ARTIFICIAL INTELLIGENCE
&
MACHINE LEARNING

Constrained Satisfaction

SESSION NO :9&10

Constraint Satisfaction Problems (CSPs)

INTRODUCTION

What is a CSP?

- A **Constraint Satisfaction Problem (CSP)** is a mathematical model used to represent problems where the goal is to assign values to a set of variables **while satisfying a set of constraints**.

- CSPs are widely used in **Artificial Intelligence**, **Operations Research**, and **Computer Science** to solve problems like scheduling, planning, and puzzle solving.

Components in the constraint satisfaction problem

There are mainly three basic components in the constraint satisfaction problem:

Variables: The things that need to be determined are variables. Variables in a CSP are the objects that must have values assigned to them in order to satisfy a particular set of constraints. Boolean, integer, and categorical variables are just a few examples of the various types of variables, for instance, could stand in for the many puzzle cells that need to be filled with numbers in a sudoku puzzle.

Domains: The range of potential values that a variable can have is represented by domains. Depending on the issue, a domain may be finite or limitless. For instance, in Sudoku, the set of numbers from 1 to 9 can serve as the domain of a variable representing a problem cell.

Constraints: The guidelines that control how variables relate to one another are known as constraints. Constraints in a CSP define the ranges of possible values for variables. Unary constraints, binary constraints, and higher-order constraints are only a few examples of the various sorts of constraints. For instance, in a sudoku problem, the restrictions might be that each row, column, and 3×3 box can only have one instance of each number from 1 to 9.

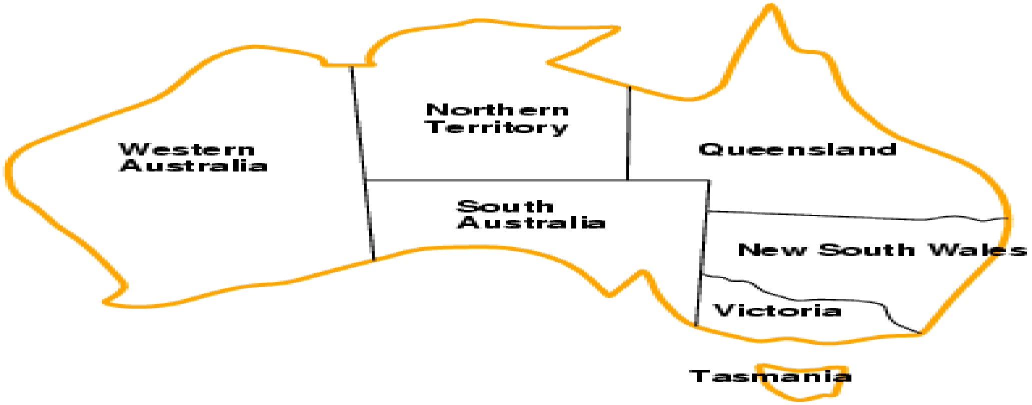
Introduction to
Constraint satisfaction problems (CSP)

- A problem described by imposing constraints on the variables related to a problem is called Constraint Satisfaction Problem (CSP).
- A constraint satisfaction problem consists of three components X, D, and C

  - X is a set of variables, $\{X_i, ., X_n,\}$.
  - D is a set of domains, $\{D_i, , D_n\}$ one for each variable.
  - C is a set of constraints that specify allowable combinations of values.
  - Each domain D_i consists of a set of allowable values {v1,v2,…vn} for variable X.
  - Each constraint C_i consists of a tuple of variables participating in the constraint.
  - A constraint is a relation defining the values that variables can take on.

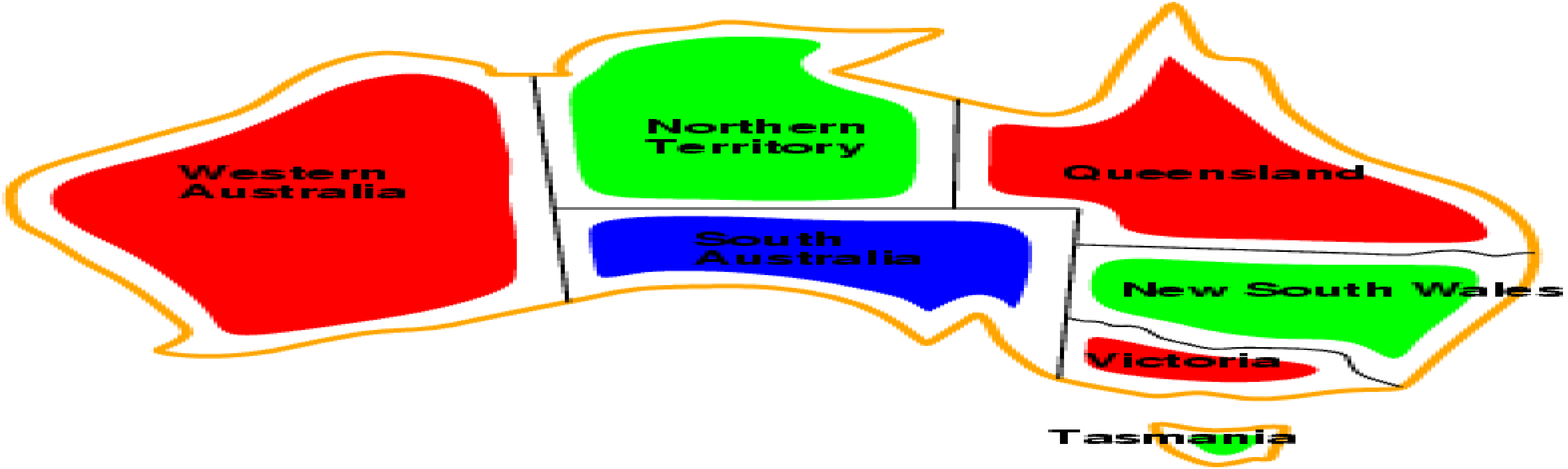
Example CSP: Map-Coloring PROBLEM

- Variables *WA, NT, Q, NSW, V, SA, T*
- Domains D_i = {red, green, blue}
- Constraints: adjacent regions must have different colors
- e.g., WA ≠ NT, or (WA, NT) in {(red, green),(red, blue),(green, red), (green, blue),(blue, red),(blue, green)}.



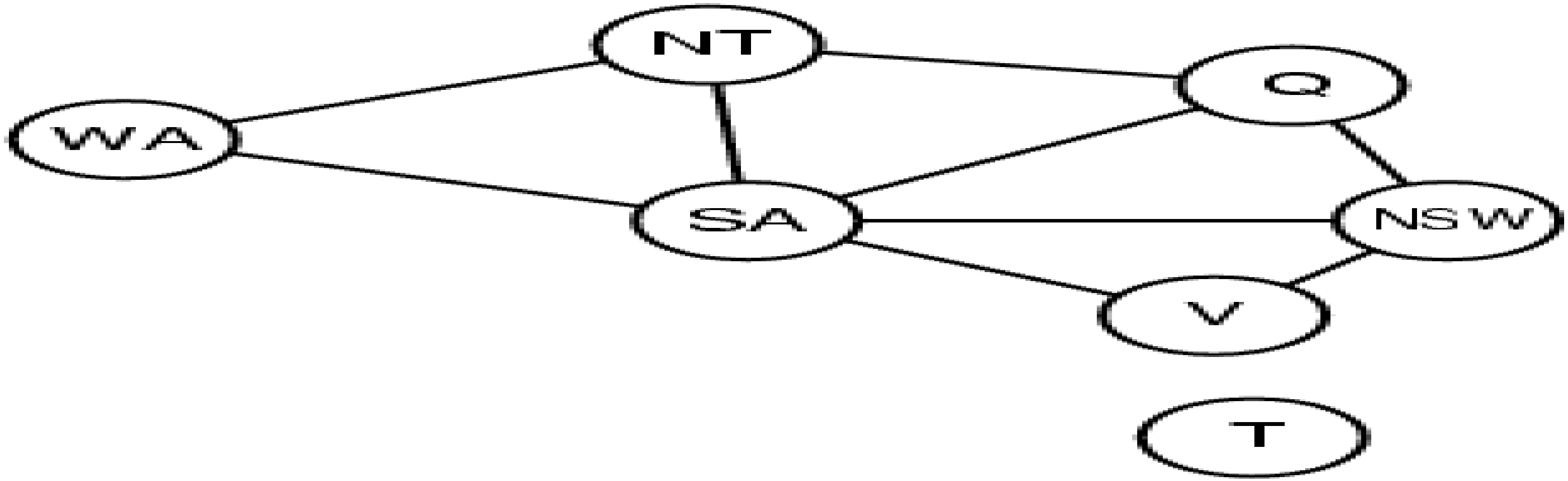
Example CSP: Map-Coloring PROBLEM

- Solutions are complete and consistent assignments (WA = red, NT = green, Q = red, NSW = green, V = red, SA = blue, T = green.



Representing Constraints as a graph

- **Constraint graph:** nodes are variables, arcs/edges are constraints



Types of Variable used in A CSP problem

- Discrete variables
- Continuous Variable

Discreet VARIABLES in CSP

- Discrete variables with Finite Domains (Color Graphing, 8 Queen)
  - If the domain size of any variable is d, then the possible number of complete assignments will be $O(d^n)$, where n is the number of variables
  - Finite domains include Boolean values

- Discrete Variables with Infinite Domains
  - Infinite domains are represented in terms of Integers and Strings
  - The number of values that can be assigned to a variable could be infinite
  - It is not possible to define constraints by considering all possible combinations of the values

Continuous variables in CSP

- Continuous Variables involve continuous domain
  - which frequently appears in many types of problems, especially in the field of operations research
  - Linear programming problems fall under the category of Continuous domain where the **constraints** are expressed using linear in-equalities

Language of constraints

- A constraint can be defined using a language must be used to define the constraints

Example: The job1, which can be taken after 5 days, must precede Job3, which can be represented using constraint language such as

$$StartJob_1 + 5 <= job3$$

Types of Constraints– CSP problem

- Linear Constraints
- Nonlinear Constraints
- Absolute Constraints
- Preference constraints

Types of constraints

- **Linear Constraints:** Linear combination of Variables expressed in Mathematical expressions (Linear Equations, Linear Inequalities, Logical Expressions)

- **Non-Linear Constraints** (Polynomial, Exponential, Trigonometric and Logarithmic)

- **Absolute Constraints:** Constraints are imposed on the variables using either absolute values or inequalities

- **Preference Constraints:** Constrained imposed based on the user preferences (Ex. In a university timetabling problem, Prof. X might prefer teaching in the morning, whereas Prof. Y prefers teaching in the afternoon).

Classification of constraints

- Unary constraints involve a single variable(SA ≠ green)
- Binary constraints involve pairs of variables (SA ≠ WA)
- Higher-order constraints involve 3 or more variables (crypt arithmetic column constraints

Example Application implemented using CSP

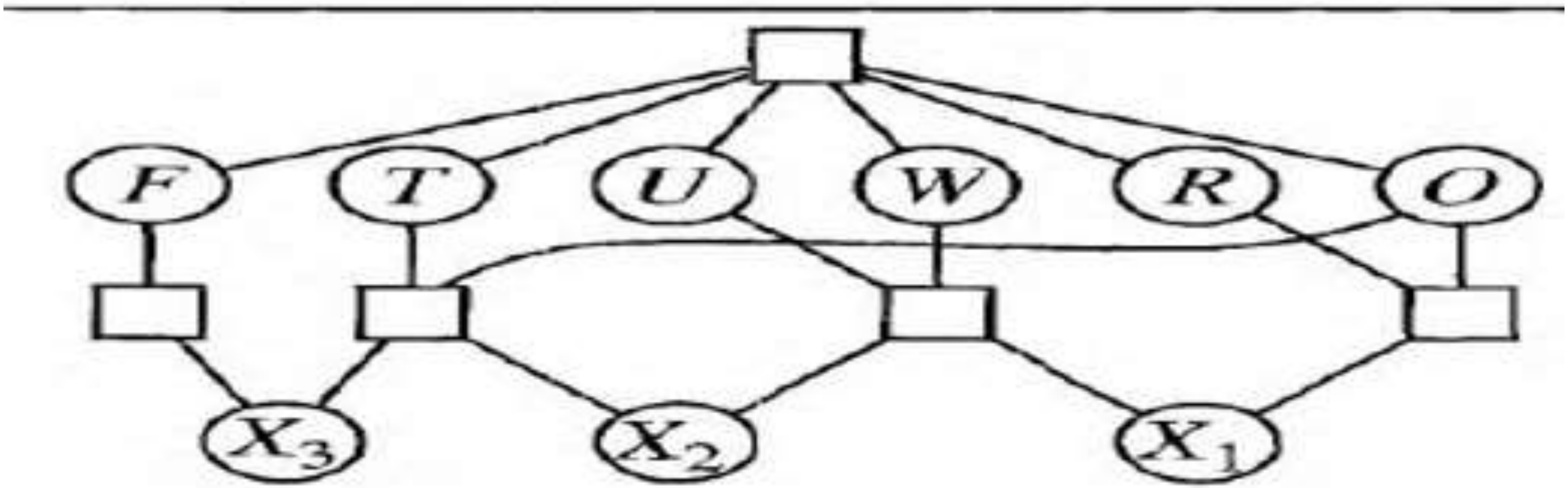
- Solve cryptographic arithmetic puzzles using Constraint Satisfaction Problem (CSP) techniques.
- A classic example is the SEND + MORE = MONEY puzzle, where each letter represents a unique digit (0-9).
- Solve this puzzle using CSP concepts.

Cryptographic Arithmetic Problem

Each Letter stands for a distinct digit. The aim is to find a substitution of digits for letters such that the resulting sum is arithmetically correct with the added restriction that no leading zeros are allowed

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

Constraint GRAPH
Cryptographic Arithmetic Problem



Identifying Constraints
Cryptographic Arithmetic Problem

The constraint Graph shows all diff constraints as well as column addition constraints. Each Square Box is a Constraint connected to the variables that it constrains.

Variables F, T, U, W, R, O

Constraints

$$O + O = R + 10 \cdot X_1$$
$$X_1 + W + W = U + 10 \cdot X_2$$
$$X_2 + T + T = O + 10 \cdot X_3$$
$$X_3 = F$$

Addition Constraints: F ≠ U and –F ≠ T

Real-world Applications requiring CSP Searches

- Assignment problems (e.g., who teaches what class)
- Timetabling problems(e.g., which class is offered when and where?)
- Transportation Scheduling
- Factory Scheduling

Standard search formulation

States are defined by the values assigned.

- Initial state: the **empty assignment { }**
- **Successor function: assign a value to an unassigned variable that does not conflict with the current assignment. Fail if no legal assignments exist**
- **Goal Test: The current assignment is complete**

This is the same for all CSPs

1. Every solution appears at depth *n* with *n* variables→ use depth-first search
2. Path is irrelevant, so we can also use complete-state formulation
3. b = (n - l)d at depth l

Components of CSP based Search

- **Initial state:** the empty assignment in which all variables are unassigned.
- **Successor function:** a value can be assigned to any unassigned variable, if it does not conflict with a previously assigned value
- **Goal test:** The current assignment is complete.
- **Path cost:** A constant cost of 1 for every step.
- Some observations
  - The Search is complete
  - The solution Appears at a depth n where n is the number of variables existing in the problem
  - Depth First Search Algorithms are best suited for solving the CSP

CSP Search Algorithms

- ❑ **Forward Checking**
- ❑ **Backward Tracking**

Forward Checking Algorithm

1. Initialize:
    - Start with the initial assignment of variables and their domains.
    - Create an empty set to store the list of constrained variables.
2. Select Variable: Choose a variable to assign a value to. This can be based on heuristics like Minimum Remaining Values (MRV) or Degree Heuristic.
3. Select Value: Choose a value from the domain of the selected variable. This can be based on heuristics like Least Constraining Value (LCV).
4. Assign Value: Assign the selected value to the selected variable.
5. Update Domain: For each unassigned variable adjacent to the variable just assigned, remove the selected value from their domains.

Forward Checking ALgorithm

6. **Check Domain Emptiness:** **If any domain becomes empty, backtrack to the previous variable and try a different value.**
7. **Check for Solution:** **If all variables are assigned values and all constraints are satisfied, you have found a solution.**
8. **Recursive Step:** **If not all variables are assigned, recursively repeat steps 2-7 for the next variable.**
9. **Backtrack:** **If you reach a point where no value can be assigned to the current variable, backtrack to the previous variable and try a different value.**
10. **Return Solution or Failure:** **If a solution is found, return it. If you reach a point where all possible assignments have been tried and none succeed, report failure.**

Forward Checking ALgorithm

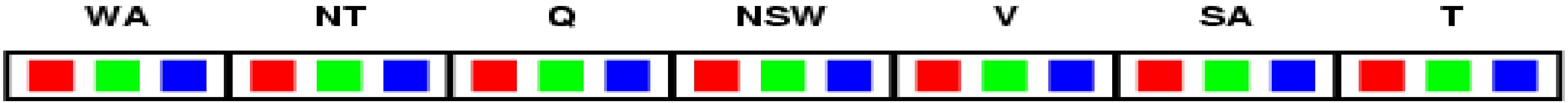
11. **Undo Assignments:** Before backtracking, undo the assignment of the current variable, and restore the domains of variables that were updated.

12. **Continue Search:** Continue the search process until a solution is found or all possible assignments have been tried.

Solving the CSP problems through Forward checking

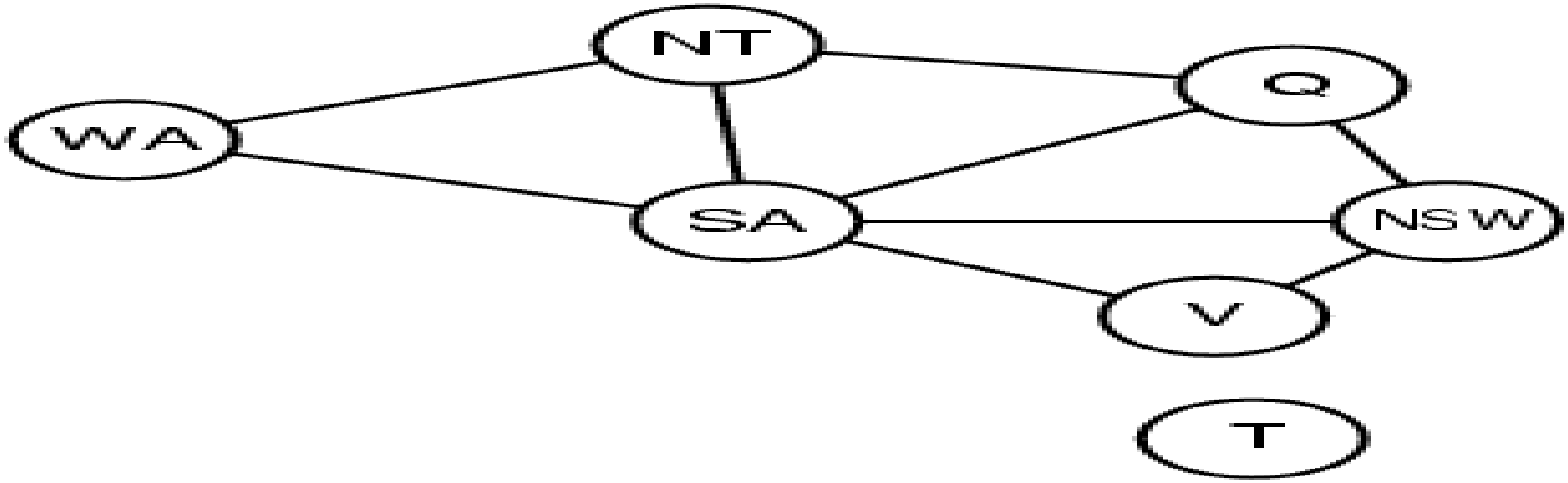
Idea

Keep track of remaining legal values for unassigned variables. Terminate the search when any variable has no legal values.



Representing Constraints as a graph

- **Constraint graph:** nodes are variables, arcs/edges are constraints

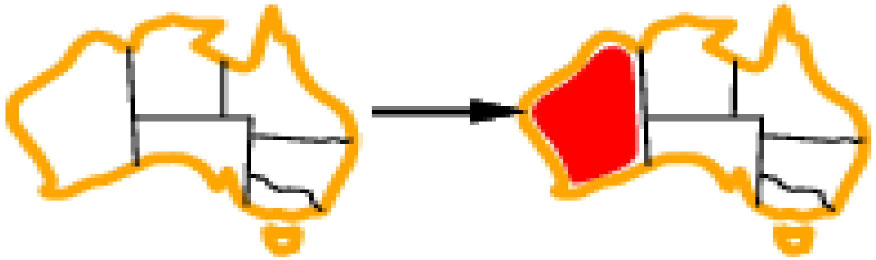


Solving the problems through Forward checking

Idea:

Keep track of remaining legal values for unassigned variables

Terminate search when any variable has no legal values

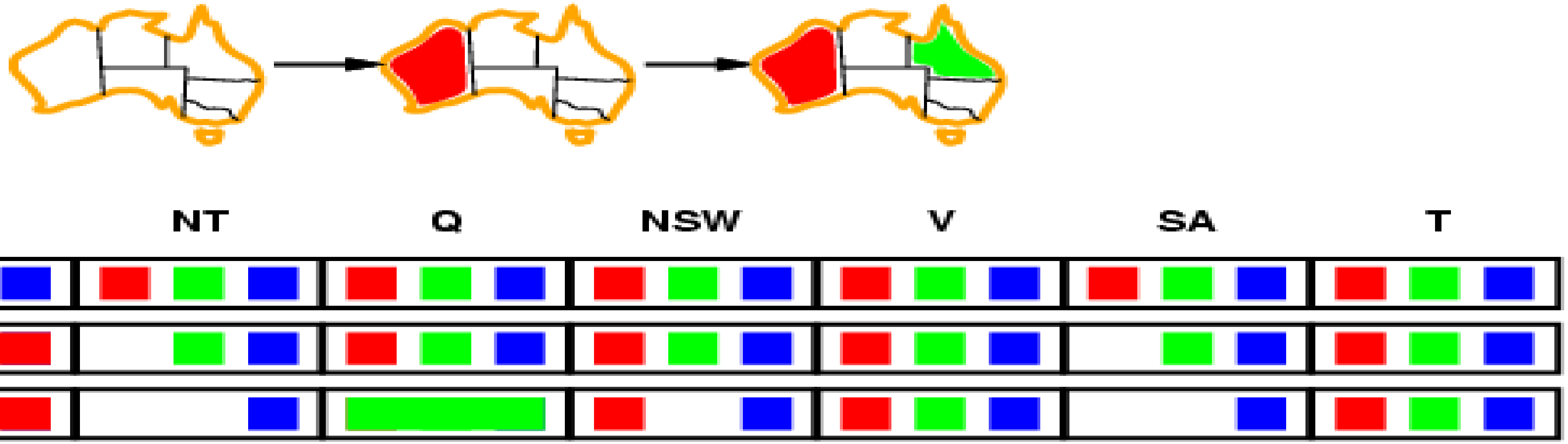


Solving the problems through Forward checking

Idea:

Keep track of remaining legal values for unassigned variables

Terminate search when any variable has no legal values

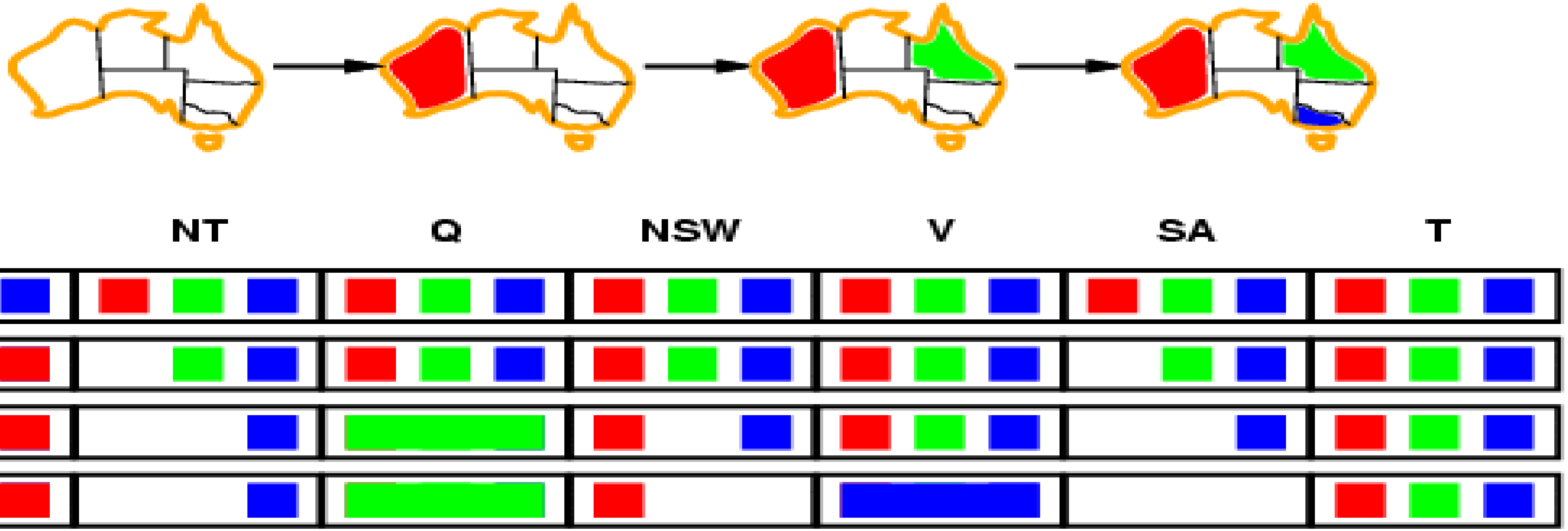


Solving the problems through Forward checking

Idea:

Keep track of remaining legal values for unassigned variables

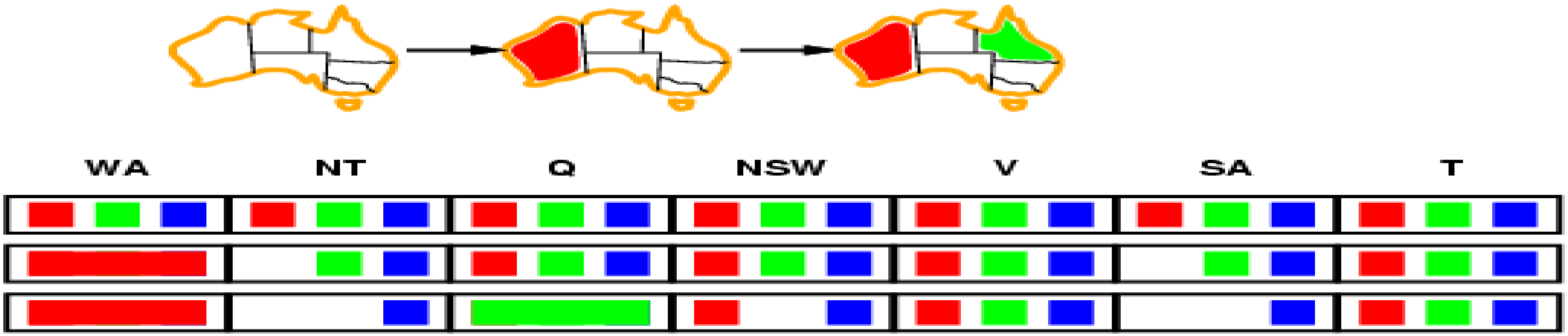
Terminate search when any variable has no legal values



Solving the problems through Forward checking

Idea:

- Forward checking propagates information from assigned to unassigned variables, but doesn't provide early detection for all failures:
- NT and SA cannot both be blue!
- Constraint propagation algorithms repeatedly enforce constraints locally



Constraint propagation

- In regular state-space search, an algorithm can do searching.
- In CSPs, an algorithm can choose a new variable value from several possibilities while searching
- In CSP, "constraint propagation" can also be made while doing the search.
- Constraint Propagation means using the constraints to reduce the number of legal values for a variable, which can reduce the legal values for another variable, and so on.
- Constraint propagation may be intertwined with the search, or it may be done as a preprocessing step, before the search starts
- Sometimes preprocessing can solve the whole problem itself without the necessity of employing the search.

Local Consistency

- **Local consistency is a property in Constraint Satisfaction Problems (CSPs) that characterizes the level of constraint propagation achieved during the search process.**
- **It refers to the extent to which the constraints in a CSP have been enforced among neighboring variables.**

Local consistency

- There are different types of local consistency.
    - Node consistency
    - Arc consistency
    - Path consistency
    - K-consistency

NODE consistency

- **This is the simplest form of consistency.**
- **It involves ensuring that each variable in the CSP satisfies its individual unary constraints (constraints that involve only one variable).**
- **For example, if a variable x has a unary constraint x > 3, then the domain of x would be reduced to values greater than 3.**

ARC consistency

- **Arc Consistency extends node consistency by considering binary constraints (constraints that involve two variables).**
- **It ensures that for every pair of variables (x, y) and for every value in the domain of x, there is at least one value in the domain of y that satisfies the binary constraint. If not, the inconsistent value is removed from the domain of x.**
- **AC-3 is a widely used algorithm for enforcing arc consistency.**

Path consistency

- **Path consistency is a stronger form of consistency that extends arc consistency.**
- **It enforces constraints over longer paths in the constraint graph.**
- **It checks for consistent values along chains of variables connected by binary constraints.**
- **This helps in propagating constraints more effectively.**

Backtracking in CSP

Introduction - Backtracking

- Backtracking is a fundamental algorithmic technique for solving Constraint Satisfaction Problems (CSPs).
- It systematically explores the search space of possible assignments to variables while using constraints to prune branches unlikely to lead to a solution.

Introduction - Backtracking

- The term backtracking search is used for a depth-first search that chooses values for one variable at a time and backtracks when a variable has no legal values left to assign.
- An unassigned variable is chosen, and then all values in that variable's domain are tried to find a solution.
- If an inconsistency is detected, then BACKTRACK returns failure, causing the previous call to try another value
- Backtracking considers assignments to a single variable at each node
- Depth-first search for CSPs with single-variable assignments is called backtracking search

Step by STEP procedure – Backtracking algorithm

1. Initialization: Start with an initial assignment of values to variables. If some variables are already assigned, this could be an empty or partial assignment.

2. Select Unassigned Variable: Choose an unassigned variable from the variables that still need to be assigned a value.

   The choice of a variable can be based on various heuristics, like the most constrained variable or the variable with the fewest legal values.

3. Order Domain Values: Order the selected variable’s domain values. The order can be based on heuristics, like the least constraining value that rules out the fewest choices for other variables.

Step by STEP procedure – Backtracking algorithm

4. Value Assignment:

For each value in the domain of the selected variable:

i. Assign the value to the variable.

ii. Check if the assignment violates any constraints with the already assigned variables. If a constraint is violated, backtrack to step 2.

5. Constraint Propagation:

After assigning a value, apply constraint propagation techniques. This could involve revising the domains of other variables based on the newly assigned variable and the constraints. Common techniques include arc consistency and domain reduction.

Step by STEP procedure – Backtracking algorithm

6. **Recursive Step or Backtrack:**

   a) If no inconsistency is found after assigning a value and applying constraint propagation, proceed recursively to the next variable and repeat steps 2 to 5.

   b) If a variable's domain becomes empty (no valid choices left) due to the assignment, backtrack to the previous variable and undo the assignment (backtrack step).

7. **Solution Found:** A solution has been found if all variables are assigned values and all constraints are satisfied. Return the assignment.

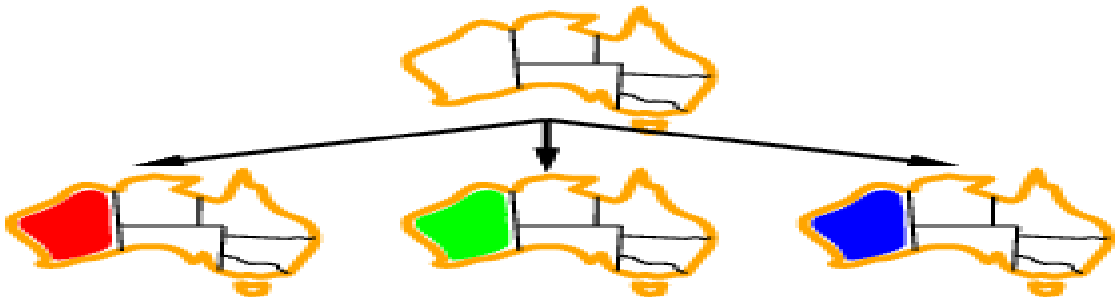
Step by STEP procedure – Backtracking algorithm

8. **Backtrack:** If the search reaches a dead-end (all values for a variable have been tried, and none lead to a solution), backtrack to the previous variable, undo its assignment, and continue the search from there.

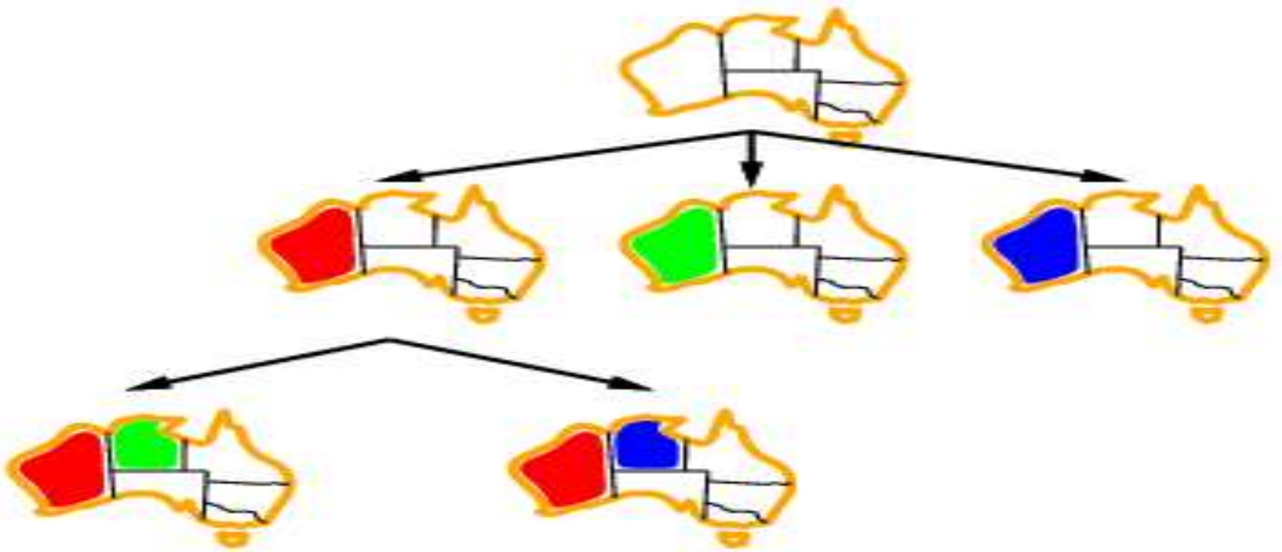
9. **Termination:** Continue until a solution is found or all possible assignments have been explored.

Backtracking example – Color Graph

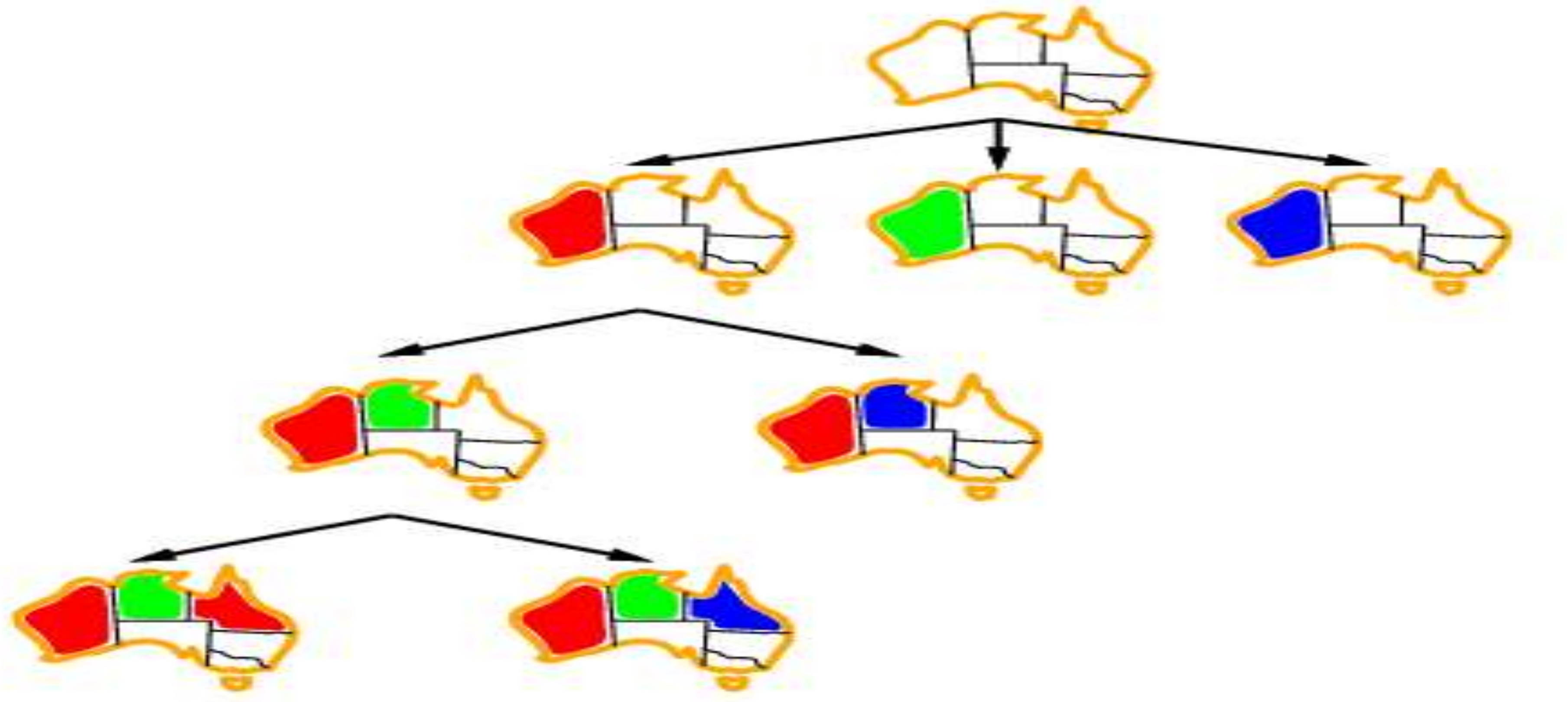
Backtracking example



Backtracking example



Backtracking example



Improving backtracking efficiency

- General-purpose methods can give huge gains in speed:
  - Which variable should be assigned next?
  - In what order should its values be tried?
  - Can we detect inevitable failure early?

Minimum Remaining Value Heuristics

- **By default, the SELECT-UNASSIGNED-VARIABLE function selects the next unassigned variable in the order given by the list**
- This static variable ordering seldom results in the most efficient search.
- For example, after the assignments for WA = *red* and *NT* = *green*, there is only one possible value for SA, so it makes sense to assign *blue to SA* next rather than to Q.
- In fact, after SA is assigned, the choices for Q, NSW, and V are all forced.
- This intuitive idea of choosing the variable with the fewest "legal" values is called the **remaining values (MRV)** heuristic.

Minimum Remaining Value

- It is also has been called the "most constrained variable" or "fail-first" heuristic because it picks a variable that is most likely to cause a failure soon, thereby pruning the search tree.

- If there is a variable X with zero legal values remaining, the MRV heuristic will select X, and failure will be detected immediately.

Variables ORDERING

- **Variable Ordering** in constraint satisfaction problems (CSPs) refers to the strategy used to decide the order in which variables are assigned values during search. Choosing a good ordering can significantly improve efficiency.
- The MRV (Minimum Remaining Values)heuristic doesn't help choose the first region to colour in Australia because, initially, every region has three legal colours.
- In this case, the **degree heuristic** comes in handy. It attempts to reduce the branching factor on future choices by selecting the variable involved in the largest number of constraints on other unassigned variables.
- SA is the variable with the highest degree, 5; the other variables have degrees 2 or 3, except for T, which has 0.

Selecting Variables with Highest Degree

- In fact, once SA is chosen, applying the degree heuristic solves the problem without any false steps-you can choose any consistent colour at each choice point and still arrive at a solution with no backtracking

- The minimum remaining values heuristic is usually a more powerful guide, but the degree heuristic can be useful as a tie-breaker.

- Once a variable has been selected, the algorithm must decide on the order in which to examine its values.

Selecting Variables with Highest Degree

- The minimum remaining values heuristic is usually a more powerful guide, but the highest degree heuristic can be useful as a tie-breaker.
- Once a variable has been selected, the algorithm must decide how to examine its values.
- For this, the **least-constraining-value** heuristic can be effective in some cases. It prefers the value that rules out the fewest choices for the neighbouring variables in the constraint graph.
- For example, suppose we have generated the partial assignment with WA = red and NT = green, and our next choice is for Q. Blue would be a bad choice because it eliminates the last legal value left for Q's neighbour, SA.

Selecting Variables with Highest Degree

- The least-constraining-value heuristic, therefore, prefers red to blue.

- In general, the heuristic is trying to leave the maximum flexibility for subsequent variable assignments. Of course, if we are trying to find all the solutions to a problem, not just the first one, then the ordering does not matter because we have to consider every value anyway. The same holds if there are no solutions to the problem.

CSP Summary

- CSPs are a special kind of problem
  - States defined by values of a fixed set of variables
  - Goal test defined by constraints on variable values
- Backtracking = depth-first search with one variable assigned per node
- Variable ordering and value selection heuristics help significantly
- Forward checking prevents assignments that guarantee later failure
- Constraint propagation (e.g., arc consistency) does additional work to constrain values and detect inconsistencies
- Iterative min-conflicts is usually effective in practice

THANK YOU