```
function BACKTRACKING-SEARCH(csp) returns a solution or failure
  return BACKTRACK(csp, { })

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

Figure 6.5 A simple backtracking algorithm for constraint satisfaction problems. The algorithm is modeled on the recursive depth-first search of Chapter 3. The functions SELECT-UNASSIGNED-VARIABLE and ORDER-DOMAIN-VALUES implement the general-purpose heuristics discussed in Section 6.3.1. The INFERENCE function can optionally impose arc-, path-, or k-consistency, as desired. If a value choice leads to failure (noticed either by INFERENCE or by BACKTRACK), then value assignments (including those made by INFERENCE) are retracted and a new value is tried.

$SA = blue$, but we would never choose between $NSW = red$ and $SA = blue$. With this restriction, the number of leaves is d^n, as we would hope. At each level of the tree we do have to choose which variable we will deal with, but we never have to backtrack over that choice.

Figure 6.5 shows a backtracking search procedure for CSPs. It repeatedly chooses an unassigned variable, and then tries all values in the domain of that variable in turn, trying to extend each one into a solution via a recursive call. If the call succeeds, the solution is returned, and if it fails, the assignment is restored to the previous state, and we try the next value. If no value works then we return failure. Part of the search tree for the Australia problem is shown in Figure 6.6, where we have assigned variables in the order $WA, NT, Q, \ldots$.

Notice that BACKTRACKING-SEARCH keeps only a single representation of a state (assignment) and alters that representation rather than creating new ones (see page 80).

Whereas the uninformed search algorithms of Chapter 3 could be improved only by supplying them with *domain-specific* heuristics, it turns out that backtracking search can be improved using *domain-independent* heuristics that take advantage of the factored representation of CSPs. In the following four sections we show how this is done:

- (6.3.1) Which variable should be assigned next (SELECT-UNASSIGNED-VARIABLE), and in what order should its values be tried (ORDER-DOMAIN-VALUES)?
- (6.3.2) What inferences should be performed at each step in the search (INFERENCE)?
- (6.3.3) Can we BACKTRACK more than one step when appropriate?
- (6.3.4) Can we save and reuse partial results from the search?