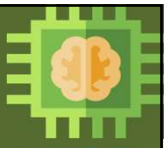
Elective Course
Course Code: CS4103
Autumn 2025-26



Lecture #17

Artificial Intelligence for Data Science

Week-5: CONSTRAINT SATISFACTION PROBLEM (CSP) [Part-III]
(Solving CSPs)

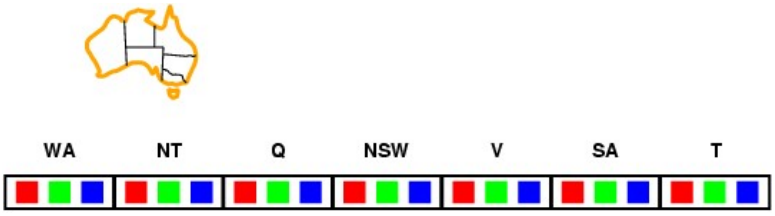
Course Instructor:
Dr. Monidipa Das
Assistant Professor
Department of Computational and Data Sciences
Indian Institute of Science Education and Research Kolkata, India 741246

Solving CSPs: Improving Efficiency

- Which variable should be assigned next?
- In what order should its values be tried?
- Can we **detect** inevitable failures early?
- Can we take advantage of the **problem structure**?

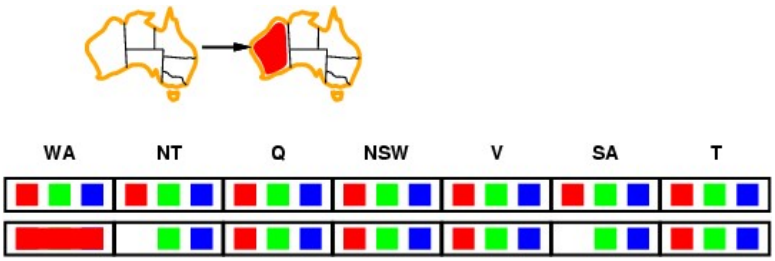
Forward checking

- Idea:
  - Keep track of remaining legal values for unassigned variables
  - Terminate search when any variable has no legal values



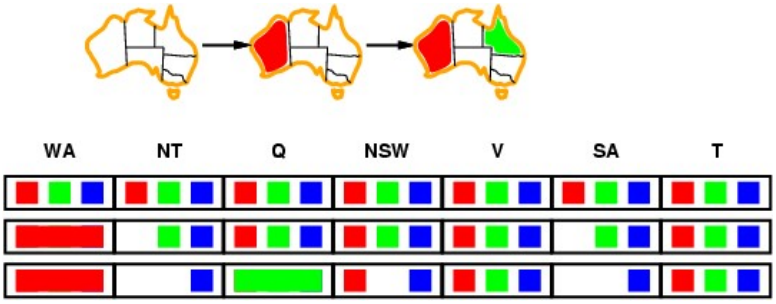
Forward checking

- Idea:
  - Keep track of remaining legal values for unassigned variables
  - Terminate search when any variable has no legal values



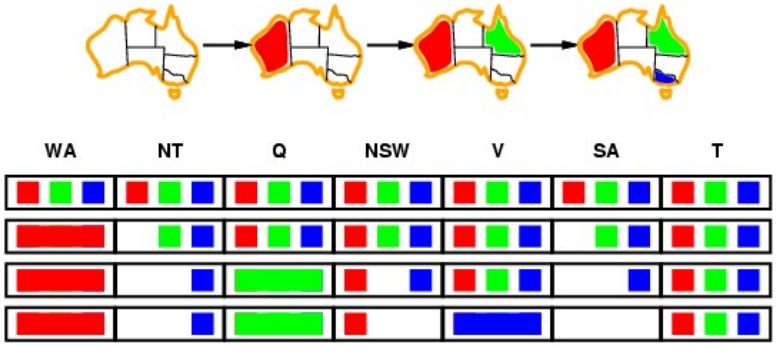
Forward checking

- Idea:
  - Keep track of remaining legal values for unassigned variables
  - Terminate search when any variable has no legal values

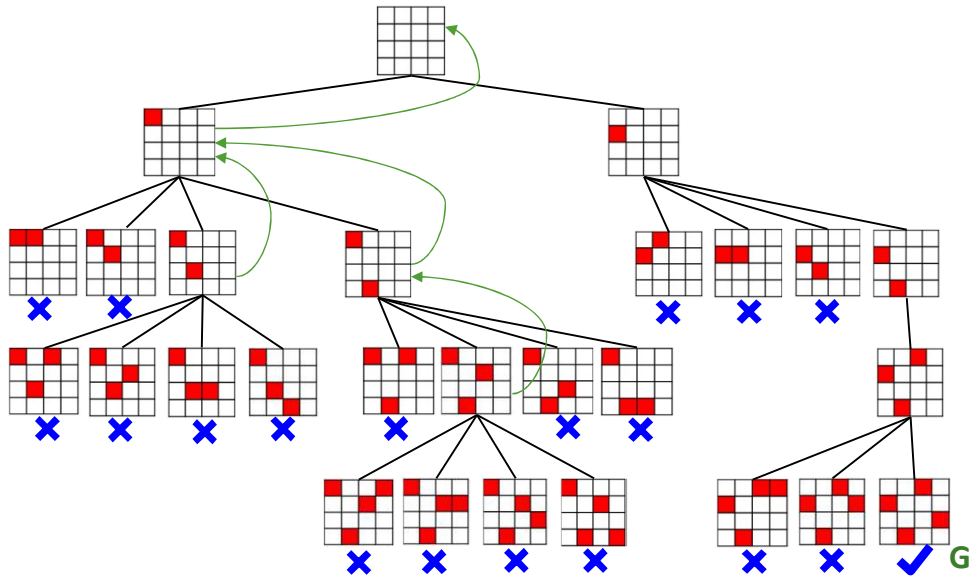


Forward checking

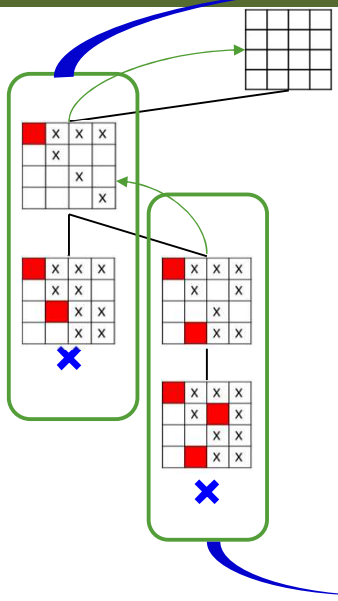
- Idea:
  - Keep track of remaining legal values for unassigned variables
  - Terminate search when any variable has no legal values



Backtracking Search Example: 4-Queens Problem



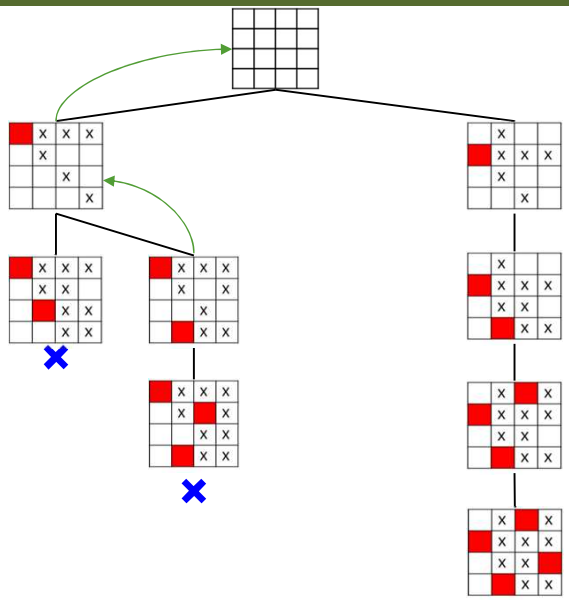
Forward checking: 4-Queens Problem



Q1	Q2	Q3	Q4
{1, 2, 3, 4}	{1, 2, 3, 4}	{1, 2, 3, 4}	{1, 2, 3, 4}
1	{3, 4}	{2, 4}	{2, 3}
1	3	{}	{2}

Q1	Q2	Q3	Q4
{1, 2, 3, 4}	{1, 2, 3, 4}	{1, 2, 3, 4}	{1, 2, 3, 4}
1	{3, 4}	{2, 4}	{2, 3}
1	4	{2}	{3}
1	4	2	{}

Forward checking: 4-Queens Problem



Q1	Q2	Q3	Q4
{1, 2, 3, 4}	{1, 2, 3, 4}	{1, 2, 3, 4}	{1, 2, 3, 4}
2	{4}	{1, 3}	{1, 3, 4}
2	4	{1}	{1, 3}
2	4	1	{3}
2	4	1	3

✔ Goal state!

Constraint propagation

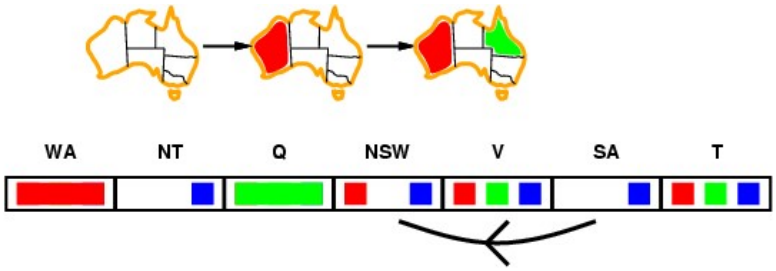
- Forward checking propagates information from assigned to unassigned variables, but doesn't provide early detection for all failures:

- NT and SA cannot both be blue!
- Constraint propagation repeatedly enforces constraints locally

Arc consistency

- Simplest form of propagation makes each arc consistent
- $X \rightarrow Y$ is consistent iff

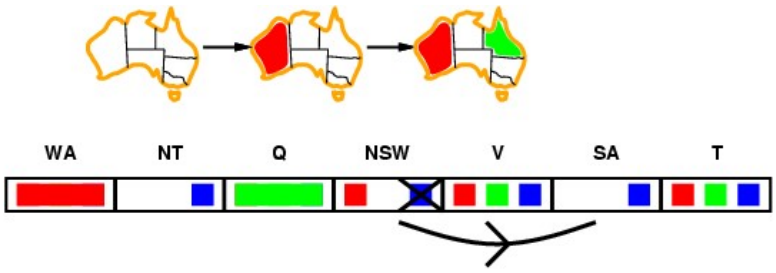
  for every value x of X there is some allowed y



Arc consistency

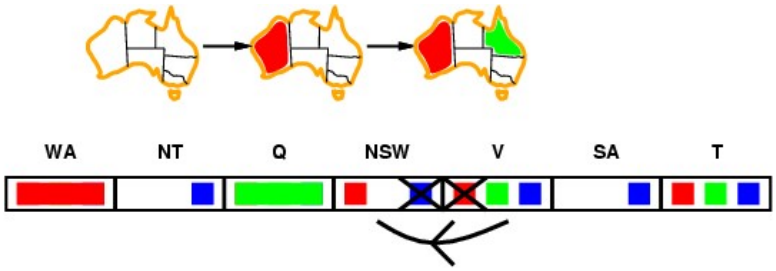
- Simplest form of propagation makes each arc consistent
- $X \rightarrow Y$ is consistent iff

  for every value x of X there is some allowed y



Arc consistency

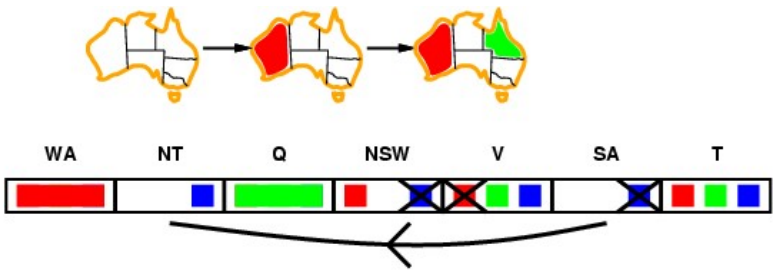
- Simplest form of propagation makes each arc consistent
- $X \rightarrow Y$ is consistent iff
  for every value x of X there is some allowed y



- If X loses a value, neighbors of X need to be rechecked

Arc consistency

- Simplest form of propagation makes each arc consistent
- $X \rightarrow Y$ is consistent iff
  for every value x of X there is some allowed y



- If X loses a value, neighbors of X need to be rechecked
- Arc consistency detects failure earlier than forward checking
- Can be run as a preprocessor or after each assignment

Arc consistency Algorithm

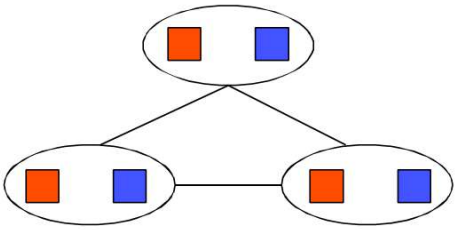

```
function AC-3( csp) returns the CSP, possibly with reduced domains
  inputs: csp, a binary CSP with variables {X1, X2, ..., Xn}
  local variables: queue, a queue of arcs, initially all the arcs in csp

  while queue is not empty do
    (Xi, Xj) ← REMOVE-FIRST(queue)
    if RM-INCONSISTENT-VALUES(Xi, Xj) then
      for each Xk in NEIGHBORS[Xi] do
        add (Xk, Xi) to queue

function RM-INCONSISTENT-VALUES( Xi, Xj) returns true iff remove a value
  removed ← false
  for each x in DOMAIN[Xi] do
    if no value y in DOMAIN[Xj] allows (x,y) to satisfy constraint(Xi, Xj)
      then delete x from DOMAIN[Xi]; removed ← true
  return removed
```

Time complexity: $O(n^2d^3)$

Limitations of Arc Consistency

- After running arc consistency:
  - Can have one solution left
  - Can have multiple solutions left
  - **Can have no solutions left (and not know it)**

What went wrong here?



k-consistency

- A CSP is *k-consistent* if, for any set of $k-1$ variables, and for any consistent assignment to those variables, a consistent value can always be assigned to any k-th variable
  - 1-consistency is **node consistency**
  - 2-consistency is **arc consistency**
  - For binary constraint networks, 3-consistency is the same as ***path consistency***
- Getting k-consistency requires time and space exponential in k

Questions?