# Constraint Programming

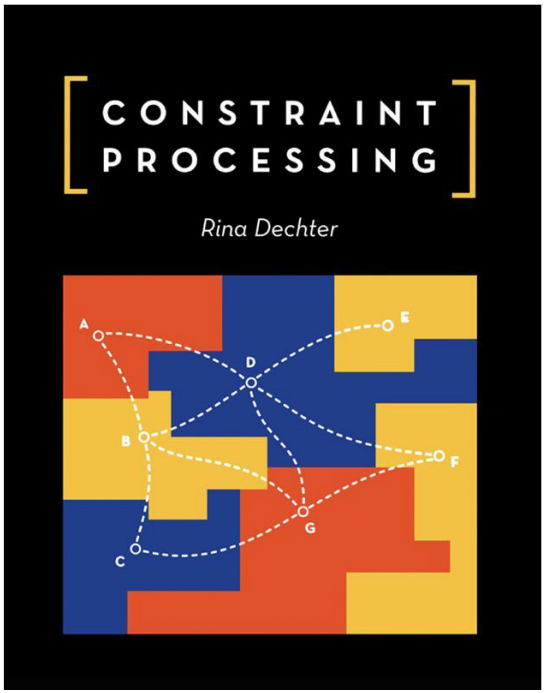
**Matteo Zavatteri**

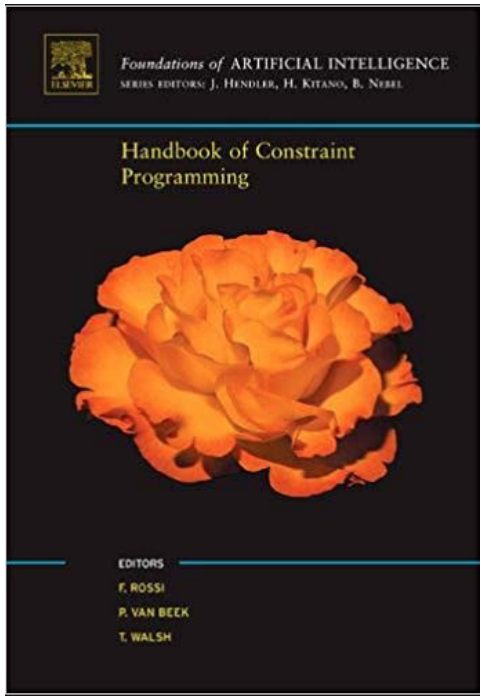
**Math Decisions 2020**

# Outline

1. Constraint networks
2. Global constraints
3. More expressiveness
4. Modeling CSPs

# Reference books




More online:
`http://www.constraint-programming.com/people/regin/papers/global.pdf`
`https://www.minizinc.org` (have a look at *MiniZinc Handbook*)

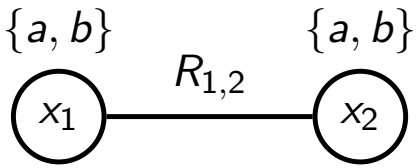
# Constraint Networks

## Formal definition

A *constraint network* is a triple $N = \langle X, D, C \rangle$, where:

1. $X = \{x_1, \ldots, x_n\}$ is a finite set of *(decision) variables*
2. $D = \{D_1, \ldots, D_n\}$ is a set of associated domains.
3. $C$ is a finite set of *constraints*. Each constraint is a relation $R_{i,\ldots,k}$ (defined over the set of variables $\{x_i, \ldots, x_k\}$ such that $R_{i,\ldots,k} \subseteq D_i \times \cdots \times D_k$.

*To ease notation, scopes and tuples are "ordered" with respect to variable indexes.*




## Formal specification

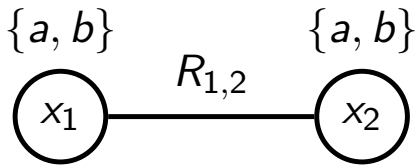
- $X = \{x_1, x_2\}$, $D = \{D_1, D_2\}$
- $D_1 = D_2 = \{a, b\}$
- $C = \{R_{1,2}\}$, where $R_{1,2} = \{(a, b), (b, a)\}$ (i.e., $x_1 \neq x_2$)

# Consistency

## Consistency

A constraint network is consistent if ther exists a solution. That is, if every variable can be assigned a value from its domain such that all constraints are eventually satisfied.

Given a solution $x_1 = v_1, \ldots, x_n = v_n$, a constraint $R_{p,\ldots,z}$ is satisfied, if $(v_p, \ldots, v_z) \in R$, where $x_p = v_p, \ldots, x_z = v_z$ are in the solution.




Solution: $x_1 = a$, $x_2 = b$

# A few problems associated to constraint networks

Given a constraint network $N$, we might address the following problems:

## Decision problems:

- Is $N$ consistent/inconsistent?
- Does $N$ admit at least/at most/exaclty $k$ different solutions?
- Is there an assignment satisfying at least $k$ constraints?

## Search problems:

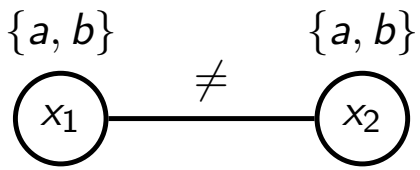
- Find a consistent assignment
- Find 2,3,etc different consistent assignments
- Find all consistent assignments
- How many different consistent assignments does $N$ admit?
- Find an assignment maximizing the number of satisfied constraints

In this part, we will only address search problems for which we need to find a fixed number of solutions.

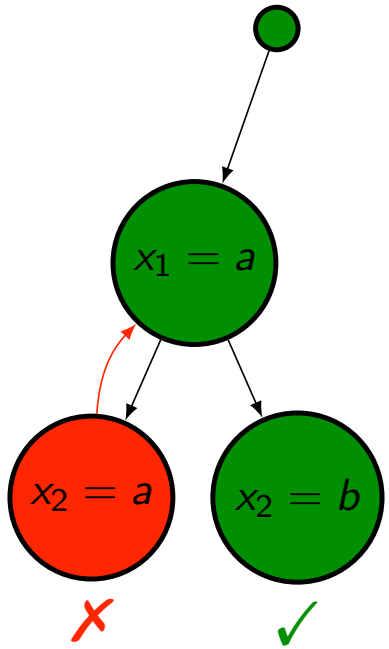
# Search for 1 solution: backtracking

Assume a recursive algorithm that assigns variables according to the order of their indexes.

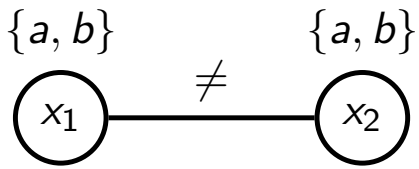
The algorithm stops *as soon as* it finds a solution



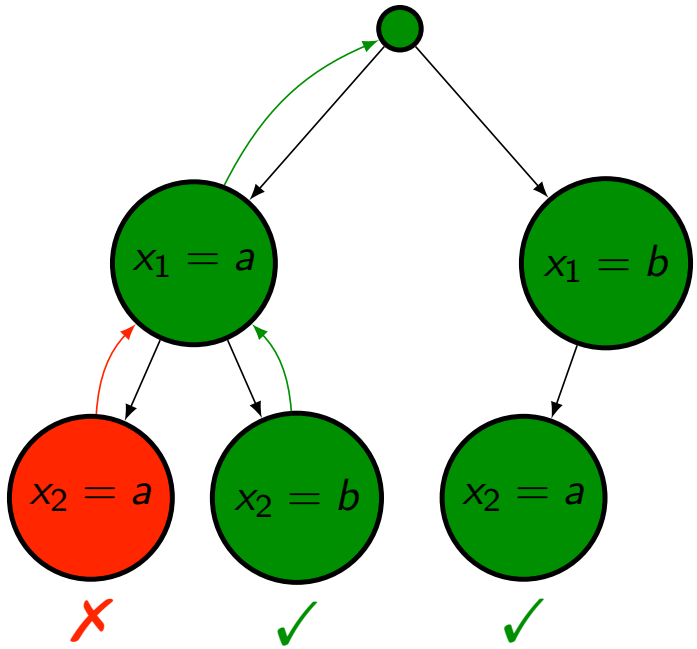

Solution 1: $x_1 = a$, $x_2 = b$



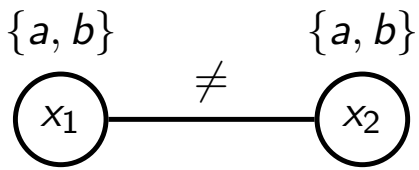
# Search for 2 solutions: keep searching up to the 2nd



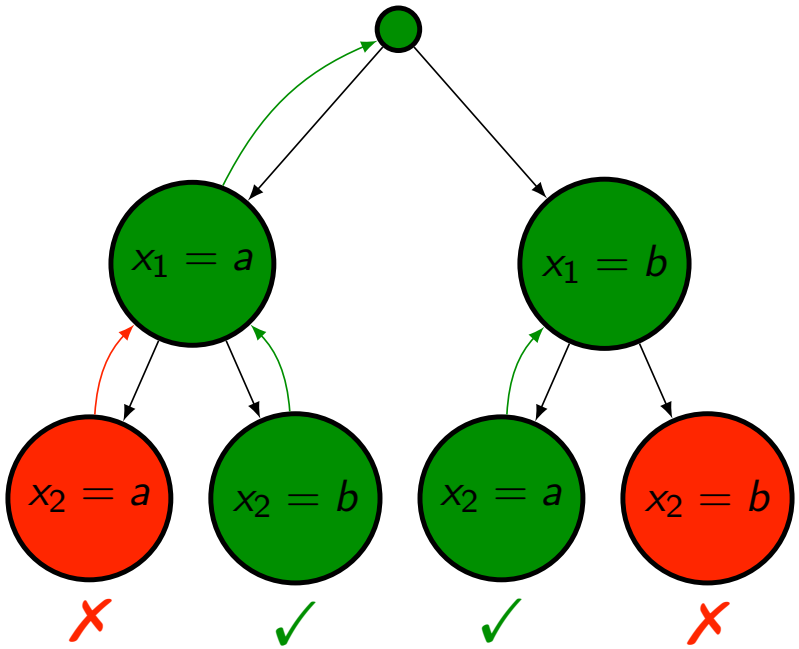

Solution 1: $x_1 = a$, $x_2 = b$
Solution 2: $x_1 = b$, $x_2 = a$



# Search for all solutions: keep searching up to the end



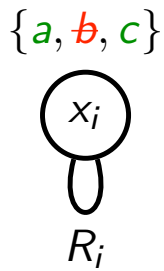

Solution 1: $x_1 = a$, $x_2 = b$
Solution 2: $x_1 = b$, $x_2 = a$



# Filtering domains: node consistency

## Node consistency

A variable $x_i$ is node consistent if for each $v \in D_i$ we have that $(v) \in R_i$.




$R_i = \{(a), (c)\}$

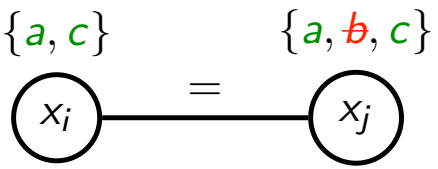
- $x_i$ is not node consistent as $b \notin R_i$
- Removing $b$ from $D_i$ makes $x_i$ node consistent

**Rationale:** *every solution must satisfy $R_i$ and $x_i = b$ just doesn't.*

# Filtering domains: arc consistency

## Arc consistency

A pair of different variables $x_i, x_j$ is arc consistent if for each $v_i \in D_i$ there exists $v_j \in D_j$ such that $(v_i, v_j) \in R_{ij}$.



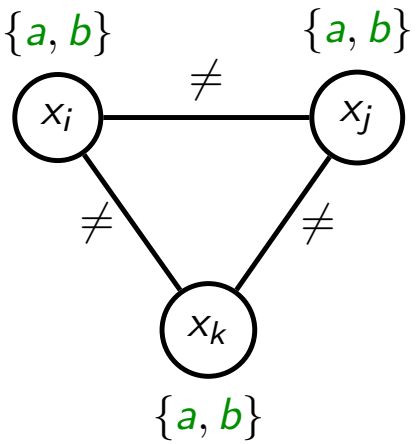

- $x_i, x_j$ are not arc consistent. $(a, b) \notin R_{i,j}$, $(c, b) \notin R_{i,j}$
- Removing $b$ from $D_j$ makes $x_i, x_j$ arc consistent.

**Rationale:** *every solution must satisfy $R_{i,j}$ and $x_j = b$ (whatever $x_i$) just doesn't.*

# Filtering domains: path consistency

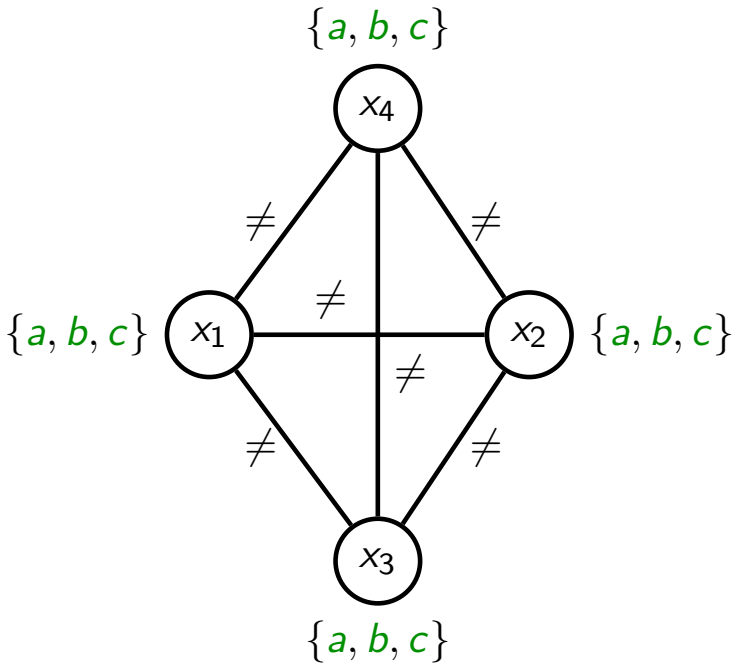
## Path consistency

A pair of variables variables $x_i, x_j$ is path consistent with another variable $x_k$ ($x_i \neq x_j \neq x_k$) if for each $v_i \in D_i$, $v_j \in D_j$ with $(v_i, v_j) \in R_{ij}$, there exists $v_k \in D_k$ such that $(v_i, v_k) \in R_{ik}$ and $(v_j, v_k) \in R_{jk}$




- Arc consistent!
- Not path consistent. $x_i = a$, $x_j = b$ cannot be extended to any $x_k = v_k$ where $v_k \in \{a, b\}$.
- The network is actually inconsistent.

# Path consistency is not enough!




- Path consistent!
- Yet, the network is actually inconsistent.
- All variables must get different values. Four variables. Three values.
- Examples like this extend to networks with $n$ variables, $n-1$ values in each domain and a "$\neq$" constraint between any pair of distinct variables.
- Enforcing consistency on $n$ variables says nothing for $n+1$ variables.

Node, arc and path consistency are *pruning techniques* to rule out (even many) values from domains. But eventually, we still need to search.

# Global constraints

Global constraints = compact constraints

- they encapsulate several constraints in a single one
- they avoid writing an explicit relation of many, many tuples
- they typically involve several variables
- they allow for the specification of "high level" constraints

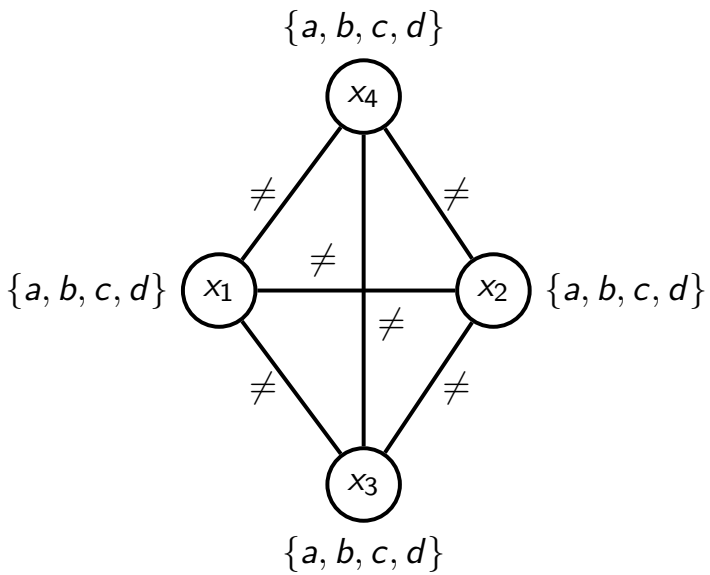
Examples:

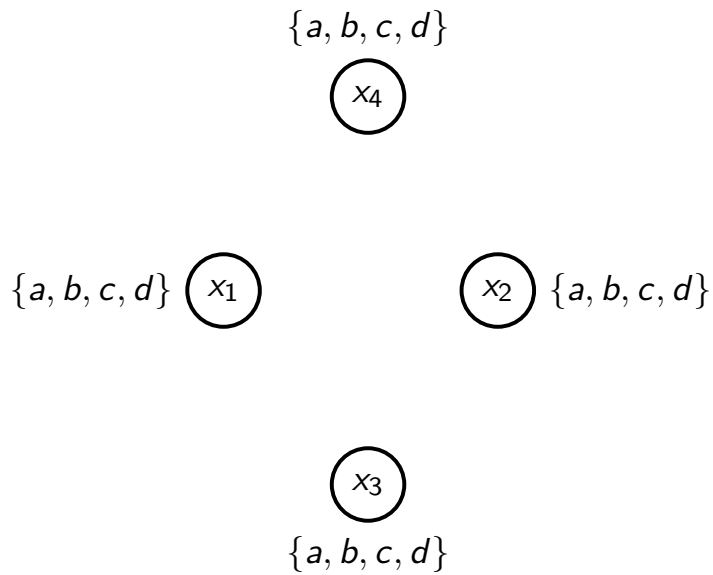
- $\texttt{all\_different}(x_1, \ldots, x_n)$
- $\ldots$

# all_different

## All different

A solution $x_1 = v_1, \ldots, x_n = v_n$ to a constraint network satisfies an $\texttt{all\_different}(x_i, \ldots, x_j)$ iff $v_i \neq \cdots \neq v_j$.



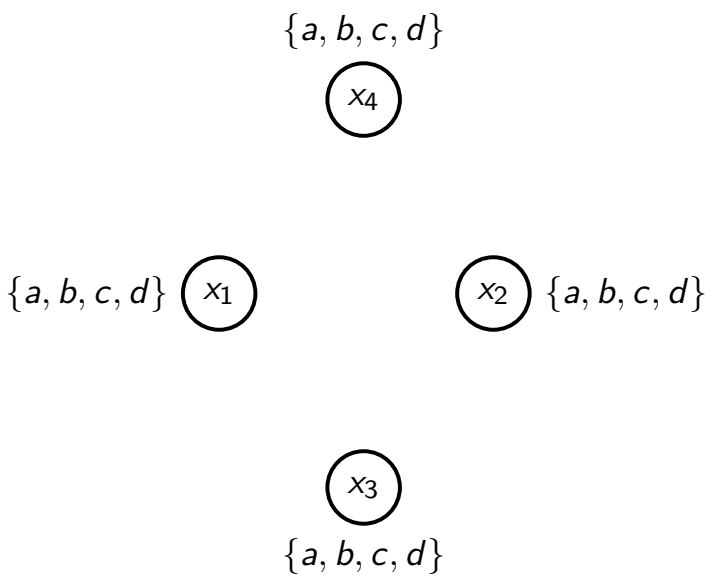

$\texttt{all\_different}(x_1, x_2, x_3, x_4)$

$x_1 = a,\ x_2 = b,\ x_3 = c,\ x_4 = d$

# `all_different`, more formally

## A possible formal definition

`all_different`$(x_1, \ldots, x_n)$ is equivalent to a relation $R_{1,\ldots,n}$ such that for each tuple $(v_1, \ldots, v_n) \in R$ , it holds that $|\{v_i \mid v_i \in (v_1, \ldots, v_n)\}| = n$.




`all_different`$(x_1, x_2, x_3, x_4)$

$R_{1,2,3,4} = \{(a,b,c,d), (a,b,d,c), (a,c,b,d), (a,c,d,b), (a,d,b,c), (a,d,c,b), (b,a,c,d), (b,a,d,c), (b,c,a,d), (b,c,d,a), (b,d,a,c), (b,d,c,a), (c,a,b,d), (c,a,d,b), (c,b,a,d), (c,b,d,a), (c,d,a,b), (c,d,b,a), (d,a,b,c), (d,a,c,b), (d,b,a,c), (d,b,c,a), (d,c,a,b), (d,c,b,a)\}$

$x_1 = c,\ x_2 = a,\ x_3 = b,\ x_4 = d$

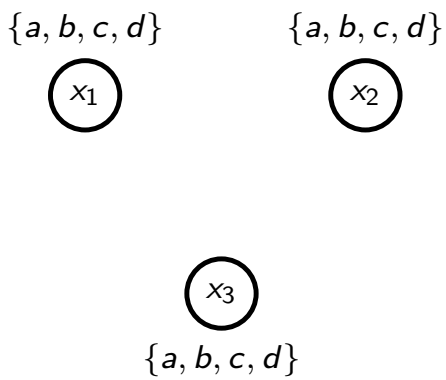
# Boosting expressiveness maintaining complexity

## Main complexity result

Deciding consistency of (classic) constraint networks is NP-complete.

- it is easy to see that the problem remains NP-complete even if we add global constraints or we turn a set of constraints into a boolean formula where global constraints and relations play the role of boolean atoms (provided that, given a solution, the satisfaction of each atom can be checked in polynomial time).

$F ::= R_{i,\dots,k} \mid \texttt{global\_constraint}(\dots) \mid \neg F \mid (F) \mid F \square F$ where $\square \in \{\wedge, \vee, \Rightarrow, \Leftrightarrow, \dots\}$




Let $x_i = v_j$ be a short for $R_i = \{(v_j)\}$ (i.e., a further constraint language improvement!). The formula:

$$\texttt{all\_different}(x_1, x_2, x_3) \wedge (x_1 = a \vee x_3 = c) \wedge x_2 = d \wedge (x_1 = a \Rightarrow x_2 = c)$$

is satisfied by the solution $x_1 = b$, $x_2 = d$, $x_3 = c$

The solution (certificate of yes) can still be checked in polynomial time!

# Humanizing relations by means of formulae

Consider the following constraint language:

$$F ::= x = v \mid (F) \mid F \wedge F \mid F \vee F$$

Wouldn't it be enough to encode a set of constraints $R_{i,\ldots,z}$? (yes!)

Consider a constraint network $N = \langle X, D, C \rangle$ where:

- $X = \{x_1, x_2, x_3\}$
- $D_1 = D_2 = D_3 = \{a, b\}$
- $C = \{R_2, R_{13}, R_{123}\}$, where $R_2 = \{(b)\}$, $R_{13} = \{(a, a), (b, a), (b, b)\}$, $R_{123} = \{(a, b, a), (b, a, b)\}$

$C$ can be encoded in a (DNF) formula $F \equiv \underbrace{F_2}_{R_2} \wedge \underbrace{F_{13}}_{R_{13}} \wedge \underbrace{F_{123}}_{R_{123}}$, where:

- $F_2 \equiv (x_2 = b)$
- $F_{13} \equiv ((x_1 = a \wedge x_3 = a) \vee (x_1 = b \wedge x_3 = a) \vee (x_1 = b \wedge x_3 = b))$
- $F_{123} \equiv ((x_1 = a \wedge x_2 = b \wedge x_3 = a) \vee (x_1 = b \wedge x_2 = a \wedge x_3 = b))$

In general $R_{i,\ldots,z}$ can be encoded in $F \equiv (\bigvee_{(v_i,\ldots,v_z) \in R_{i,\ldots,z}} (x_i = v_i \wedge \cdots \wedge x_z = v_z))$

# Modeling constraint satisfaction problems (CSP)

In what follows, we will:

1. start with the definition of some problem in natural and formal language
2. model it in the input language of MiniZinc
3. push a button to search for one (or more) solution(s)

In this order. This is what we are going to do.

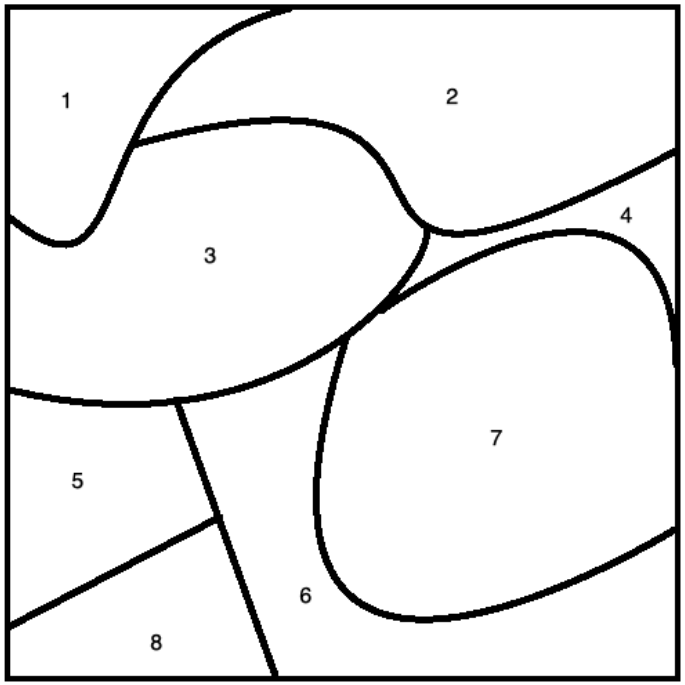
# MiniZinc


`https://www.minizinc.org`

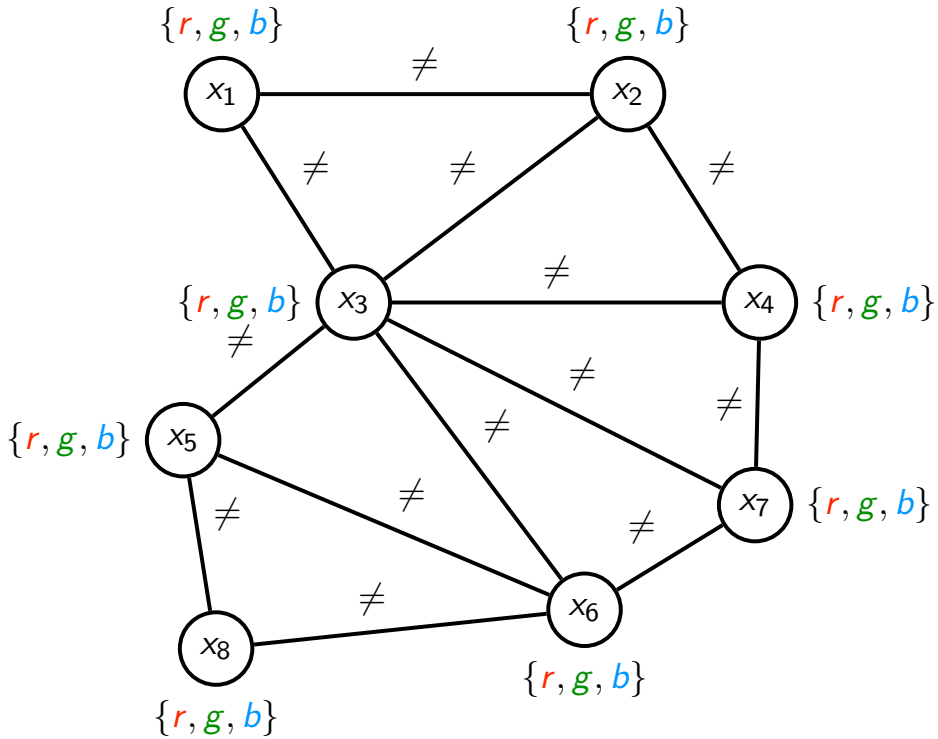
- `MiniZinc` is a free and open-source constraint modeling language that allows you to write models that are compiled into `FlatZinc`: an input language that is understood by a wide range of solvers.
- `MiniZinc` is developed at item Monash University `MonashUniversity` in collaboration with Data61 Decision Sciences `https://research.csiro.au/data61/tag/decision-sciences/` and the University of Melbourne `https://unimelb.edu.au`.
- `MiniZinc` is available for Windows, Linux and MacOS. Have a look at `https://www.minizinc.org/software.html`, download and install it on your computer.

# Modeling CSPs: Map coloring

Can you color this map by using red, green and blue so that any two adjacent regions have different colors?

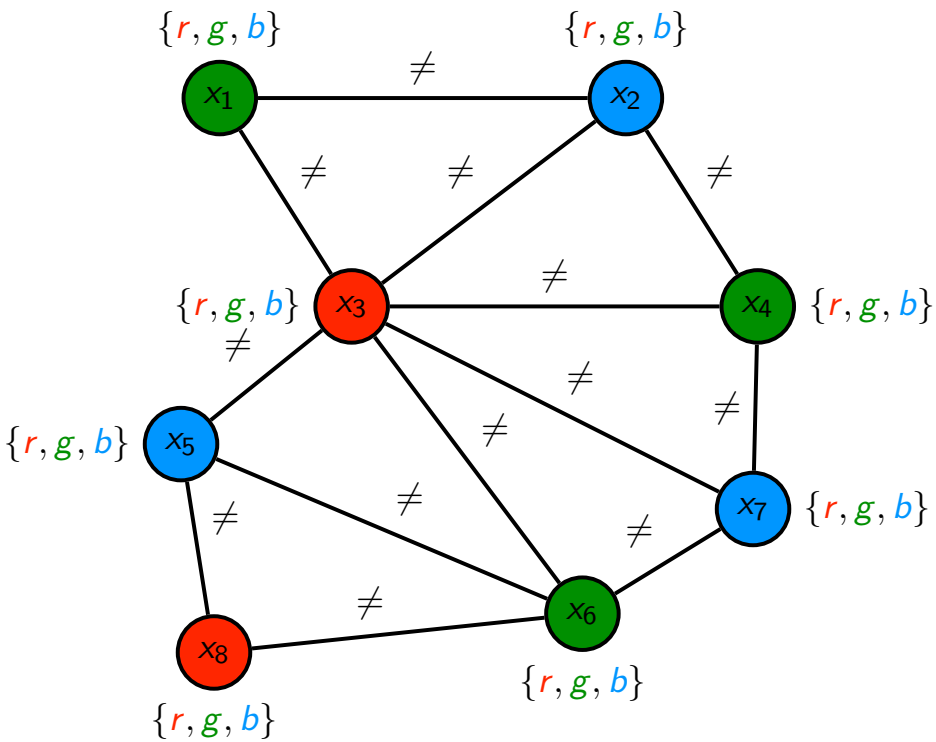



Constraint Network formulation

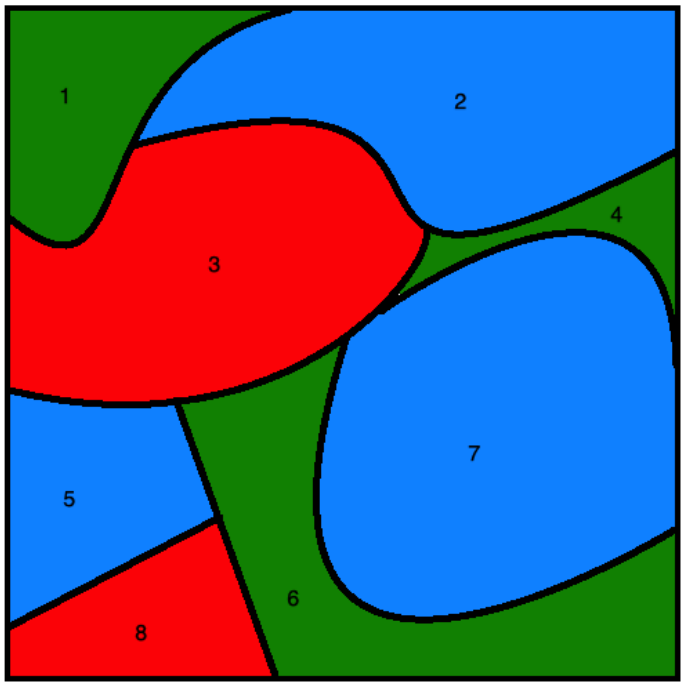


# Modeling CSPs: Map coloring

Solution to the corresponding constraint network




Back to the map!

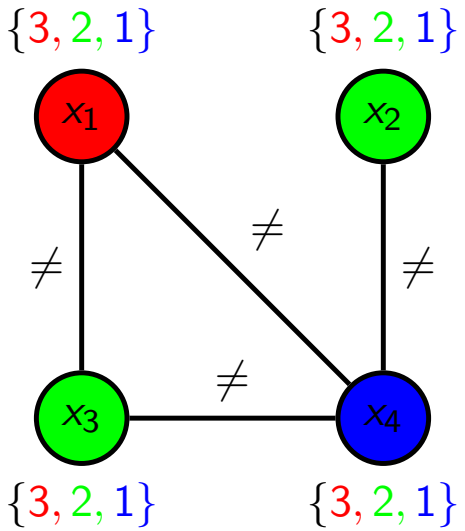


# Graph $K$-Coloring Problem

**Input.** A graph $G = (V, E)$ and a positive integer $K$.
**Output.** yes iff there exists $f : V \to \{1, .., K\}$ s.t. $f(u) \neq f(v)$ for each $(u, v) \in E$

**Example.** $G = (V, E)$, where $V = \{x_1, x_2, x_3, x_4\}$ and

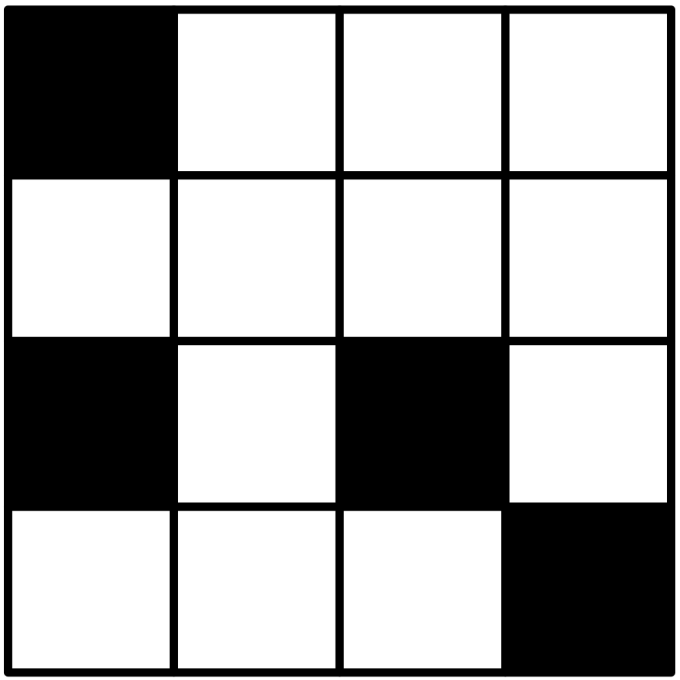
$E = \{(x_1, x_3), (x_1, x_4), (x_3, x_4), (x_2, x_4)\}$ and $K = 3$.



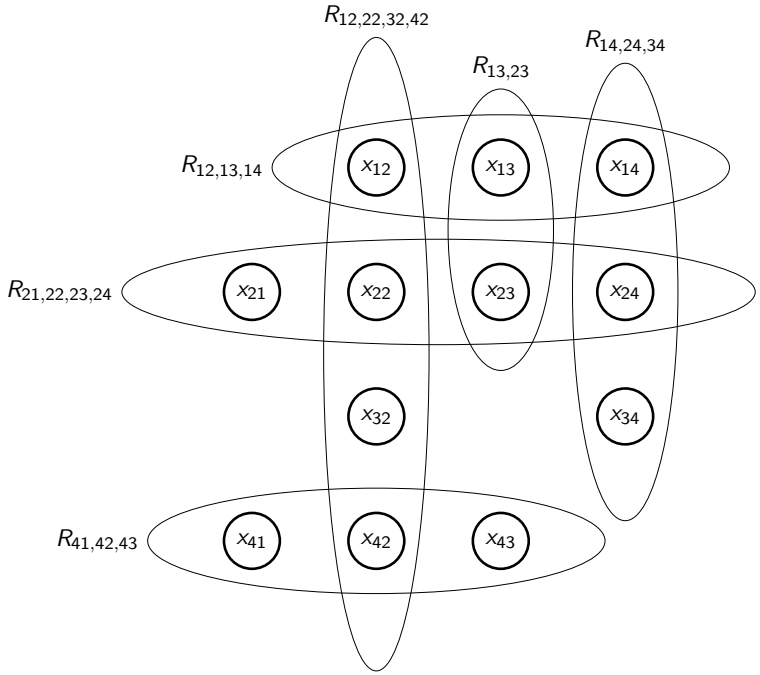

$f(x_1) = r$, $f(x_2) = g$, $f(x_3) = g$, $f(x_4) = b$.

# Modeling CSPs: Crossword puzzle

Can you fill the crossword puzzle, by only using the words `CAT`, `LATE`, `TED`, `TEA`, `CAKE`, `AT` (repetitions allowed)?



Constraint Network formulation

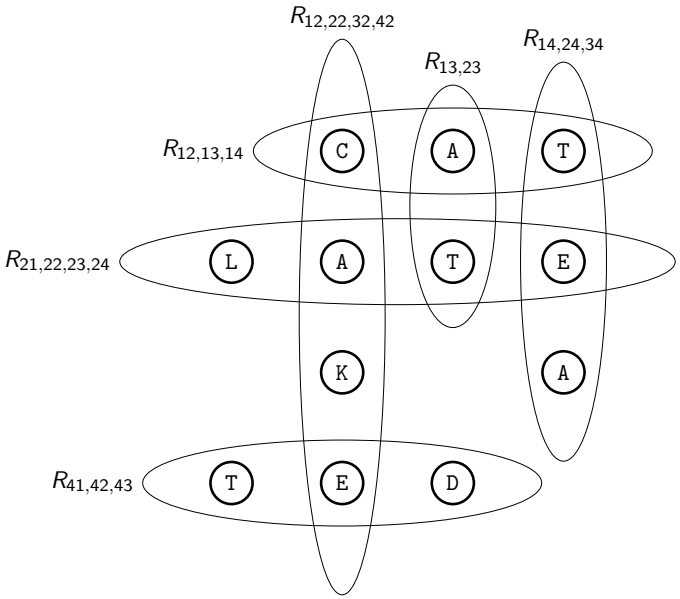



For each $x_{ij}$, $D_{ij} = \{\mathtt{A}, \mathtt{C}, \mathtt{D}, \mathtt{E}, \mathtt{K}, \mathtt{L}, \mathtt{T}\}$

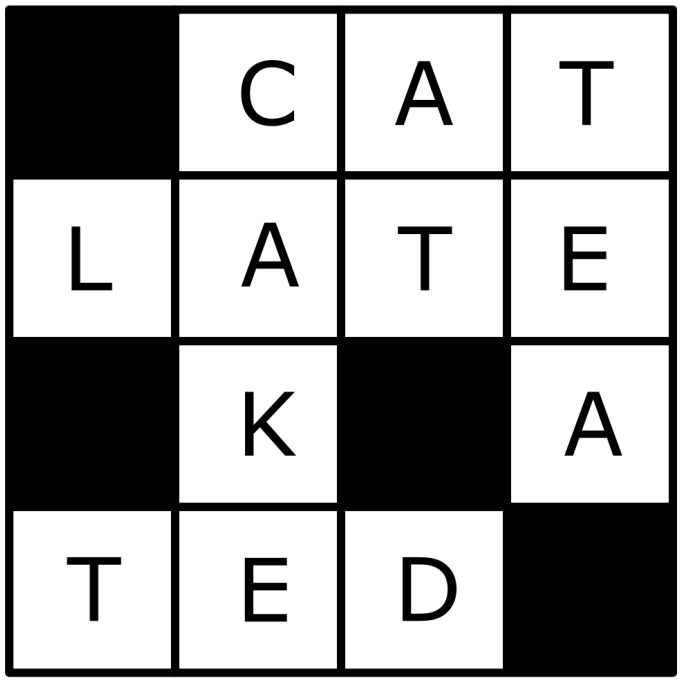
$R_{12,13,14} = \{(\mathtt{C},\mathtt{A},\mathtt{T}), (\mathtt{T},\mathtt{E},\mathtt{D}), (\mathtt{T},\mathtt{E},\mathtt{A})\}$, $R_{21,22,23,24} = \{(\mathtt{L},\mathtt{A},\mathtt{T},\mathtt{E}), (\mathtt{C},\mathtt{A},\mathtt{K},\mathtt{E})\}$,
$R_{41,42,43} = \{(\mathtt{C},\mathtt{A},\mathtt{T}), (\mathtt{T},\mathtt{E},\mathtt{D}), (\mathtt{T},\mathtt{E},\mathtt{A})\}$, $R_{12,22,32,42} = \{(\mathtt{L},\mathtt{A},\mathtt{T},\mathtt{E}), (\mathtt{C},\mathtt{A},\mathtt{K},\mathtt{E})\}$,
$R_{13,23} = \{\{(\mathtt{A},\mathtt{T})\}$, $R_{14,24,34} = \{(\mathtt{C},\mathtt{A},\mathtt{T}), (\mathtt{T},\mathtt{E},\mathtt{D}), (\mathtt{T},\mathtt{E},\mathtt{A})\}$

# Modeling CSPs: Crossword puzzle

Solution to the corresponding constraint network




Back to crossword puzzle




For each $x_{ij}$, $D_{ij} = \{\mathtt{A}, \mathtt{C}, \mathtt{D}, \mathtt{E}, \mathtt{K}, \mathtt{L}, \mathtt{T}\}$

$R_{12,13,14} = \{(\mathtt{C}, \mathtt{A}, \mathtt{T}), (\mathtt{T}, \mathtt{E}, \mathtt{D}), (\mathtt{T}, \mathtt{E}, \mathtt{A})\}$, $R_{21,22,23,24} = \{(\mathtt{L}, \mathtt{A}, \mathtt{T}, \mathtt{E}), (\mathtt{C}, \mathtt{A}, \mathtt{K}, \mathtt{E})\}$,
$R_{41,42,43} = \{(\mathtt{C}, \mathtt{A}, \mathtt{T}), (\mathtt{T}, \mathtt{E}, \mathtt{D}), (\mathtt{T}, \mathtt{E}, \mathtt{A})\}$, $R_{12,22,32,42} = \{(\mathtt{L}, \mathtt{A}, \mathtt{T}, \mathtt{E}), (\mathtt{C}, \mathtt{A}, \mathtt{K}, \mathtt{E})\}$,
$R_{13,23} = \{(\mathtt{A}, \mathtt{T})\}$, $R_{14,24,34} = \{(\mathtt{C}, \mathtt{A}, \mathtt{T}), (\mathtt{T}, \mathtt{E}, \mathtt{D}), (\mathtt{T}, \mathtt{E}, \mathtt{A})\}$

# Sudoku

**Input.** A 9x9 grid in which each cell $(i, j)$ must be filled with digits from 1 to 9. Some cells are prefilled (coherently with what the solution should look like).
**Output.** A filling of all empty cells of the grid such that each digit appears exaclty once in each row, each column and each 3x3 subsquare.

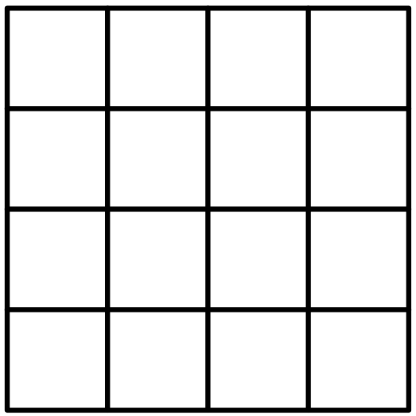
**Example.**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 5 | 9 | 8 | | | | 7 | |
| | | | | 1 | | | | |
| 3 | | | | | 2 | 5 | | 8 |
| | | | 6 | 2 | | | | 1 |
| | 8 | 2 | | 3 | | 4 | | |
| | | 6 | | | | | 3 | |
| 9 | | | | 7 | | | | 3 |
| | | | | 9 | | | 2 | |
| | 4 | 8 | | | 6 | 7 | | |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 6 | 5 | 9 | 8 | 4 | 3 | 1 | 7 | 2 |
| 8 | 2 | 7 | 5 | 1 | 9 | 3 | 4 | 6 |
| 3 | 1 | 4 | 7 | 6 | 2 | 5 | 9 | 8 |
| 4 | 7 | 3 | 6 | 2 | 5 | 9 | 8 | 1 |
| 5 | 8 | 2 | 9 | 3 | 1 | 4 | 6 | 7 |
| 1 | 9 | 6 | 4 | 8 | 7 | 2 | 3 | 5 |
| 9 | 6 | 1 | 2 | 7 | 4 | 8 | 5 | 3 |
| 7 | 3 | 5 | 1 | 9 | 8 | 6 | 2 | 4 |
| 2 | 4 | 8 | 3 | 5 | 6 | 7 | 1 | 9 |

# Modeling CSPs: 4-queens problem

Can you place 4 queens on a $4 \times 4$ board such that no queen attacks any other? I.e., no row, column, nor diagonal sharing.

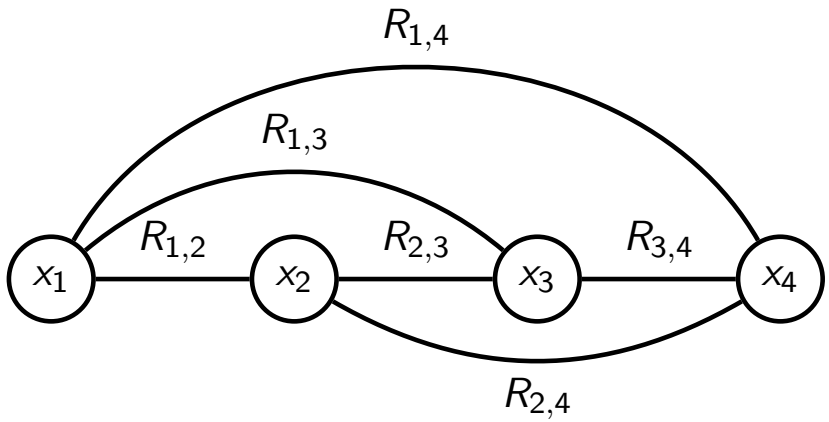


Constraint Network formulation

$j$ variables, one for each column.

$D_i = \{1, 2, 3, 4\}$ for $i = 1, \ldots, 4$ (rows).

$x_j = i$ means that there is a queen in $(i, j)$




$R_{1,2} = \{(1,3),(1,4),(2,4),(3,1),(4,1),(4,2)\}$,

$R_{1,3} = \{(1,2),(1,4),(2,1),(2,3),(3,2),(3,4),(4,1),(4,2),(4,3)\}$,

$R_{1,4} = \{(1,2),(1,3),(2,1),(2,3),(2,4),(3,1),(3,2),(3,4),(4,2),(4,3)\}$,

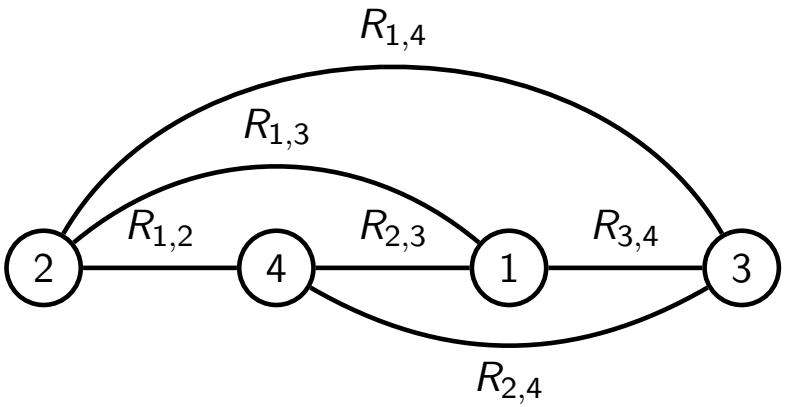
$R_{2,3} = \{(1,3),(1,4),(2,4),(3,1),(4,1),(4,2)\}$

$R_{2,4} = \{(1,2),(1,4),(2,1),(2,3),(3,2),(3,4),(4,1),(4,3)\}$

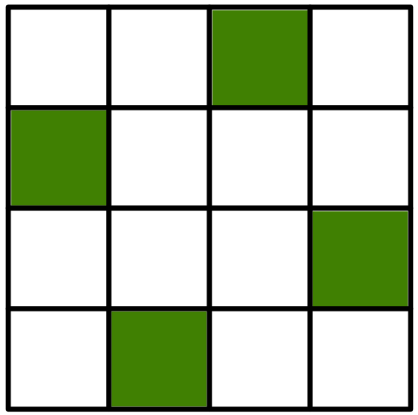
$R_{3,4} = \{(1,3),(1,4),(2,4),(3,1),(4,1),(4,2)\}$

# Modeling CSPs: 4-queens problem

Solution to the constraint network




Back to the board.



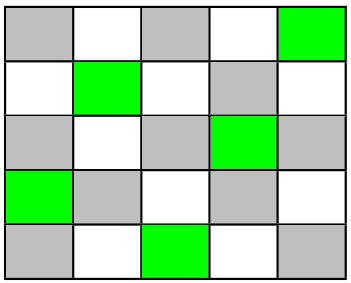
$R_{1,2} = \{(1,3),(1,4),(2,4),(3,1),(4,1),(4,2)\}$,
$R_{1,3} = \{(1,2),(1,4),(2,1),(2,3),(3,2),(3,4),(4,1),(4,2),(4,3)\}$,
$R_{1,4} = \{(1,2),(1,3),(2,1),(2,3),(2,4),(3,1),(3,2),(3,4),(4,2),(4,3)\}$,
$R_{2,3} = \{(1,3),(1,4),(2,4),(3,1),(4,1),(4,2)\}$
$R_{2,4} = \{(1,2),(1,4),(2,1),(2,3),(3,2),(3,4),(4,1),(4,3)\}$
$R_{3,4} = \{(1,3),(1,4),(2,4),(3,1),(4,1),(4,2)\}$

**Interpretation of constraints:** each pair $(i,j) \in R_{a,b}$ means that the placement of two queens in positions $(i,a)$ and $(j,b)$ is permitted.

# $N$ queens

**Input.** An $N \times N$ board.
**Output.** A placement of $N$ queens such that no queen can attack any other in one move.
**Note.** A queen can attack another queen if the two queens are on the same row, column or diagonal.

**Example.** $N = 5$



$(4,1), (2,2), (5,3), (3,4), (1,5)$

Two queens are on the same diagonal if one of the following conditions hold:

- The row number plus the column number for each of the two queens are equal
- The row number minus the column number for each of the two queens are equal

# Modeling CSPs: (very) simple activity scheduling

4 tasks $A, B, C, D$ to be scheduled.
Suppose any task is scheduled at a specific hour and takes that hour to be executed

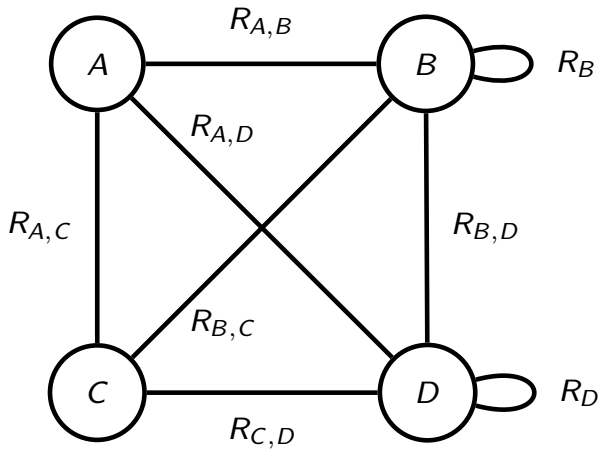
Constraints:

- $B$ is either the 1st or the 4th to be executed
- $D$ is either the 3rd or the 4th to be executed
- One task at a time (no 2 tasks are executed simultaneosly)
- $B$ before $A$, $A$ before $D$, $C$ after $B$

Constraint network formulation

Assume $A, B, C, D$ resemble the indexes $1, 2, 3, 4$.

$D_I = \{1, 2, 3, 4\}$ for $I = A, B, C, D$.



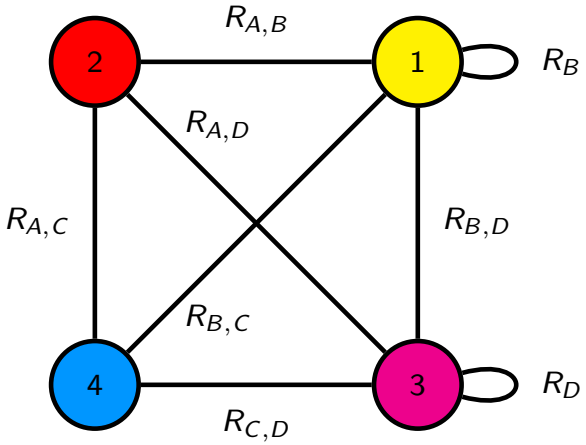

$R_B = \{(1), (4)\}$, $R_D = \{(3), (4)\}$
$R_{I,J} = \{(1,2), (1,3), (1,4), (2,1), (2,3), (2,4), (3,1), (3,2), (3,4)\}$ for $I, J = A, B, C, D$ with $I \neq J$
$R_{A,B} = \{(2,1), (3,1), (3,2), (4,3), (4,2), (4,1)\}$
$R_{A,D} = \{(1,2), (1,3), (1,4), (2,3), (2,4), (3,4)\}$
$R_{B,C} = \{(1,2), (1,3), (1,4), (2,3), (2,4), (3,4)\}$

# Modeling CSPs: (very) simple activity scheduling

Solution to the constraint

$A, B, C, D$ resemble the indexes $1, 2, 3, 4$.

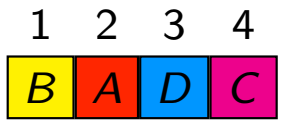
$D_I = \{1, 2, 3, 4\}$ for $I = A, B, C, D$.




Back to the scheduling problem

Constraints:

- $B$ is either the 1st or the 4th to be executed
- $D$ is either the 3rd or the 4th to be executed
- One task at a time (no 2 tasks are executed simultaneosly)
- $B$ before $A$, $A$ before $D$, $C$ after $B$




$R_B = \{(1), (4)\}$, $R_D = \{(3), (4)\}$

$R_{I,J} = \{(1,2), (1,3), (1,4), (2,1), (2,3), (2,4), (3,1), (3,2), (3,4)\}$ for $I, J = A, B, C, D$ with $I \neq J$ (e.g., $R_{A,B}$)

$R_{A,B} = \{(2,1), (3,1), (3,2), (4,3), (4,2), (4,1)\}$

$R_{A,D} = \{(1,2), (1,3), (1,4), (2,3), (2,4), (3,4)\}$

$R_{B,C} = \{(1,2), (1,3), (1,4), (2,3), (2,4), (3,4)\}$

# Modeling CSPs: 3SAT

SAT: A boolean formula $\varphi$ in CNF (=conjunctive normal form) over a set of variables $x_1, \ldots, x_n$

A CNF is a conjunction of clauses, where each clause is a disjunction of literals. E.g., $(x_1 \vee x_2) \wedge (\neg x_3 \vee x_4 \vee \neg x_5) \ldots$

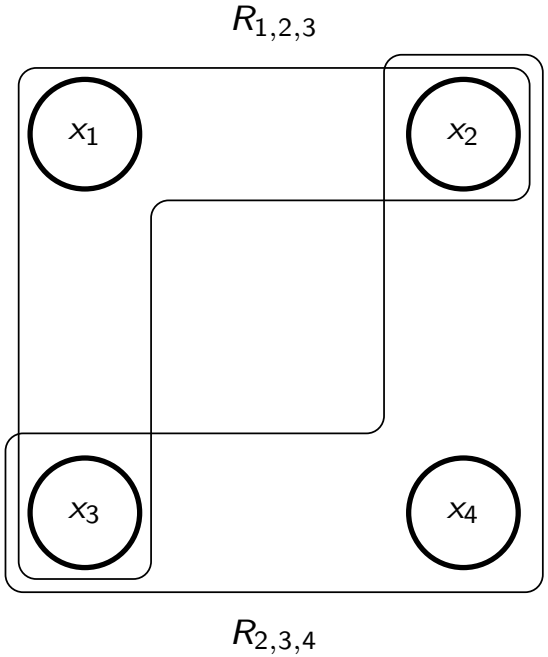
3SAT: A SAT instance in which each clause has exactly 3 literals.

Consider

$$\varphi \equiv \underbrace{(x_1 \vee \neg x_2 \vee x_3)}_{\text{First clause}} \wedge \underbrace{(x_2 \vee \neg x_3 \vee \neg x_4)}_{\text{Second clause}}$$

Constraint Network formulation

$D_i = \{0, 1\}$ for $i = 1, \ldots, 4$.



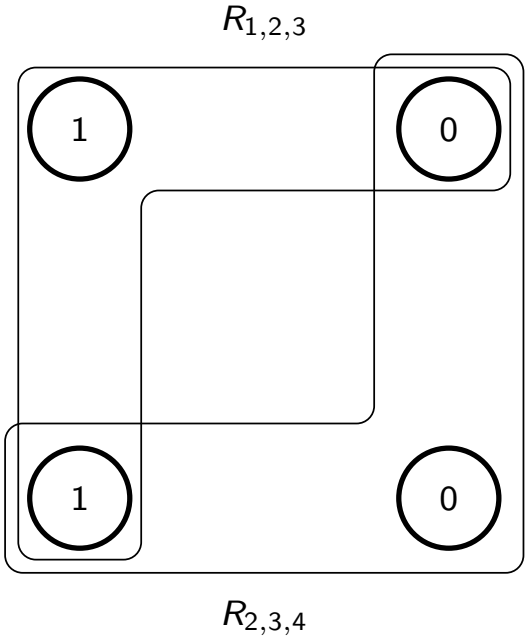

$$R_{1,2,3} = \{(0,0,0), (0,0,1), (0,1,1), (1,0,0), (1,0,1), (1,1,0), (1,1,1)\}$$

$$R_{2,3,4} = \{(0,0,0), (0,0,1), (0,1,0), (1,0,0), (1,0,1), (1,1,0), (1,1,1)\}$$

# Modeling CSPs: 3SAT

Constraint Network formulation

$D_i = \{0, 1\}$ for $i = 1, \ldots, 4$.




Back to 3SAT

$$\varphi \equiv \underbrace{(x_1 \vee \neg x_2 \vee x_3)}_{\text{First clause}} \wedge \underbrace{(x_2 \vee \neg x_3 \vee \neg x_4)}_{\text{Second clause}}$$

Model: $x_1 = 1$, $x_2 = 0$, $x_3 = 1$, $x_4 = 0$,

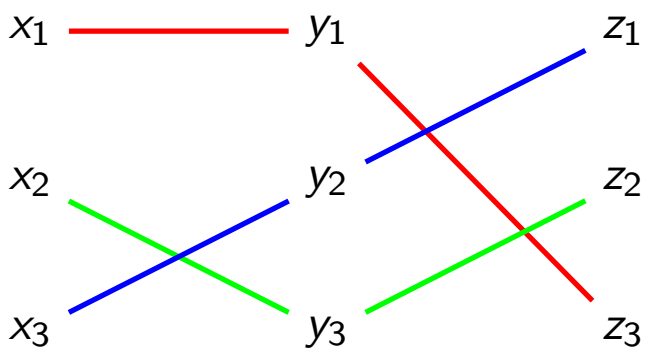
$$R_{1,2,3} = \{(0,0,0), (0,0,1), (0,1,1), (1,0,0), (1,0,1), (1,1,0), (1,1,1)\}$$

$$R_{2,3,4} = \{(0,0,0), (0,0,1), (0,1,0), (1,0,0), (1,0,1), (1,1,0), (1,1,1)\}$$

# Three Dimensional Matching

**Input.** 3 disjoint sets $X, Y, Z$ each of size $n$ and a set $T \subseteq X \times Y \times Z$.
**Output.** yes iff there exists $T_1 \subseteq T$ such that $|T_1| = n$ and for each $t \in X \cup Y \cup Z$, $t$ is contained exacly in one of the triples of $T_1$.
**Example** $X = \{x_1, x_2, x_3\}$, $Y = \{y_1, y_2, y_3\}$, $Z = \{z_1, z_2, z_3\}$

$T = \{(x_1, y_1, z_3), (x_2, y_1, z_1), (x_2, y_3, z_2), (x_3, y_1, z_3), (x_3, y_2, z_1)\}$




$T_1 = \{(x_1, y_1, z_3), (x_2, y_3, z_2), (x_3, y_2, z_1)\}$

# Knapsack Problem

**Input.** A finite set $I$ of items where for each $i \in I$, $w(i) \in \mathbb{Z}^+$ is the weight and $v(i) \in \mathbb{Z}^+$ is the value of item $i$, and two positive integers $W$ and $V$.
**Output.** yes iff there exists $I_1 \subseteq I$ such that $\sum_{i \in I_1} w(i) \leq W$ and $\sum_{i \in I_1} v(i) \geq V$.

**Example**: $W = 17$, $V = 32$

| Item | Weight | Value |
|---|---|---|
| 1 | 1 | 7 |
| 2 | 8 | 7 |
| 3 | 3 | 8 |
| 4 | 6 | 10 |
| 5 | 5 | 7 |
| 6 | 8 | 1 |
| 7 | 7 | 4 |
| 8 | 2 | 3 |
| 9 | 9 | 5 |
| 10 | 10 | 1 |

# Set Packing

**Input.** A collection $C$ of finite sets and a positive integer $K \leq |C|$ **Output.** yes iff $C$ contains at least $K$ mutually disjoint sets.

**Example** $C = \{\{2,4\},\{1,3,5\},\{1,2,4\},\{1,2,5\},\{2,3\},\{6,8\},\{1,3,9\},\{10,7\},\{10,2,3\},\{3,5,8\}\}$, $K = 4$;

# Set Splitting

**Input.** A collection $C$ of subsets of a finite set $S$.
**Output.** yes iff there exists a partition of $S$ in two subsets $S_1$ and $S_2$ such that for each $C_i \in C$, $C_i$ is neither a subset of $S_1$ nor $S_2$.
**Example** $S = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$,
$C = \{\{1, 2, 4\}, \{1, 2, 5\}, \{2, 3\}, \{10, 7\}, \{10, 2, 3\}, \{3, 5, 8\}\}$

$S_1 = \{3, 4, 5, 10\}$, $S_2 = \{1, 2, 6, 7, 8, 9\}$

# Subset Sum

**Input.** A finite set $X$ of elements where for each $x \in X$, $s(x) \in \mathbb{Z}$, and an integer $S$.
**Output.** yes iff there exists $X_1 \subseteq X$ such that $\sum_{x \in X_1} s(x) = S$.

**Example** $S = 13$

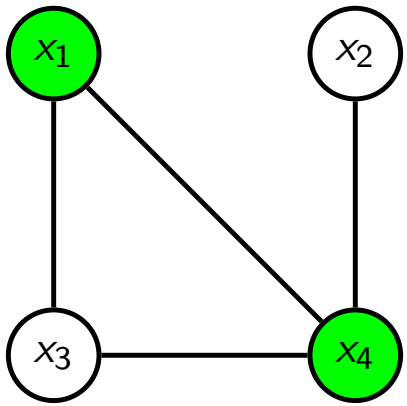
| S | s(x) |
|---|---|
| 1 | $-1$ |
| 2 | $-8$ |
| 3 | 3 |
| 4 | $-6$ |
| 5 | $-5$ |
| 6 | 8 |
| 7 | $-7$ |
| 8 | 2 |
| 9 | $-9$ |
| 10 | 10 |

# Vertex Cover

**Input.** A graph $G = (V, E)$ and a positive integer $K$.
**Output.** yes iff there exists $V_1 \subseteq V$ such that $|V_1| \leq K$ and for each $(u, v) \in E$, $u$ or $v$ is in $V_1$ (might also be both)

**Example.** $G = (V, E)$, where $V = \{x_1, x_2, x_3, x_4\}$,

$E = \{(x_1, x_3), (x_1, x_4), (x_3, x_4), (x_4, x_2)\}$ and $K = 2$.

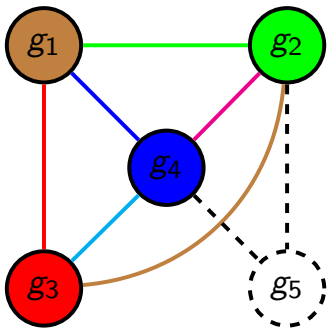


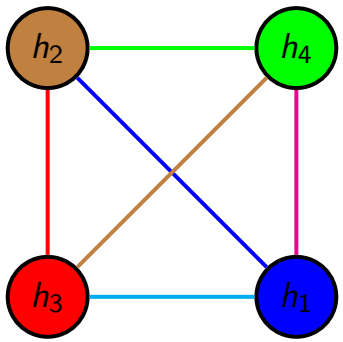

$V = \{x_1, x_4\}$

# SubGraph Isomorphism

**Input.** Two (undirected) graphs $G = (V_G, E_G)$ and $H = (V_H, E_H)$
**Output.** yes iff there exists a graph $I = (V_I, E_I)$ and an injection $f : V_H \to V_I$ such that $I$ is a subgraph of $G$ (i.e., $V_I \subseteq V_G$ and $E_I \subseteq E_G$), $|V_I| = |V_H|$, $|E_I| = |E_H|$, $(f(v), f(u)) \in E_I$ iff $(u, v) \in E_H$.

**Example.**



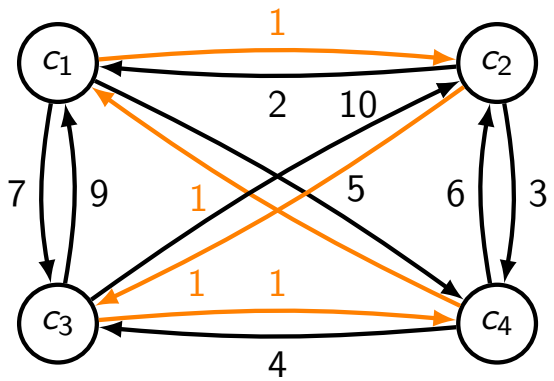




$f(h_1) = g_4$, $f(h_2) = g_3$, $f(h_3) = g_2$, $f(h_4) = g_1$

# Traveling Salesman Problem

**Input.** A set $C$ of $N$ cities, a positive integer $B$ and a distance matrix $D$ $N \times N$ where each $D[i,j]$ is a postive integer containing the distance to go from city $i$ to city $j$.
**Output.** yes iff there exists an injection $f : \{1,..,N\} \to C$ such that $(\sum_{i=1}^{N-1} D[f(i), f(i+1)]) + D[f(N), f(1)]) \leq B$

**Example.** $C = \{c_1, c_2, c_3, c_4\}$, $B = 5$ and $D$ (see below).




$$D = \begin{bmatrix} 0 & 1 & 7 & 5 \\ 2 & 0 & 1 & 3 \\ 9 & 10 & 0 & 1 \\ 1 & 6 & 4 & 0 \end{bmatrix}$$

$f(1) = c_2,\ f(2) = c_3,\ f(3) = c_4,\ f(4) = c_1$
$(c_2 \to c_3 \to c_4 \to c_1 \to c_2)$