```
function AND-OR-SEARCH(problem) returns a conditional plan, or failure
   return OR-SEARCH(problem, problem.INITIAL, [])

function OR-SEARCH(problem, state, path) returns a conditional plan, or failure
   if problem.IS-GOAL(state) then return the empty plan
   if IS-CYCLE(state, path) then return failure
   for each action in problem.ACTIONS(state) do
      plan ← AND-SEARCH(problem, RESULTS(state, action), [state] + path)
      if plan ≠ failure then return [action] + plan
   return failure

function AND-SEARCH(problem, states, path) returns a conditional plan, or failure
   for each $s_i$ in states do
      $plan_i$ ← OR-SEARCH(problem, $s_i$, path)
      if $plan_i$ = failure then return failure
   return [if $s_1$ then $plan_1$ else if $s_2$ then $plan_2$ else ... if $s_{n-1}$ then $plan_{n-1}$ else $plan_n$]
```

Figure 4.11 An algorithm for searching AND–OR graphs generated by nondeterministic environments. A solution is a conditional plan that considers every nondeterministic outcome and makes a plan for each one.

4.3.3 Try, try again

Consider a *slippery* vacuum world, which is identical to the ordinary (non-erratic) vacuum world except that movement actions sometimes fail, leaving the agent in the same location. For example, moving *Right* in state 1 leads to the belief state $\{1,2\}$. Figure 4.12 shows part of the search graph; clearly, there are no longer any acyclic solutions from state 1, and AND-OR-SEARCH would return with failure. There is, however, a **cyclic solution**, which is to keep trying *Right* until it works. We can express this with a new **while** construct:

Cyclic solution

$$[Suck, \textbf{while}\ State = 5\ \textbf{do}\ Right, Suck]$$

or by adding a **label** to denote some portion of the plan and referring to that label later:

$$[Suck, L_1 :\ Right, \textbf{if}\ State = 5\ \textbf{then}\ L_1\ \textbf{else}\ Suck]\,.$$

When is a cyclic plan a solution? A minimum condition is that every leaf is a goal state and that a leaf is reachable from every point in the plan. In addition to that, we need to consider the cause of the nondeterminism. If it is really the case that the vacuum robot's drive mechanism works some of the time, but randomly and independently slips on other occasions, then the agent can be confident that if the action is repeated enough times, eventually it will work and the plan will succeed. But if the nondeterminism is due to some unobserved fact about the robot or environment—perhaps a drive belt has snapped and the robot will never move—then repeating the action will not help.

One way to understand this decision is to say that the initial problem formulation (fully observable, nondeterministic) is abandoned in favor of a different formulation (partially observable, deterministic) where the failure of the cyclic plan is attributed to an unobserved property of the drive belt. In Chapter 12 we discuss how to decide which of several uncertain possibilities is more likely.