






# CSPs: definitions

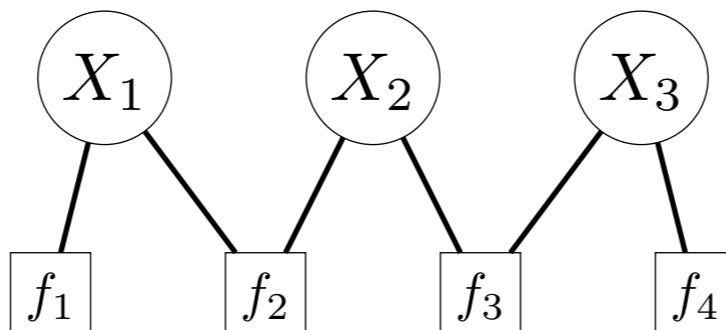
A 9x9 grid representing a CSP problem, drawn on a chalkboard background. The grid contains numbers in some cells, representing a partially filled-in state. The numbers are:

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

- In this module, I will formally define constraint satisfaction problems as well as the more general notion of a factor graph.

# Factor graph example: voting

*definitely blue*  B or R?    
 *must agree*  B or R?    
 *tend to agree*  B or R?    
 *leaning red*



$x_1$	$f_1(x_1)$
R	0
B	1

$x_1$	$x_2$	$f_2(x_1, x_2)$
R	R	1
R	B	0
B	R	0
B	B	1

$x_2$	$x_3$	$f_3(x_2, x_3)$
R	R	3
R	B	2
B	R	2
B	B	3

$x_3$	$f_4(x_3)$
R	2
B	1

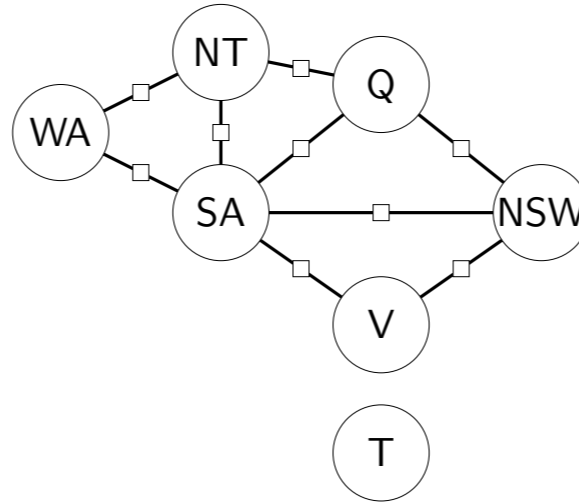
$$f_1(x_1) = [x_1 = \text{B}] \quad f_2(x_1, x_2) = [x_1 = x_2] \quad f_3(x_2, x_3) = [x_2 = x_3] + 2 \quad f_4(x_3) = [x_3 = \text{R}] + 1$$

[demo]

- Let us provide an example of a factor graph.
- Suppose there are three people, each of which will vote for a color, red or blue. We know that Person 1 is dead set on blue, while Person 3 is leaning red. Person 1 and Person 2 are close friends and must vote on the same color, while Person 2 and Person 3 are acquaintances who only weakly prefer to have the same color. The question is how each person will vote given their influences on each other?
- We can model this situation as a factor graph consisting of three **variables**,  $X_1, X_2, X_3$ , each of which must be assigned red (**R**) or blue (**B**).
- We encode each of the constraints/preferences as a **factor**, which assigns a non-negative number based on the assignment to a subset of the variables.
- We can either describe the factor as an explicit table, or via a function (e.g.,  $[x_1 = x_2]$ ).
- Notation: we use  $[condition]$  to represent the indicator function which is equal to 1 if the condition is true and 0 if not. Normally, this is written  $1[condition]$ , but we drop the **1** for succinctness.



## Example: map coloring



Variables:

$$X = (\text{WA}, \text{NT}, \text{SA}, \text{Q}, \text{NSW}, \text{V}, \text{T})$$

$$\text{Domain}_i \in \{\text{R}, \text{G}, \text{B}\}$$

Factors:

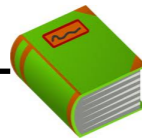
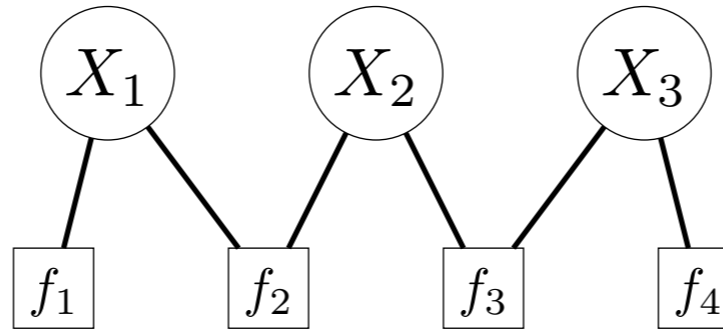
$$f_1(X) = [\text{WA} \neq \text{NT}]$$

$$f_2(X) = [\text{NT} \neq \text{Q}]$$

...

- Let's revisit the map coloring example.
- For each province, we have a variable, whose domain is the three colors.
- We have one factor for each pair of neighboring provinces which returns 1 (okay) if the two colors are not equal and 0 otherwise.

# Factor graph



## Definition: factor graph

Variables:

$$X = (X_1, \dots, X_n), \text{ where } X_i \in \text{Domain}_i$$

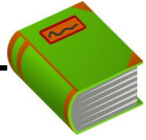
Factors:

$$f_1, \dots, f_m, \text{ with each } f_j(X) \geq 0$$

- Now we proceed to the general definition. A factor graph consists of a set of variables and a set of factors: (i)  $n$  variables  $X_1, \dots, X_n$ , which are represented as circular nodes in the graphical notation; and (ii)  $m$  factors (also known as potentials)  $f_1, \dots, f_m$ , which are represented as square nodes in the graphical notation.
- Each variable  $X_i$  can take on values in its **domain**  $\text{Domain}_i$ . Each factor  $f_j$  is a function that takes an assignment  $x$  to all the variables and returns a non-negative number representing how good that assignment is (from the factor's point of view). Usually, each factor will depend only on a small subset of the variables.



# Factors



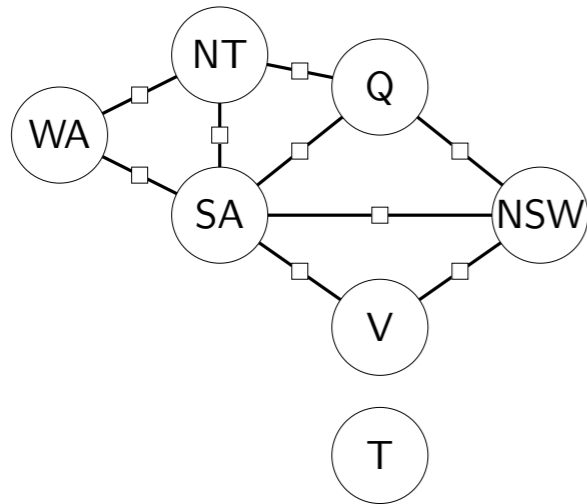
## Definition: scope and arity

**Scope** of a factor  $f_j$ : set of variables it depends on.

**Arity** of  $f_j$  is the number of variables in the scope.

**Unary** factors (arity 1); **Binary** factors (arity 2).

**Constraints** are factors that return 0 or 1.



## Example: map coloring

Scope of  $f_1(X) = [WA \neq NT]$  is  $\{WA, NT\}$

$f_1$  is a binary constraint

- The key aspect that makes factor graphs useful is that each factor  $f_j$  only depends on a subset of variables, called the **scope**.
- The arity of the factors is generally small (think 1 or 2).
- Factors that return 0 or 1 are called constraints. A constraint is satisfied iff a constraint returns 1.

# Assignment weights example: voting

$x_1$	$f_1(x_1)$
R	0
B	1

$x_1$	$x_2$	$f_2(x_1, x_2)$
R	R	1
R	B	0
B	R	0
B	B	1

$x_2$	$x_3$	$f_3(x_2, x_3)$
R	R	3
R	B	2
B	R	2
B	B	3

$x_3$	$f_4(x_3)$
R	2
B	1

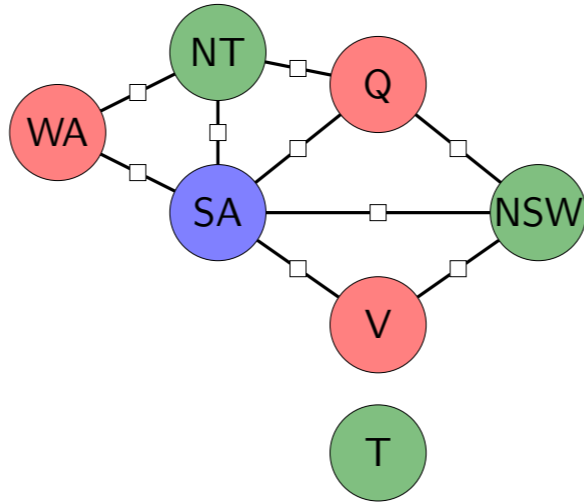
$x_1$	$x_2$	$x_3$	Weight
R	R	R	$0 \cdot 1 \cdot 3 \cdot 2 = 0$
R	R	B	$0 \cdot 1 \cdot 2 \cdot 1 = 0$
R	B	R	$0 \cdot 0 \cdot 2 \cdot 2 = 0$
R	B	B	$0 \cdot 0 \cdot 3 \cdot 1 = 0$
B	R	R	$1 \cdot 0 \cdot 3 \cdot 2 = 0$
B	R	B	$1 \cdot 0 \cdot 2 \cdot 1 = 0$
B	B	R	$1 \cdot 1 \cdot 2 \cdot 2 = 4$
B	B	B	$1 \cdot 1 \cdot 3 \cdot 1 = 3$

[demo]

- An **assignment** specifies a value for each variable, which is a candidate solution.
- Recall that the factors specify local interactions between variables.
- For each assignment, we get its weight, which is defined to be the product over each factor evaluated on that assignment.
- Each factor makes a contribution to the weight. Note that any factor has veto power: if it returns zero, then the weight of the entire assignment is irrecoverably zero.
- Think of all the factors chiming in on their opinion of  $x$ . We multiply all these opinions together to get the global opinion.
- In this setting, the maximum weight assignment is (B, B, R), which has a weight of 4. This is the assignment we wish to return.



## Example: map coloring



Assignment:

$$x = \{WA : R, NT : G, SA : B, Q : R, NSW : G, V : R, T : G\}$$

Weight:

$$\text{Weight}(x) = 1 \cdot 1 \cdot 1 \cdot 1 \cdot 1 \cdot 1 \cdot 1 \cdot 1 \cdot 1 = 1$$

Assignment:

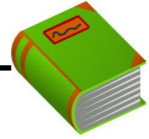
$$x' = \{WA : R, NT : R, SA : B, Q : R, NSW : G, V : R, T : G\}$$

Weight:

$$\text{Weight}(x') = 0 \cdot 0 \cdot 1 \cdot 1 \cdot 1 \cdot 1 \cdot 1 \cdot 1 \cdot 1 = 0$$

- Consider the map coloring example. Here we are writing an assignment as a dictionary from variable (name) to value.
- For the first assignment, all the constraints (factors) are satisfied and evaluates to 1.
- For the second assignment, WA and NT have the same color (red), so  $[WA \neq NT] = 0$ . This zeros out the weight for the entire assignment.

# Assignment weights



## Definition: assignment weight

Each **assignment**  $x = (x_1, \dots, x_n)$  has a **weight**:

$$\text{Weight}(x) = \prod_{j=1}^m f_j(x)$$

An assignment is **consistent** if  $\text{Weight}(x) > 0$ .

**Objective**: find the maximum weight assignment

$$\arg \max_x \text{Weight}(x)$$

A CSP is **satisfiable** if  $\max_x \text{Weight}(x) > 0$ .

- Formally, the **weight** of an assignment  $x$  is the product of all the factors applied to that assignment ( $\prod_{j=1}^m f_j(x)$ ). We say that an assignment is consistent if it has a non-zero weight.
- The objective in constraint satisfaction problem (what it means to solve a CSP) is to find the **maximum weight assignment**. A CSP is satisfiable if there exists a consistent assignment.
- Note: strictly speaking, a CSP only contains factors which are constraints (that return 0 or 1), but we consider a more general version of CSPs where weights can be arbitrary.
- Note: do not confuse the term "weight" in the context of factor graphs with the "weight vector" in machine learning.



# Constraint satisfaction problems

Boolean satisfiability (SAT):

variables are booleans, factors are logical formulas  $[X_1 \vee \neg X_2 \vee X_5]$

Linear programming (LP):

variables are reals, factors are linear inequalities  $[X_2 + 3X_5 \leq 1]$

Integer linear programming (ILP):

variables are integers, factors are linear inequalities

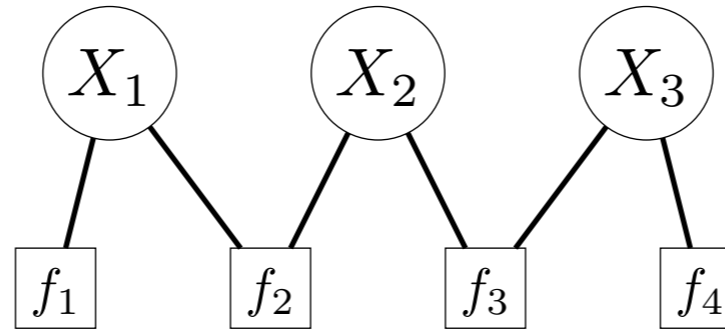
Mixed integer programming (MIP):

variables are reals and integers, factors are linear inequalities

- Constraint satisfaction problems are a general umbrella term that captures several important special cases, which are widely studied in the mathematical programming community.
- In SAT, all variables are boolean-valued and factors (constraints) are logical formulas. The goal is just to find any consistent assignment. While SAT is NP-complete, there has been extraordinary progress in SAT solving, and we can routinely solve SAT instances much larger than theory would predict.
- In linear programming, the variables are real-valued, and factors are linear inequalities. These problems can be solved efficiently using specialized methods (e.g., the simplex algorithm)
- ILPs and MIPs are hard to solve in general because they include integer values.



# Summary



Variables, factors: specify locally

$$\text{Weight}(\{X_1 : \mathbf{B}, X_2 : \mathbf{B}, X_3 : \mathbf{R}\}) = 1 \cdot 1 \cdot 2 \cdot 2 = 4$$

Assignments, weights: optimize globally

- In summary, we have formally defined factor graphs, where variables represent unknown quantities, and factors specify preferences for partial assignments. These allow us to specify preferences in a modular way: just "throw in" any desiderata you have.
- The weight of an assignment is the product of all the factors. The objective in solving a CSP is to find the maximum weight assignment, which is a global notion that must take into account all the factors at once.