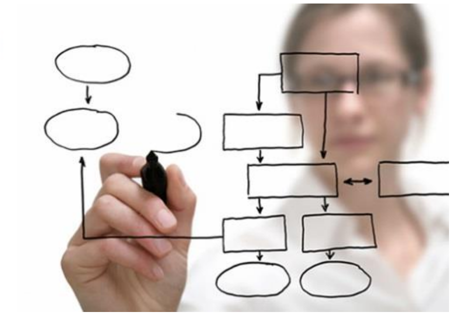
CORK INSTITUTE OF TECHNOLOGY
INSTITIÚID TEICNEOLAÍOCHTA CHORCAÍ

Computing@CIT

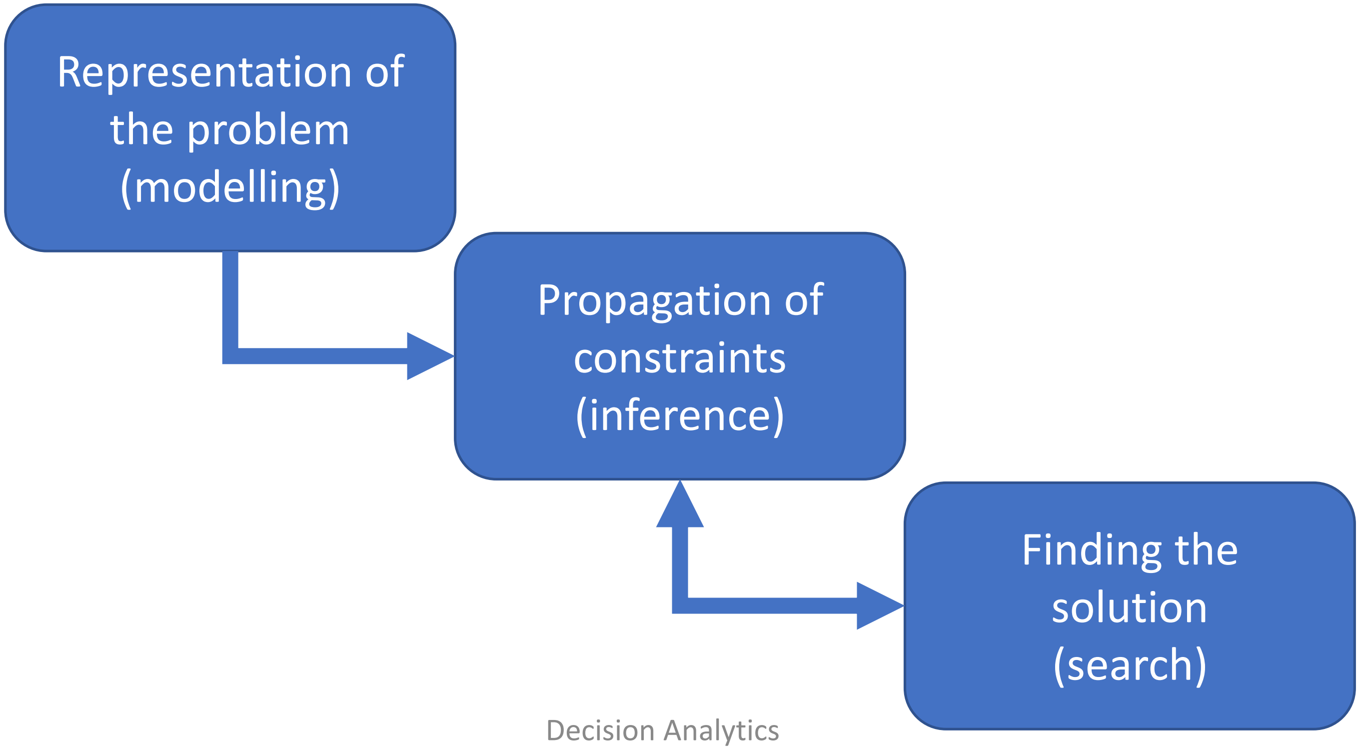


Decision Analytics

Lecture 13: Domain and Arc consistency

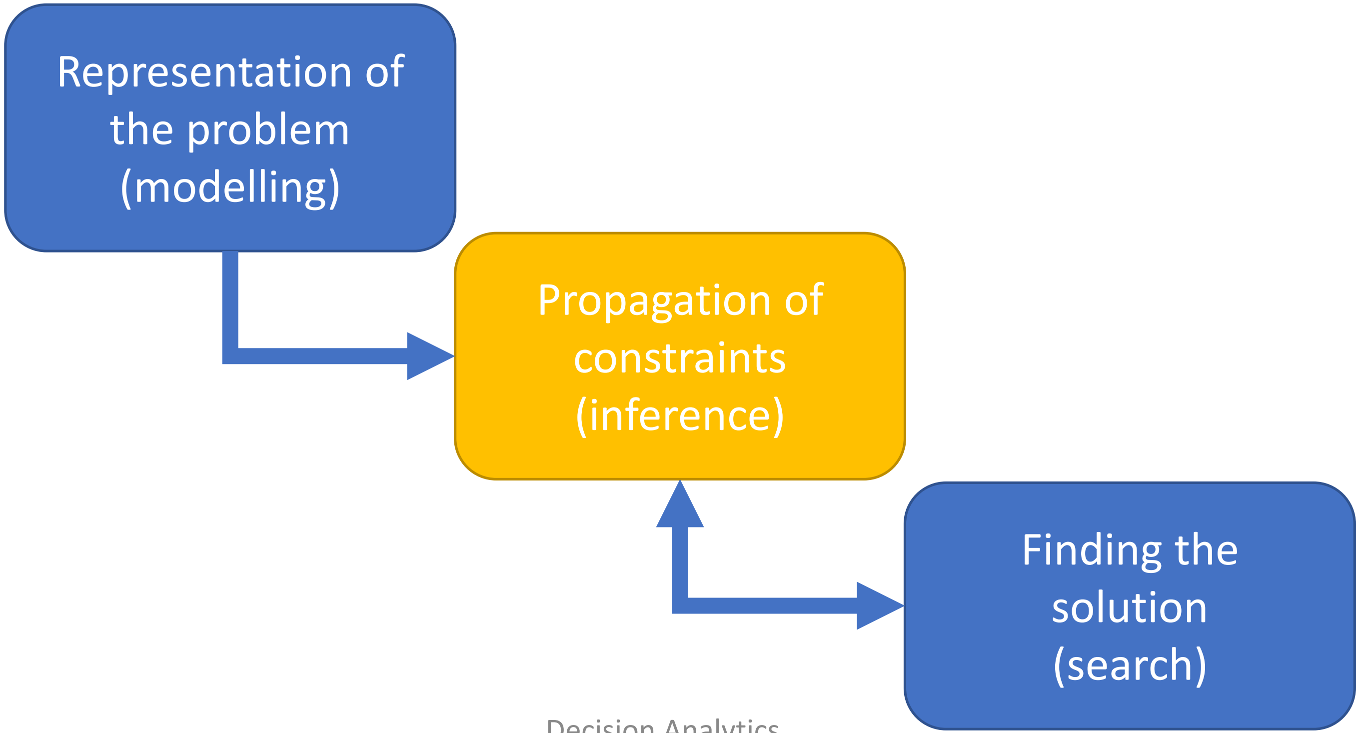
Constraint Programming

- Constraint Programming (CP) is a paradigm for solving combinatorial constraint satisfaction and constrained optimisation problems using a combination of modelling, propagation, and search



Constraint Programming

- Constraint Programming (CP) is a paradigm for solving combinatorial constraint satisfaction and constrained optimisation problems using a combination of modelling, propagation, and search
- This lecture is about **constraint propagation**



Constraint network

- A constraint network (X, D, C) is defined by
  - A sequence of n **variables**
$$X = (x_1, \dots, x_n)$$
  - A **domain** for X defined by the domains of the individual variables
$$D = D(x_1) \times \cdots \times D(x_n)$$
  - A set of **constraints**
$$C = \{c_1, \dots, c_e\}$$
- A network is **normalised** if two different constraints do not contain exactly the same variables, i.e. $c_i \neq c_j \Rightarrow X(c_i) \neq X(c_j)$

Tightening of a network

- The goal of constraint propagation is to transform a network to reduce the number of solution possibilities that need to be searched

- The **tightening** $\mathcal{P}_N$ of a network $N = (X, D, C)$ is defined as the set of networks $N' = (X, D', C')$ that are more restricted both in terms of domain as well as constraints, i.e. $D' \subset D$ and $\forall c \in C \exists c' \in C': X(c') = X(c) \wedge c' \subset c$
- We denote the subset of **solution preserving tightenings**, i.e. those where $sol(N') = sol(N)$, with $\mathcal{P}_N^{sol}$

- A network $G_N \in \mathcal{P}_N^{sol}$ that is the "smallest" with respect to $\preccurlyeq$ in the set can be shown to be globally consistent, i.e. all locally consistent instantiations can be extended to a solution
- Finding a network "close" to G_N is the goal of constraint propagation

Domain based tightening

- If we keep the same constraints, i.e. only shrink the domains, then we can define **domain-based tightenings** $\mathcal{P}_{ND}$ of a network $N = (X, D, C)$ as the set of networks $N' = (X, D', C)$ with $D' \subset D$

- Again, we are more interested in the subset of **solution preserving domain-based tightenings**, i.e. those where $sol(N') = sol(N)$, which we denote with $\mathcal{P}_{ND}^{sol}$

- The network $G_{ND} = (X, D_G, C)$ that is the smallest element in $\mathcal{P}_{ND}^{sol}$ with respect to $\preccurlyeq$ is NP-hard to find
- All domain-based constraint propagation procedures are trying to approximate G_{ND} with the domain D_G being a lower bound to the approximation

$$D_G \preccurlyeq \text{N'} \preccurlyeq N$$

Domain based tightening

- Example: Let $N = (X, D, C)$ be defined as

$$X = (x_1, x_2, x_3)$$
$$D(x_i) = \{1,2,3\}$$
$$C = \{x_1 < x_2, x_2 < x_3\}$$

- The network $N' = (X, D', C)$ with

$$D'(x_1) = \{1\}$$
$$D'(x_2) = \{2,3\}$$
$$D'(x_3) = \{2,3\}$$

  is a solution-preserving domain based tightening of N

- The network $N'' = (X, D'', C)$ with

$$D'(x_1) = \{1\}$$
$$D'(x_2) = \{2\}$$
$$D'(x_3) = \{3\}$$

  is a the smallest solution-preserving domain based tightening of N (and at the same time it makes the solution obvious)

Node consistency

- Every unary constraint, i.e. every constraint on only one variable $X(c) = (x_i)$, can be applied directly to the domain

$$D'(x_i) = D(x_i) \cap c$$

- If all domains are tightened so that the no unary constraints reduce domain sizes any further this is called **node consistency**

- Because achieving node consistency is trivial, we can usually assume that $|X(c)| \geq 2$ after all unary constraints have been absorbed into the domains

Node consistency

- Example: The network $N = (X, D, C)$ with

$$X = (x_1, x_2)$$
$$D(x_1) = D(x_2) = \{1,2,3\}$$
$$C = \{x_1 > 1, x_2 < x_1\}$$

- Can be made node consistent by tightening the domain

$$D'(x_1) = \{2,3\}$$

- The constraint $x_1 > 1$ has become redundant and can be removed from the network without changing the solution

$$C' = \{x_2 < x_1\}$$

Arc consistency

- While node consistency only looked at unary constraints, arc consistency ensures that every domain is compatible with every constraint

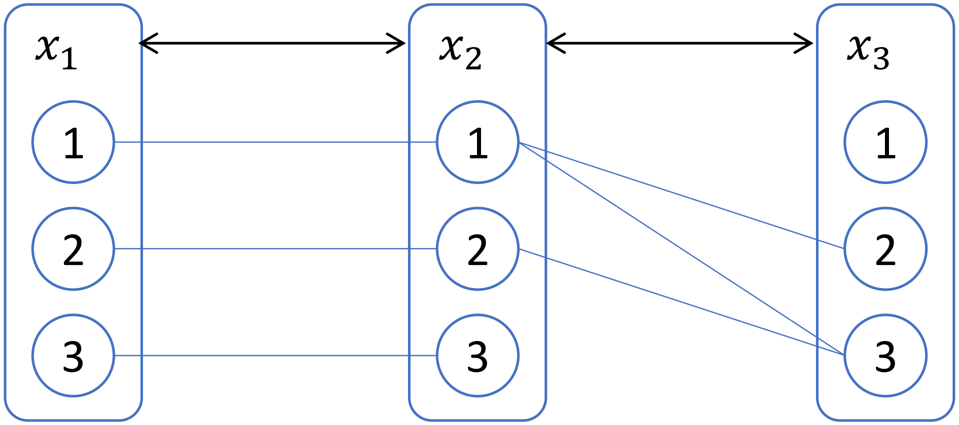
- A network is $N = (X, D, C)$ is **arc consistent** if all for all variable domains and all constraints

$$D(x_i) \subset \pi_{\{\ x_i\}}(c \cap \pi_{X(c)}(D))$$

Arc consistency

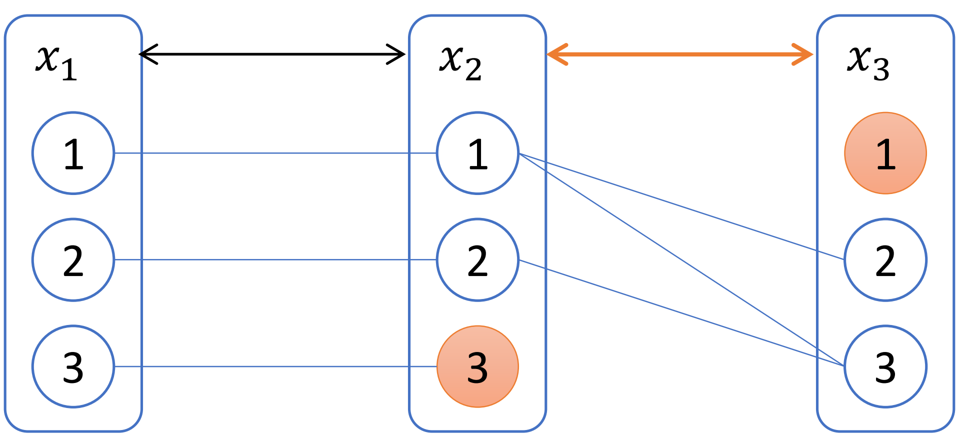
- Example: The network $N = (X, D, C)$ with

$$X = (x_1, x_2, x_3)$$
$$D(x_1) = D(x_2) = D(x_3) = \{1,2,3\}$$
$$C = \{x_1 = x_2, x_2 < x_3\}$$



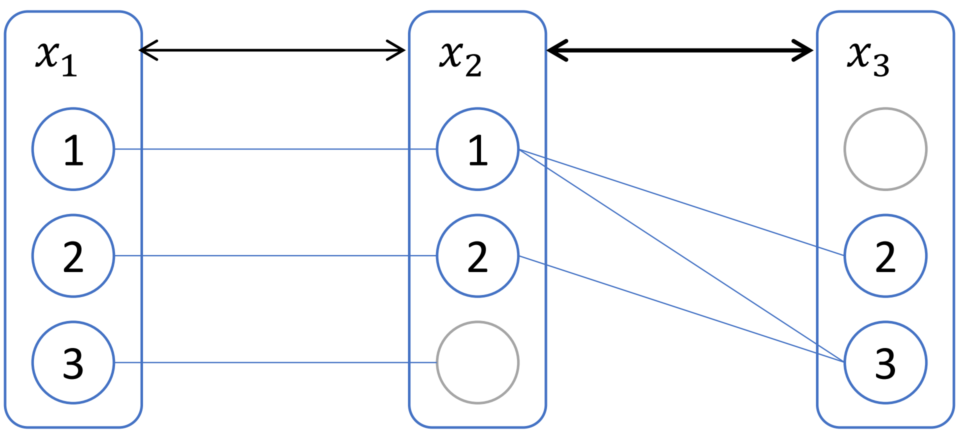
Arc consistency

- Example: The network $N = (X, D, C)$ with
$$X = (x_1, x_2, x_3)$$
$$D(x_1) = D(x_2) = D(x_3) = \{1,2,3\}$$
$$C = \{x_1 = x_2, x_2 < x_3\}$$
- We start be removing values from $D(x_2)$ and $D(x_3)$ that are locally inconsistent with $x_2 < x_3$



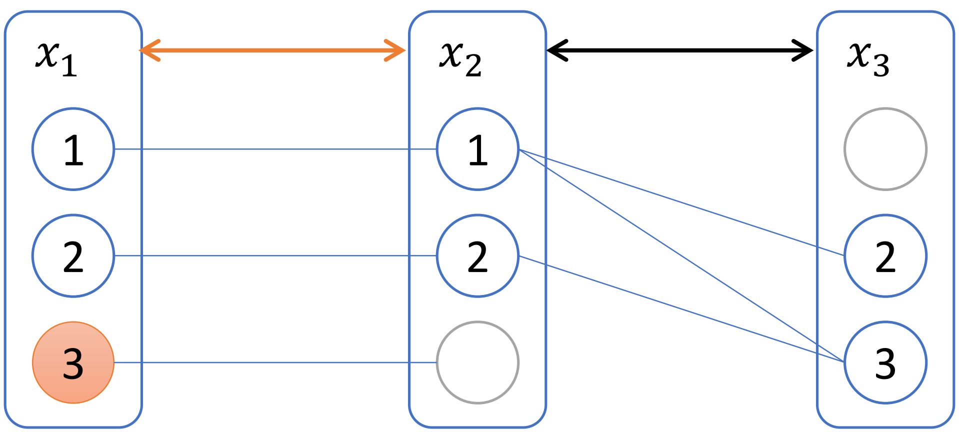
Arc consistency

- Example: The network $N = (X, D, C)$ with
$$X = (x_1, x_2, x_3)$$
$$D(x_1) = D(x_2) = D(x_3) = \{1,2,3\}$$
$$C = \{x_1 = x_2, x_2 < x_3\}$$
- We start be removing values from $D(x_2)$ and $D(x_3)$ that are locally inconsistent with $x_2 < x_3$



Arc consistency

- Example: The network $N = (X, D, C)$ with
$$X = (x_1, x_2, x_3)$$
$$D(x_1) = D(x_2) = D(x_3) = \{1,2,3\}$$
$$C = \{x_1 = x_2, x_2 < x_3\}$$
- Next, we remove values from $D(x_1)$ that are locally inconsistent with $x_1 = x_2$



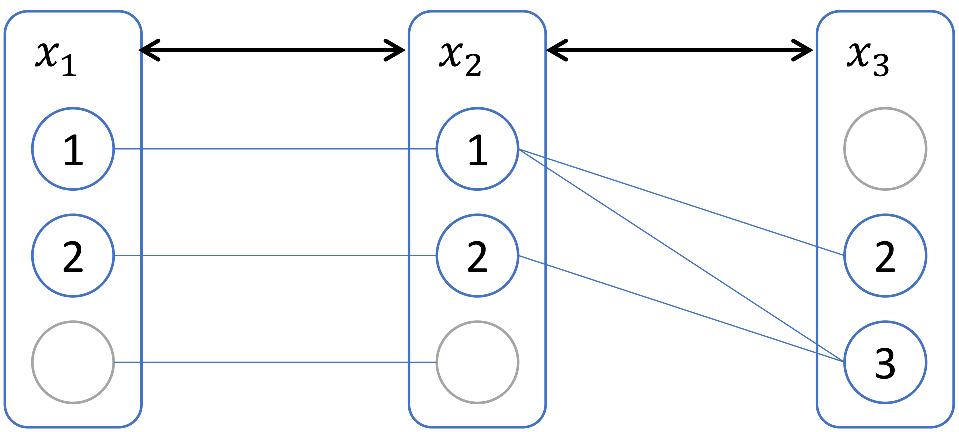
Arc consistency

- Example: The network $N = (X, D, C)$ with

$$X = (x_1, x_2, x_3)$$
$$D(x_1) = D(x_2) = D(x_3) = \{1,2,3\}$$
$$C = \{x_1 = x_2, x_2 < x_3\}$$

- The network is arc consistent now by tightening the domains to

$$D'(x_1) = \{1,2\}$$
$$D'(x_2) = \{1,2\}$$
$$D'(x_3) = \{2,3\}$$



AC3 Algorithm

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

Thank you for your attention!