SEARCH WITH NONDETERMINISTIC ACTIONS

SEARCH

- After identifying the path agent should continue paying attention to percept or close it?

1. Observable, Deterministic and Known Environment: Close percept

   I'm in state S_1 and if I do action a, I'll end up in state S_2.

2. Partially Observable and/or Nondeterministic: Continue looking at percept

PARTIALLY OBSERVABLE V/S NONDETERMINISTIC

- Environment is partially observable => the agent doesn't know for sure what state it is in.

- Environment is nondeterministic => the agent doesn't know what state it transitions to after taking an action.

I'm either in state S_1 or S_2 , and if I do action a, I'll end up in state S_3, S_4 or S_5.

- A set of physical states that the agent believes are possible are **belief state**.

CONDITIONAL PLAN (STRATEGY)

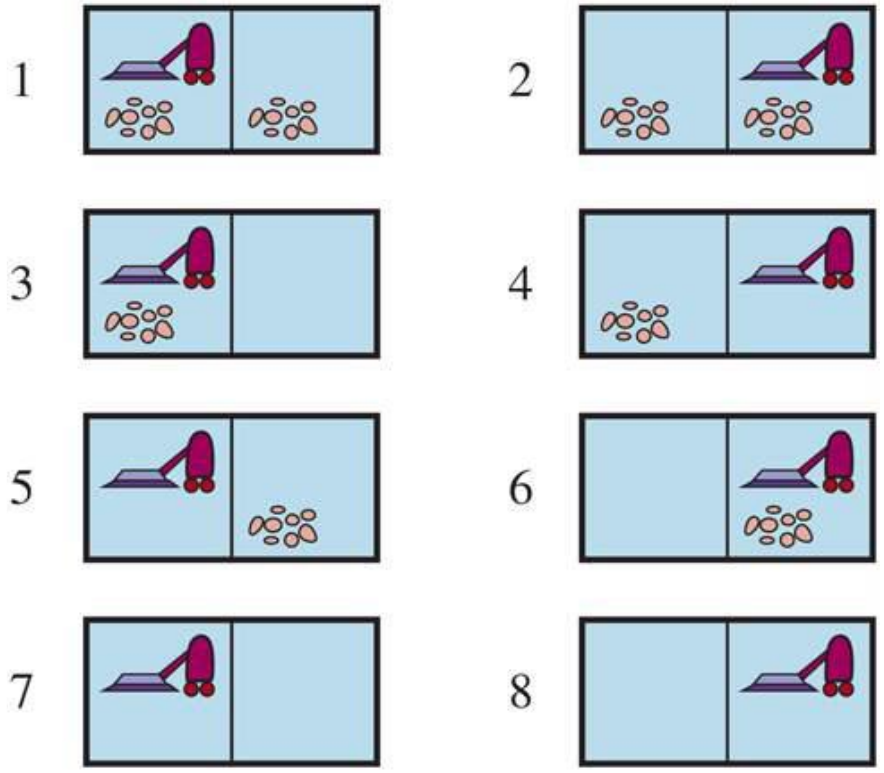
- In partially observable and nondeterministic environments, the solution to a problem is no longer a sequence, but rather a conditional plan (sometimes called a contingency plan or a strategy) that specifies what to do depending on what percepts agent receives while executing the plan.
- Search in Nondeterministic environments

1. The erratic vacuum world
2. AND–OR search trees
3. Try, try again

THE ERRATIC VACUUM WORLD

- Suck action works as follows:

1. When applied to a dirty square the action cleans the square and sometimes cleans up dirt in an adjacent square, too.
2. When applied to a clean square the action sometimes deposits dirt on the carpet.



TRANSITION MODEL & CONDITIONAL PLAN

- Suck action in state 1

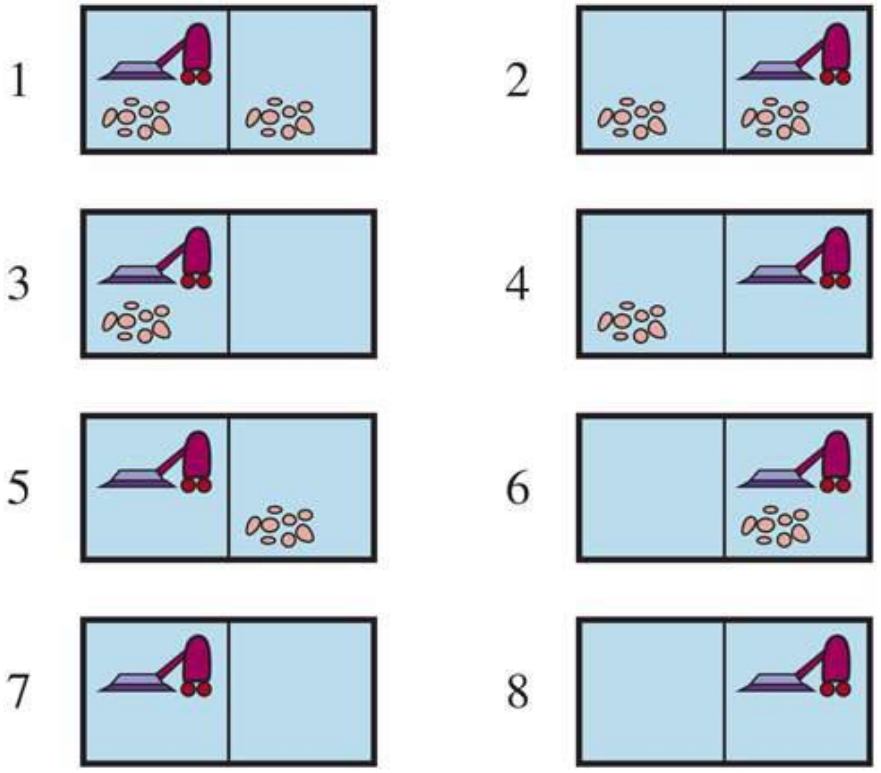
  RESULTS(1, Suck) = {5,7}

- Conditional plan

[Suck, if State = 5 then [Right, Suck] else [ ]]

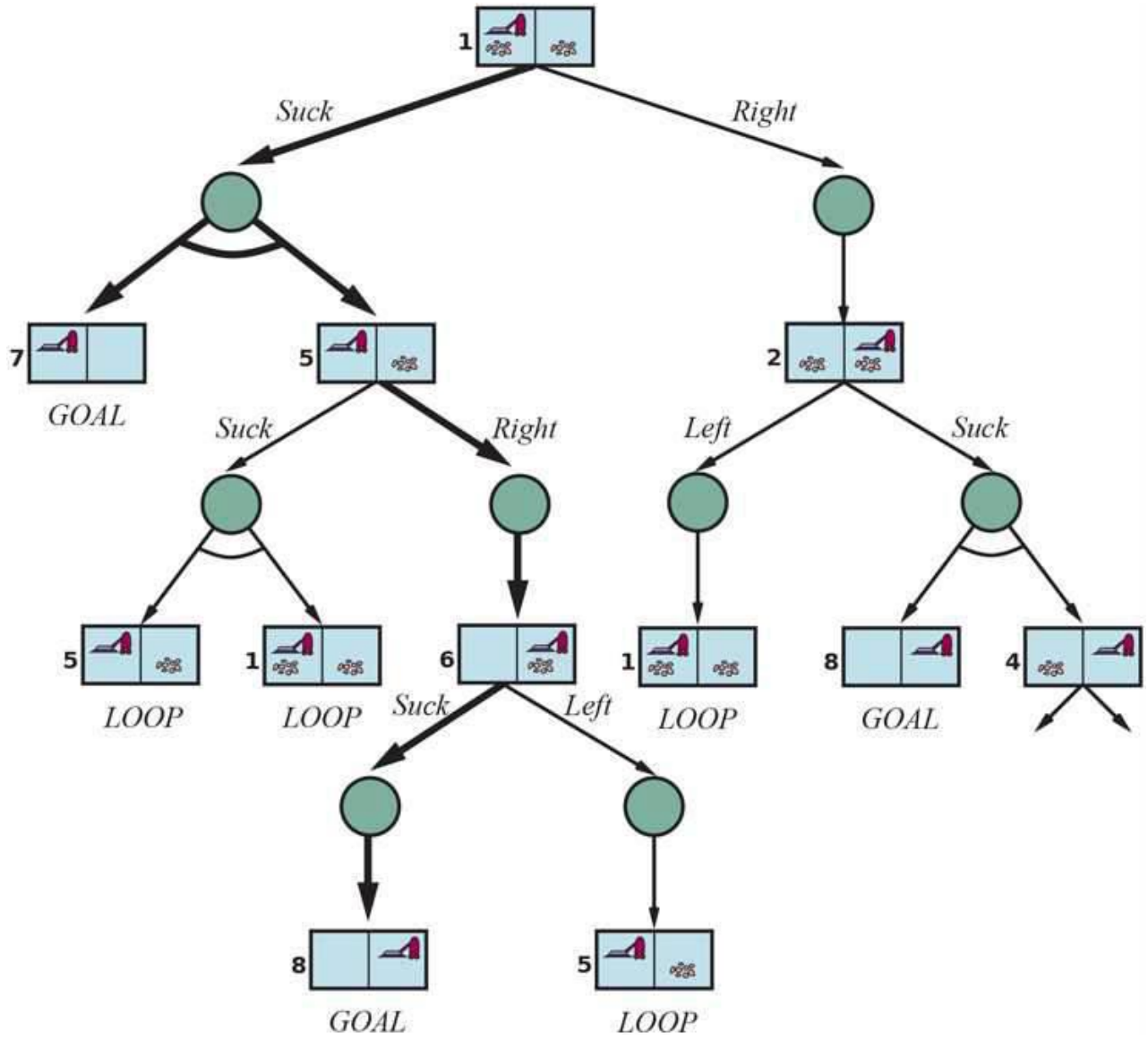
if–then–else steps

Solutions are trees rather than sequences.



AND–OR SEARCH TREES

- In a deterministic environment, the only branching is introduced by the agent's own choices in each state: I can do this action or that action. We call these nodes **OR nodes**.
- In the vacuum world, for example, at an OR node, the agent chooses Left or Right or Suck.
- In a nondeterministic environment, branching is also introduced by the environment's choice of outcome for each action. We call these nodes **AND nodes**.
- For example, the Suck action in state 1 results in the belief state , so the agent would need to find a plan for state 5 and for state 7.



AND–OR SEARCH TREES

- Solution is subtree of the complete search tree that
  1. Has a goal node at every leaf.
  2. Specifies one action at each of its OR nodes.
  3. Includes every outcome branch at each of its AND nodes.

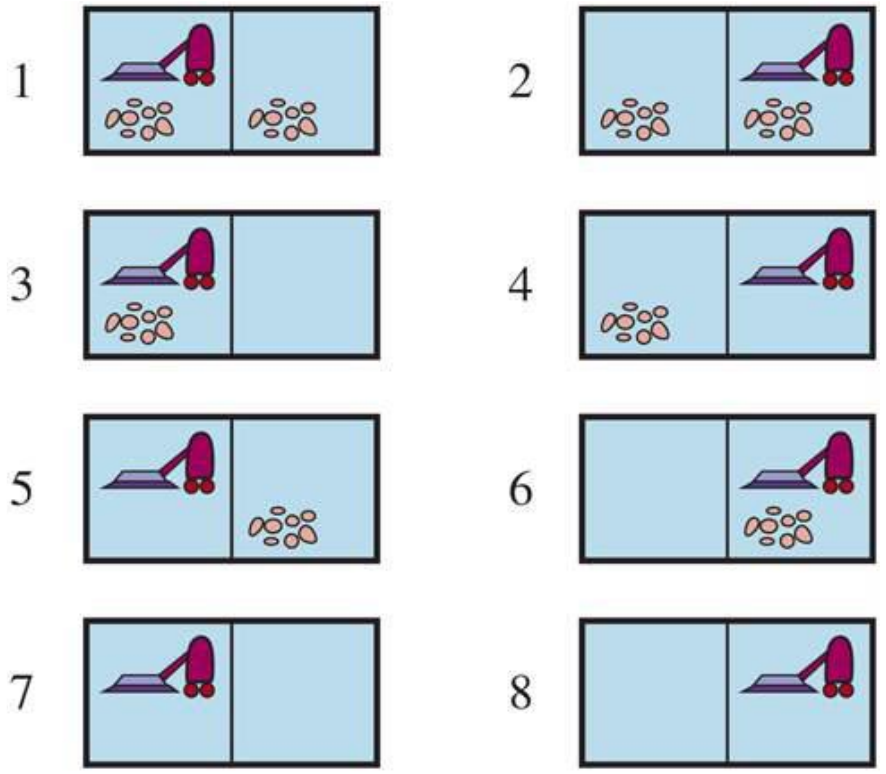
```
function AND-OR-SEARCH(problem) returns a conditional plan, or failure
  return OR-SEARCH(problem, problem.INITIAL, [])

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(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]
```

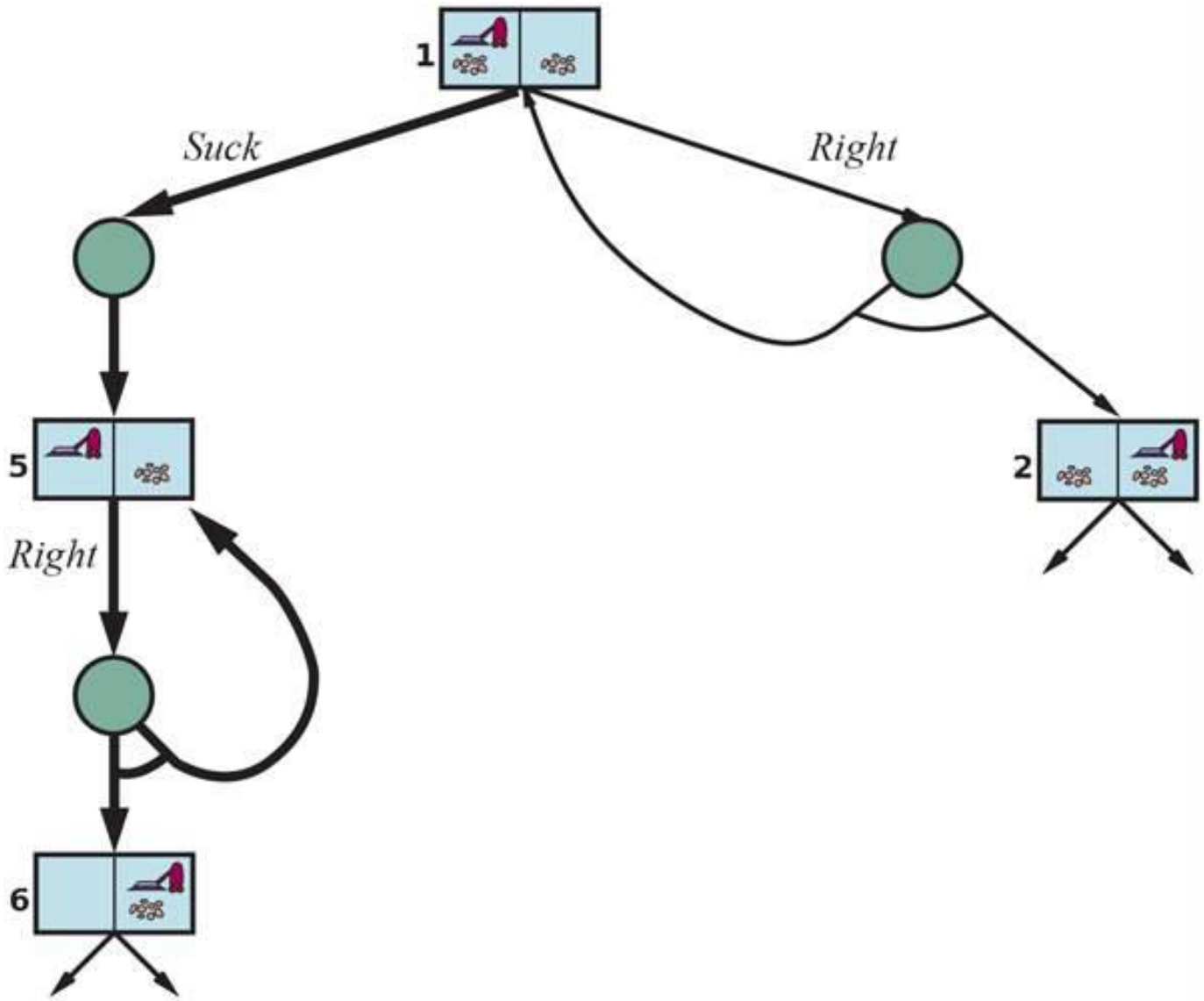
ALGORITHM FOR SEARCHING AND-OR GRAPHS

- Recursive, depth-first algorithm.
- If the current state is identical to a state on the path from the root, then it returns with failure.
- Goal, a dead end, or a repeated state

TRY, TRY AGAIN

- Slippery vacuum world
- Right in state 1 leads to the belief state {1,2}.
- AND OR- SEARCH would return with failure. (No Acyclic solutions)





AND-OR SEARCH TREES

- **Cyclic solution:**

Keep trying Right until it works.

[Suck, while State = 5 do Right, Suck]

[Suck,L1 : Right, if State = 5 then L1 else Suck]