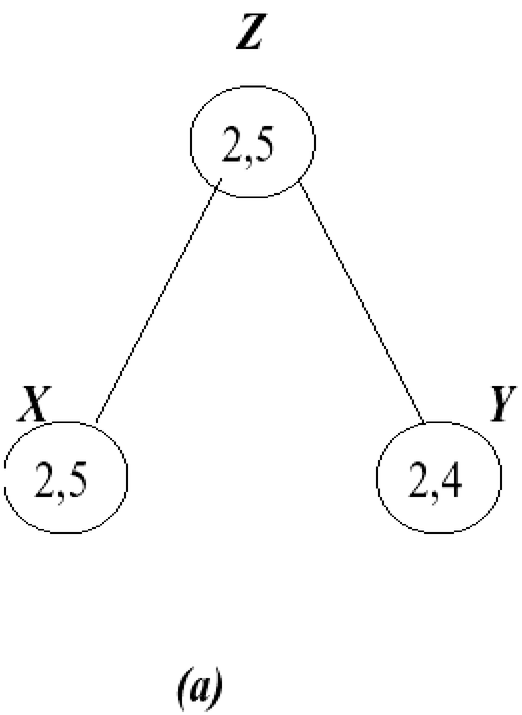
AC-3($\mathcal{R}$)

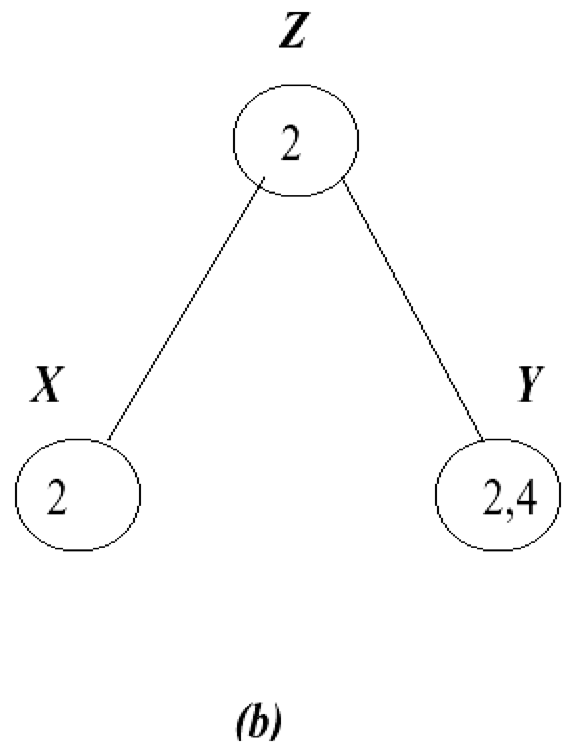
input: a network of constraints $\mathcal{R} = (X, D, C)$
output: $\mathcal{R}'$ which is the largest arc-consistent network equivalent to $\mathcal{R}$
for every pair $\{x_i, x_j\}$ that participates in a constraint $R_{ij} \in \mathcal{R}$
    $queue \leftarrow queue \cup \{(x_i, x_j), (x_j, x_i)\}$
endfor
while $queue \neq \{\}$
    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
        then $queue \leftarrow queue \cup \{(x_k, x_i), i \neq k\}$
    endif
endwhile

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

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

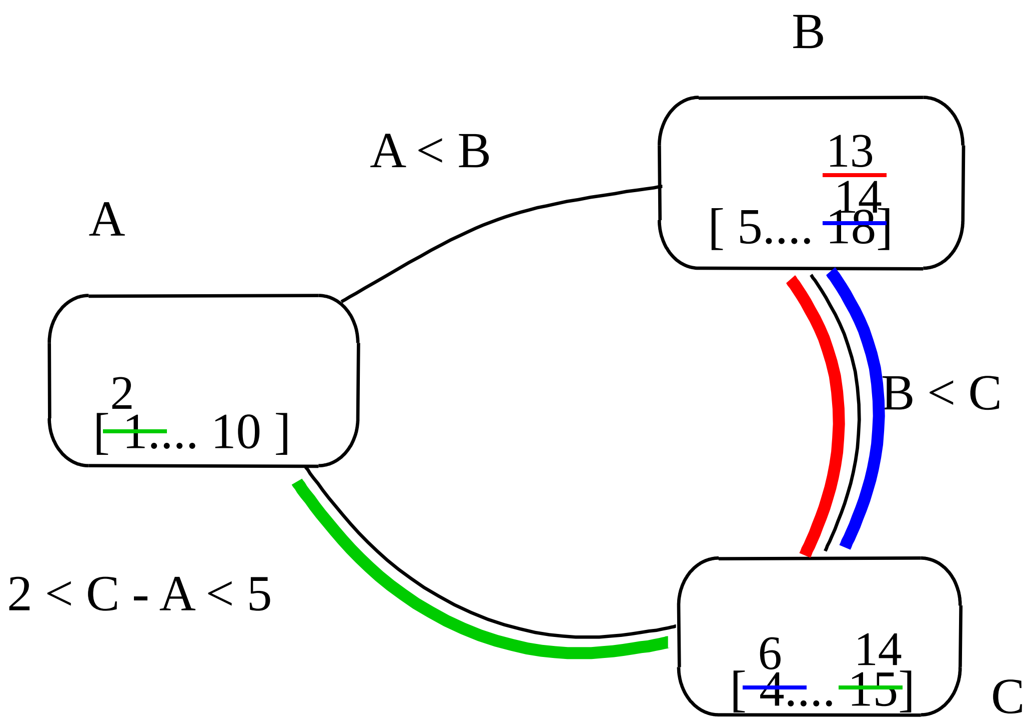
Example: A 3 variables network with 2 constraints: z divides x and z divides y (a) before and (b) after AC-3 is applied.





Constraint checking

→ Arc-consistency



1- B: [5 .. 14]

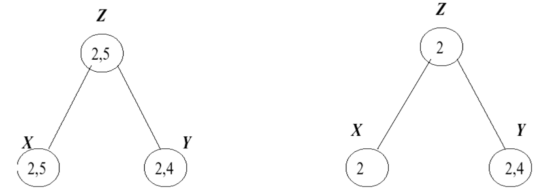
C: [6 .. 15]

2- A: [2 .. 10]

C: [6 .. 14]

3- B: [5 .. 13]

AC-4



(a) (b)

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

input: a network of constraints $\mathcal{R}$

output: An arc-consistent network equivalent to $\mathcal{R}$

Initialization: $M \leftarrow \emptyset$,
  initialize $S_{(x_i,c_i)}$, $counter(i, a_i, j)$ for all R_{ij}
  for all counters
    if $counter(x_i, a_i, x_j) = 0$ (if $< x_i, a_i >$ is unsupported by x_j)
      then add $< x_i, a_i >$ to $LIST$
    endif
  endfor
while $LIST$ is not empty
  choose $< x_i, a_i >$ from LIST, remove it, and add it to M
  for each $< x_j, a_j >$ in $S_{(x_i,a_i)}$
    decrement $counter(x_j, a_j, x_i)$
    if $counter(x_j, a_j, x_i) = 0$
      then add $< x_j, a_j >$ to $LIST$
    endif
  endfor
endwhile

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

- Complexity: $O(ek\,)$
- (Counter is the number of supports to ai in xi from xj. S_(xi,ai) is the set of pairs that (xi,ai) supports)

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 $O(nek^3)$
- **AC-3**, queue-based $O(ek^3)$
- **AC-4**, context-based, optimal $O(ek^2)$
- **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)…

Using constraint tightness in analysis

t = number of tuples bounding a constraint

- **AC-1**: brute-force, $O(nek^3)$ $O(nekt)$
- **AC-3**, queue-based $O(ek^3)$ $O(ekt)$
- **AC-4**, context-based, optimal $O(et)$
- **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)…

DRAC on the dual join-graph

R_1

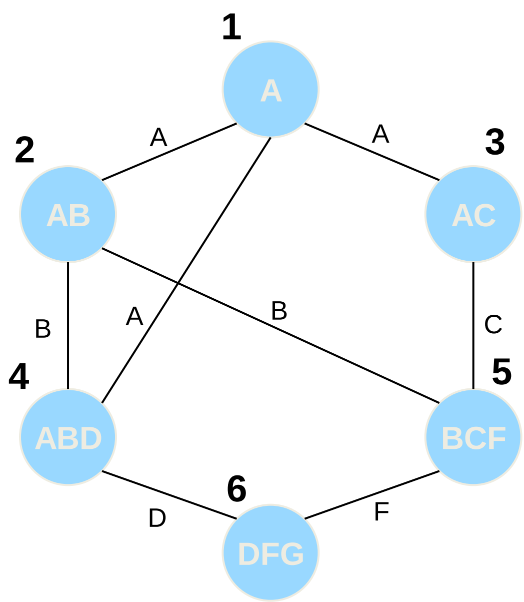
A
1
2
3

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

Distributed Relational Arc-Consistency

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

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

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

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

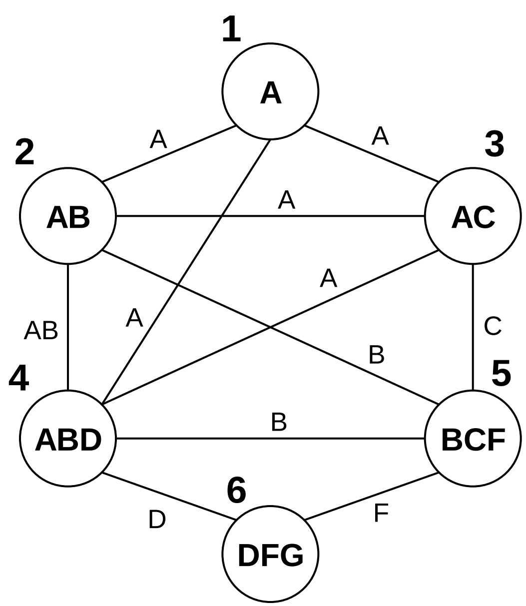
A
1
3

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

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

Iteration 1

R_1

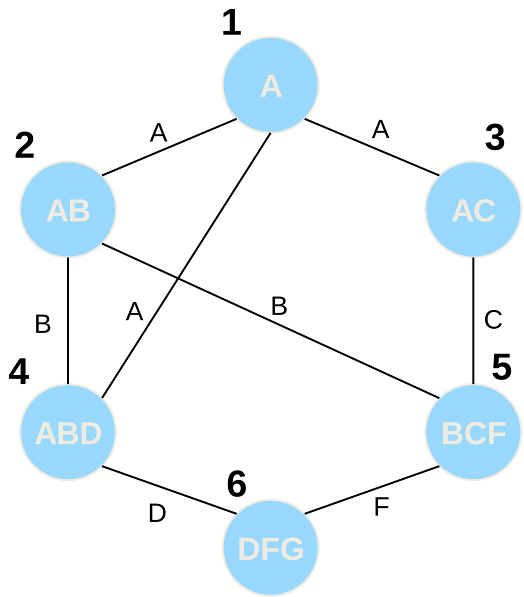
A
1
3

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

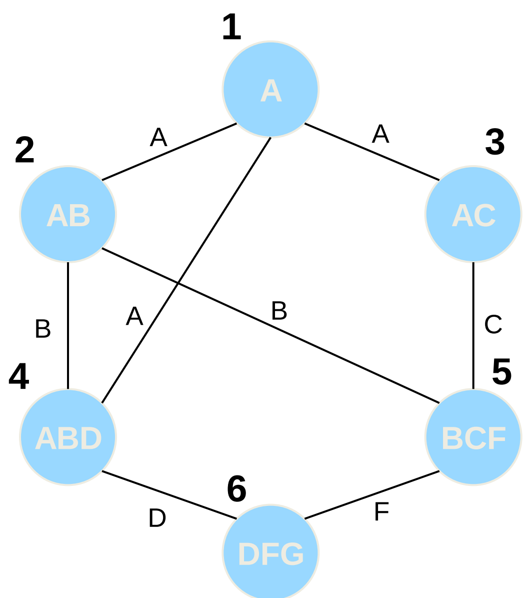
A	C
1	2
3	2

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

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

Iteration 2

R_1

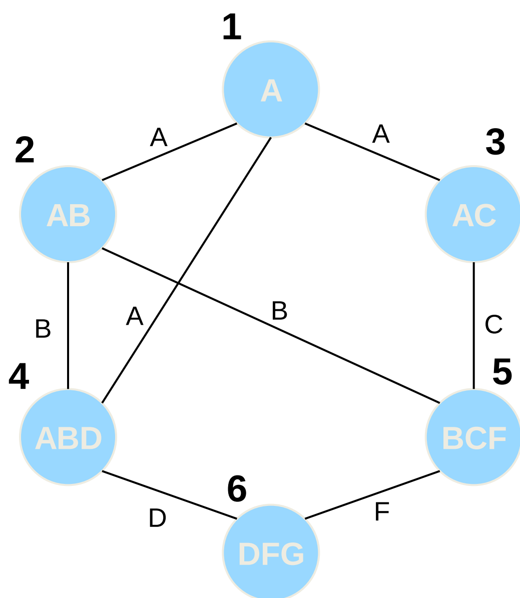
A
1
3

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

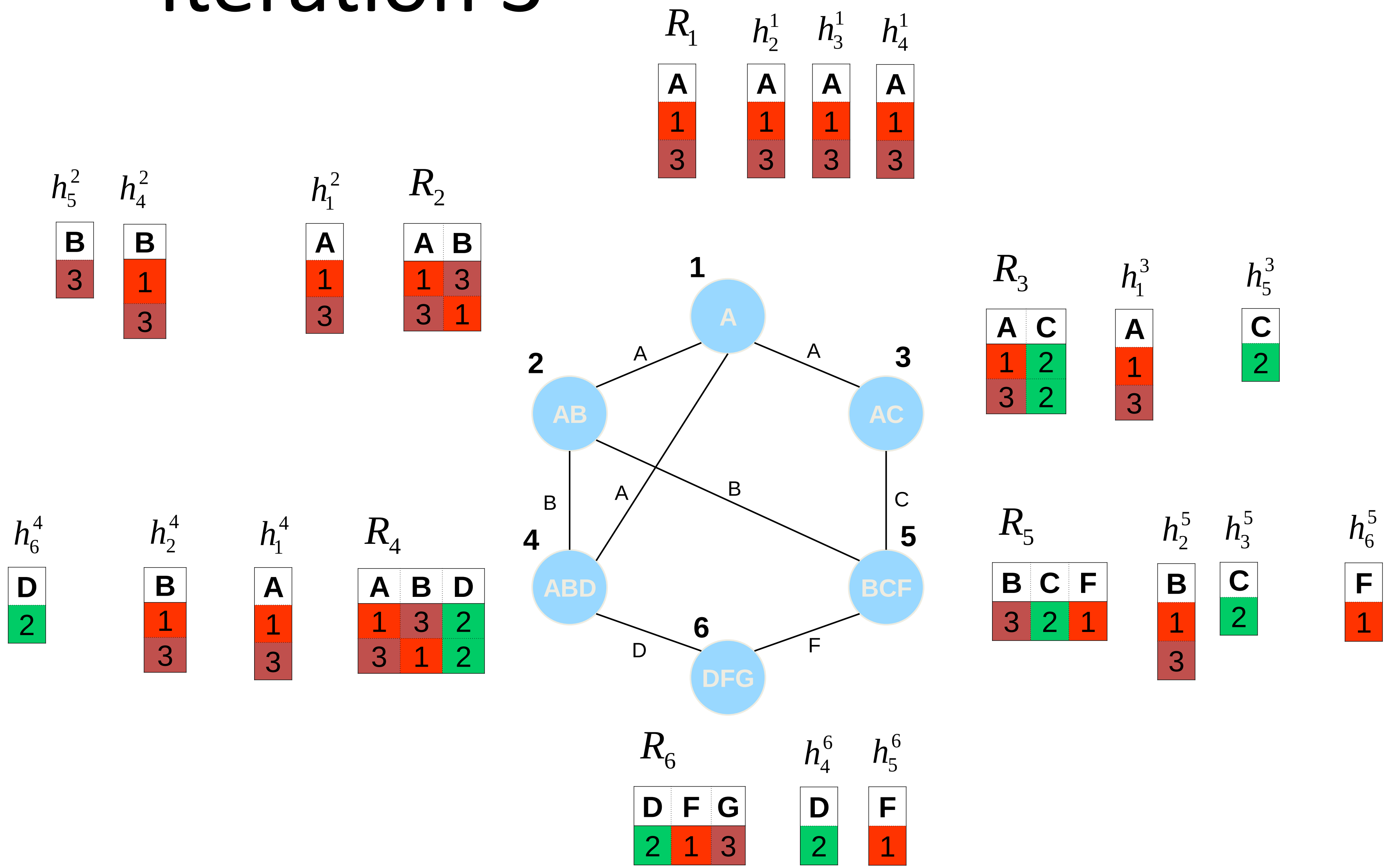
B	C	F
3	2	1

R_6

D	F	G
2	1	3

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

Iteration 3



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

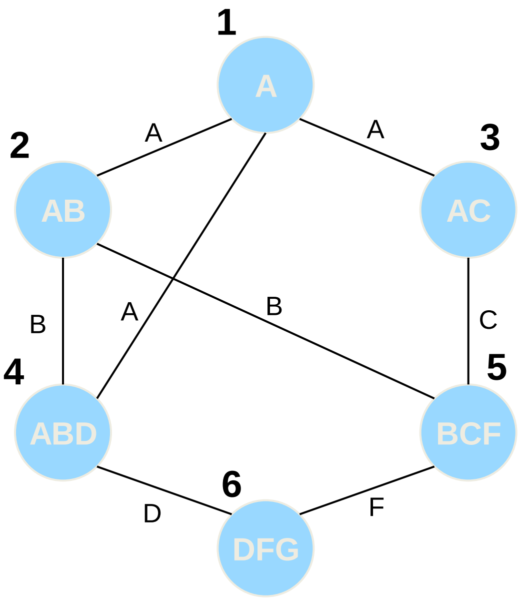
Iteration 3

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

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

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

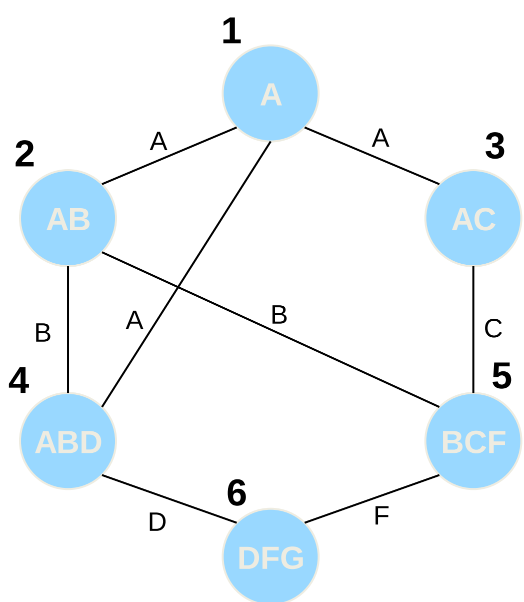
B
1
3

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

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

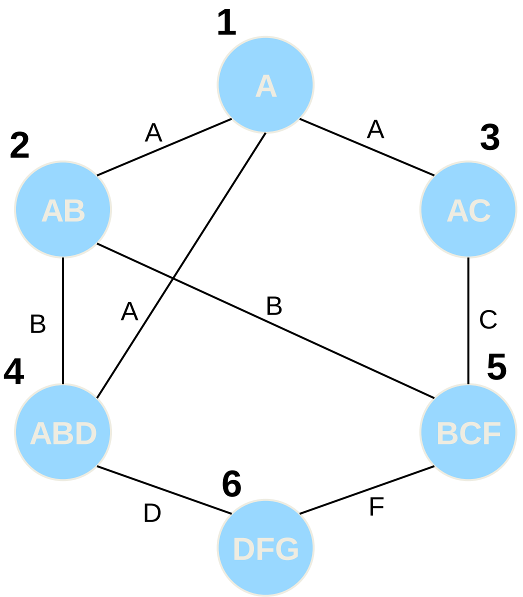
Iteration 4

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

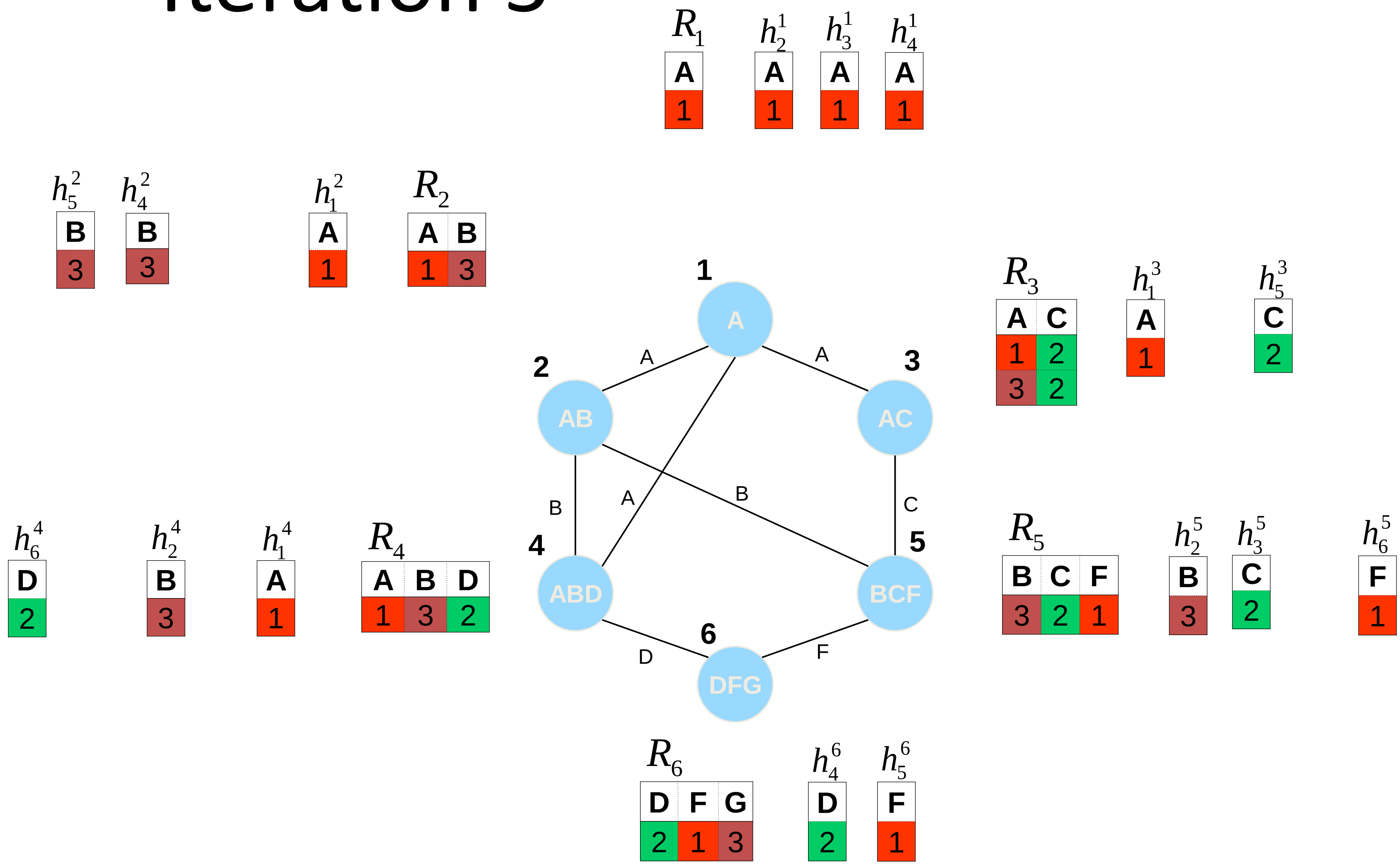
B	C	F
3	2	1

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



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

Iteration 3

R_1

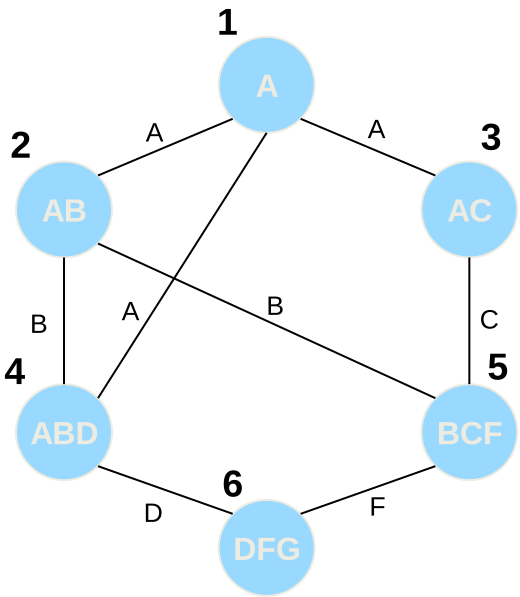
A
1

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

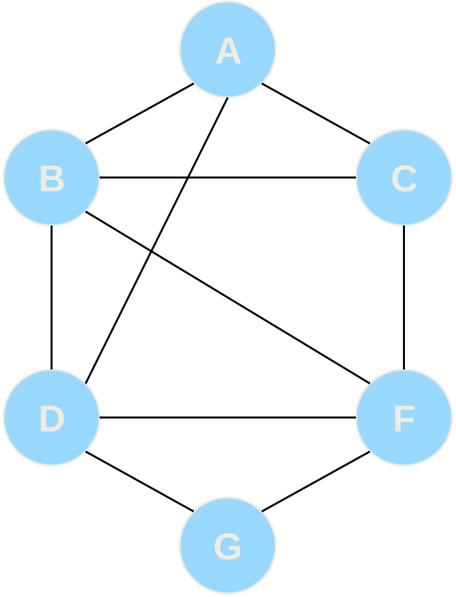
D	F	G
2	1	3

Distributed Arc-Consistency

– Arc-consistency can be formulated as a distributed algorithm:

$$D_i^j \leftarrow \pi_j(R_{ij} \bowtie D_i) \quad (1)$$

$$D_i \leftarrow D_i \cap (\bowtie_{k \in ne(i)} D_k^i) \quad (2)$$



a Constraint network

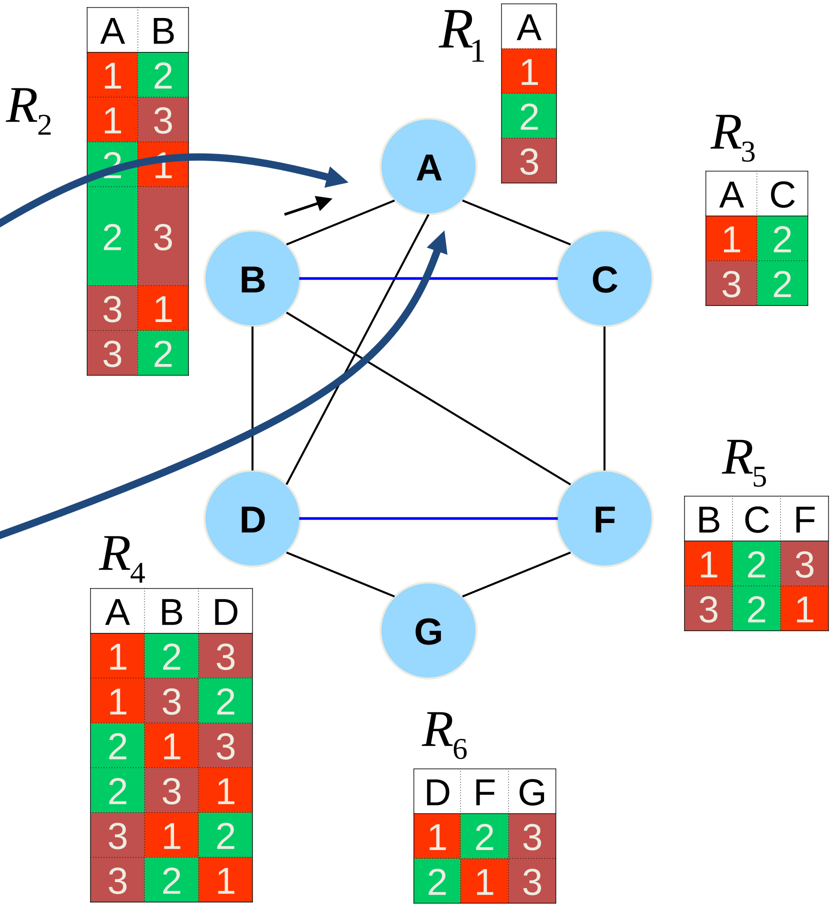
Relational Arc-consistency

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_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

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.

Path-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.

Path-consistency

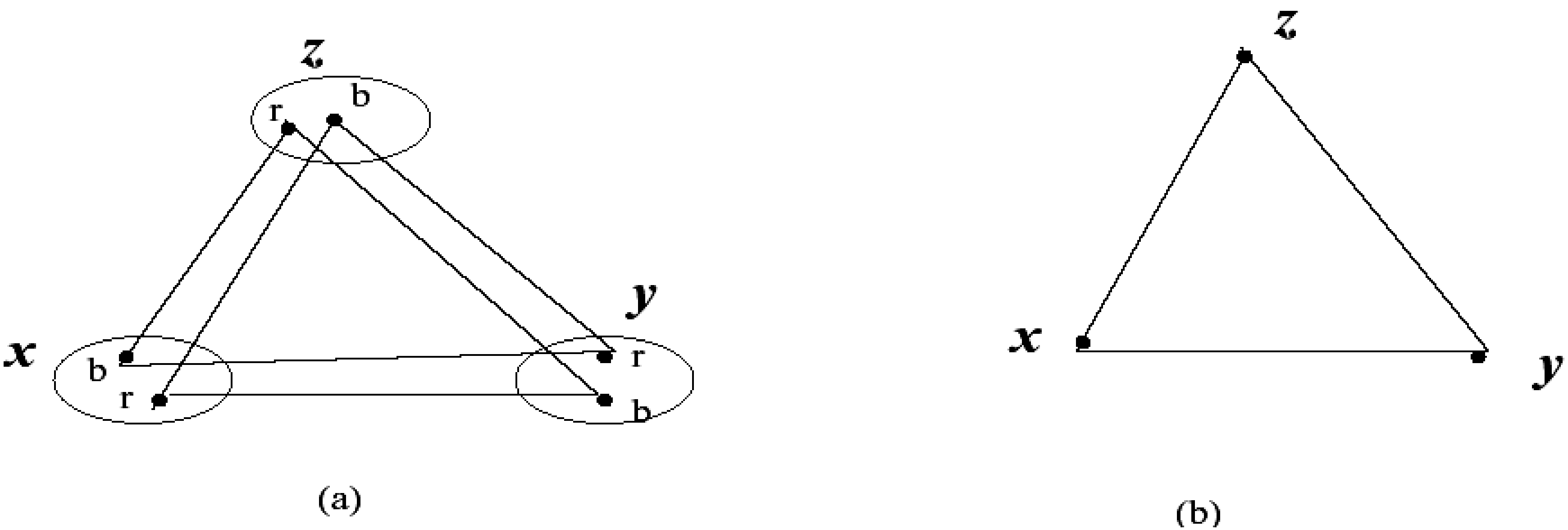


Figure 3.8: (a) The matching diagram of a 2-value graph coloring problem. (b) Graphical picture of path-consistency using the matching diagram.

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

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

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

PC-1

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

input: a network $\mathcal{R} = (X, D, C)$.

output: a path consistent network equivalent to $\mathcal{R}$.

repeat
    for $k \leftarrow 1$ to n
        for $i, j \leftarrow 1$ to n
            $R_{ij} \leftarrow R_{ij} \cap \pi_{ij}(R_{ik} \bowtie D_k \bowtie R_{kj})$/* $(Revise - 3((i,j),k))$
        endfor
    endfor
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-3($\mathcal{R}$)

input: a network $\mathcal{R} = (X, D, C)$.
output: $\mathcal{R}'$ a path consistent network equivalent to $\mathcal{R}$.

$Q \leftarrow \{(i,k,j) \mid 1 \leq i < j \leq n, 1 \leq k \leq n, k \neq i, k \neq j\ \}$
while Q is not empty
  select and delete a 3-tuple (i,k,j) from Q
  $R_{ij} \leftarrow R_{ij} \cap \pi_{ij}(R_{ik} \bowtie D_k \bowtie R_{kj})$ /* (Revise-3($(i,j),k$))
  if R_{ij} changed then
  $Q \leftarrow Q \cup \{(l,i,j)(l,j,i) \mid 1 \leq l \leq n, l \neq i, l \neq j\}$
endwhile

Figure 3.11: Path-consistency-3 (PC-3)

- Complexity: $O(n^3k^5)$
- Optimal PC-4: $O(n^3k^3)$
- (each pair deleted may add: 2n-1 triplets, number of pairs: O(n^2 k^2) → size of Q is O(n^3 k^2), processing is O(k^3))

Example: before and after path-consistency

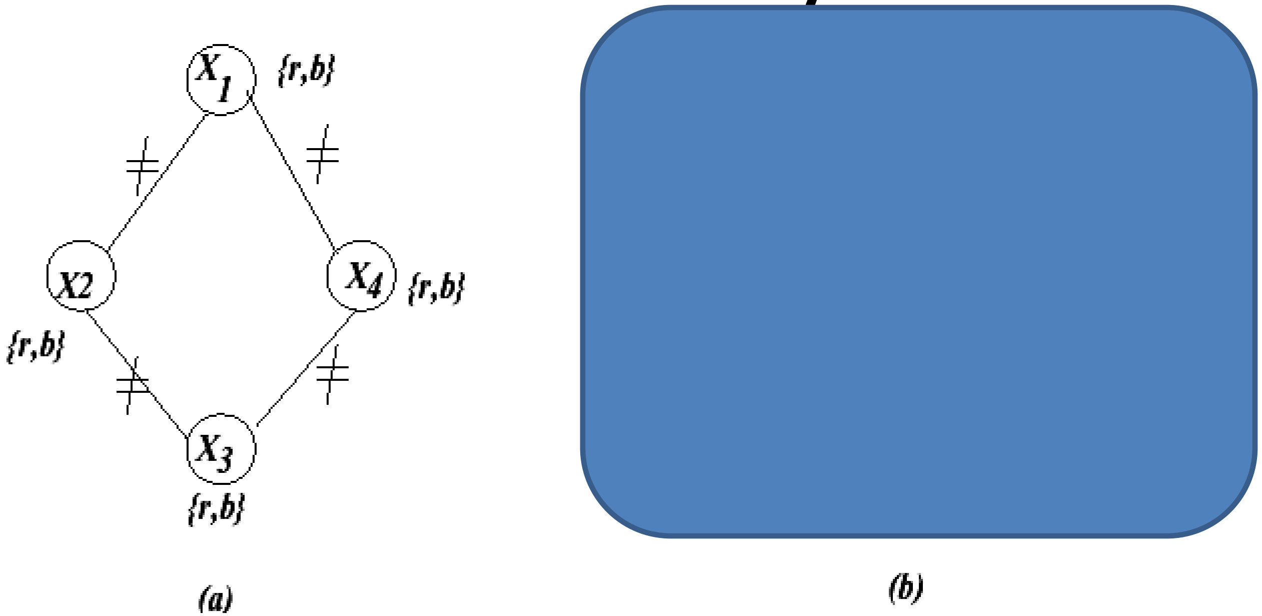


Figure 3.12: A graph-coloring graph (a) before path-consistency (b) after path-consistency

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

Example: before and after path-consistency

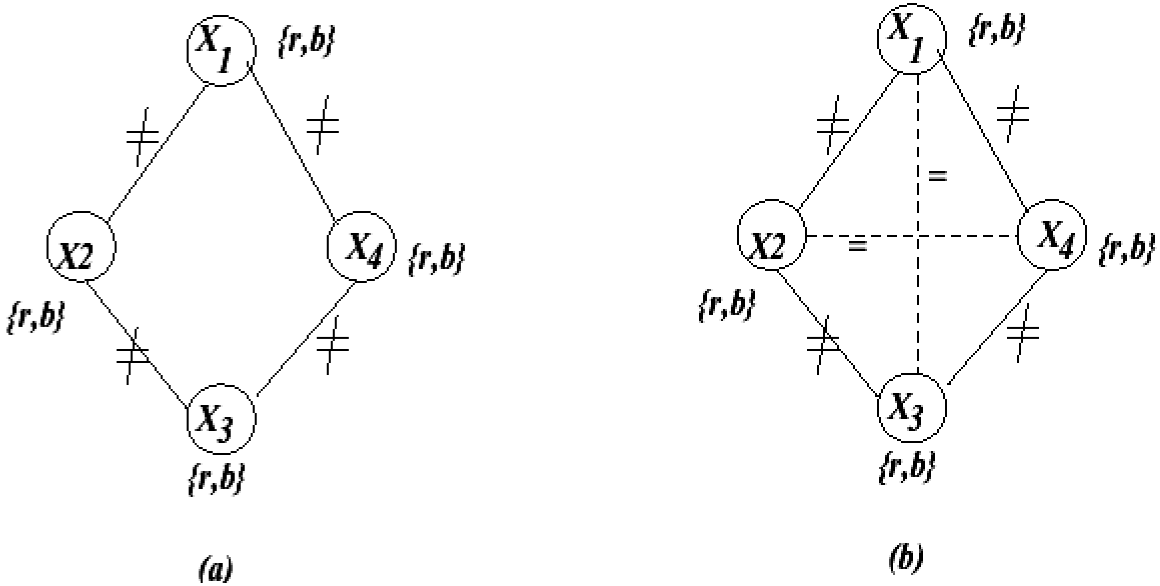


Figure 3.12: A graph-coloring graph (a) before path-consistency (b) after path-consistency

- 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

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

- 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

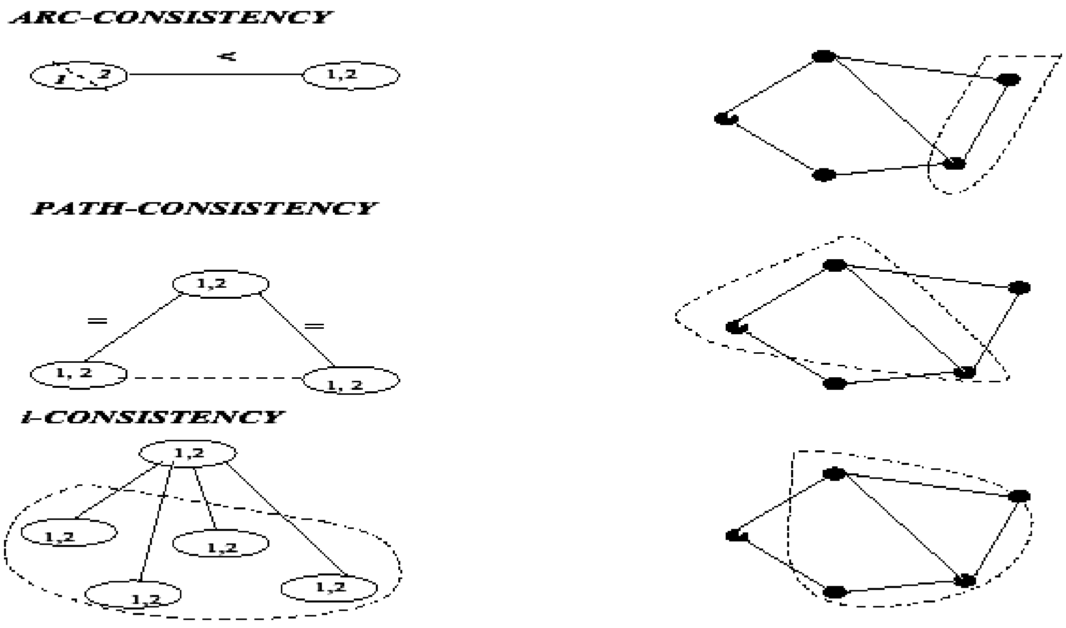


Figure 3.17: The scope of consistency enforcing: (a) arc-consistency, (b) path-consistency, (c) i-consistency

Higher levels of consistency, global-consistency

Definition 3.4.1 (i-consistency, global consistency) *Given a general network of constraints $\mathcal{R} = (X, D, C)$, a relation $R_S \in C$ where $|S| = i - 1$ is i-consistent relative to a variable y not in S iff for every $t \in R_S$, there exists a value $a \in D_y$, s.t. (t, a) is consistent. 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 $j \leq i$. A strongly n-consistent network, where n is the number of variables in the network, is called* globally consistent.

Revise-i

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

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
- For arbitrary constraints: $O((2k)^i)$

4-queen example

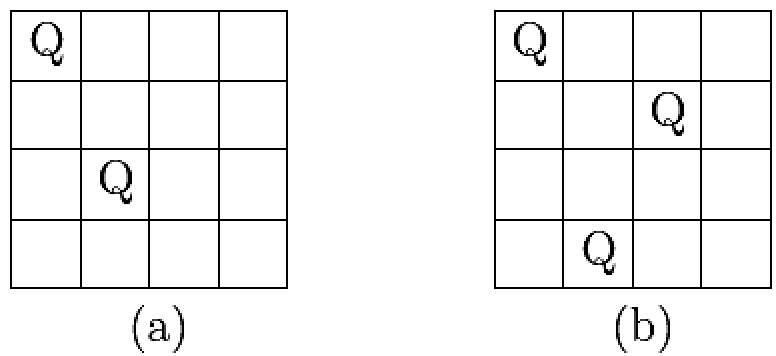


Figure 3.13: (a) Not 3-consistent; (b) Not 4-consistent

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. $R_S \leftarrow R_S \cap \pi_S(\Join_{S' \in \mathcal{S}} R_{S'})$ (this is Revise-i(S, x_i))
6. **endfor**
7. **until** no constraint is changed.

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 $O(n^ik^i)$, respectively. A lower bound for enforcing i-consistency is $\Omega(n^ik^i)$.* □

Arc-consistency for non-binary constraints:

Generalized arc-consistency

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.*

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

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

Relational arc-consistency:

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

Examples of generalized arc-consistency

- x+y+z <= 15 and z >= 13 implies

x<=2, y<=2

- Example of relational arc-consistency

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

- x+y <= 2

What is SAT?

Given a sentence:

- ***Sentence***: conjunction of clauses

$$\left(c_1 \vee \neg c_4 \vee c_5 \vee c_6\right) \wedge \left(c_2 \vee \neg c_3\right) \wedge \left(\neg c_4\right)$$

- ***Clause***: disjunction of literals $\left(c_2 \vee \neg c_3\right)$

- ***Literal***: a term or its negation $c_1, \neg c_6$

- ***Term***: Boolean variable $c_1 = 1 \Leftrightarrow \neg c_1 = 0$

Question: Find an assignment of truth values to the Boolean variables such the sentence is satisfied.

Boolean constraint propagation

Example: party problem

- If Alex goes, then Becky goes: $\mathbf{A} \rightarrow \mathbf{B}$ (or, $\neg\mathbf{A} \vee \mathbf{B}$)
- If Chris goes, then Alex goes: $\mathbf{C} \rightarrow \mathbf{A}$ (or, $\neg\mathbf{C} \vee \mathbf{A}$)
- **Query:**

Is it possible that Chris goes to the party but Becky does not?

$$\Updownarrow$$

Is *propositional theory*

$$\varphi = \{\neg\mathbf{A} \vee \mathbf{B},\ \neg\mathbf{C} \vee \mathbf{A},\ \neg\mathbf{B},\ \mathbf{C}\} \text{ satisfiable?}$$

CSP is NP-Complete

- Verifying that an assignment for all variables is a solution
  - Provided constraints can be checked in polynomial time
- Reduction from 3SAT to CSP
  - Many such reductions exist in the literature (perhaps 7 of them)

Problem reduction

Example: CSP into SAT (*proves nothing, just an exercise*)

Notation: variable-value pair = **vvp**

- vvp $\rightarrow$ term
  - V_1 = {a, b, c, d} yields $x_1 = (V_1, a)$, $x_2 = (V_1, b)$, $x_3 = (V_1, c)$, $x_4 = (V_1, d)$,
  - V_2 = {a, b, c} yields $x_5 = (V_2, a)$, $x_6 = (V_2, b)$, $x_7 = (V_2, c)$.
- The vvp's of a variable $\rightarrow$ disjunction of terms
  - V_1 = {a, b, c, d} yields
- (Optional) At most one VVP per variable

$$x_1 \vee x_2 \vee x_3 \vee x_4$$

$$\left(x_1 \wedge \neg x_2 \wedge \neg x_3 \wedge \neg x_4\right) \vee \left(\neg x_1 \wedge x_2 \wedge \neg x_3 \wedge \neg x_4\right) \vee$$
$$\left(\neg x_1 \wedge \neg x_2 \wedge x_3 \wedge \neg x_4\right) \vee \left(\neg x_1 \wedge \neg x_2 \wedge \neg x_3 \wedge x_4\right)$$

CSP into SAT (cont.)

Constraint: $C_{V_1V_2} = \{(a,a),(a,b),(b,c),(c,b),(d,a)\}$

1. Way 1: Each inconsistent tuple $\rightarrow$ one disjunctive clause
   – For example: $\neg x_1 \vee \neg x_7$ how many?

2. Way 2:
   a) Consistent tuple $\rightarrow$ conjunction of terms $x_1 \wedge x_5$
   b) Each constraint $\rightarrow$ disjunction of these conjunctions

$$\left(x_1 \wedge x_5\right) \vee \left(x_1 \wedge x_6\right) \vee \left(x_2 \wedge x_7\right)$$
$$\vee \left(x_3 \wedge x_6\right) \vee \left(x_4 \wedge x_5\right)$$

$\rightarrow$ transform into conjunctive normal form (CNF)

Question: find a truth assignment of the Boolean variables such that the sentence is satisfied

Constraint propagation for Boolean constraints: Unit 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 $\gamma = \textbf{\textit{resolve}}(\beta, T)$.
   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

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

$$x + y = 10$$

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

$$by - adding\ - x + y = 10, -y \le -5$$

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

$$y + z \le 3$$

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

$$obtained\ - by - adding\, , x + y = 10, -y - z \ge -3$$

More arc-based consistency

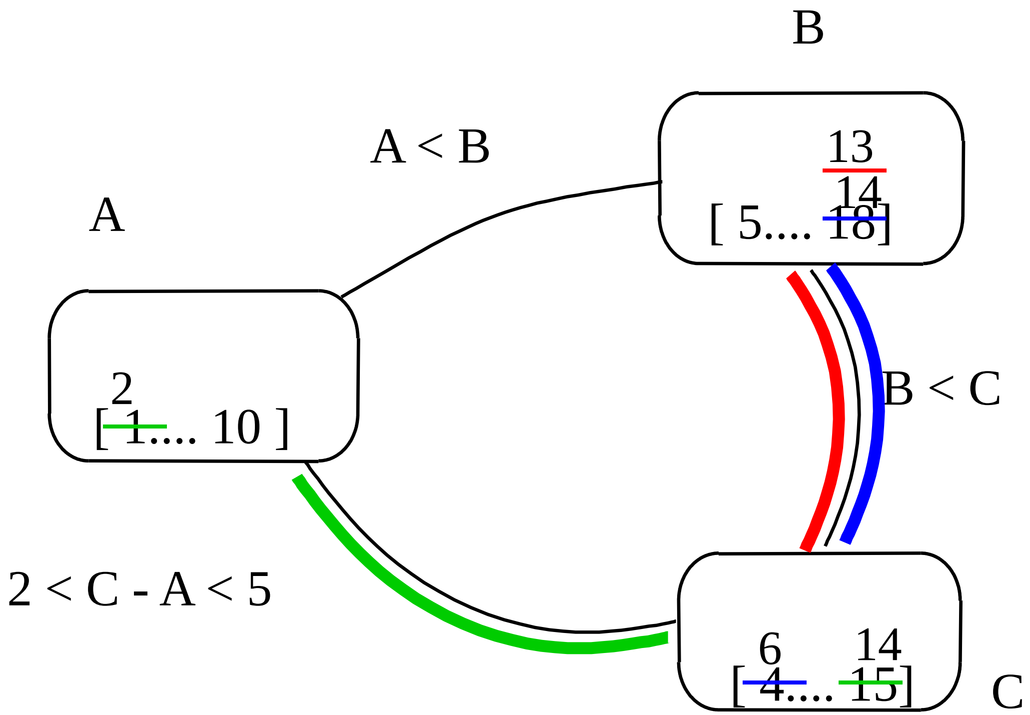
- Global constraints: e.g., all-different constraints
  - Special semantic constraints that appears often in practice and a specialized constraint propagation. Used in constraint programming.
- Bounds-consistency: pruning the boundaries of domains

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.*

Constraint checking

→ Arc-consistency



1- B: [5 .. 14]

C: [6 .. 15]

2- A: [2 .. 10]

C: [6 .. 14]

3- B: [5 .. 13]

Bounds consistency for Alldifferent constraints

Example 3.5.5 Consider the constraint problem with variables $x_1, ...x_6$, each with domains $1, ..., 6$, 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? □

For alldiff bounds consistency can be enforced in O(nlog n)

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.

Changes in the network graph as a result of arc-consistency, path-consistency and 4-consistency.