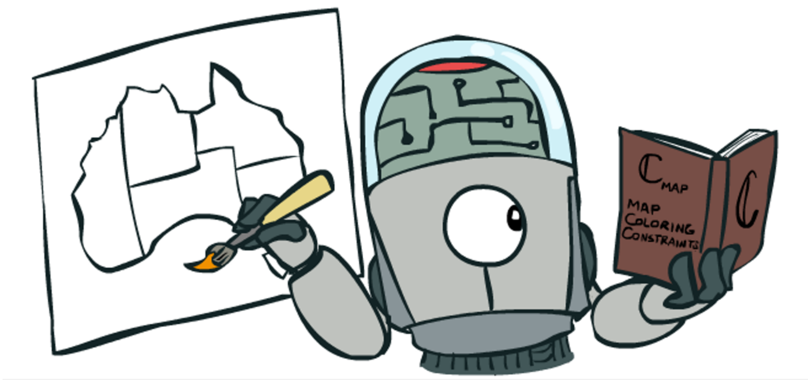
CS440 – Constraint Satisfaction Problems



Purpose: Consider a problem like coloring a map where you are given a set of constraints. For the map coloring problem the constraints are that you don't want two regions in the map which are adjacent to have the same color. This is a special case of a search problem.

The purpose of this module is to present algorithms for solving constraint satisfaction problems.

Knowledge: This module will help you become familiar with the following content knowledge:

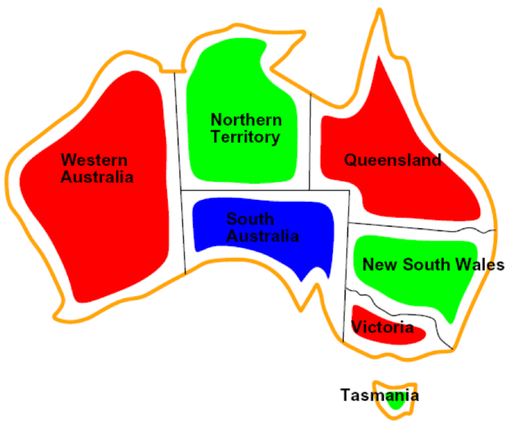
- The components of a Constraint Satisfaction Problem
- The basic backtracking algorithm for solving CSPs
- Refinements to the backtracking algorithm

Activity1 - Constraint Satisfaction Problems:
With your group perform the following tasks and answer the questions. You will be reporting your answers back to the class in 20 minutes.

1. A **Constraint Satisfaction Problem (CSP)** is a special case of a search problem:

   - States are made up of variables X_i with values from a *domain* D
   - The goal test is a set of constraints specifying allowable combinations of values for subsets of variables

   An example is map coloring.

   The variables are WA, NT, Q, NSW, V, SA, T
   The domain is red, green, blue
   What would be the constraints?

   Another example is the N-Queens problem where we want to place N queens on an NxN chess board so that no queen attacks another (is on the same row, column, or diagonal).

   We could let the variables be X_{ij} representing the cells at row i column j.
   The domain is 0,1 indicating whether there is a queen there or not.
   The constraints would be:
   $\forall i, j, k(X_{ij}, Xik) \in \{(0,0),(0,1),(1,0)\}$
   $\forall i, j, k(X_{ij}, Xkj) \in \{(0,0),(0,1),(1,0)\}$
   $\forall i, j, k(X_{ij}, Xi+k, j+k) \in \{(0,0),(0,1),(1,0)\}$
   $\forall i, j, k(X_{ij}, Xi+k, j-k) \in \{(0,0),(0,1),(1,0)\}$

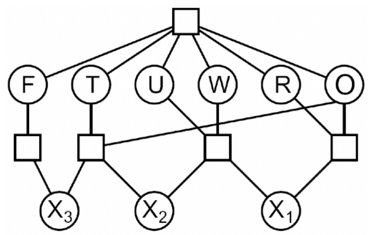
   Explain the meaning of each of these constraints.

   We could simplify this CSP formulation by changing the variables to $Q_1, Q_2, ...Q_N$

   What would be the domain and constraints now? (You may explain the constraint in english rather than mathematical notation).

Another example of a CSP is Cryptoarithmetic where you are given an arithmetic problem with variables representing digits.

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

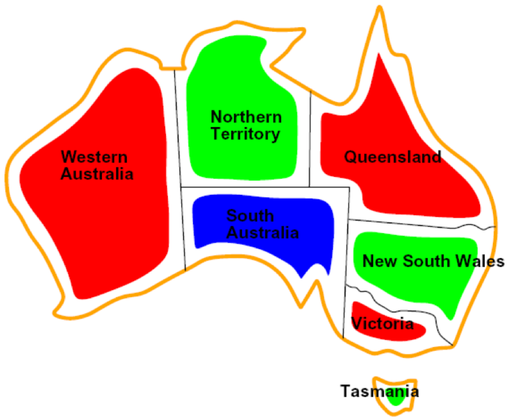


The variables are $F, T, U, W, R, O, X_1, X_2, X_3$
The domain is 0,1,2,3,4,5,6,7,8,9,0
The constraints include:
alldiff(F,T,U,W,R,O) which says that all of the values for those variables must be different
$O + O = R + 10X_1$
etc.

What do the variable X_1, X_2, and X_3 represent?

Activity2 - Search Methods:
With your group perform the following tasks and answer the questions. You will be reporting your answers back to the class in 40 minutes.

1. In a *Binary CSP*, each constraint related at most two variables. You can then draw a graph in which nodes are variables and edges represent the constraints.

   Draw the 7 nodes constraint graph for our map coloring problem of Australia.

2. Now consider using the search techniques we have already learned where the states are defined by the values assigned so far (partial assignments).
   The initial state is the empty assignment
   The successor function assigns a value to an unassigned variable.
   The goal test determines if the current assignment is complete and satisfies all the constraints.

   Draw part of the BFS tree for the Australian map color problem. What are the difficulties with using BFS? Draw part of the DFS tree for the Australian map color problem. This probably doesn't solve the all the difficulties with the search.

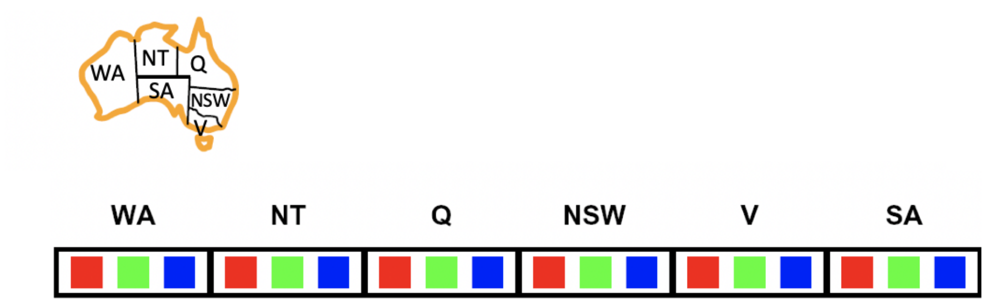
3. We can improve the search by checking the constraints as we go to get a backtracking search. This search is basically DFS with variable ordering and a fail on a constraint violation.

```
function BACKTRACKING-SEARCH(csp) returns solution/failure
   return RECURSIVE-BACKTRACKING({ }, csp)

function RECURSIVE-BACKTRACKING(assignment, csp) returns soln/failure
   if assignment is complete then return assignment
   var ← SELECT-UNASSIGNED-VARIABLE(VARIABLES[csp], assignment, csp)
   for each value in ORDER-DOMAIN-VALUES(var, assignment, csp) do
      if value is consistent with assignment given CONSTRAINTS[csp] then
         add {var = value} to assignment
         result ← RECURSIVE-BACKTRACKING(assignment, csp)
         if result ≠ failure then return result
         remove {var = value} from assignment
   return failure
```

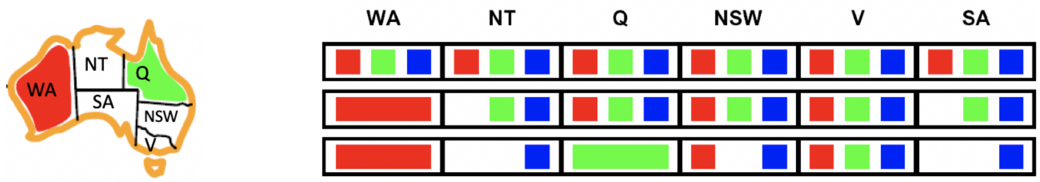
What are the choice points in this algorithm? We can improve things by picking wisely.

4. One technique to improve the backtracking performance is filtering. Filtering with *forward checking* keeps track of domains for unassigned variables and crosses off bad options. Forward checking crosses off values that violate a constraint when adding to the existing assignment.



What would get filtered with forward checking after we assign WA=red?
What would get filtered next after we assign Q=green?
What would get filtered next after we assign V=blue?
What would we do if a domain became empty after filtering?

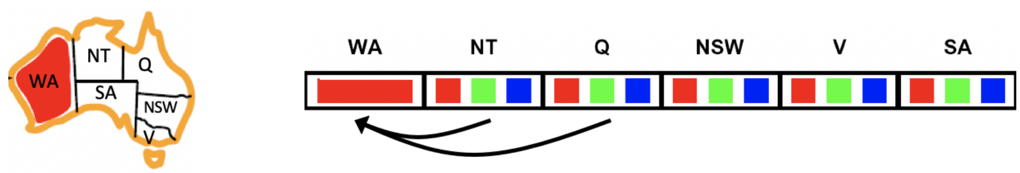
5. Forward checking propagates information from assigned to unassigned variables, but does not provide early detection for all failures:



What is the problem at this point?
Forward checking didn't catch this. What would it take to detect this?

6. *Constraint propagation* requires checking lots of pairs of variables throughout the graph to see if anything is broken. We will examine how to do that.

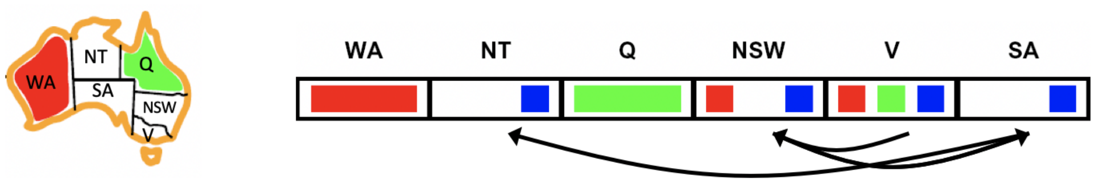
An arc (or edge) in the constraint graph is *consistent* if and only if for every x in the tail of the arc there is some y in the head which could be assigned without violating a constraint.



Is the arc from NT to WA consistent? If not, what would you have to delete from NT to make it consistent?
Is the arc from Q to WA consistent? If not, what would you have to delete from Q to make it consistent?

We can now make forward checking enforce the consistency arc pointing to each new assignment.

7. We can also check the consistency of an entire CSP by checking that all of the arcs are consistent.

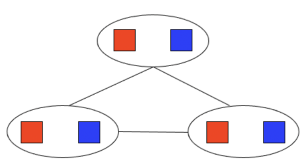


We will check if the arcs are consistent. Remember to delete from the tail of the arcs.

Check if V to NSW is consistent. What needs to be done if not?
Check if SA to NSW is consistent. What needs to be done if not?
Check if NSW to SA is consistent. What needs to be done if not?
Why would we need to recheck V to NSW at this point?

This is time intensive. In fact it is NP-hard since it is equivalent to the satisfiability problem.

8. There are limitations to arc consistency since we are checking consistency by pairs.



What went wrong here in this consistent graph?

9. We can choose the next variable to assign wisely. One way to do so is MRV (Minimum Remain Values). This says that the next variable to assign is the one with the minimum possible choices. Why should we use minimum rather than maximum?

Lastly, we can choose the value to assign wisely. One way is Least Constraining Value where we choose a value we causes the least constraints.

To give you a feeling for the performance, the plain vanilla backtracking algorithm can easily solve n-queens fro an n around 25. Combining the techniques we just described allows for n=1000 for the n-queens problem.