CORK INSTITUTE OF TECHNOLOGY
INSTITIÚID TEICNEOLAÍOCHTA CHORCAÍ

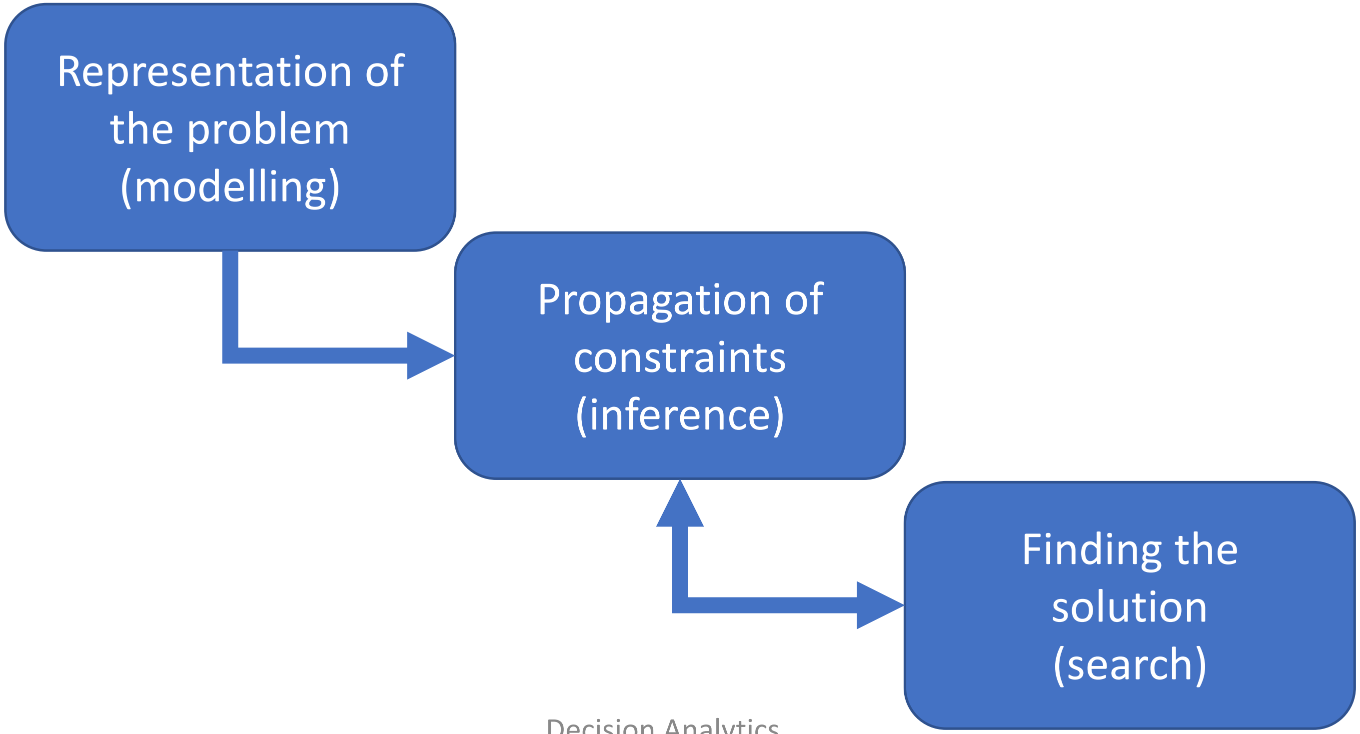
Computing@CIT

# Decision Analytics

## Lecture 12: Constraint networks

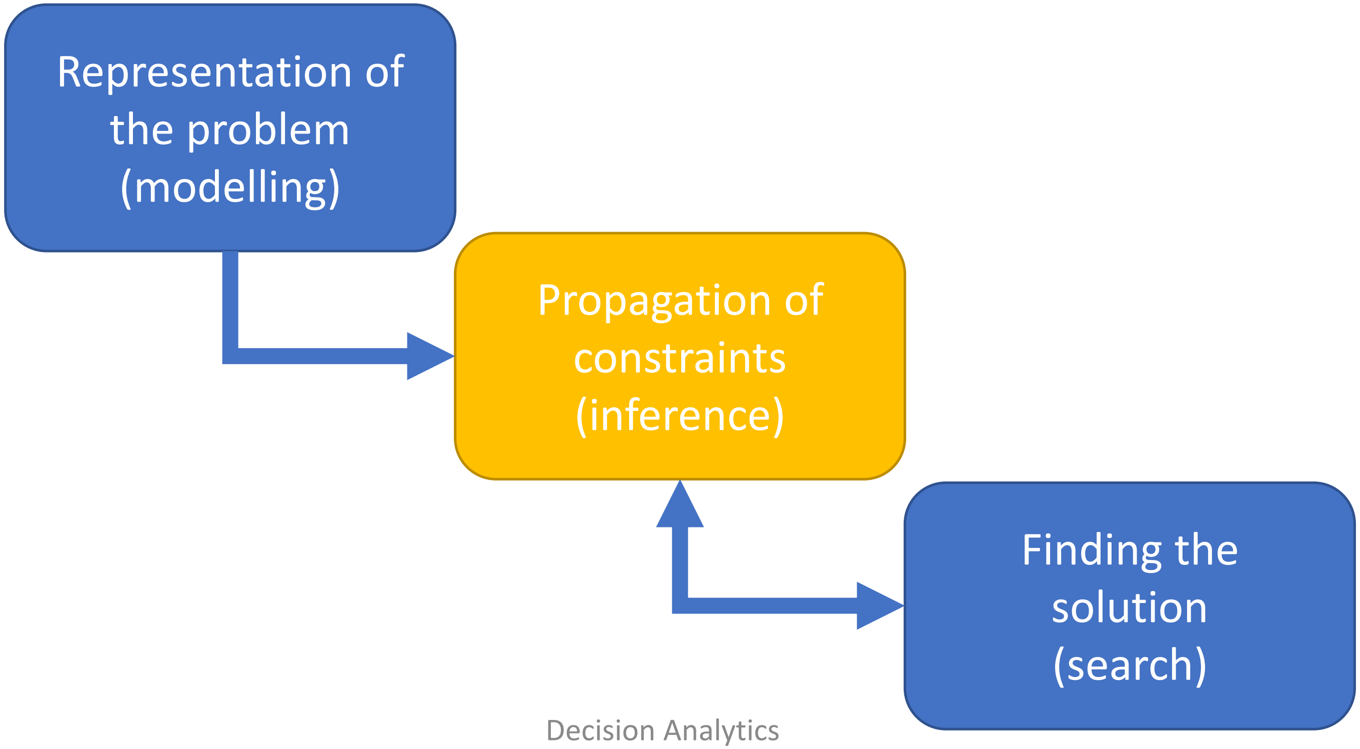
# Constraint Programming

- 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 Propagation

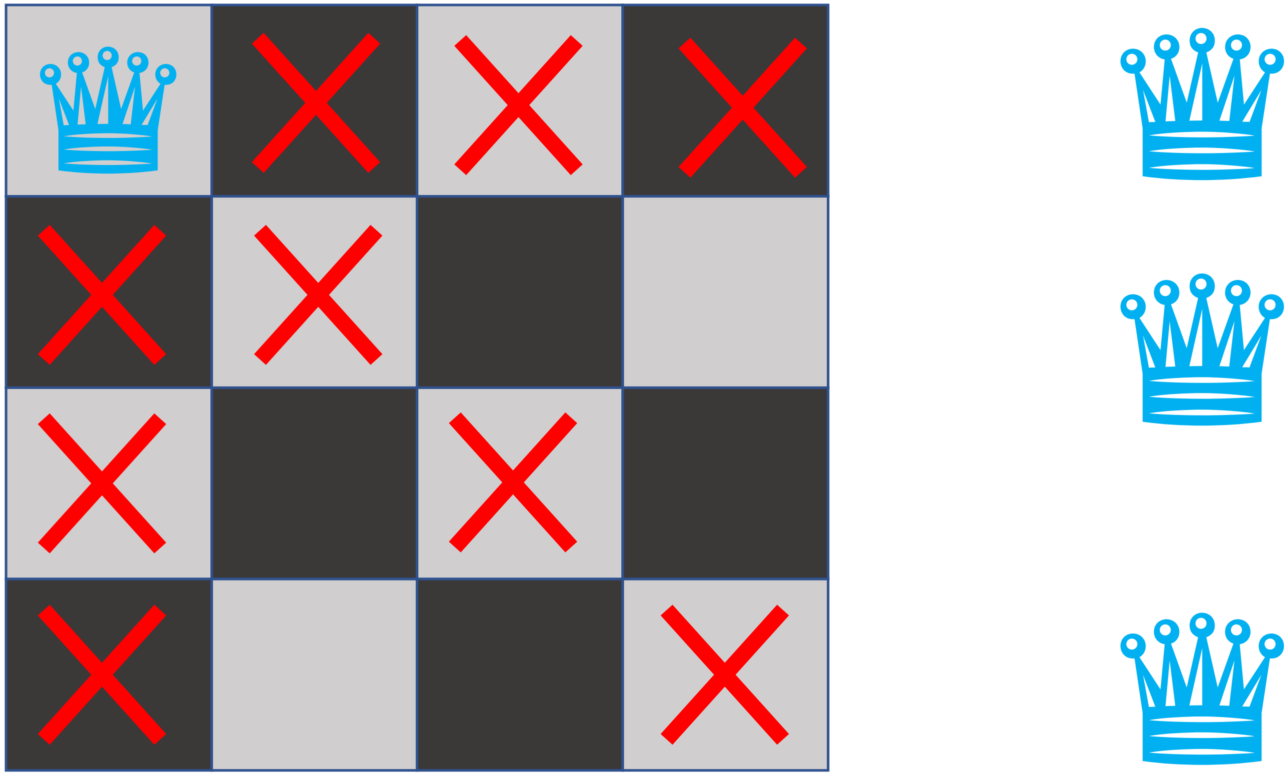
- The purpose of constraint propagation is to disallow assignments of variables or combinations of variables because subsets of constraints cannot be satisfied otherwise

- While search is inevitable for most problems, constraint propagation can be used to at least reduce the size of the search space

- There are two ways to characterise constraint propagation:
  - By describing **local consistencies**, that need to be fulfilled after each iteration of constraint propagation
  - By describing **rules iteration**, i.e. the process of propagation itself

- We will look at both concepts

# Constraint propagation for the 4-queens



# Constraint

- A constraint $c$ is a relation defined on a sequence of variables

$$X(c) = (x_{i_1}, \dots, x_{i_k})$$

- The sequence of variables is called the **scheme** of the constraint $c$
- The number of variables $k$ is called the **arity** of the constraint $c$
- Testing if a **tuple** $\tau \in \mathbb{Z}^k$ satisfies the constraint $c$ is called a **constraint check**

# Constraint

- For example, let the variables be $X = (x_1, x_2, x_3, x_4)$

- The constraint $c_1 \equiv x_2 < x_3$
  - The scheme of $c_1$ is $X(c_1) = (x_2, x_3)$
  - The arity of $c$ is $|X(c_1)| = 2$
  - The tuple $(1,2)$ satisfies $c_1$
  - The tuple $(2,2)$ does not satisfy $c_1$

- Constraints can also be defined by listing all valid tuples
- For instance we can define a constraint $c_2$ on $X(c_2) = (x_1, x_3, x_4)$ as

$$c_2 = \{((1,1,1), (2,2,2), (1,2,3))\}$$

# Constraint

- For a subset of variables $W \subset X(c)$ we denote the **projection** of $c$ on $W$ as $\pi_W(c)$ being the constraint with variables W that can be extended to tuples satisfying $c$

- We denote the **intersection** of constraints $c_1 \cap c_2$ being the relation with schema $X(c_1 \cap c_2) = X(c_1) = X(c_2)$ that contains the tuples present in both $c_1$ and $c_2$
- Similar we denote the **union** of constraints as $c_1 \cup \mathrm{c}_2$ being the relation containing tuples present in either $c_1$ or $c_2$

- The **join** of two constraints $c_1 \bowtie \mathrm{c}_2$ is the relation on the union of variables $X(c_1) \cup X(c_2)$ that contains the tuples $\tau$ that are consistent with both constraints, i.e. $\tau\big(X(c_1)\big) \in c_1$ and $\tau\big(X(c_2)\big) \in c_2$

# Constraint

- For example, let the variables be $X = (x_1, x_2, x_3)$ and the constraint $c$ defined on $X(c) = (x_1, x_2, x_3)$ be

$$c = \{(1,1,1), (1,1,2), (1,3,1)\}$$

- Then some projections of $c$ are

$$\pi_{\{x_1\}} = \{(1)\}$$

$$\pi_{\{x_2\}} = \{(1), (3)\}$$

$$\pi_{\{x_1,x_3\}} = \{(1,1), (1,2)\}$$

# Constraint

- For example, let the variables be $X = (x_1, x_2, x_3)$ and the constraints defined on $X(c) = (x_1, x_2, x_3)$ be

$$c_1 = \{(1,1,1), (1,1,2), (1,3,1)\}$$
$$c_2 = \{(1,1,1), (1,2,3)\}$$

- Then the union of $c_1$ and $c_2$ is

$$c_1 \cup c_2 = \{(1,1,1), (1,1,2), (1,3,1), (1,2,3)\}$$

- the intersection of $c_1$ and $c_2$ is

$$c_1 \cap c_2 = \{(1,1,1)\}$$

# Constraint

- For example, let the variables be $X = (x_1, x_2, x_3, x_4)$ and the constraints defined on $X(c_1) = (x_1, x_2, x_3)$ and $X(c_1) = (x_2, x_3, x_4)$ be

$$c_1 = \{(3,1,1), (4,2,2), (5,3,3)\}$$
$$c_2 = \{(2,2,6), (3,3,7), (4,4,8)\}$$

- Then the join of $c_1$ and $c_2$ is

$$c_1 \bowtie c_2 = \{(4,2,2,6), (5,3,3,7)\}$$

# 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. $c_i \neq c_j \Rightarrow X(c_i) \neq X(c_j)$

# Constraint network

- The constraint network can be seen as a hypergraph

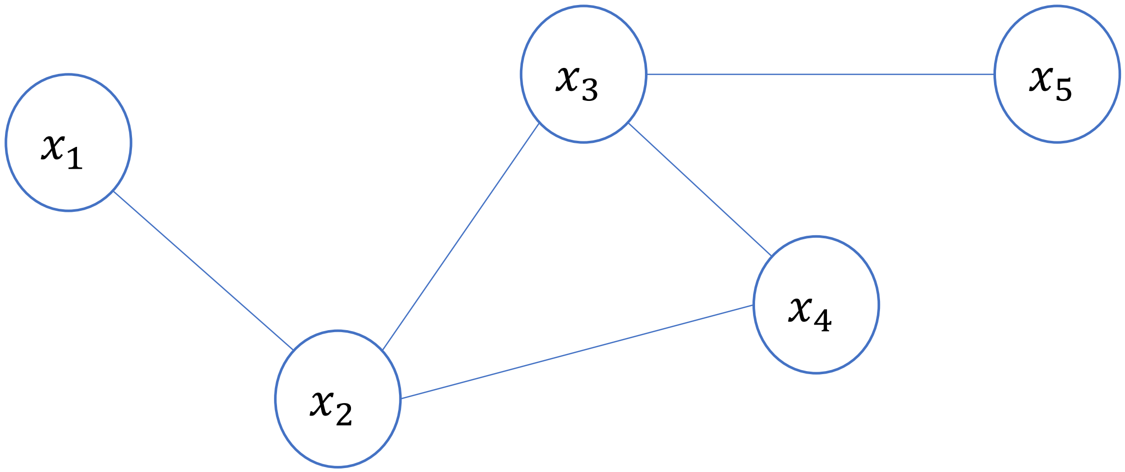
- The vertices of the constraint network graph are then
$$X = (x_1, \dots, x_n)$$
- The edges of the constraint network graph are defined by the schemes of the constraints
$$X(c_1), \dots, X(c_e)$$

- It is a graph, if all constraints have arity 2, i.e. $|X(c_i)| = 2$
- In this case it is called a **binary constraint network**

# Constraint network

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

$$X = (x_1, x_2, x_3, x_4, x_5)$$
$$D(x_i) = \{1,2,3,4,5\}$$
$$C = \{x_1 < x_2, x_2 = x_3, x_3 \geq x_4, x_3 \geq x_5, x_2 \leq x_4\}$$



# Instantiation

- An instantiation of a network $N = (X, D, C)$ is an assignment of values to a subset of variables
$$Y = (x_1, \dots, x_k) \subset X$$
- Denoted as tuple
$$I = \big((x_1, v_1), \dots, (x_k, v_k)\big)$$

- Note, that the assignment is only for a subset of the variables of the whole network
- This is to facilitate backtracking search, which assigns variables one-by-one until all variables are assigned

# Instantiation

- An instantiation $I$ on $Y$ is **valid** if all assigned values are in the respective domains of the network, i.e. $\forall x_i \in Y: I[x_i] \in D(x_i)$

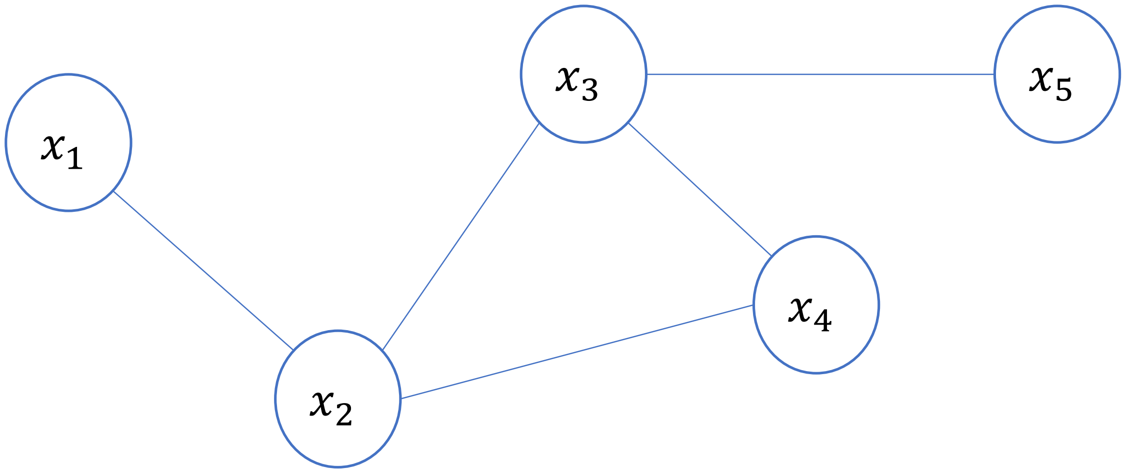
- An instantiation $I$ on $Y$ is **locally consistent** if it is valid and satisfies all constraints that are defined on the subset $Y$**,** $X(c) \subset Y \Rightarrow I[X(c)] \in c$

- A locally consistent instantiation $I$ on all variables of the network $X$ is a **solution** of the network, denoted as $sol(N)$

- An instantiation $I$ on a subset of variables $Y$ that can be extended to a solution on all variables, i.e. $\exists s \in sol(N): I = s[Y]$, is called **globally consistent**

# Constraint network

- Example: Let $N = (X, D, C)$ be defined as
  $X = (x_1, x_2, x_3, x_4, x_5)$
  $D(x_i) = \{1,2,3,4,5\}$
  $C = \{x_1 < x_2, x_2 = x_3, x_3 \geq x_4, x_3 \geq x_5, x_2 \leq x_4\}$
- The instantiation $I = \{(x_1, 1), (x_2, 2)\}$ is globally consistent
- The instantiation $I = \{(x_1, 1), (x_3, 1)\}$ is locally consistent, but not globally consistent
- The instantiation $I = \{(x_2, 1), (x_3, 2)\}$ is locally inconsistent



# No-good instantiations

- When searching for a solution we need to exclude search paths based on partial instantiations

- We can define a quasi-order on networks $N' \preccurlyeq N$ which holds when every instantiation of $I$ on a subset $Y$ that is locally inconsistent in $N$ is locally inconsistent in $N$' as well
- In this case, if we identify a **no-good instantiation** for $N$, i.e. one that will not lead to a solution, we do not need to search for a solution in $N$' anymore
- If the relation holds both ways $\mathrm{N} \preccurlyeq N' \preccurlyeq N$ we call the two networks **no-good equivalent**

Thank you for your attention!