Chapter 3

Consistency algorithms
Consistency methods

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

1 ≤ X, Y, Z, T ≤ 3
X < Y
Y = Z
T < Z
X ≤ T
Arc-consistency

1 ≤ X, Y, Z, T ≤ 3
X < Y
Y = Z
T < Z
X ≤ T
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 \( R_{ij} \in \mathcal{C} \), 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.
**Arc-consistency**

Only domain constraints are recorded: \( R_A \leftarrow \prod A R_{AB} \Join D_B \)

**Example:** \( R_X = \{1, 2, 3\}, R_Y = \{1, 2, 3\}, \) constriant \( X < Y \)
reduces domain of \( X \) to \( R_X = \{1, 2\} \).
Revise for arc-consistency

\texttt{REVISE}((x_i), x_j)

\textbf{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 \(R_{ij}\)

\textbf{output:} \(D_i\), such that, \(x_i\) arc-consistent relative to \(x_j\)

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

\textbf{Figure 3.2:} The Revise procedure

\[ D_i \leftarrow D_i \cap \pi_i (R_{ij} \otimes D_j) \]
Figure 3.3: (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})

\textbf{input:} a network of constraints \( \mathcal{R} = (X, D, C) \)

\textbf{output:} \( \mathcal{R'} \) which is the loosest arc-consistent network equivalent to \( \mathcal{R} \)

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

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

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

AC-3(\mathcal{R})

\textbf{input:} a network of constraints \( \mathcal{R} = (X, D, C) \)

\textbf{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. \( \text{queue} \leftarrow \text{queue} \cup \{(x_i, x_j), (x_j, x_i)\} \)
3. endfor
4. while \( \text{queue} \neq \{\} \)
5. select and delete \( (x_i, x_j) \) from \text{queue}
6. \( \text{Revise}((x_i), x_j) \)
7. if \( \text{Revise}((x_i), x_j) \) causes a change in \( D_i \)
8. then \( \text{queue} \leftarrow \text{queue} \cup \{(x_k, x_i), i \neq k\} \)
9. endif
10. endwhile

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

- Complexity: \( O(ek^3) \)
- Best case \( O(ek) \), since each arc may be processed in \( O(2k) \)
Example: A three variable network, with two constraints: \( z \) divides \( x \) and \( z \) divides \( y \) (a) before and (b) after AC-3 is applied.
AC-4

AC-4(ℜ)

**input:** a network of constraints ℜ
**output:** An arc-consistent network equivalent to ℜ

1. Initialization: \( M \leftarrow \emptyset \),
2. initialize \( S_{(x_i,c_i)} \), \( \text{counter}(i,a_i,j) \) for all \( R_{ij} \)
3. for all counters
4. \hspace{1cm} if \( \text{counter}(x_i,a_i,x_j) = 0 \) (if \( <x_i,a_i> \) is unsupported by \( x_j \))
5. \hspace{1cm} then add \( <x_i,a_i> \) to LIST
6. \hspace{1cm} 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 \( S_{(x_i,c_i)} \)
11. \hspace{1cm} decrement \( \text{counter}(x_j,a_j,x_i) \)
12. \hspace{1cm} if \( \text{counter}(x_j,a_j,x_i) = 0 \)
13. \hspace{1cm} then add \( <x_j,a_j> \) to LIST
14. \hspace{1cm} 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 \( (xi,ai) \) supports)
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:
  \[ D_i \leftarrow D_i \cap \pi_i (R_{ij} \otimes D_j) = D_i \leftarrow D_i \cap h_i^j \]

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

\textbf{Revise-3}((x, y), z)

\textbf{input}: a three-variable subnetwork over \((x, y, z), R_{xy}, R_{yz}, R_{xz}\).
\textbf{output}: revised \(R_{xy}\) path-consistent with \(z\).

1. \textbf{for} each pair \((a, b) \in R_{xy}\)
2. \quad \textbf{if} no value \(c \in D_z\) exists such that \((a, c) \in R_{xz}\) and \((b, c) \in R_{yz}\)
3. \quad \textbf{then} delete \((a, b)\) from \(R_{xy}\).
4. \quad \textbf{endif}
5. \textbf{endfor}

\begin{equation}
R_{ij} \leftarrow R_{ij} \cap \pi_{ij} (R_{ik} \otimes D_k \otimes R_{kj})
\end{equation}

- Complexity: \(O(k^3)\)
- Best-case: \(O(t)\)
- Worst-case \(O(tk)\)
PC-1

PC-1(\(R\))

**input:** a network \(R = (X, D, C)\).

**output:** a path consistent network equivalent to \(R\).

1. repeat
2. for \(k \leftarrow 1\) to \(n\)
3. for \(i, j \leftarrow 1\) to \(n\)
4. \(R_{ij} \leftarrow R_{ij} \cap \pi_{ij}(R_{ik} \times D_k \times R_{kj})/* (Revise - 3((i, j), k))\)
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 \(\rightarrow O(n^3 k^3)\)
- 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 \(\mathcal{R}\).

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

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

- **Complexity:** \(O(n^3 k^5)\)
- **Optimal PC-4:** \(O(n^3 k^3)\)
- (each pair deleted may add: 2n-1 triplets, number of pairs: \(O(n^2 k^2)\) \(\rightarrow\)
- size of \(Q\) is \(O(n^3 k^2)\), processing is \(O(k^3)\)
Example: before and after path-consistency

![Graph Coloring Example](image)

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?
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 \( R = (X, D, C) \), 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, \ldots, x_{i-1}\}, x_i)

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

**output:** a constraint $R_S$, $S = \{x_1, \ldots, x_{i-1}\}$ $i$-consistent relative to $x_i$.

1. for each instantiation $\bar{a}_{i-1} = (\langle x_1, a_1 \rangle, \langle x_2, a_2 \rangle, \ldots, \langle x_{i-1}, a_{i-1} \rangle)$ do,
2. if no value of $a_i \in D_i$ exists s.t. $(\bar{a}_{i-1}, a_i)$ is consistent
   \hspace{1em} \text{then} delete $\bar{a}_{i-1}$ from $R_S$
   \hspace{1em} (Alternatively, let $S$ be the set of all subsets of $\{x_1, \ldots, x_i\}$ that contain $x_i$ and appear as scopes of constraints of $\mathcal{R}$, then $R_S \leftarrow R_S \cap \pi_S(\forall_{S' \subseteq S} R_{S'})$)
3. endfor

\textbf{Figure 3.14: Revise-i}

- **Complexity:** for binary constraints \( O(k^i) \)
- **For arbitrary constraints:** \( O((2k)^i) \)
4-queen example

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

\textsc{i-consistency}(\mathcal{R})

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

\textbf{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 \( S \) be the set of all subsets in of \( \{x_1, \ldots, x_i\} \) \text{scheme}(\mathcal{R}) \) that contain \( x_i \)
4. \( R_S \leftarrow R_S \cap \pi_S (\bigwedge_{S' \in S} R_{S'}) \) (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 \( O(n^i k^i) \), respectively. A lower bound for enforcing i-consistency is \( \Omega(n^i k^i) \). \( \square \)
Definition 3.5.1 (generalized arc-consistency) Given a constraint network $\mathcal{R} = (\mathcal{X}, \mathcal{D}, \mathcal{C})$, with $R_S \in \mathcal{C}$, 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.

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

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

Relational arc-consistency:

$$R_{S \setminus \{x\}} \leftarrow \pi_{S \setminus \{x\}} (R_S \otimes D_x)$$
Examples of generalized arc-consistency

- \{x + y + z \leq 15, z \geq 13\} \Rightarrow x \leq 2, y \leq 2

- Example of relational arc-consistency
  \{A \land B \rightarrow G, \neg G\}, \Rightarrow \neg A \lor \neg B
More arc-based consistency

- Global constraints: e.g., all-different constraints
  - Special semantic constraints that appear often in practice and a specialized constraint propagation. Used in constraint programming.
- Bounds-consistency: pruning the boundaries of domains
- Do exercise 16
Example for alldiff

- $A = \{3,4,5,6\}$
- $B = \{3,4\}$
- $C = \{2,3,4,5\}$
- $D = \{2,3,4\}$
- $E = \{3,4\}$
- Alldiff $(A,B,C,D,E)$
- Arc-consistency does nothing
- Apply GAC to sol$(A,B,C,D,E)$?
  - $\rightarrow A = \{6\}$, $F = \{1\}$.
- 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
- Global cardinality constraint (a value can be assigned a bounded number of times)
- The cumulative constraint (related to scheduling tasks)
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 for AllDifferent constraints

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

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

\[ C_5 : \text{alldifferent}\{x_1, x_2, x_3, x_4, x_5\} \]

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?
Boolean constraint propagation

- \((A \lor \neg B)\) and \((B)\)
  - \(B\) is arc-consistent relative to \(A\) but not vice-versa
- Arc-consistency achieved by resolution:
  \(\text{res}((A \lor \neg B), B) = A\)

Given also \((B \lor C)\), path-consistency means:
\(\text{res}((A \lor \neg B), (B \lor C)) = (A \lor C)\)

What can generalized arc-consistency do to cnfs?
Relational arc-consistency rule = unit-resolution
Boolean constraint propagation

Example: party problem

- If Alex goes, then Becky goes: \( A \rightarrow B \) (or, \( \neg A \lor B \) )
- If Chris goes, then Alex goes: \( C \rightarrow A \) (or, \( \neg C \lor A \) )
- Query:

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

\[ \Downarrow \]

Is propositional theory
\[ \varphi = \{ \neg A \lor B, \neg C \lor A, \neg B, C \} \] satisfiable?
Constraint propagation for Boolean constraints: Unit propagation

Procedure UNIT-PROPAGATION

Input: A cnf theory, $\varphi$, $d = Q_1, ..., Q_n$.
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 $\beta$ containing $T$ or $\neg T$
5. if $\beta$ contains $T$ delete $\beta$ (subsumption elimination)
6. else, For each clause $\gamma = resolve(\beta, T)$.
   if $\gamma$, the resolvent, is empty, the theory is unsatisfiable.
7. else, add the resolvent $\gamma$ to the theory and delete $\beta$.
   if $\gamma$ is a unit clause, add to Queue.
8. endfor.
9. endwhile.

Theorem 3.6.1 Algorithm UNIT-PROPAGATION has a linear time complexity.
Algorithms for relational and generalized arc-consistency

Think about the following:

- GAC-i apply AC-i to the dual problem when singleton variables are explicit: the bi-partite representation.
- What is the complexity?
- Relational arc-consistency: imitate unit propagation.
- Apply AC-1 on the dual problem where each subset of a scope is presented.
- Is unit propagation equivalent to AC-4?
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 \leq -5 \]

\[ z \in [-10, 10], \]
\[ y + z \leq 3 \]

.path \ - \ consistency \ \Rightarrow \ x - z \geq 7 \]

.obtained \ - \ by \ - \ adding, \ x + y = 10, -y - z \geq -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.
Changes in the network graph as a result of arc-consistency, path-consistency and 4-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 \( x_i \)

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

  \[ D_i \leftarrow D_i \cap h_i^j \]

- Node \( x_i \) updates its domain:

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

  \[ R_{S \setminus \{x\}} \leftarrow \pi_{S \setminus \{x\}} (R_S \otimes D_x) \]
The message that R2 sends to R1 is

R1 updates its relation and domains and sends messages to neighbors
Distributed Arc-Consistency

DR-AC can be applied to the dual problem of any constraint network.

\[ h_i^j \leftarrow \pi_k (R_i \times (\bigtimes_{k \in \text{ne}(i)} h_k^i)) \]  \hspace{1cm} (1)

\[ D_i \leftarrow D_i \cap (\bigtimes_{k \in \text{ne}(i)} D_k^i) \]  \hspace{1cm} (2)

b) Constraint network
DR-AC on a dual join-graph
Iteration 1

\[ h^j_i \leftarrow \pi_{h^j_i}(R_i \Join \bigotimes_{k \in \text{ne}(i)} h^j_k) \] (1)
$R_i \leftarrow R_i \cap \left( \bigotimes_{h \in \text{ne}(i)} h^i \right)$ (2)

Iteration 1

$R_1$
- A
  - 1
  - 3

$R_2$
- A
  - 1
- B
  - 3
- 2
  - 1
- 2
  - 3
- 3
  - 1

$R_3$
- A
  - 1
- C
  - 2
- 3
  - 2

$R_4$
- A
  - 1
- B
  - 3
- D
  - 2
- 2
  - 3
- 1
- 3
  - 1
- 3
  - 2
- 1

$R_5$
- B
  - 1
- C
  - 2
- F
  - 3
- 2
  - 1

$R_6$
- D
  - 2
- F
  - 1
- G
  - 3
Iteration 2

\[ h^j_i \leftarrow \pi_{h^j_i}(R_i \bowtie \bigotimes_{k \in \text{ne}(i)} h^j_k) \] (1)
Iteration 2

\[ R_i \leftarrow R_i \cap \left( \bigotimes_{h \in \text{ne}(i)} h^i \right) \]  \hspace{1cm} (2)
\[ h_6^j \leftarrow \pi_{I_4}(R_i \Join \bigotimes_{k \in n_{ec}(i)} h_k^j) \]
Iteration 3

\[ R_i \leftarrow R_i \cap \bigcap_{h \in \text{ne}(i)} h^i \]  (2)
\[ h^*_b \leftarrow \pi_{h^*_b}(R_i \bowtie (\bowtie_{k \in ne(i)} h^*_k)) \] (1)

Iteration 4

\[ h^2_A \]
\[ h^2_B \]
\[ h^2_C \]
\[ h^2_D \]
\[ h^2_E \]
\[ h^2_F \]
\[ R_1 \]
\[ R_2 \]
\[ R_3 \]
\[ R_4 \]
\[ R_5 \]
\[ R_6 \]

\[ h^*_A \]
\[ h^*_B \]
\[ h^*_C \]
\[ h^*_D \]
\[ h^*_E \]
\[ h^*_F \]
\[ R_i \leftarrow R_i \cap \bigcap_{h \in \text{ne}(i)} h^i \] (2)

**Iteration 4**
\[ h^i_b \leftarrow \pi_{I_y}(R_i \bowtie (\bowtie_{k \in \text{ne}(i)} h^i_b)) \] (1)

**Iteration 5**

\[
\begin{array}{c|c|c|c|c|c}
R_1 & h^1_2 & h^1_3 & h^1_4 \\
A & 1 & 1 & 1 & 1 \\
\end{array}
\]

\[
\begin{array}{c|c|c|c}
R_2 & h^2_1 & h^2_2 \\
A & 1 & A B \\
\end{array}
\]

\[
\begin{array}{c|c|c}
R_3 & h^3_1 & h^3_5 \\
A C & 1 & 3 \\
A & 2 & 2 \\
C & 1 & 2 \\
\end{array}
\]

\[
\begin{array}{c|c|c|c|c|c}
R_4 & h^4_1 & h^4_2 & h^4_3 \\
A B D & 1 & 3 & 2 \\
\end{array}
\]

\[
\begin{array}{c|c|c|c|c|c|c}
R_5 & h^5_2 & h^5_3 & h^5_6 \\
B C F & 3 & 2 & 1 \\
B & 2 & 3 \\
C & 2 & 1 \\
F & 1 & 3 \\
\end{array}
\]

\[
\begin{array}{c|c|c|c|c|c|c}
R_6 & h^6_4 & h^6_5 & h^6_6 \\
D F G & 2 & 1 & 3 \\
D & 2 & 1 \\
F & 1 & 3 \\
\end{array}
\]
Iteration 5

\[ R_i \leftarrow R_i \cap \left( \bigcap_{h \in \text{ne}(i)} h^i \right) \] (2)