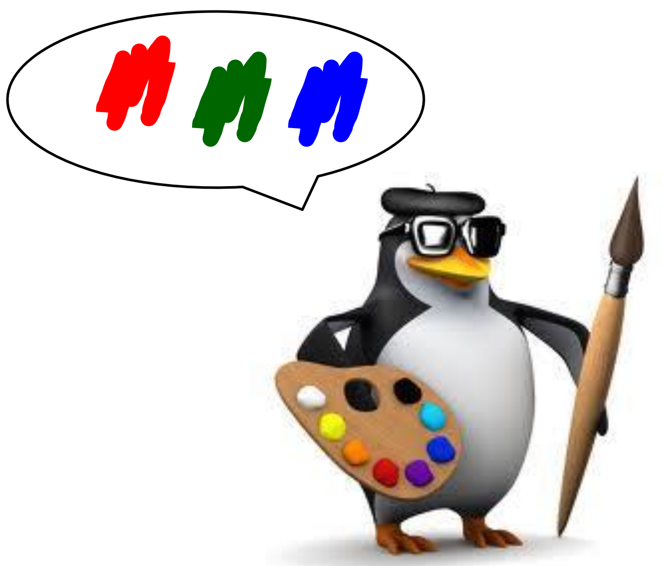
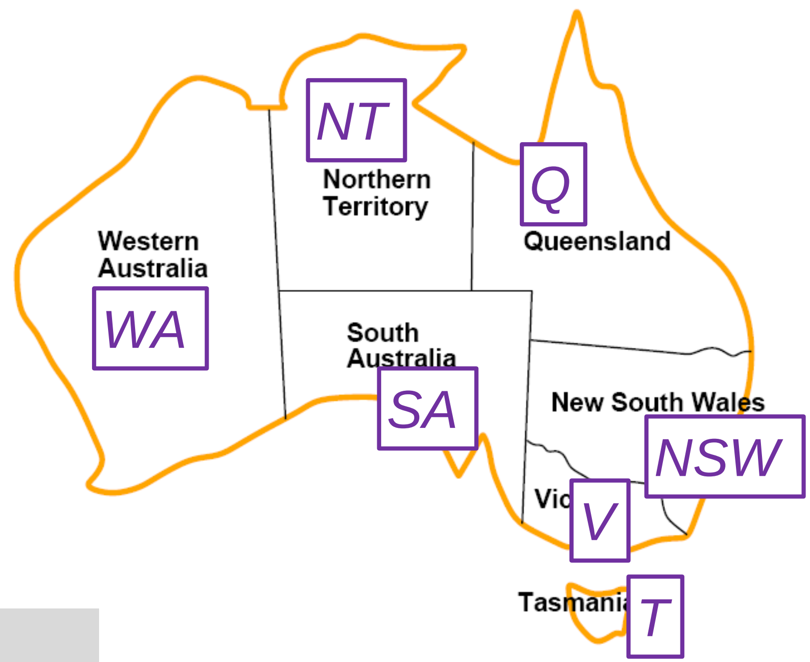
Constraint Satisfaction Problems (CSP)

Example: Map-Coloring Problems

Goal: To color the provinces of Australia as red, green, or blue, with no adjacent provinces having the same color.





Variables: *WA, NT, Q, NSW, V, SA, T*

Domains: *{red, green, blue}*

Constraints: Adjacent regions must have different colors: *WA≠NT*, etc.

Example: Cryptarithmetic Puzzles

Goal: To assign (all different) digits to the symbols.

$$\begin{array}{cccc} & T & W & O \\ + & T & W & O \\ \hline F & O & U & R \end{array}$$

Variables: $F, T, U, W, R, O, C_1, C_2, C_3$

Domains: $\{0,1,2,3,4,5,6,7,8,9\}$ for F, T, U, W, R, O

$\{0,1\}$ for C_1, C_2, C_3

Constraints: $F > 0$

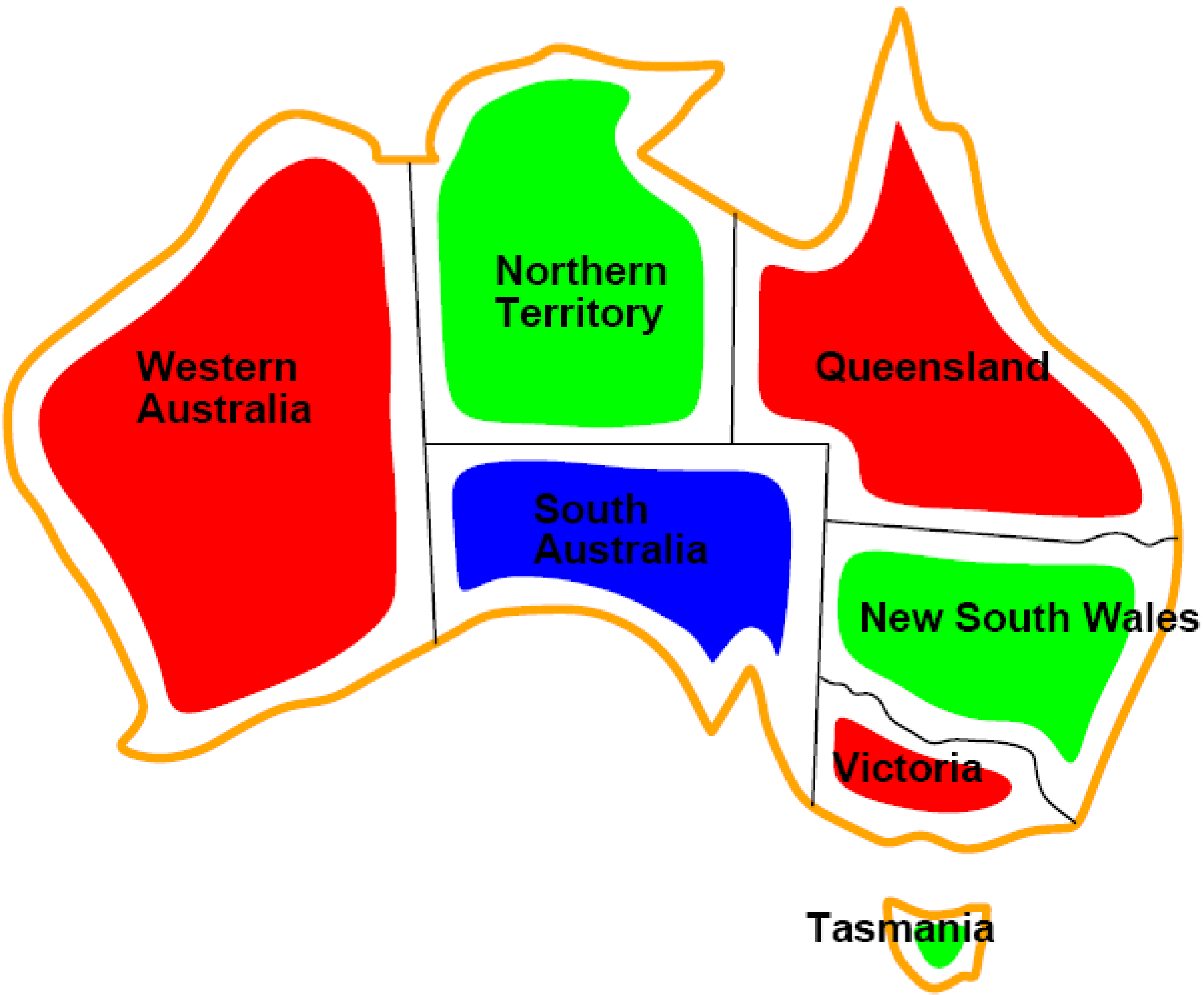
alldiff(F, T, U, W, R, O)

$O + O = R + 10^* C_1$, etc.

Constraint Satisfaction Problems

- Information about the state space:
  - Variables: X_i
  - Domains (allowed values for the variables): D_i (one for each variable)
- Goal test: Whether the variable assignments satisfy the given set of constraints C.
- Solution: Variable assignments that satisfy all the constraints.
- Search algorithms can be used to find solutions. (Only the final state, not the path, is needed).

Example Solution for Map-Coloring



Varieties of CSPs

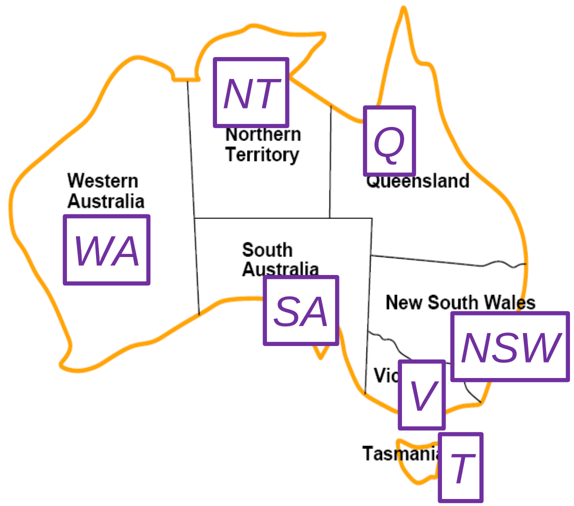
- Discrete variables with finite domains (covered here):
  - n variables, domain size = d: $O(d^n)$ complete assignments
- Discrete variables with infinite domains:
  - Example: job scheduling with the variables being the start/end days for the jobs.
  - Such CSPs with linear constraints are solvable. For such CSPs with nonlinear constraints, the problem is undecidable.
- Continuous variables:
  - Example: job scheduling with the variable being the start/end times for the jobs.
  - Such CSPs with linear constraints are solvable in polynomial time by linear programming methods.

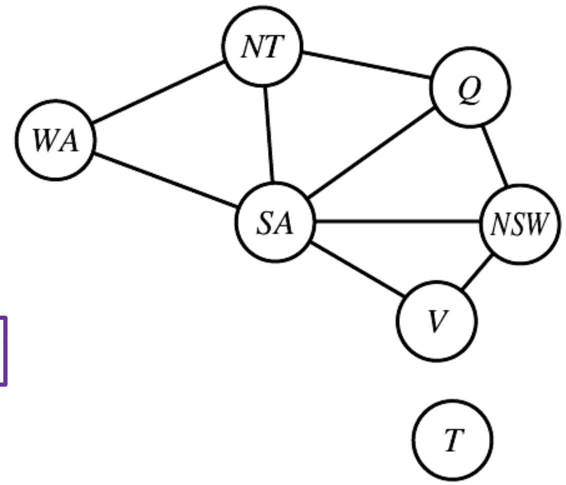
Types of Constraints

- Unary constraints: involving a single variable
  - e.g., $SA \neq green$
- Binary constraints: involving pairs of variables
  - e.g., $SA \neq WA$
- Higher-order constraints: involving 3 or more variables
  - e.g., cryptarithmetic column constraints
- Global constraints: involve an arbitrary number of variables
  - e.g., *alldiff*, *atmost*, etc.
  - Some are better solved with specialized methods.
- Preferences (soft constraints)
  - e.g., *SA* likes *red* more than *green*
  - often represented by a cost for each variable assignment ➔ optimization problems

Constraint Graph

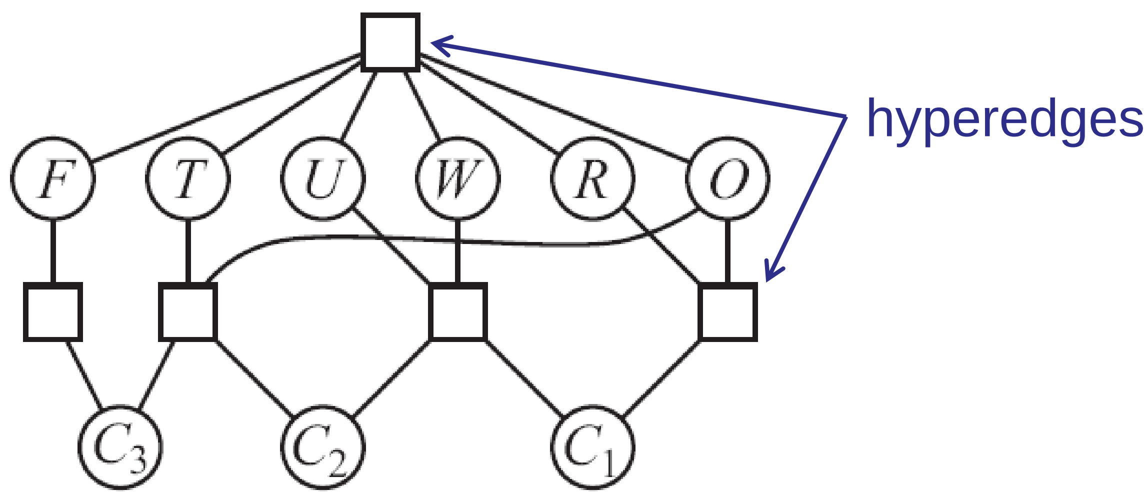
- Binary CSPs: Variables as vertices and edges as constraints





- Higher-order CSPs: Represented as **hypergraphs**

$$\begin{array}{cccc} & T & W & O \\ + & T & W & O \\ \hline F & O & U & R \end{array}$$



Constraint Propagation

Inference in CSP: Given the constraints and possible assigned values of some variables, determine the domains (allowed values) of the other variables.

- **Node consistency**: A variable is node-consistent if each possible value satisfies its unary constraints.
- **Arc consistency**: An arc ($X_i \rightarrow X_j$) is arc-consistent if each possible value of X_i leaves some valid values for X_j according to the binary constraints for $\{X_i, X_j\}$.
- **Path consistency**: A path ($X_i \rightarrow X_m \rightarrow X_j$) is path-consistent if each pair of possible values for X_i and X_j leaves some valid values for X_m according to the binary constraints for $\{X_i, X_m\}$ and $\{X_j, X_m\}$.

AC-3 Algorithm

- Goal: To make every arc arc-consistent. (It attempts to reduce the domains by removing values that cause violation of arc-consistency.)
- Initialization: A set comtaining every arc in a CSP.
- In each step, an arc ($X_i \rightarrow X_j$) is popped off the set for consideration.
  - Make ($X_i \rightarrow X_j$) arc-consistent by removing from D_i those values that would leave no valid value for X_j according to the binary constraints for $\{X_i, X_j\}$.
  - If D_i is changed, then for each neighbor X_k of X_i, add the arc ($X_k \rightarrow X_i$) to the set.
  - If D_i becomes empty, terminate the processing; there is no solution.

AC-3 Example

Example: X, Y are digits, and $Y = X^2$.

- Initial domains:
  - $D_X = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
  - $D_Y = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
- After checking the arc ($X \rightarrow Y$):
  - $D_X = \{0, 1, 2, 3\}$
- Next, after checking the arc ($Y \rightarrow X$):
  - $D_Y = \{0, 1, 4, 9\}$
- No more change; the process terminates.

AC-3 Example (Sudoku)

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

Initially, all the unassigned blanks have domain {1,2,…,9}.

An example problem:

- 81 variables
- 32 unary constraints
- Q: How many binary constraints?

AC-3 Example (Sudoku)

						5		
3		2		7		9	1	
6			9					
							2	6
	2		3			1	5	9
7	9		6		5		8	
1	6	9	7			48	4	458
4	5	67				2	3	178
2	3	8	4	5		6	79	127

Some updated domains after checking arcs involving already assigned blanks.

						5		
3		2		7		9	1	
6			9					
							2	6
	2		3			1	5	9
7	9		6		5		8	
1	6	9	7			48	4	458
4	5	67				2	3	178
2	3	8	4	5		6	79	127

Further reduction of some domains.

AC-3 Example (Sudoku)

						5		
3		2		7		9	1	
6			9					
							2	6
	2		3			1	5	9
7	9		6		5		8	
1	6	9	7			48	4	458
4	5	67				2	3	178
2	3	8	4	5		6	79	127

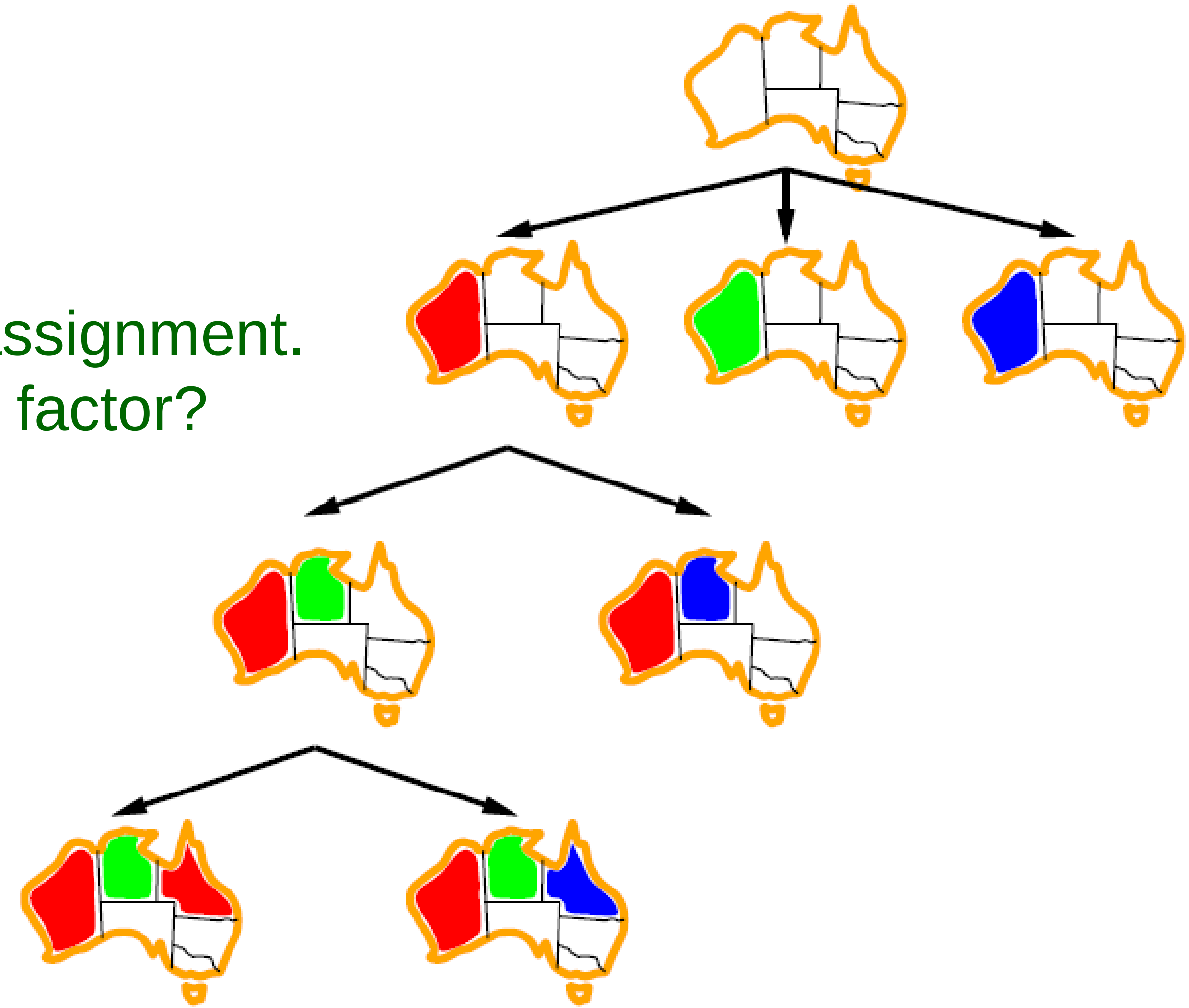
						5		
3		2		7		9	1	
6			9					
							2	6
	2		3			1	5	9
7	9		6		5		8	
1	6	9	7			48	4	458
4	5	67				2	3	178
2	3	8	4	5		6	79	127

AC-3 can occasionally solve a problem by itself.

Solving CSPs by Searching

Backtracking search: depth-first, incremental formulation (assigning one variable in each step).

Example:



Q: Consider the first assignment. What is the branching factor?

Backtracking Search

General-purpose methods can give huge gains in speed:

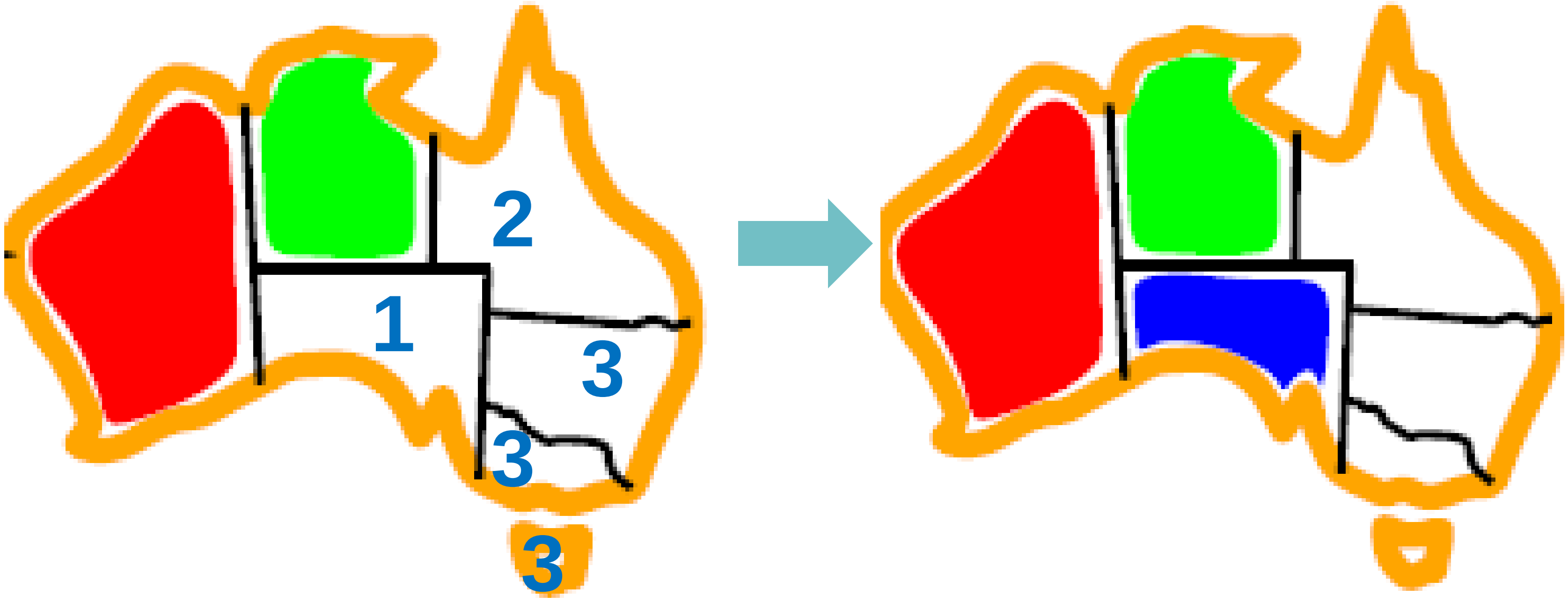
- Which variable should be assigned next?
  - **Minimum remaining values (MRV) heuristic**
  - **Degree heuristic**
- In what order should its values be tried?
  - **Least constraining value (LCV) heuristic**
- Can we detect an inevitable failure early?
- Can we take advantage of the problem structure?

Usually applied in the order of MRV→degree→LCV.

Minimum Remaining Values (MRV)

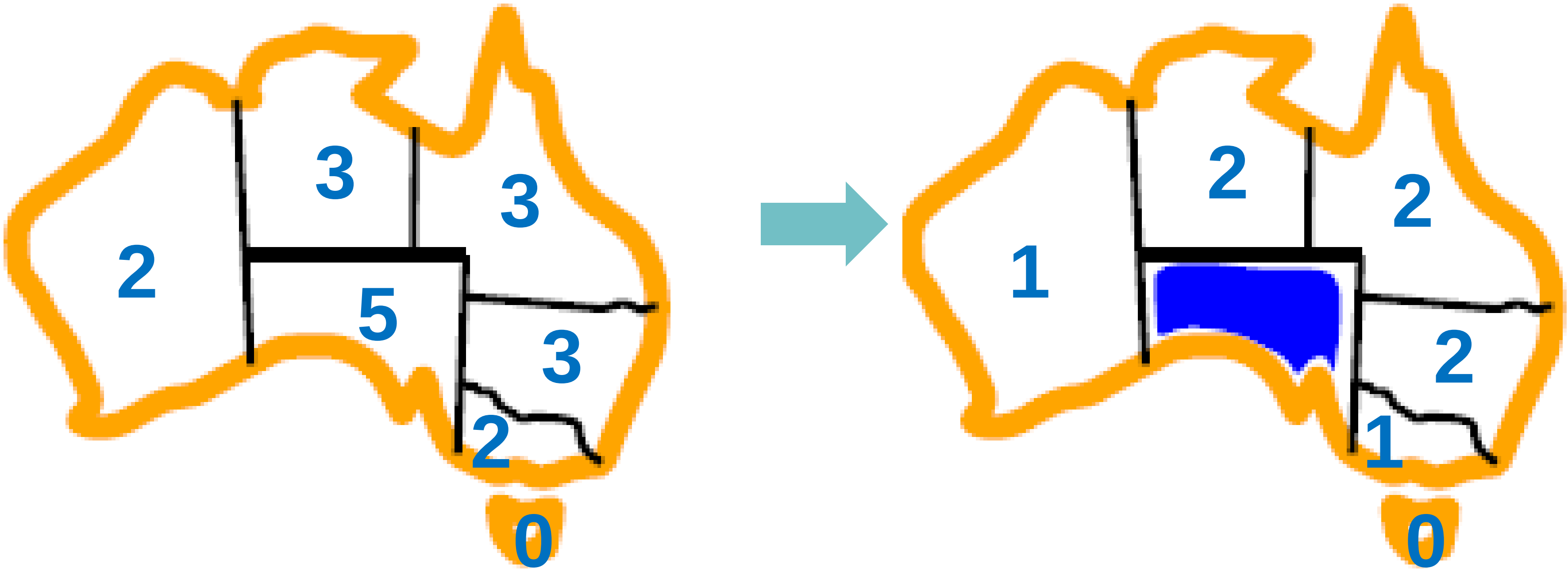
Idea: Choose the variable with the fewest legal values

Example: Which region to color next?



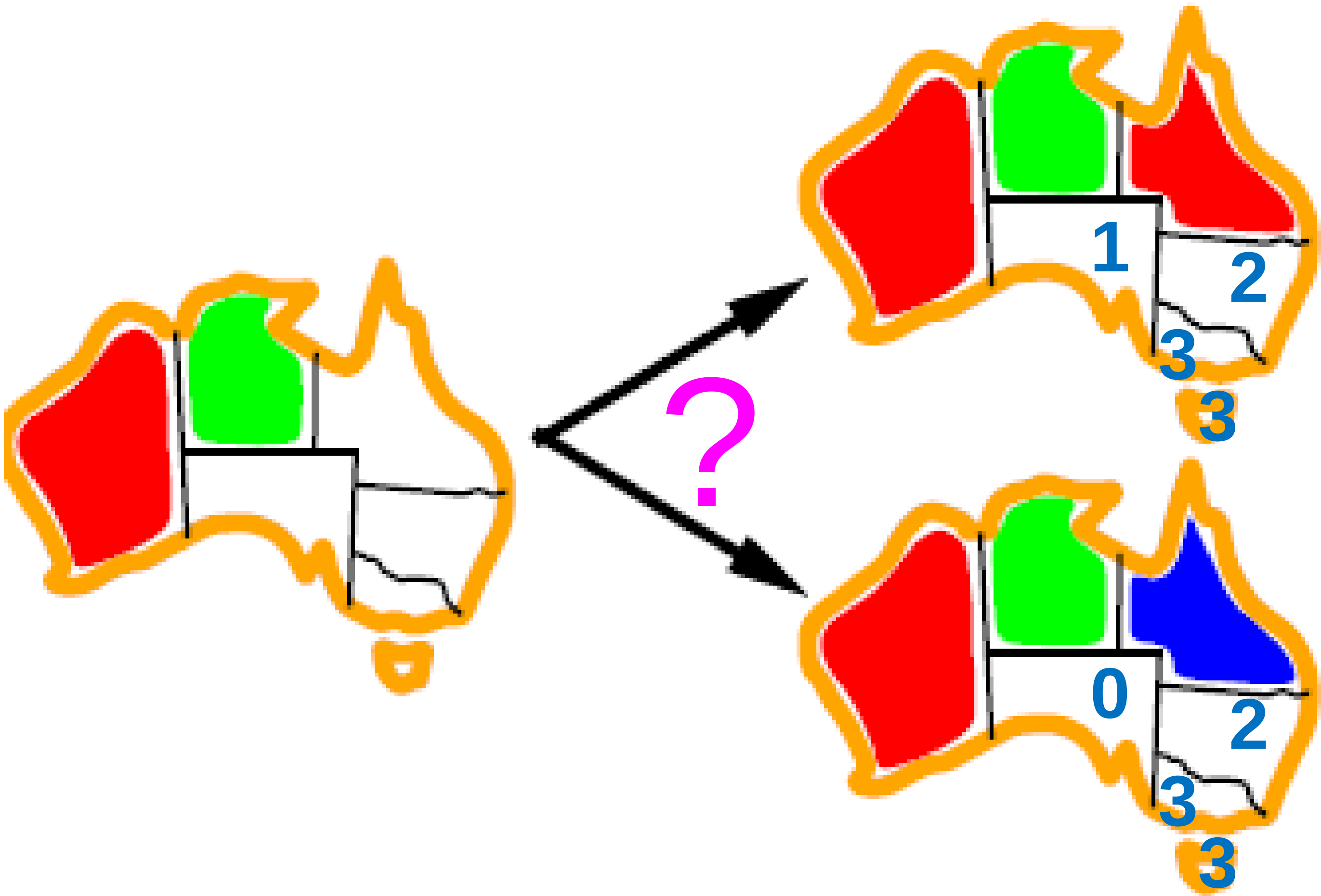
Degree Heuristic

Idea: Choose the variable with the most constraints on the remaining variables



Least Constraining Value (LCV)

Given a variable, choose the least constraining value: the one that rules out the fewest values in the remaining variables.



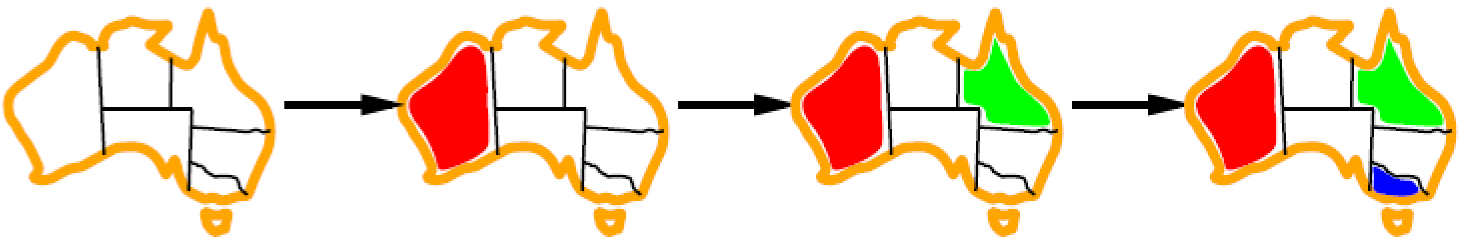
Interleaving Searching and Inference

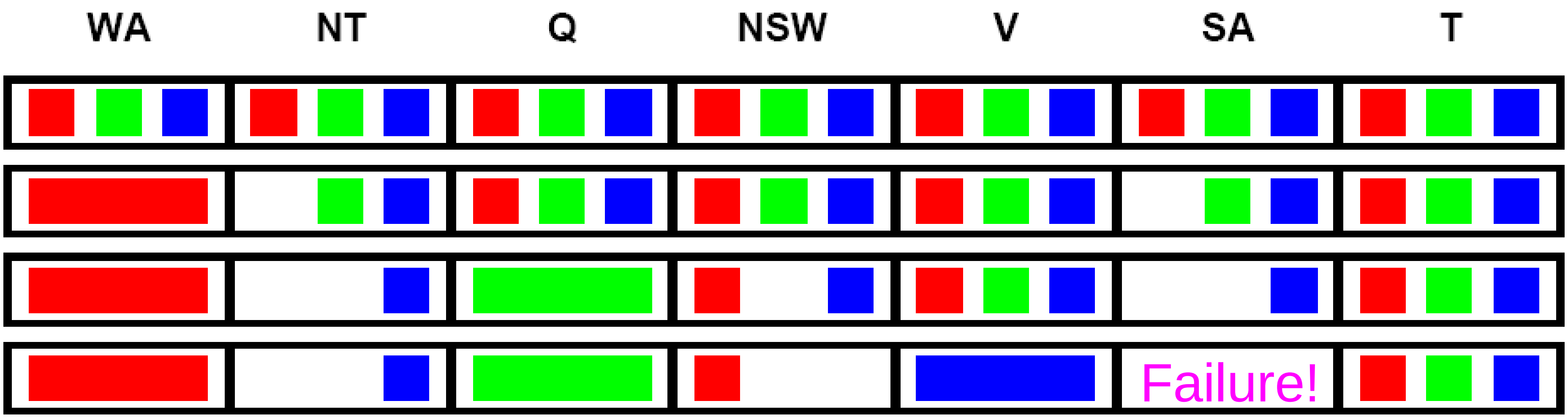
Standard backtracking detects a failure only when it is assigning a variable with an empty domain.

Q: Can we detect an inevitable failure earlier?

- **Forward checking**:
  - Idea: After each assignment, update the domains of all the neighbors of the assigned variable.
  - If any domain becomes empty, then backtrack.
- **Maintaining arc consistency**:
  - Detects more upcoming failures than forward checking.
  - A common method is to run **AC-3** (only updating the domains of unassigned variables) after each assignment.

Forward Checking Example





Min-Conflicts Local Search for CSPs

- Start from a complete configuration (each variable assigned by randomly selecting a value from its domain).
- Iteratively select a variable (randomly), and reassign its value to minimize conflicts.
- Example (4 queens):

1			
2	**Q**	**Q**	
2			**Q**
1			

Q	3		
	2	**Q**	
	2		**Q**
	0		

Q		1	
		1	
		2	**Q**
	Q	1	

Q		**Q**	2
			2
			0
	Q		1

1		**Q**	
0			
1			**Q**
1	**Q**		

		Q	
Q			
			Q
	Q		

- This can solve million-queen problems!