Fall 2021 | Lecture 5

Constraint Satisfaction Problems

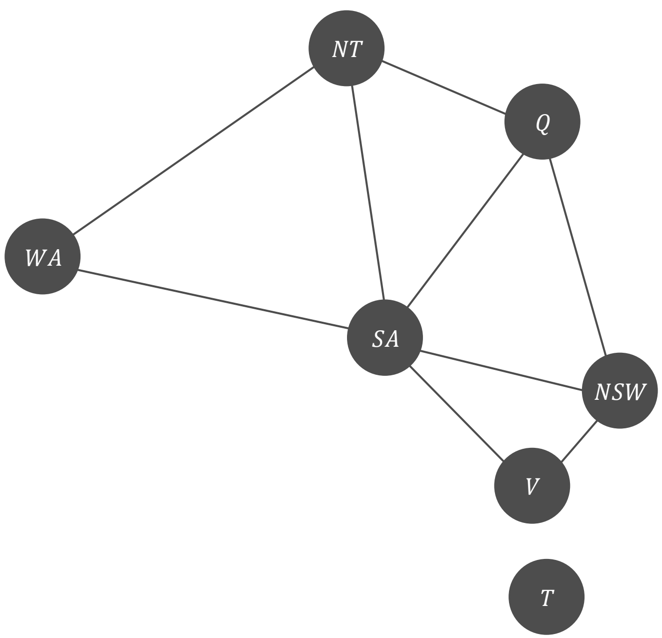
Ariel Procaccia | Harvard University

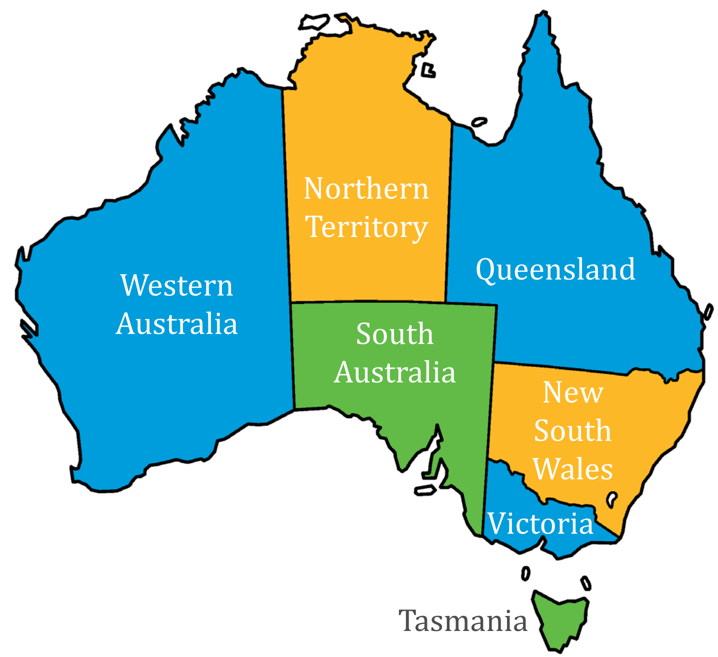
WHAT ARE CSPS?

- A constraint satisfaction problem (CSP):
  - Variables $\{X_1, \ldots, X_n\}$
  - Domains $\{D_1, \ldots, D_n\}$
  - A set of constraints: defined on subsets of variables, give allowable tuples of values
- Consider (possibly partial) assignments of values to variables
- A solution is a complete and consistent assignment
- CSP are search problems but their structure allows general purpose algorithms that don’t require domain knowledge

EXAMPLE: MAP COLORING

- Variables = {WA,NT,SA,Q,NSW,V,T}
- $D_i = \{$  $\}$
- Constraints: adjacent regions have different colors



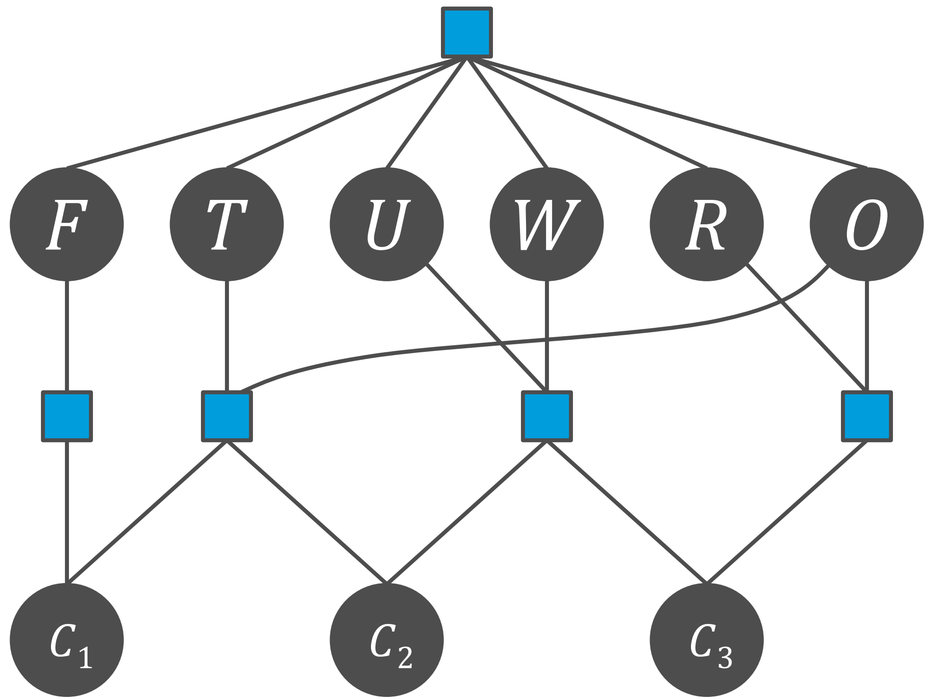


TYPES OF CONSTRAINTS

- A unary constraint restricts the value of a single variable
- A binary constraint relates two variables
- A CSP with only unary and binary constraints (such as map coloring) is called a binary CSP and can be represented using a constraint graph
- Constraints involving more than two variables can be represented using a hypergraph

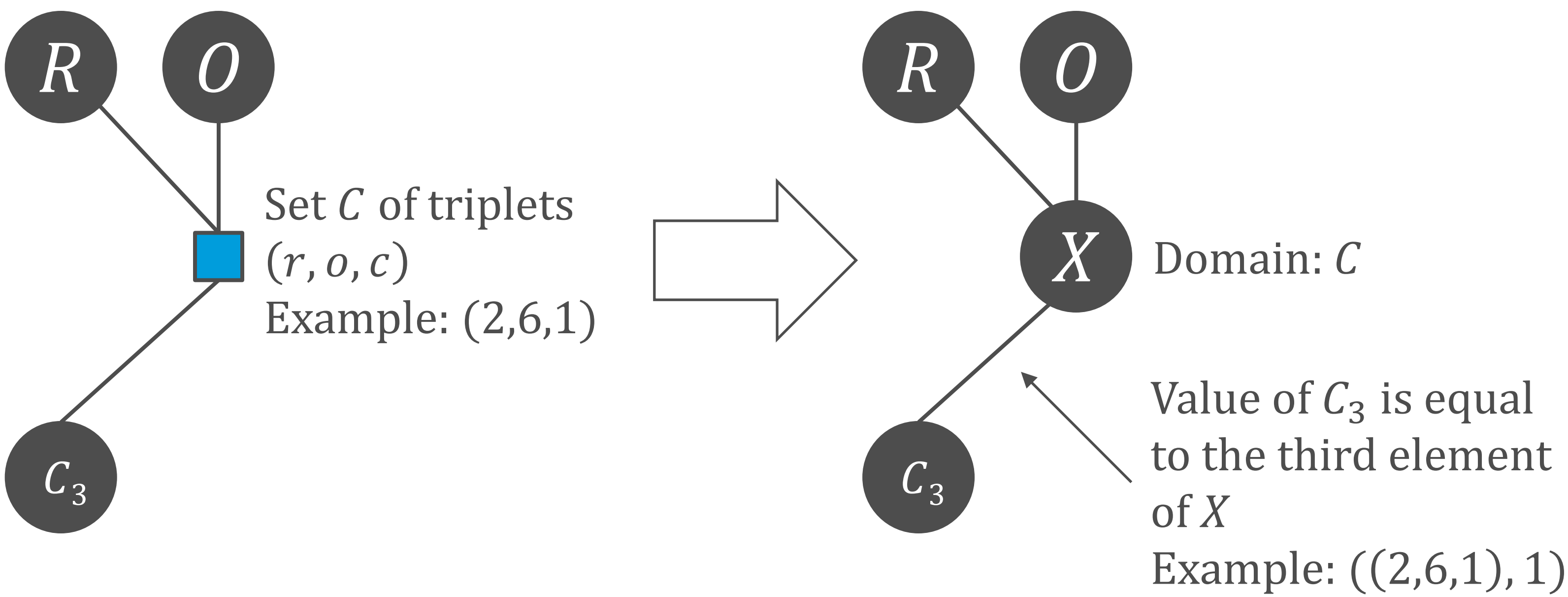
EXAMPLE: CRYPTARITHMETIC

```
   T W O
 + T W O
 -------
 F O U R
```



REDUCTION TO BINARY

- Any constraint can be reduced to a set of binary constraints by introducing additional variables
- So we can assume that the given CSP is binary



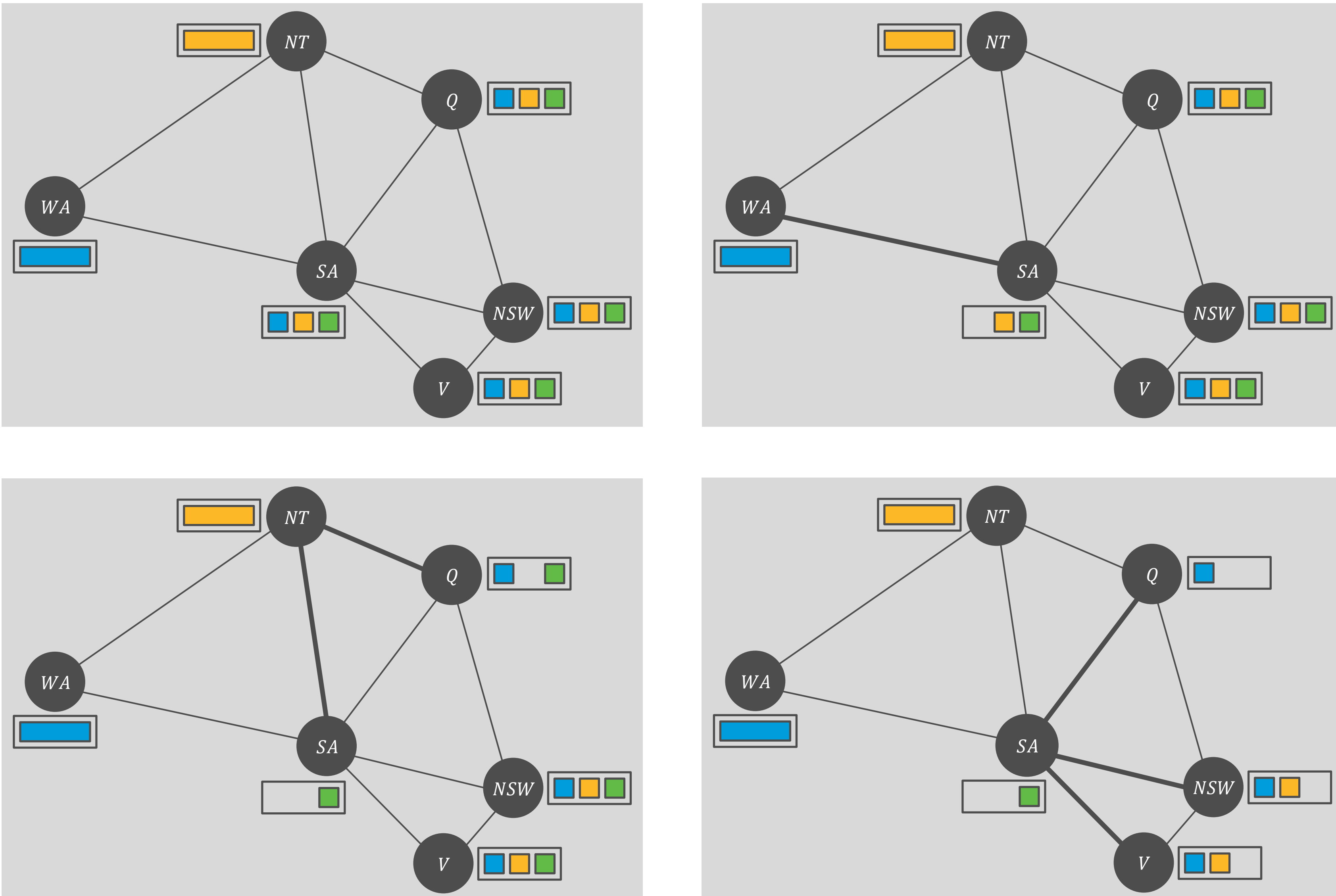
INFERENCE

- We can use the constraints to directly eliminate potential assignments of variables
- This can be done as a preprocessing step before search, or it can be interleaved with search
- Sometimes search isn't needed at all!

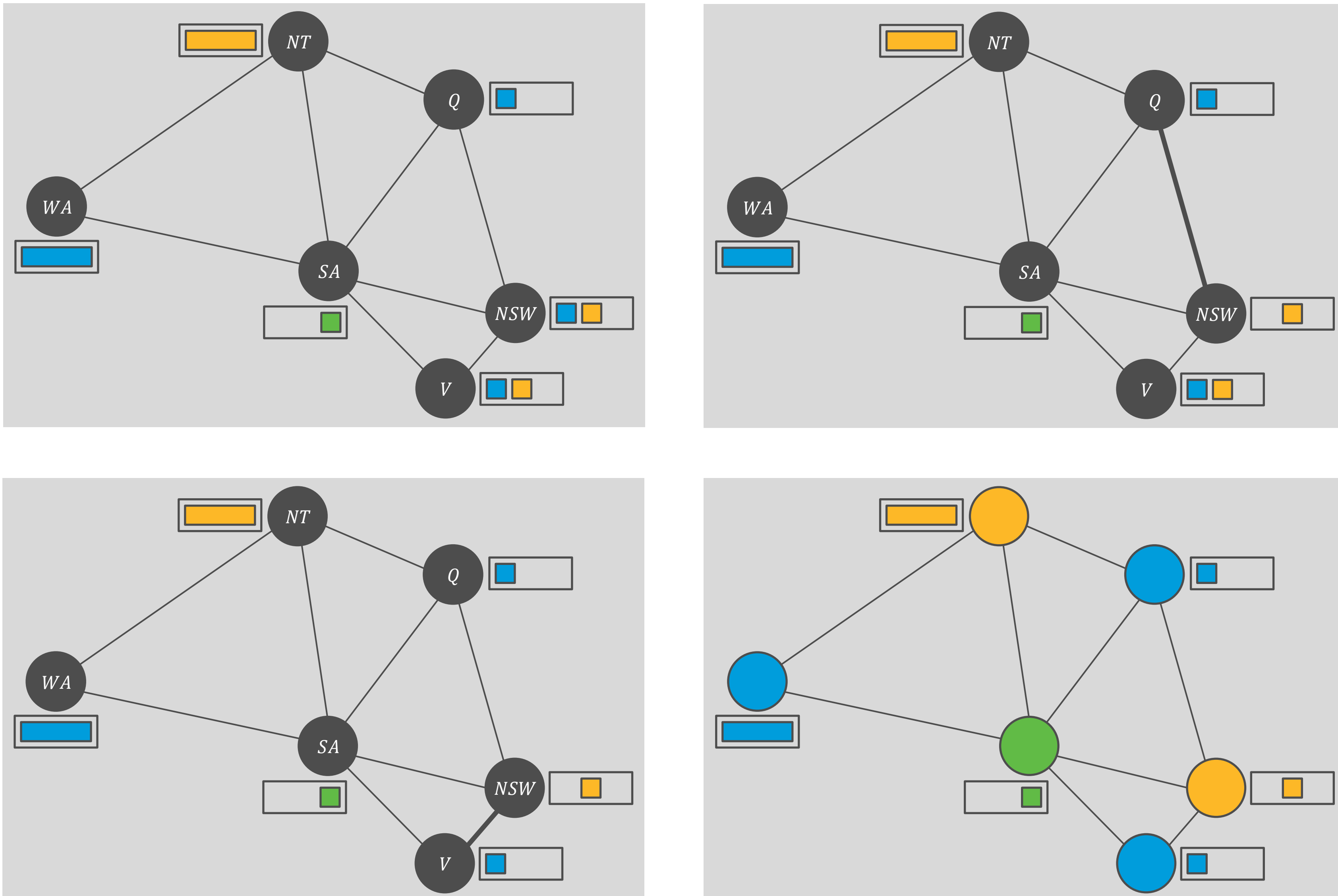
ARC CONSISTENCY

- A variable X_i is arc-consistent with respect to variable X_j if for every value in D_i there is a value in D_j that satisfies the binary constraint on (X_i, X_j)
- The CSP is arc-consistent if every variable is arc-consistent with respect to every other variable

ENFORCING ARC CONSISTENCY



ENFORCING ARC CONSISTENCY

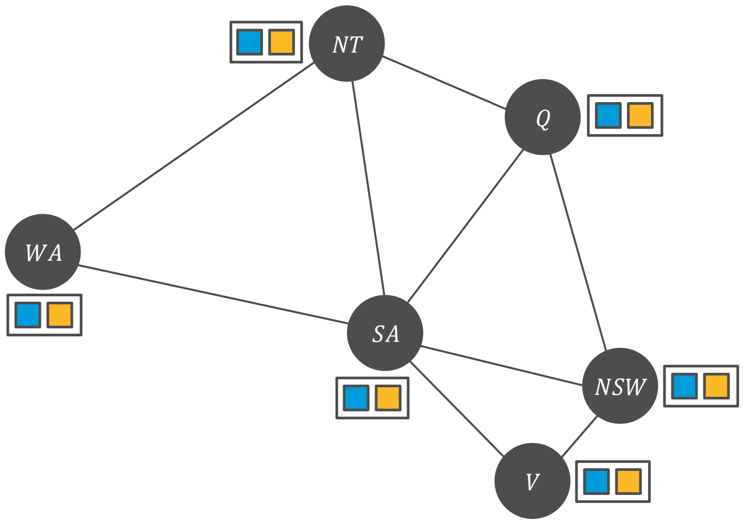


THE AC-3 ALGORITHM

```
function AC-3 (csp)
    queue ← all arcs in csp
    while queue is not empty do
        (X_i, X_j) ← POP(queue)
        if REVISE(csp, X_i, X_j) then
            if |D_i| = 0 then return false
            for each neighbor X_k ≠ X_j of X_i
                add (X_k, X_i) to queue
    return true

function REVISE(csp, X_i, X_j)
    revised ← false
    for each x in D_i do
        if no value y in D_j allows (x, y) to satisfy the
        constraint between X_i and X_j then
            delete x from D_i
            revised ← true
    return revised
```

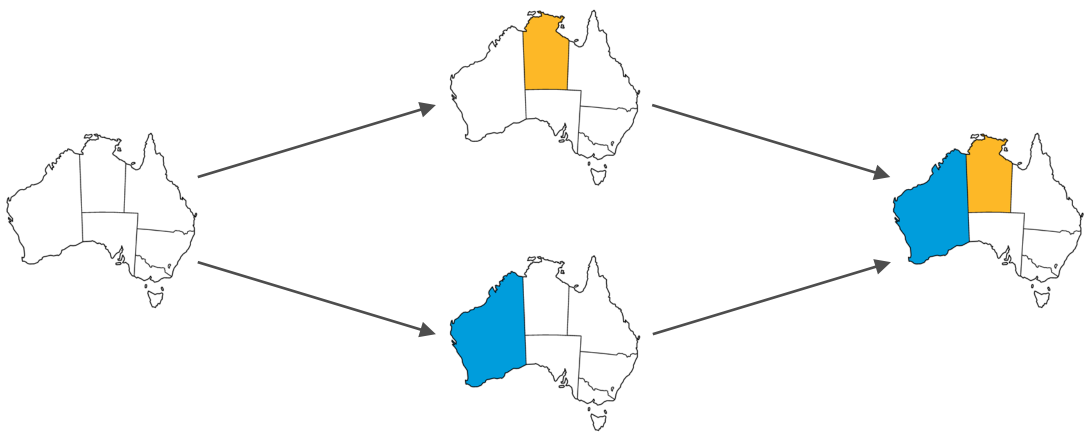
SHORTCOMING OF ARC CONSISTENCY



- There is no solution but AC-3 does nothing!
- Enforcing consistency for subsets of variables of size $k \geq 3$ is sufficient here

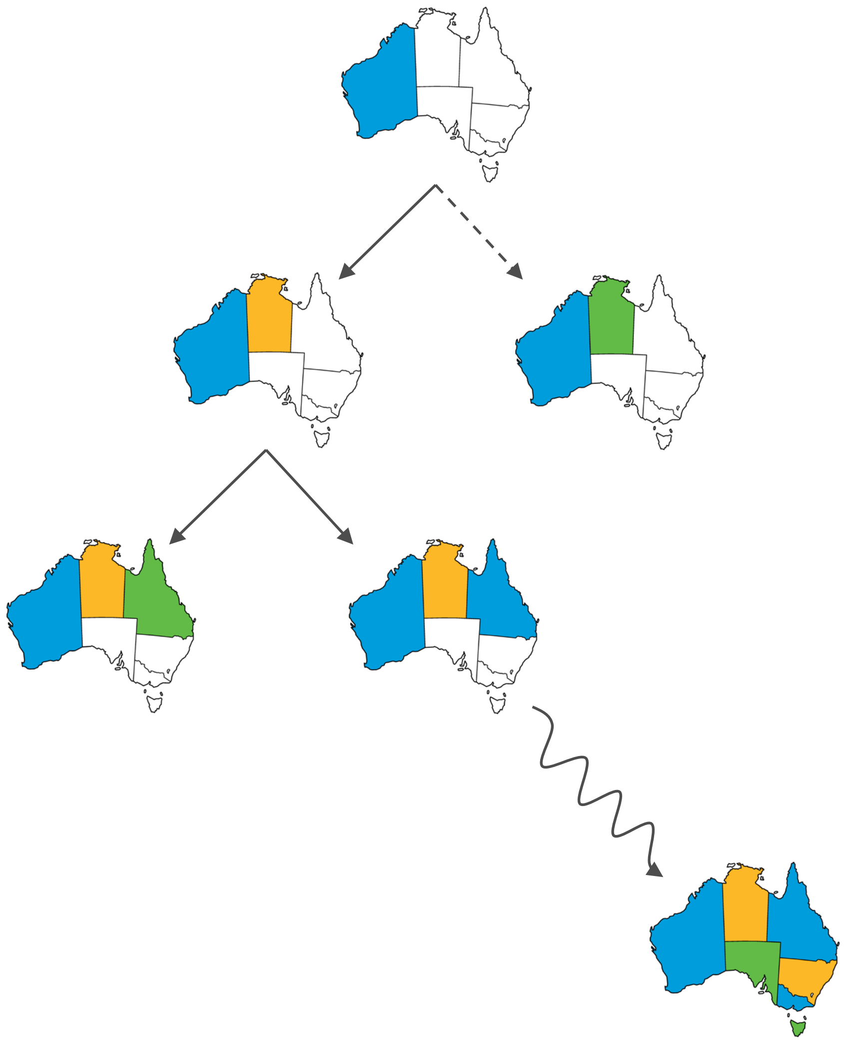
BACKTRACKING SEARCH

- Typically, inference would leave variables with multiple possible values; we need to search!
- A useful property of CSPs is commutativity: the order in which we assign variables doesn't matter



- Backtracking search assigns the variables one by one, and backtracks if needed

BACKTRACKING SEARCH



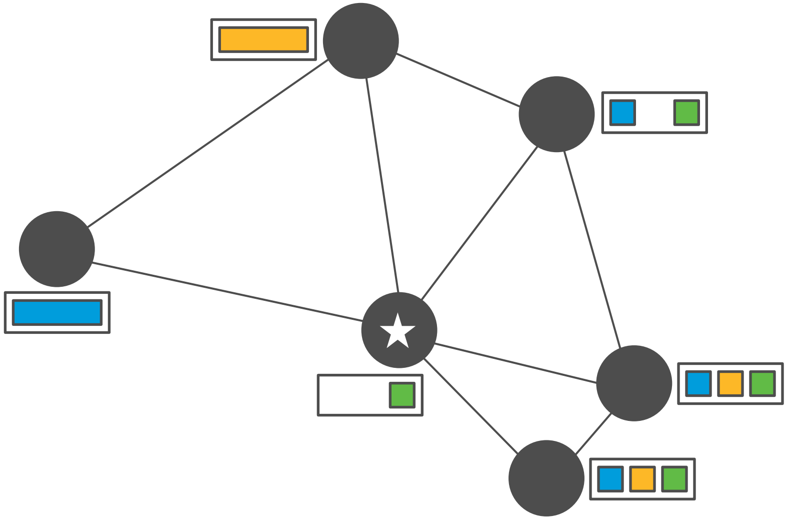
BACKTRACKING SEARCH

```
function BACKTRACK(csp, assign)
    if assign is complete then return assign
    var ←SELECT-UNASSIGNED-VARIABLE(csp, assign)
    for each value in ORDER-DOMAIN-VALUES(csp, var, assign) do
        if value is consistent with assign then
            add {var = value} to assign
            inferences ←INFERENCE(csp, var, assign)
            if inferences ≠ failure then
                add inferences to csp
                result ← BACKTRACK(csp, assign)
                if result ≠ failure then return result
                remove inferences from csp
            remove {var = value} from assign
    return failure
```

Poll 1: What are some potentially good heuristics for SELECT-UNASSIGNED-VARIABLE?

VARIABLE ORDERING

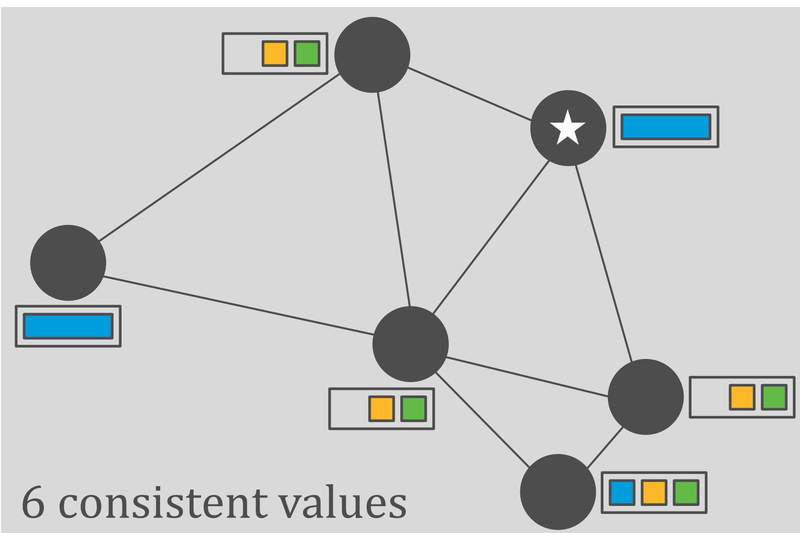
- Backtracking search includes the function SELECT-UNASSIGNED-VARIABLE
- Minimum remaining values heuristic: choose the variable with the fewest legal values

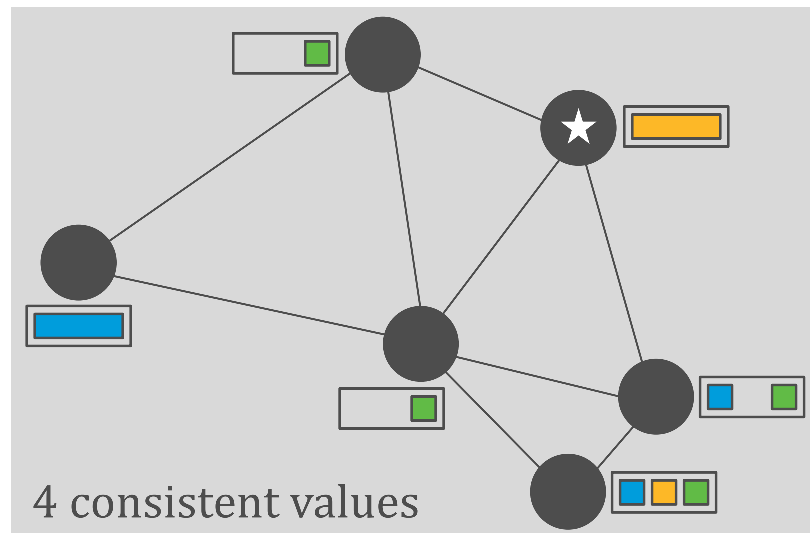


- Logic: prune as early as possible

VALUE ORDERING

- Backtrack search includes the function ORDER-DOMAIN-VALUES
- Least constraining value heuristic: choose value that rules out fewest choices for other variables

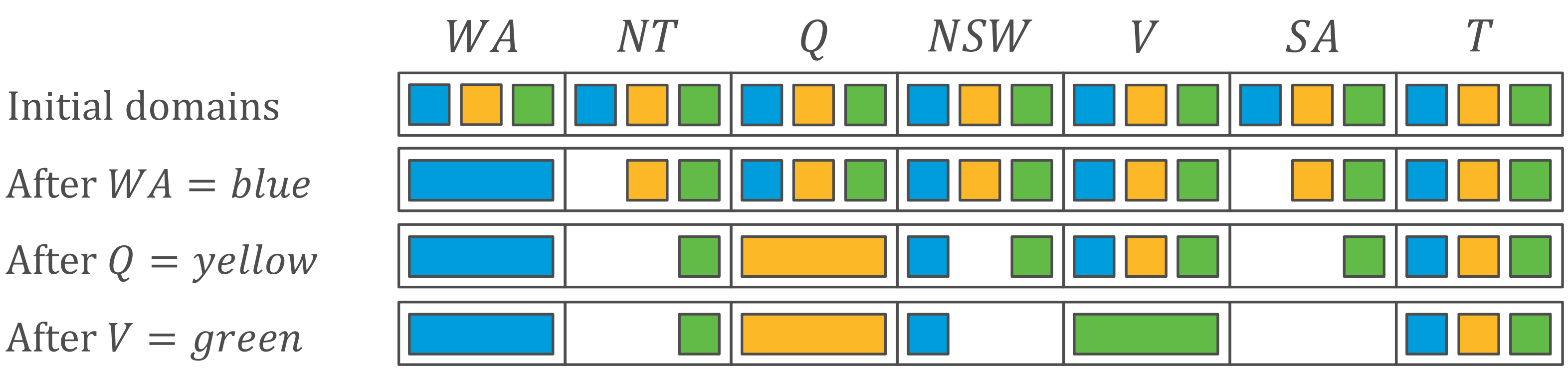




- Logic: consider value that's most likely to succeed

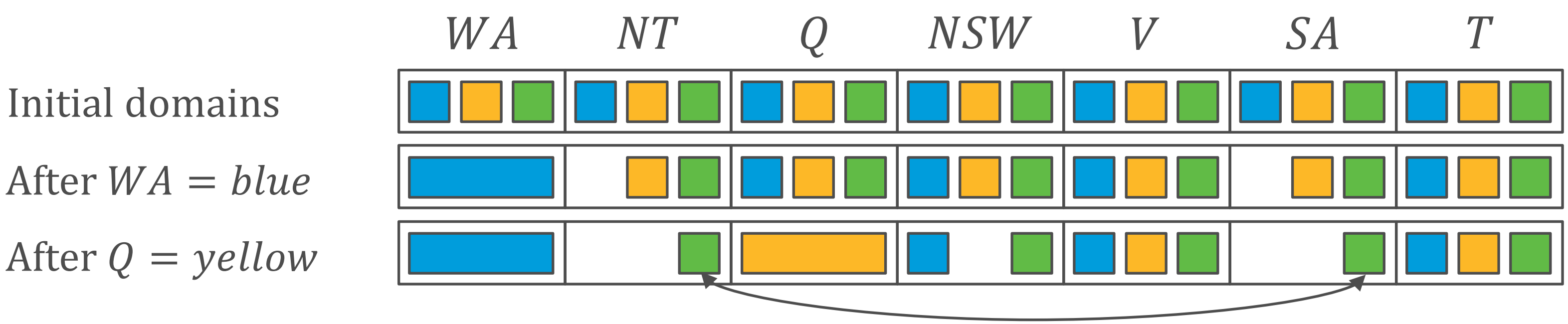
FORWARD CHECKING

- The function INFERENCE gives us an opportunity to interleave inference and search
- The simplest inference is forward checking, which enforces arc consistency for each variable that's assigned a value



MAINTAINING ARC CONSISTENCY

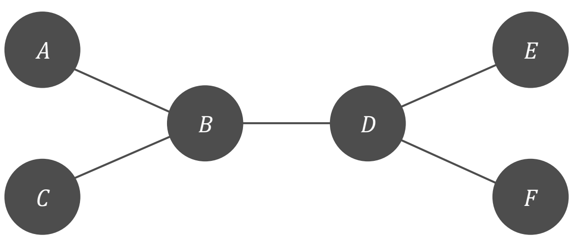
- In our forward checking example, more powerful arc consistency enforcement would have terminated earlier

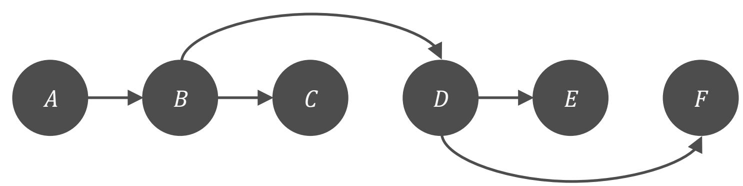


- The INFERENCE function can implement AC-3, but only arcs (X_j, X_i) between the newly assigned variable X_i and unassigned neighbors X_j are initially added to the queue

THE STRUCTURE OF PROBLEMS

- Solving a CSP whose constraint graph is structured as a tree is very easy
- We say that the CSP is directionally arc-consistent under an ordering of the variables $X_1, \ldots, X_n$ if and only if for every $i < j$, X_i is arc-consistent with X_j
- The idea is to topologically sort the tree and make the it directionally arc-consistent with respect to the $n-1$ edges





- Then we go down the list of variables and choose any remaining value

TREE CSP SOLVER

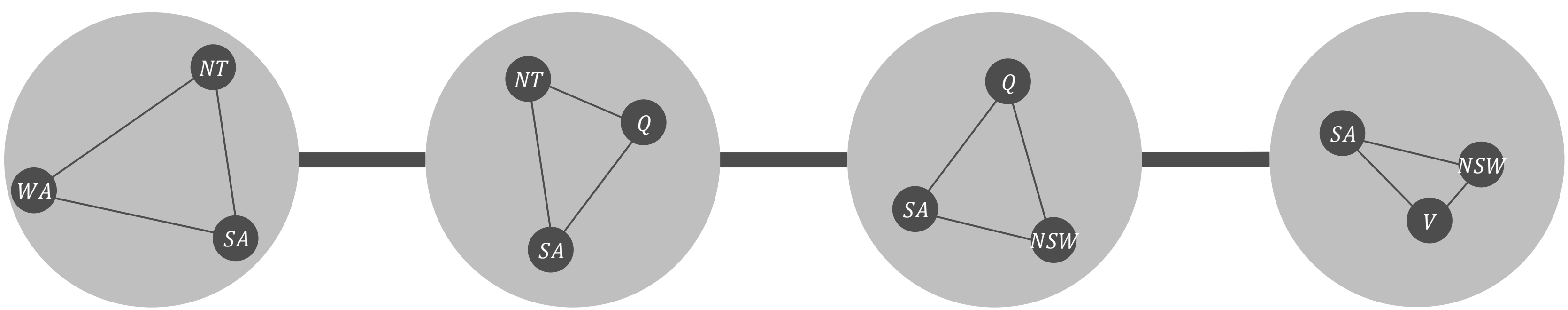
```
function TREE-CSP-SOLVER(csp)
    n ← number of variables in X
    assign ← ∅
    root ← any variable in X
    X ← TOPOLOGICAL-SORT(X, root)
    for j = n down to 2 do
        MAKE-ARC-CONSISTENT(PARENT(X_j), X_j)
        if consistency fails then return failure
    for i = 1 to n do
        if D_i has no consistent values then return failure
        assign[X_i] ← any consistent value from D_i
    return assign
```

Running time $O(nd^2)$ for $|D_i| \leq d$

TREE DECOMPOSITION

Any CSP can be converted into a tree:

- Every variable appears in at least one node
- If two variables are connected by a constraint, they must appear together in at least one node
- If one variable appears in two nodes, there must be a path between them and it must appear in every node on the path



TREE DECOMPOSITION

- Poll 2: For a CSP with n variables and domains of size d, what is the minimum number of nodes in a tree decomposition of the CSP?
  1. $\Theta(1)$ ✓
  2. $\Theta(d)$
  3. $\Theta(n/d)$
  4. $\Theta(n)$
- If a tree decomposition has m nodes and the largest node has k variables, the running time of TREE-CSP-SOLVER is $O(md^{2k})$