DM826 – Spring 2012

Modeling and Solving Constrained Optimization Problems

# Lecture 6
# Constraint Propagation and Local Consistency

Marco Chiarandini

Department of Mathematics & Computer Science
University of Southern Denmark

# Resume

- Definitions
  (CSP, restrictions, projections, istantiation, local consistency, global consistency)
- Tigthtenings
- Global consistent (any instantiation local consistent can be extended to a solution) needs exponential time
  $\rightsquigarrow$ local consistency defined by conditions $\Phi$ on instantiations
- Tightenings by constraint propagation: reduction rules + rules iterations
  - reduction rules $\Leftrightarrow \Phi$
  - rules iteration: reach fixed point, that is, closure of all tightenings that are $\Phi$ consistent
- Domain-based $\Phi$: (generalized) arc consistency

An arbitrary CSP can be polynomially converted in an equivalent binary CSP.
Proof as exercise

Filtering algorithms have focused on binary. However recently focus on efficiency issues and non-binary constraints as well.

# Outline

1. Algorithms to enforce consistency

2. Higher Order Consistencies

# AC1 – Reduction rule

Revision: making a constraint arc consistent by propagating the domain from a variable to anohter

general

$$D(x_i) \leftarrow D(x_i) \cap \pi_{\{x_i\}}(C \cap \pi_{X(C)}(\mathcal{DE}))$$

binary

$$D(x_i) \leftarrow D(x_i) \cap \pi_{\{x_i\}}(\text{join}(C, D(x_i)))$$

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**

Complexity: $O(d^2)$ or $O(rd^r)$

$d$ values, $r$ arity

# AC1 – Rules Iteration

## Binary case

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 $\mathcal{R}$

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

- Complexity (Mackworth and Freuder, 1986): $O(end^3)$
  $e$ number of arcs, $n$ variables
  ($ed^2$ each loop, $nd$ number of loops)
- best-case = $O(ed)$
- Arc-consistency is at least $O(ed^2)$ in the worst case

# AC3 (Macworth, 1977)

## General case

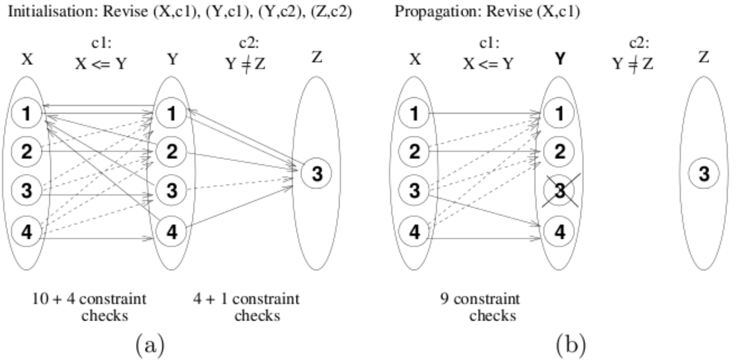
**function** Revise3(**in** $x_i$: variable; $c$: constraint): **Boolean** ;
**begin**
CHANGE $\leftarrow$ **false**;
**foreach** $v_i \in D(x_i)$ **do**
**if** $\nexists \tau \in c \cap \pi_{X(c)}(D)$ *with* $\tau[x_i] = v_i$ **then**
remove $v_i$ from $D(x_i)$;
CHANGE $\leftarrow$ **true**;
return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

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

$O(er^3 d^{r+1})$ time
$O(er)$ space

# AC3
## Example

$$\mathcal{P} = \langle X = (x, y, z),\ \mathcal{DE} = \{D(x) = D(y) = \{1, 2, 3, 4\}, D(z) = \{3\}\},$$
$$\mathcal{C} = \{C_1 \equiv x \leq y, C_2 \equiv y \neq z\}\}\rangle$$



# AC4

## Binary case

**function** AC4(**in** $X$: set): **Boolean** ;<br>
  **begin**<br>
    /* initialization */;<br>
    $Q \leftarrow \emptyset$; $S[x_j, v_j] = 0, \forall v_j \in D(x_j), \forall x_j \in X$;<br>
    **foreach** $x_i \in X, c_{ij} \in C, v_i \in D(x_i)$ **do**<br>
      initialize counter$[x_i, v_i, x_j]$ to $|\{v_j \in D(x_j) \mid (v_i, v_j) \in c_{ij}\}|$;<br>
      **if** *counter*$[x_i, v_i, x_j] = 0$ **then** remove $v_i$ from $D(x_i)$ and add $(x_i, v_i)$ to $Q$;<br>
      add $(x_i, v_i)$ to each $S[x_j, v_j]$ s.t. $(v_i, v_j) \in c_{ij}$;<br>
      **if** $D(x_i) = \emptyset$ **then** return **false** ;<br>
    /* propagation */;<br>
    **while** $Q \neq \emptyset$ **do**<br>
      select and remove $(x_j, v_j)$ from $Q$;<br>
      **foreach** $(x_i, v_i) \in S[x_j, v_j]$ **do**<br>
        **if** $v_i \in D(x_i)$ **then**<br>
          counter$[x_i, v_i, x_j]$ = counter$[x_i, v_i, x_j] - 1$;<br>
          **if** *counter*$[x_i, v_i, x_j] = 0$ **then**<br>
            remove $v_i$ from $D(x_i)$; add $(x_i, v_i)$ to $Q$;<br>
            **if** $D(x_i) = \emptyset$ **then** return **false** ;<br>
    return **true** ;<br>
  **end**

# AC4
## Example

$$\mathcal{P} = \langle X = (x, y, z),\ \mathcal{DE} = \{D(x) = D(y) = \{1, 2, 3, 4\}, D(z) = \{3\}\},$$
$$\mathcal{C} = \{C_1 \equiv x \leq y, C_2 \equiv y \neq z\}\}\rangle$$

$$\begin{array}{lll}
\texttt{counter}[x,1,y] = 4 & \texttt{counter}[y,1,x] = 1 & \texttt{counter}[y,1,z] = 1 \\
\texttt{counter}[x,2,y] = 3 & \texttt{counter}[y,2,x] = 2 & \texttt{counter}[y,2,z] = 1 \\
\texttt{counter}[x,3,y] = 2 & \texttt{counter}[y,3,x] = 3 & \texttt{counter}[y,3,z] = 0 \\
\texttt{counter}[x,4,y] = 1 & \texttt{counter}[y,4,x] = 4 & \texttt{counter}[y,4,z] = 1 \\
 & & \texttt{counter}[z,3,y] = 3
\end{array}$$

$$\begin{array}{ll}
S[x,1] = \{(y,1),(y,2),(y,3),(y,4)\} & S[y,1] = \{(x,1),(z,3)\} \\
S[x,2] = \{(y,2),(y,3),(y,4)\} & S[y,2] = \{(x,1),(x,2),(z,3)\} \\
S[x,3] = \{(y,3),(y,4)\} & S[y,3] = \{(x,1),(x,2),(x,3)\} \\
S[x,4] = \{(y,4)\} & S[y,4] = \{(x,1),(x,2),(x,3),(x,4),(z,3)\} \\
 & S[z,3] = \{(y,1),(y,2),(y,4)\}
\end{array}$$

# AC6
## Binary case



**function** AC6(**in** $X$: set)**: Boolean** ;<br>
**begin**<br>
/* initialization */;<br>
$Q \leftarrow \emptyset$; $S[x_j, v_j] = 0, \forall v_j \in D(x_j), \forall x_j \in X$;<br>
**foreach** $x_i \in X, c_{ij} \in C, v_i \in D(x_i)$ **do**<br>
$v_j \leftarrow$ smallest value in $D(x_j)$ s.t. $(v_i, v_j) \in c_{ij}$;<br>
**if** $v_j$ *exists* **then** add $(x_i, v_i)$ to $S[x_j, v_j]$;<br>
**else** remove $v_i$ from $D(x_i)$ and add $(x_i, v_i)$ to $Q$;<br>
**if** $D(x_i) = \emptyset$ **then** return **false** ;<br>
/* propagation */;<br>
**while** $Q \neq \emptyset$ **do**<br>
select and remove $(x_j, v_j)$ from $Q$;<br>
**foreach** $(x_i, v_i) \in S[x_j, v_j]$ **do**<br>
**if** $v_i \in D(x_i)$ **then**<br>
$v'_j \leftarrow$ smallest value in $D(x_j)$ greater than $v_j$ s.t. $(v_i, v_j) \in c_{ij}$;<br>
**if** $v'_j$ *exists* **then** add $(x_i, v_i)$ to $S[x_j, v'_j]$;<br>
**else**<br>
remove $v_i$ from $D(x_i)$; add $(x_i, v_i)$ to $Q$;<br>
**if** $D(x_i) = \emptyset$ **then** return **false** ;<br>
return **true** ;<br>
**end**

# AC6

## Example

$$\mathcal{P} = \langle X = (x, y, z),\ \mathcal{DE} = \{D(x) = D(y) = \{1, 2, 3, 4\}, D(z) = \{3\}\},$$
$$\mathcal{C} = \{C_1 \equiv x \leq y, C_2 \equiv y \neq z\}\}\rangle$$

$$S[x,1] = \{(y,1),(y,2),(y,3),(y,4)\}$$
$$S[x,2] = \{\}$$
$$S[x,3] = \{\}$$
$$S[x,4] = \{\}$$

$$S[y,1] = \{(x,1),(z,3)\}$$
$$S[y,2] = \{(x,2)\}$$
$$S[y,3] = \{(x,3)\}$$
$$S[y,4] = \{(x,4)\}$$
$$S[z,3] = \{(y,1),(y,2),(y,4)\}$$

# Reverse2001

## Binary case

**function** `Revise2001`**(in** $x_i$: variable; $c_{ij}$: constraint**): Boolean** ;
**begin**
    CHANGE $\leftarrow$ **false**;
    **foreach** $v_i \in D(x_i)$ *s.t.* *Last*$(x_i, v_i, x_j) \notin D(x_j)$ **do**
        $v_j \leftarrow$ smallest value in $D(x_j)$ greater than `Last`$(x_i, v_i, x_j)$ s.t. $(v_i, v_j) \in c_{ij}$;
        **if** $v_j$ *exists* **then** `Last`$(x_i, v_i, x_j) \leftarrow v_j$;
        **else**
            remove $v_i$ from $D(x_i)$;
            CHANGE $\leftarrow$ **true**;
    return CHANGE ;
**end**

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

# Reverse2001
## Example



$$\mathcal{P} = \langle X = (x, y, z),\ \mathcal{DE} = \{D(x) = D(y) = \{1,2,3,4\}, D(z) = \{3\}\},$$
$$\mathcal{C} = \{C_1 \equiv x \leq y, C_2 \equiv y \neq z\}\}\rangle$$

$$\begin{array}{lll}
\texttt{Last}[x,1,y] = 1 & \texttt{Last}[y,1,x] = 1 & \texttt{Last}[y,1,z] = 3 \\
\texttt{Last}[x,2,y] = 2 & \texttt{Last}[y,2,x] = 1 & \texttt{Last}[y,2,z] = 3 \\
\texttt{Last}[x,3,y] = 3 & \texttt{Last}[y,3,x] = 1 & \texttt{Last}[y,3,z] = nil \\
\texttt{Last}[x,4,y] = 4 & \texttt{Last}[y,4,x] = 1 & \texttt{Last}[y,4,z] = 3 \\
 & & \texttt{Last}[z,3,y] = 1
\end{array}$$

# Limitation of arc consistency

**Example**

$$\langle x < y, y < z, z < x; x, y, z \in \{1..100000\}\rangle$$

is inconsistent. Proof: Apply revise to $(x, x < y)$

$$\langle x < y, y < z, z < x; x \in \{1..99999\}, y, z \in \{1..100000\}\rangle,$$

ecc. we end in a fail.

- Disadvantage: large number of steps.
  Run time depends on the size of the domains!
- Note: we could proof fail by transitivity of $<$.
  $\leadsto$ Path consitency involves two constraints together

# Outline

# Path consistency

Given $\mathcal{P} = \langle X, \mathcal{DE}, \mathcal{C} \rangle$ normalized and $x_i, x_j$:

- the pair $(v_i, v_j) \in D(x_i) \times D(x_j)$ is $p$-path consistent iff forall $Y = (x_i = x_{k_1}, \ldots, x_{k_p} = x_j)$ with $C_{k_q, k_{q+1}} \in \mathcal{C}$ $\exists \tau : \tau[Y] = (v_i = v_{k_1}, \ldots, v_{k_{q+1}} = v_j) \in \pi_Y(\mathcal{DE})$ and $(v_{k_q}, v_{k_{q+1}}) \in C_{k_p, k_{q+1}}$, $q = 1, \ldots, p$
- the CSP $\mathcal{P}$ is $p$-path consistent iff for any $(x_i, x_j), i \neq j$ any locally consistent pair of values is path consistent.

Example

$$\mathcal{P} = \langle X = (x, x_2, x_3), D(x_i) = \{1, 2\}, \mathcal{C} \equiv \{x_1 \neq x_2, x_2 \neq x_3\} \rangle$$

Not path consistent: e.g., $(x_1, 1), (x_3, 2)$
$\mathcal{P} = \langle X, \mathcal{DE}, \mathcal{C} \cup \{x_1 = x_3\} \rangle$ is path consistent

2-path consistency if the path has length 2

# References

Bessiere C. (2006). **Constraint propagation**. In *Handbook of Constraint Programming*, edited by F. Rossi, P. van Beek, and T. Walsh, chap. 3. Elsevier. Also as Technical Report LIRMM 06020, March 2006.