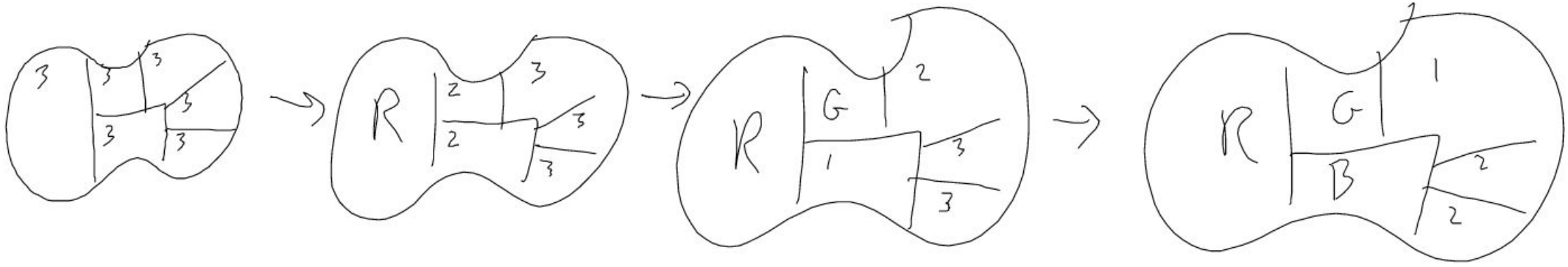
Efficiency

1. Which variable should I work on next?
2. What order should I try to assign domain values? (how should I order my successors?)
3. Can I detect failure early?
4. Can I take advantage of global problem structure?

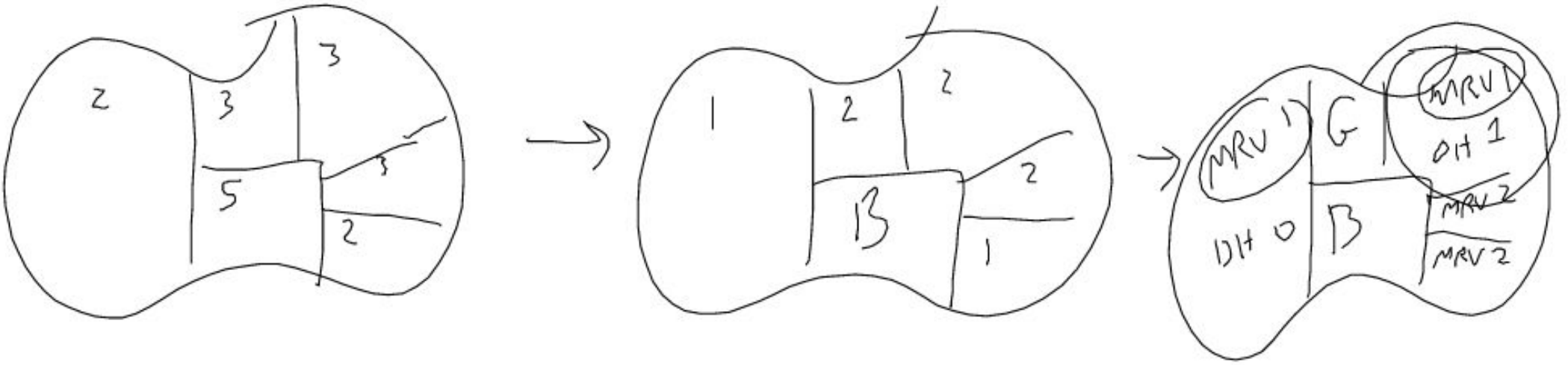
Which variable next?

- **Minimum Remaining Values (MRV)**
  - Choose variable with fewest legal values



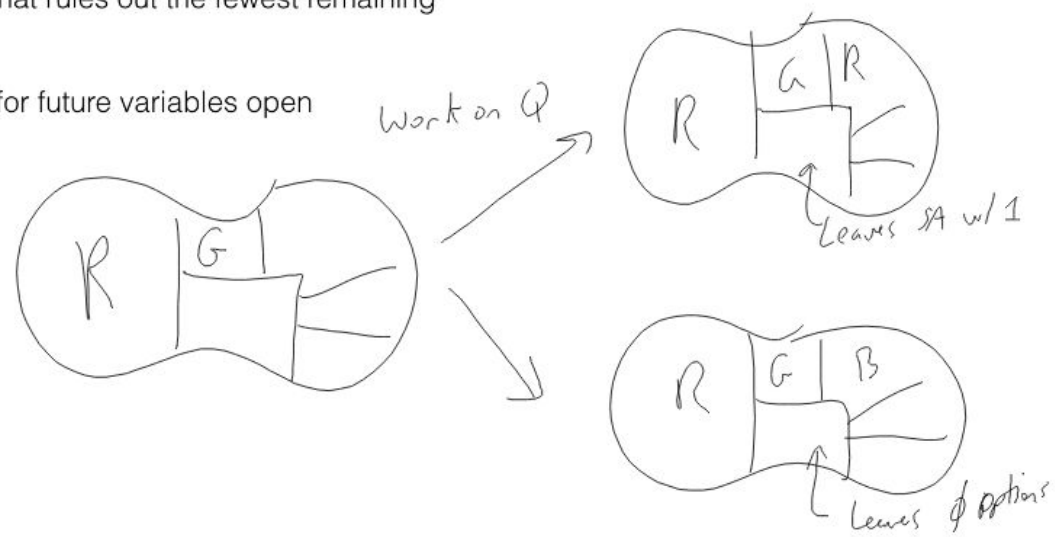
Which variable next?

- **Degree Heuristic**
  - MRV leaves ties
  - To break ties: choose variable with most constraints on remaining variables



Which value next?

- After you have picked which variable to work on, how should you sort successors?
- **Least Constraining Value**
  - Pick the value that rules out the fewest remaining variables
  - Keeps options for future variables open



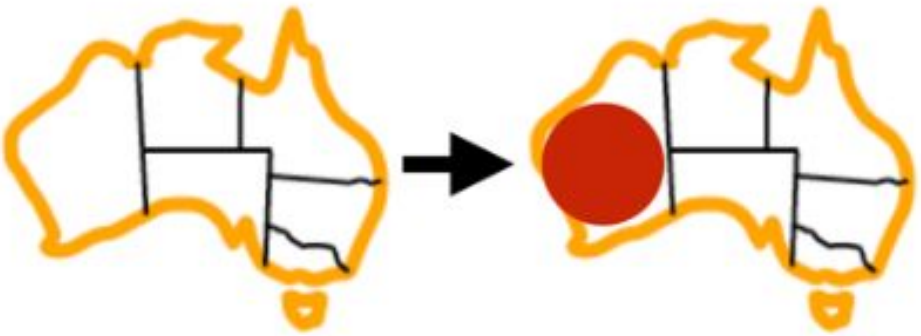
Variable choice: fail early - prunes tree as much as possible

value choice: fail late - leave options open
-> we only need 1 solution

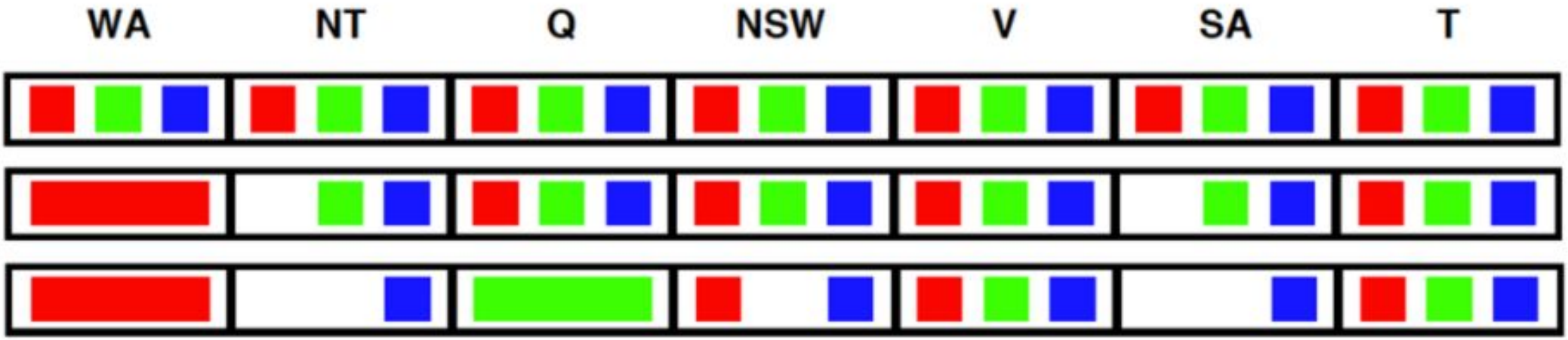
Detect failures early

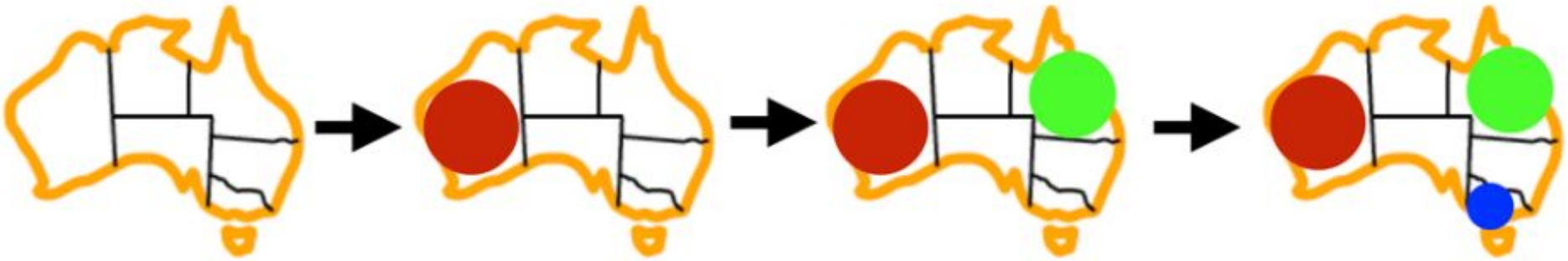
- **Forward checking**
  - Keep track of remaining possible values for each unassigned variable
  - Use constraints to remove values from variable domains as you move down the tree
  - Check for empty domains

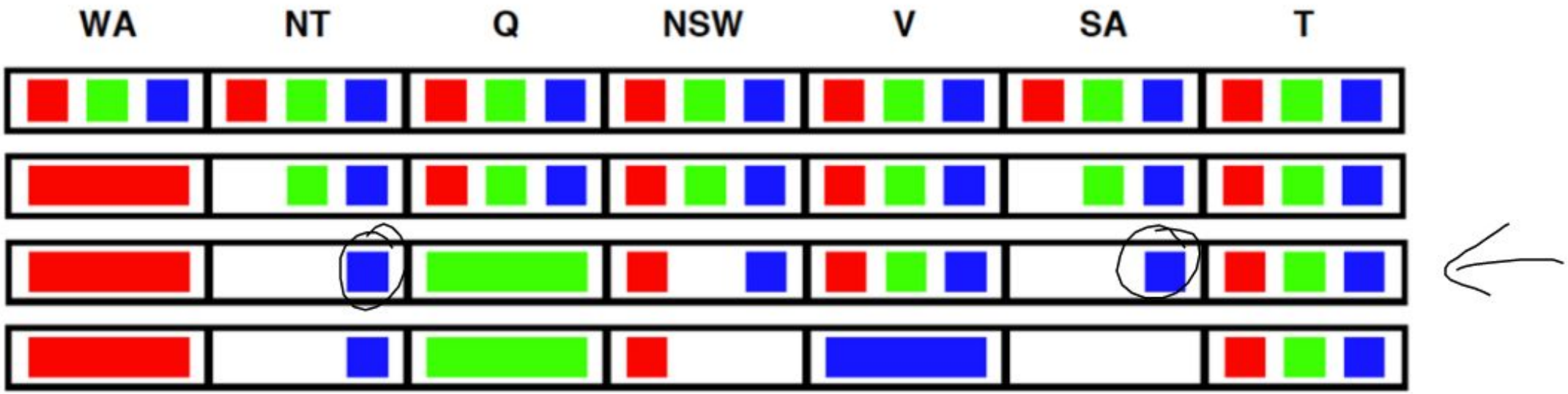

WA
NT
Q
NSW
V
SA
T




WA
NT
Q
NSW
V
SA
T

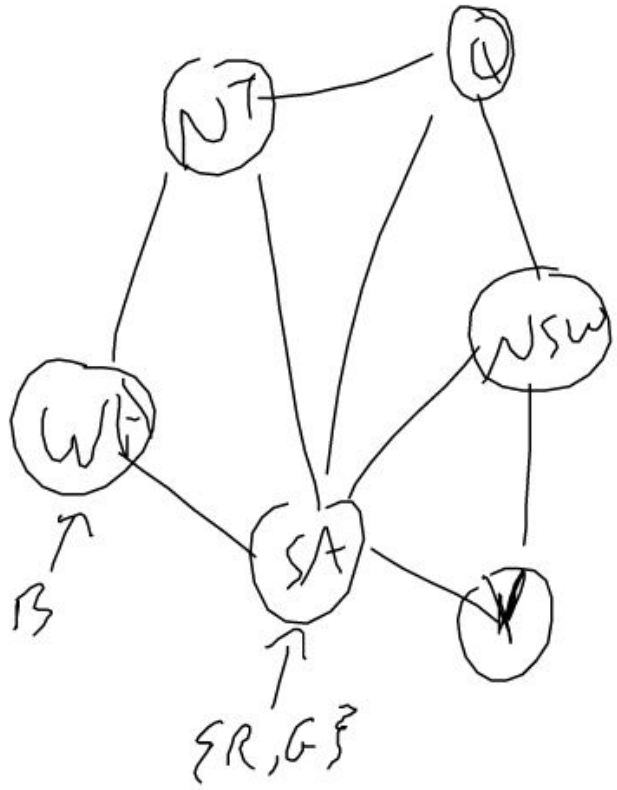

WA
NT
Q
NSW
V
SA
T

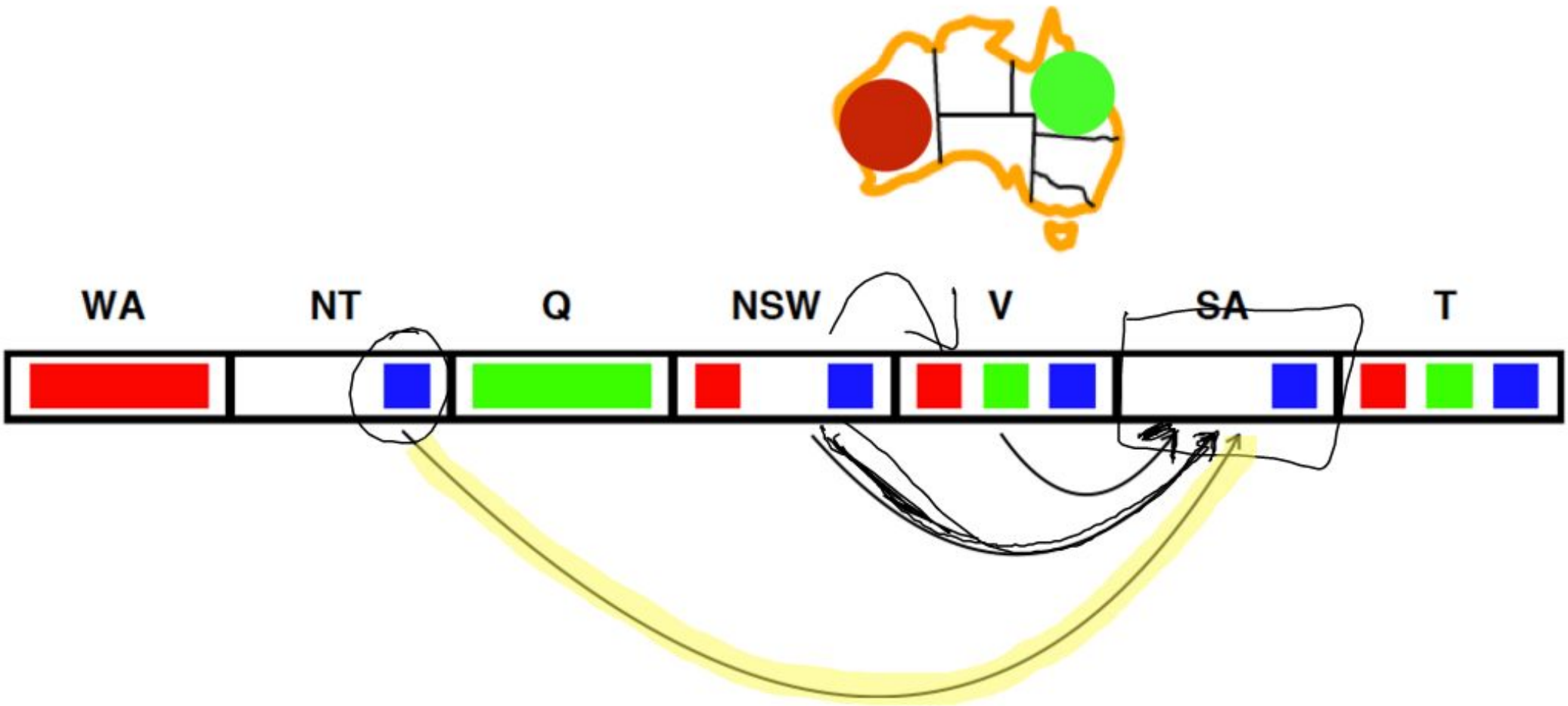



WA
NT
Q
NSW
V
SA
T

Detect failures early

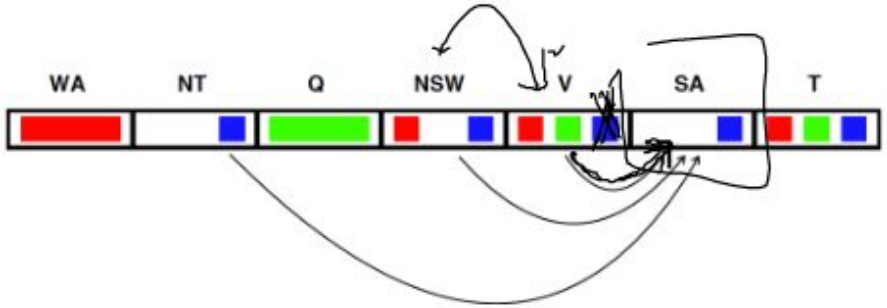
- **Maintaining Arc Consistency**
  - Consider the constraint graph (make directional arcs)
  - An arc (X_i, X_j) is consistent if for every value in D_i, there is is some value in D_j that satisfies the binary constraint on (X_i, Xj)
  - A graph is consistent if every arc is consistent



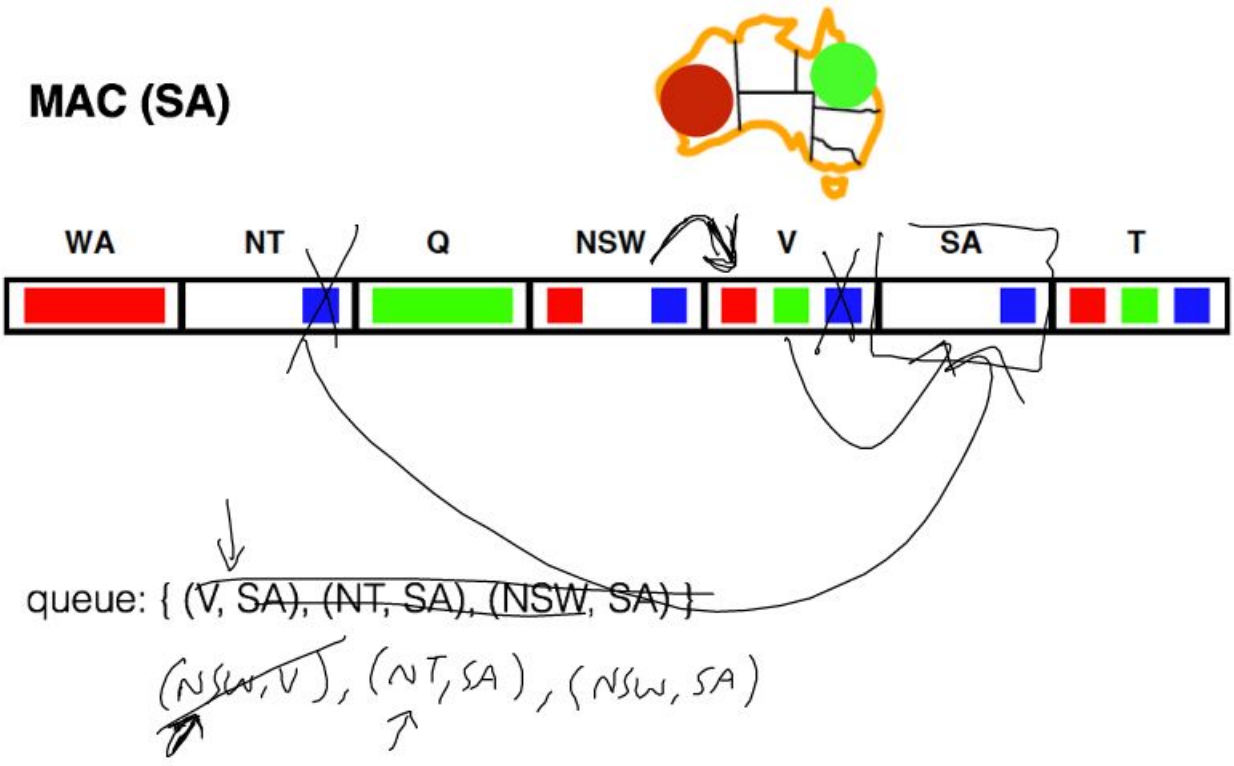

WA
NT
Q
NSW
V
SA
T

Maintain-Arc-Consistency (CSP, X_{start})

- queue <— {set of all arcs (X_j, X_{start})}
- WHILE queue is not empty DO:
  - (X_i, X_j) <— pop(queue)
  - Make X_i arc consistent with respect to X_j
  - IF $|D_i|$ decreased DO:
    - IF $|D_i| = 0$ THEN RETURN false
    - FOREACH X_k in neighbors(X_i) - {X_j} DO:
      - queue <— queue + {(X_k, X_i)}
    - END FOREACH
  - END WHILE
  - RETURN true



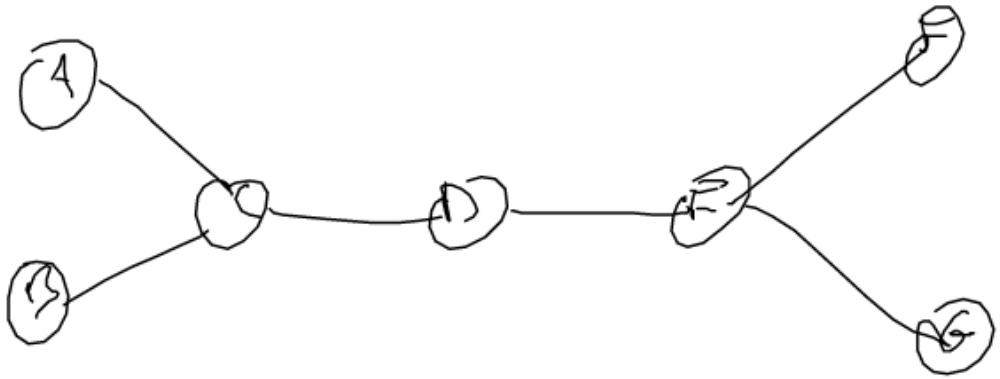
(V, SA)


MAC (SA)
WA
NT
Q
NSW
V
SA
T
queue: { (V, SA), (NT, SA), (NSW, SA) }
(NSW, V), (NT, SA), (NSW, SA)

AC-3

- MAC before you start the search
- Preprocess constraint graph
- Reduces some variable domains, making search fast
- MAC but put all $ arcs into the queue

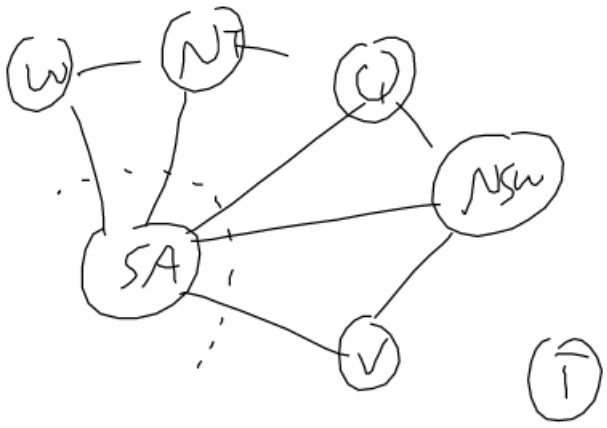
Tree - Structured CSPs



- Forward checking $\longrightarrow$ $O(n^2 d^2)$ (better $O(d^2)$)

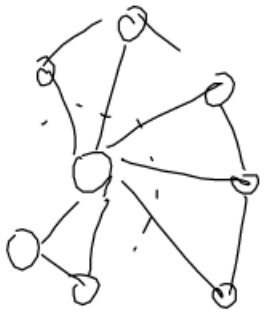
Nearly - Tree Structured

- Small # of variables
  Can be removed to
  create 1 tree and
  1 non-tree
- Solve both & merge solutions



(1) Find the tree & non-tree graphs

- cycle-cut-set : smallest # of links to create the largest tree



(2) Solve the non-tree

- Remove values from domain of variables in tree

(3) Solve tree