CSCE A405/A605 (Adv) Artificial Intelligence

Search in Complex Environments

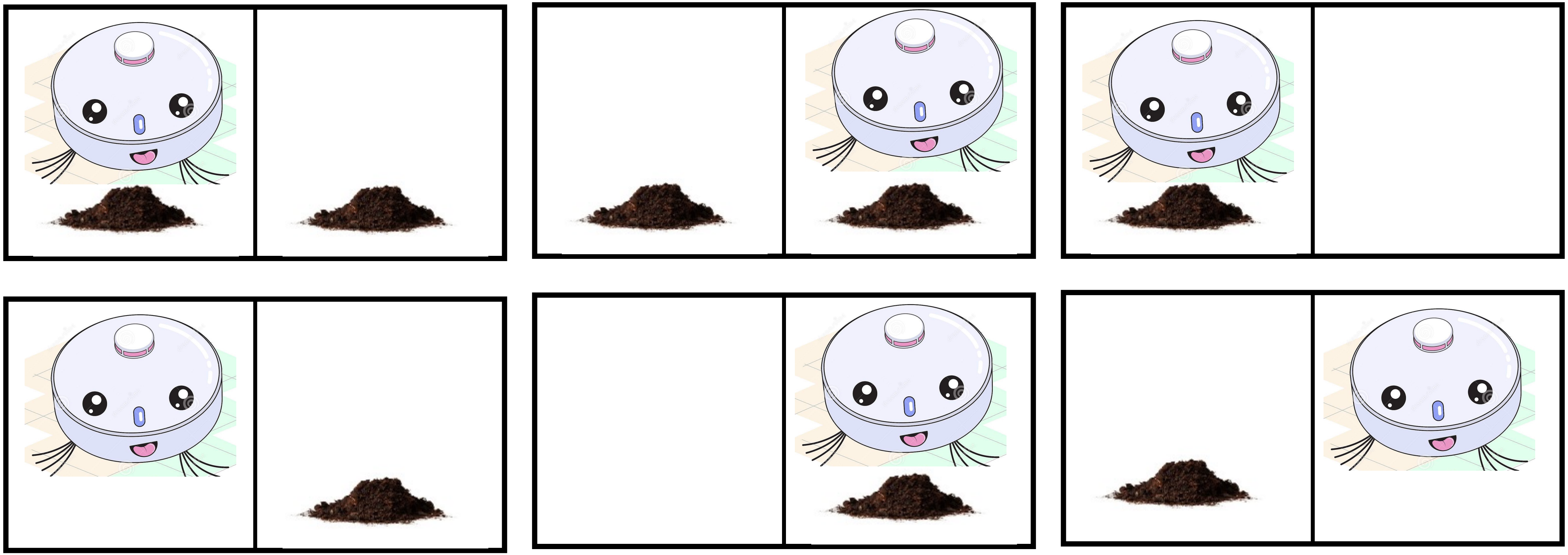
Ref: Artificial Intelligence: A Modern Approach, 4th ed by Stuart Russell and Peter Norvig, chapter 4

Instructor: Masoumeh Heidari (mheidari2@Alaska.edu)

UAA College of Engineering
UNIVERSITY of ALASKA ANCHORAGE

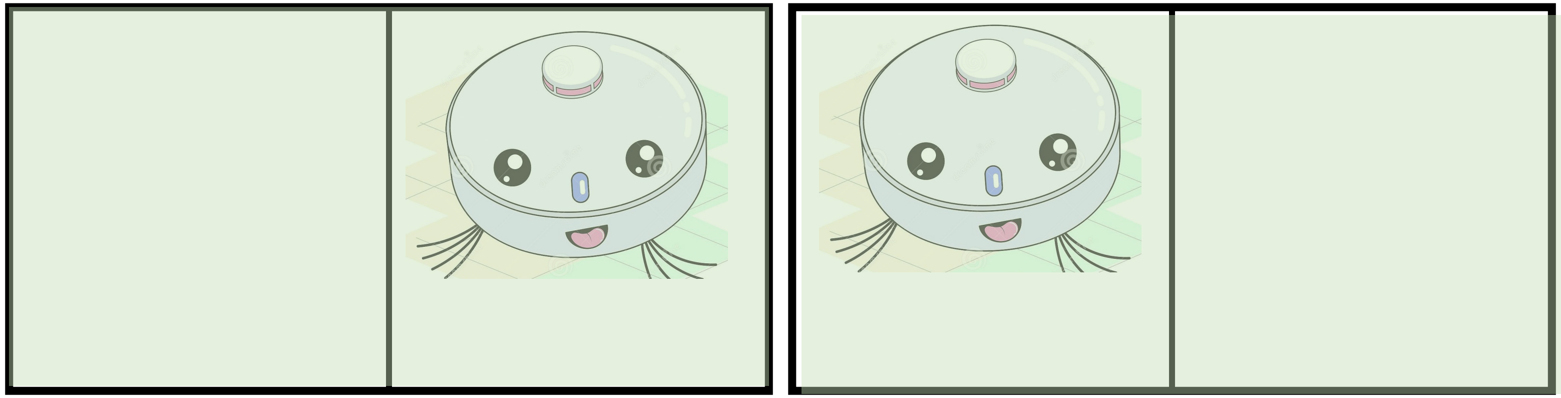
The erratic vacuum world

Eight states



Actions:

- *Right*
- *Left*
- *Suck*
- *Do nothing*



Goal states

The erratic vacuum world

- In the erratic vacuum world, the suck action works as follows:
  - When applied to a dirty square the action cleans the square and sometimes cleans up dirt in an adjacent square, too.
  - When applied to a clean square the action sometimes deposits dirt on the carpet.

?

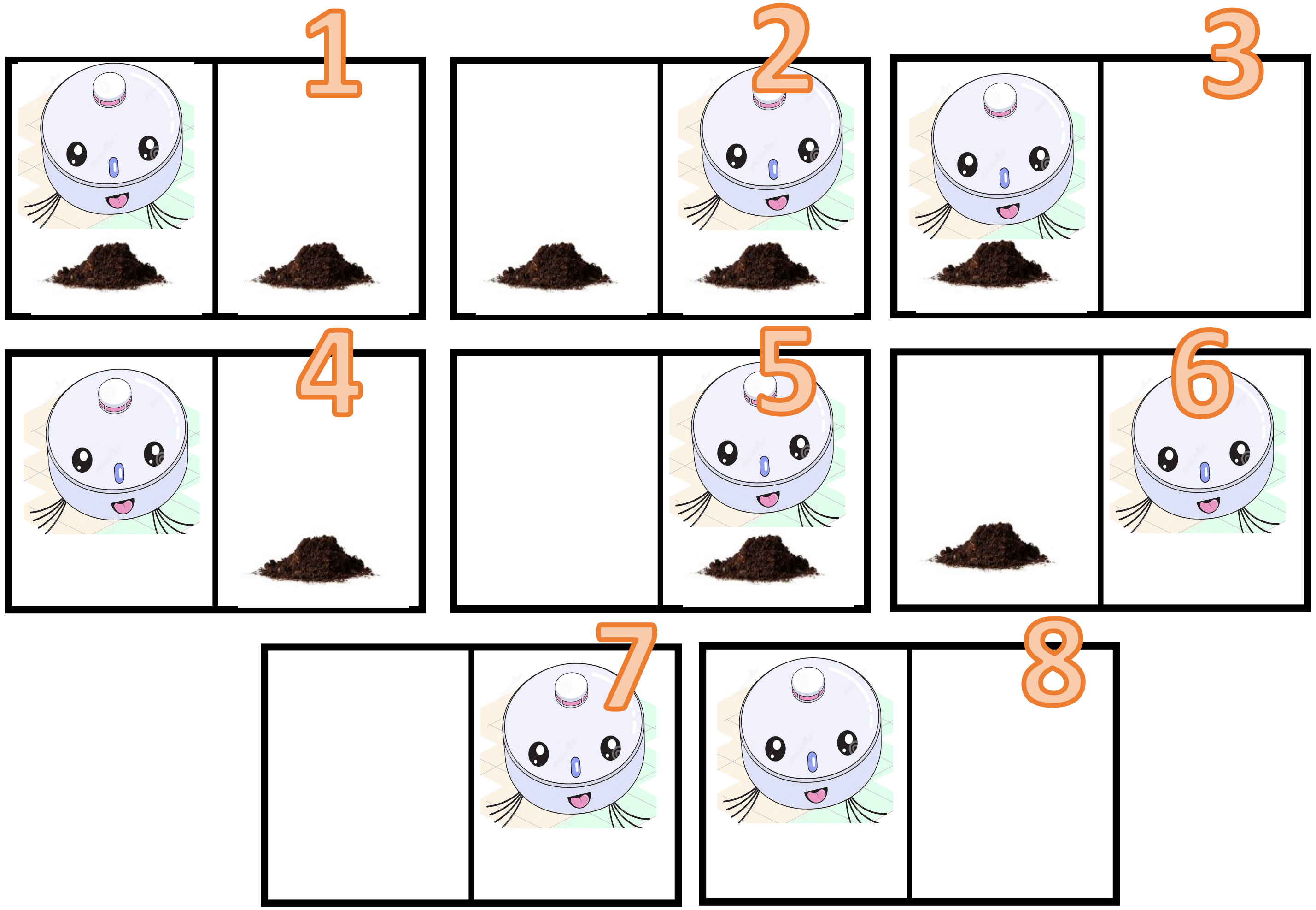
State space
Initial state
Goal state(s)
Actions
Transition model
Action cost function

Properties	
Fully observable	Partially observable
Single-agent	Multiagent
Deterministic	Nondeterministic
Episodic	Sequential
Static	Dynamic
Discrete	Continues
Known	Unknown

?

UAA College of Engineering
UNIVERSITY of ALASKA ANCHORAGE

The erratic vacuum world



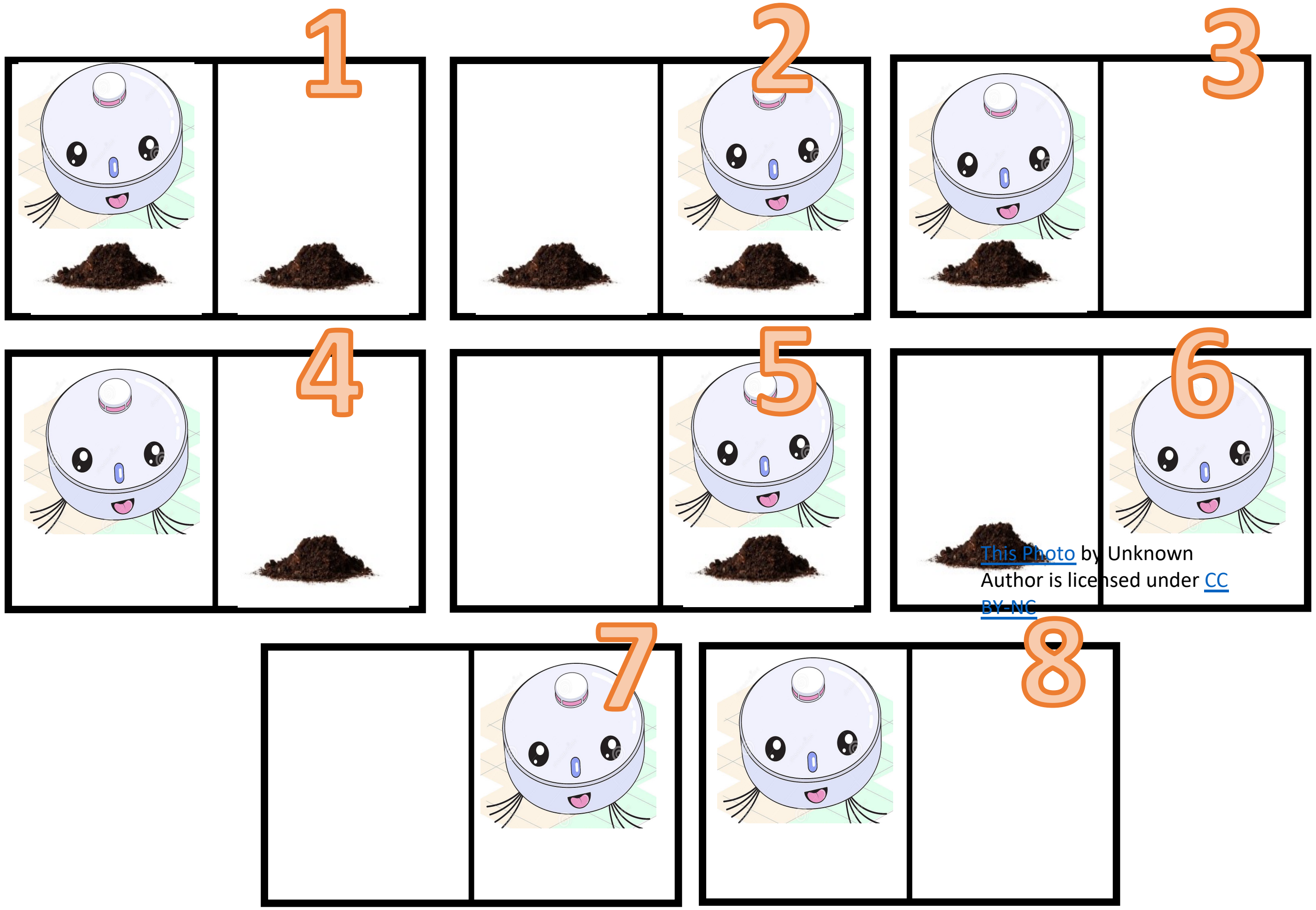
Results (1, Suck) = 4

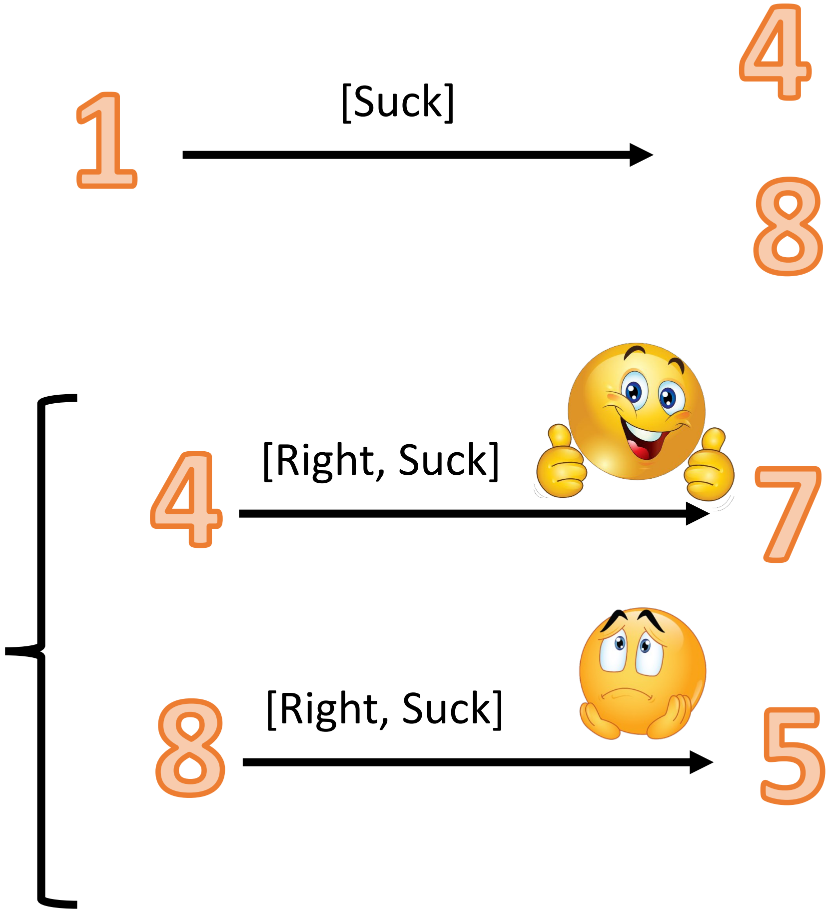
Results (1, Suck) = {4,8}

$1 \xrightarrow{\text{[Suck, Right, Suck]}} 7$

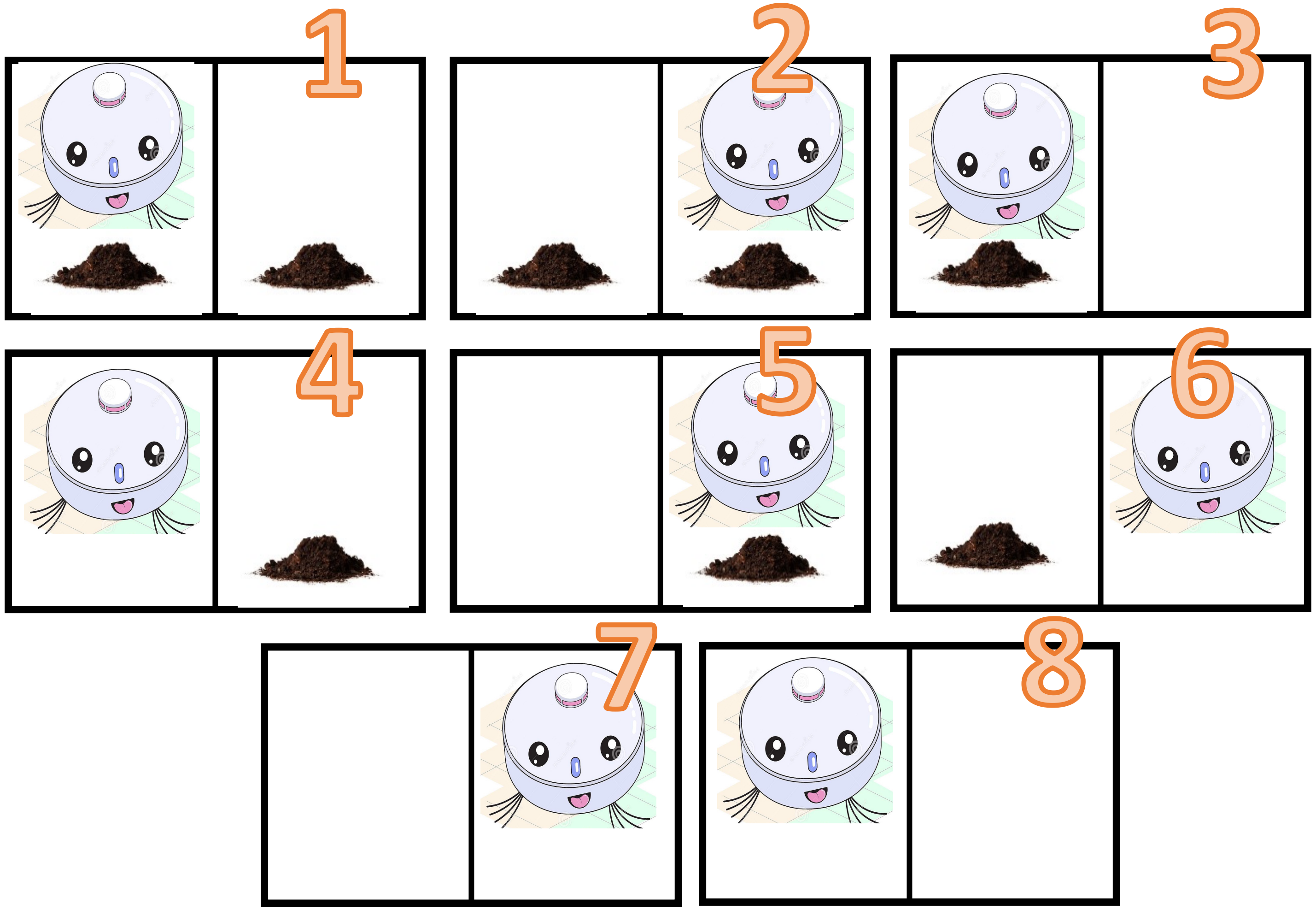
$1 \xrightarrow{\text{[?]}}$

The erratic vacuum world





The erratic vacuum world



[Suck, **if** State = 4 **then** [Right , Suck] **else** []]

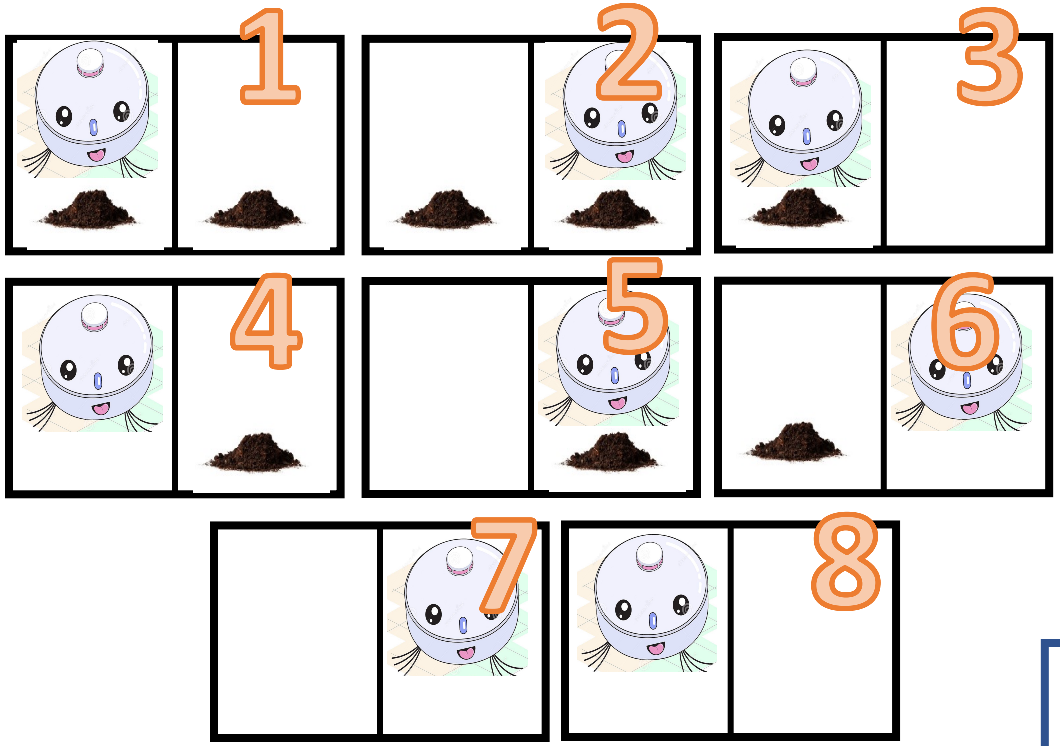
1 → 7

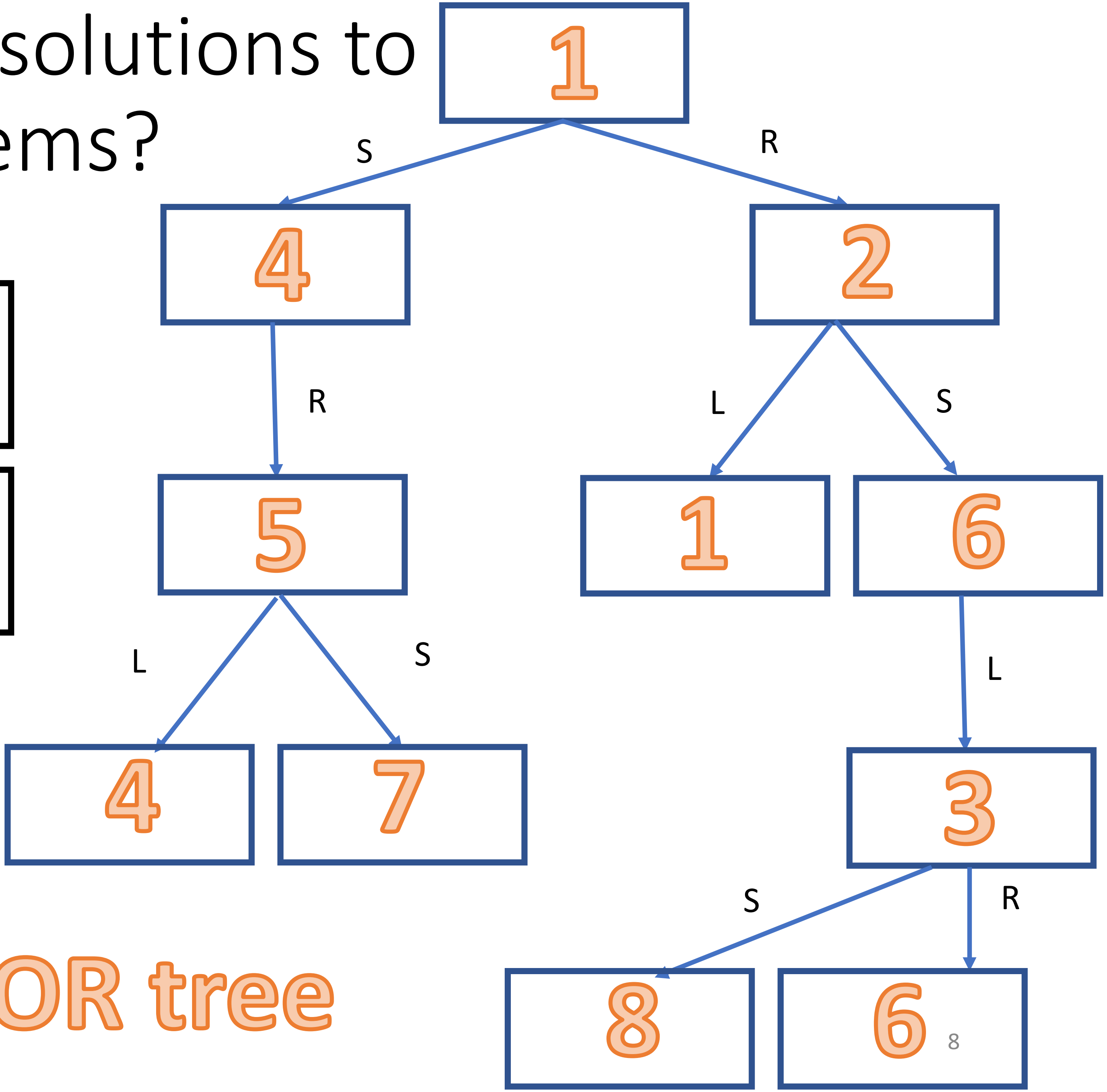
Conditional Plan

Conditional plan

- A conditional plan can contain **if-then-else** steps.
- Solutions are **trees** rather than sequences.
- The conditional in the if statement tests to see what the current state is; this is something the agent <u>will be able to observe at runtime, but doesn't know at planning time</u>.
- Many problems in the real, physical world are contingency problems, because exact prediction of the future is impossible.

How to find contingent solutions to nondeterministic problems?





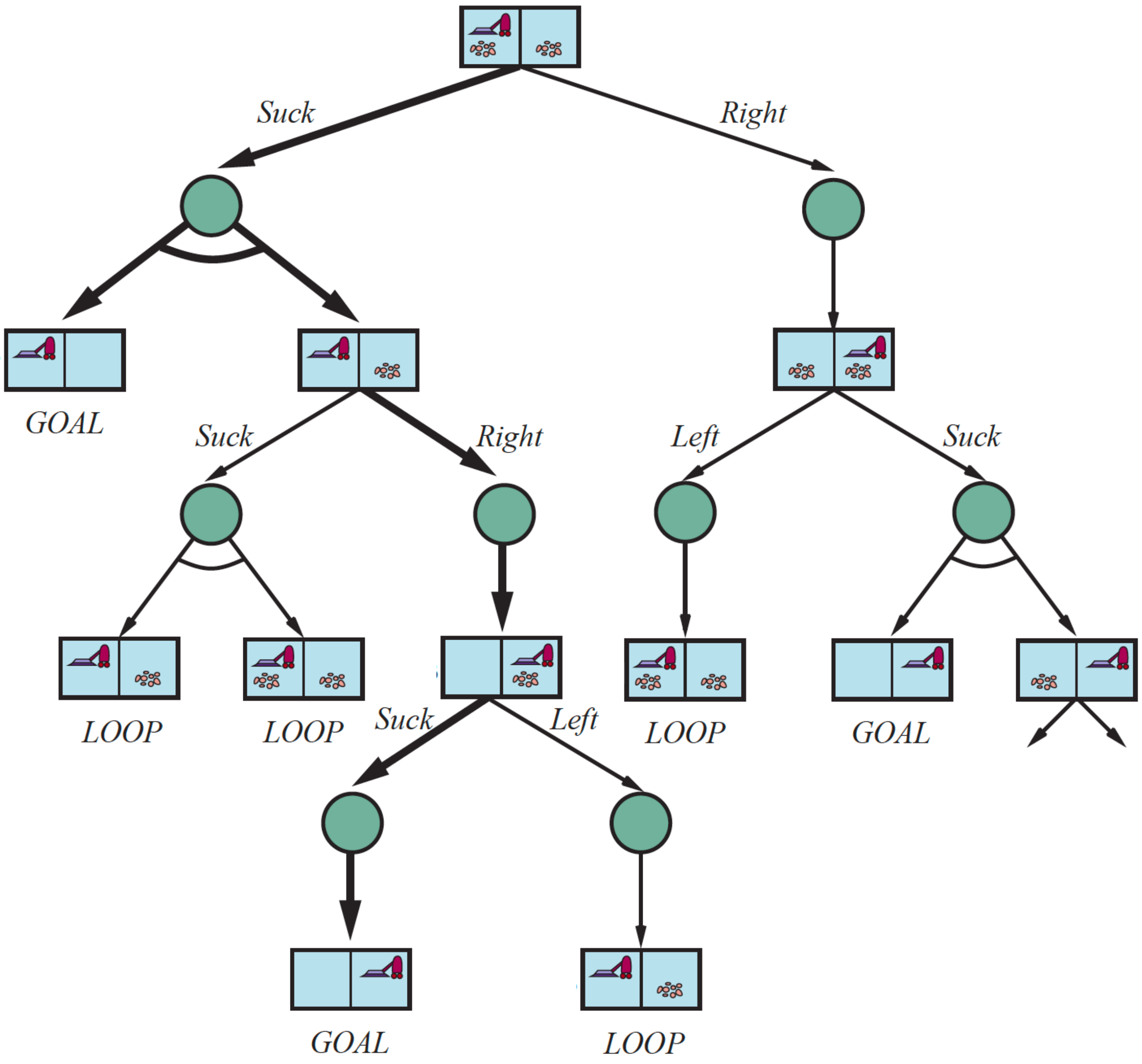
OR tree

How to find contingent solutions to nondeterministic problems?

- 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.
- 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 {4,8}, so the agent would need to find a plan for state 4 and for state 8.

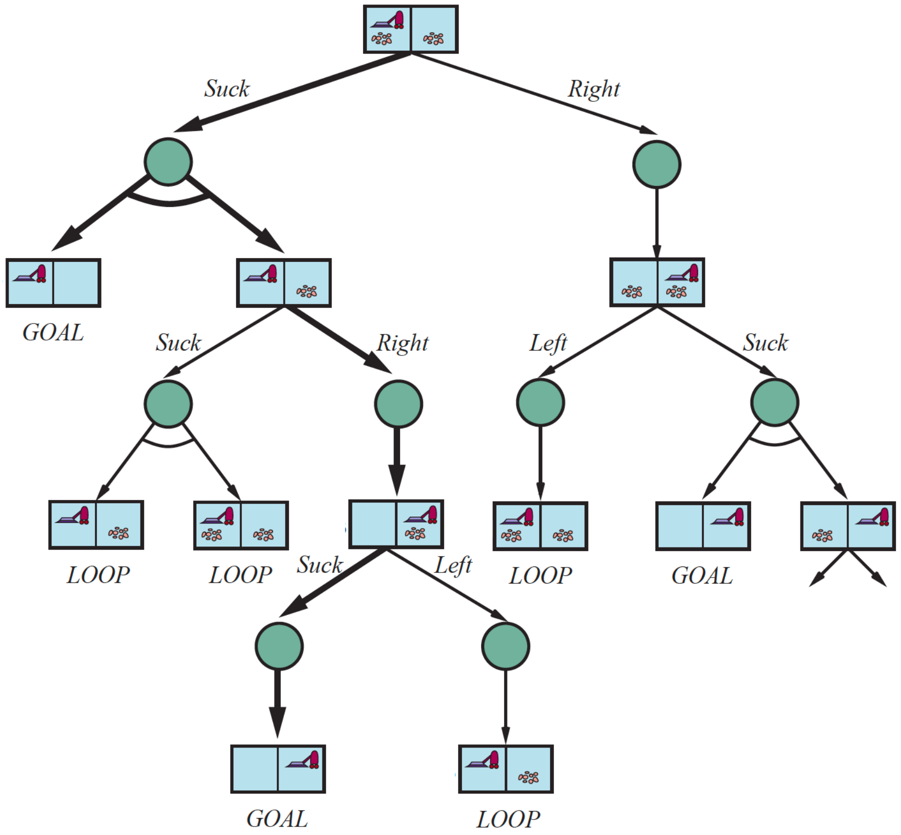
AND–OR tree

At the AND nodes, shown as circles, every outcome must be handled, as indicated by the arc linking the outgoing branches.



AND-OR search

- A solution for an AND–OR search problem is a 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.

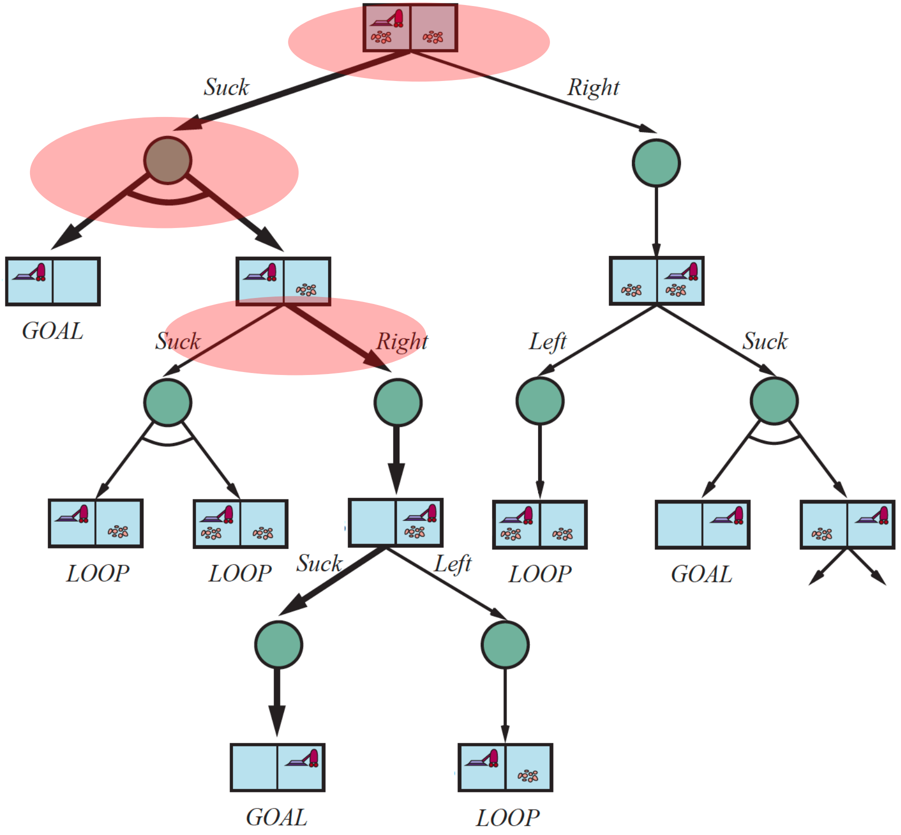


An algorithm for searching AND–OR graphs

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** $plan_n$]

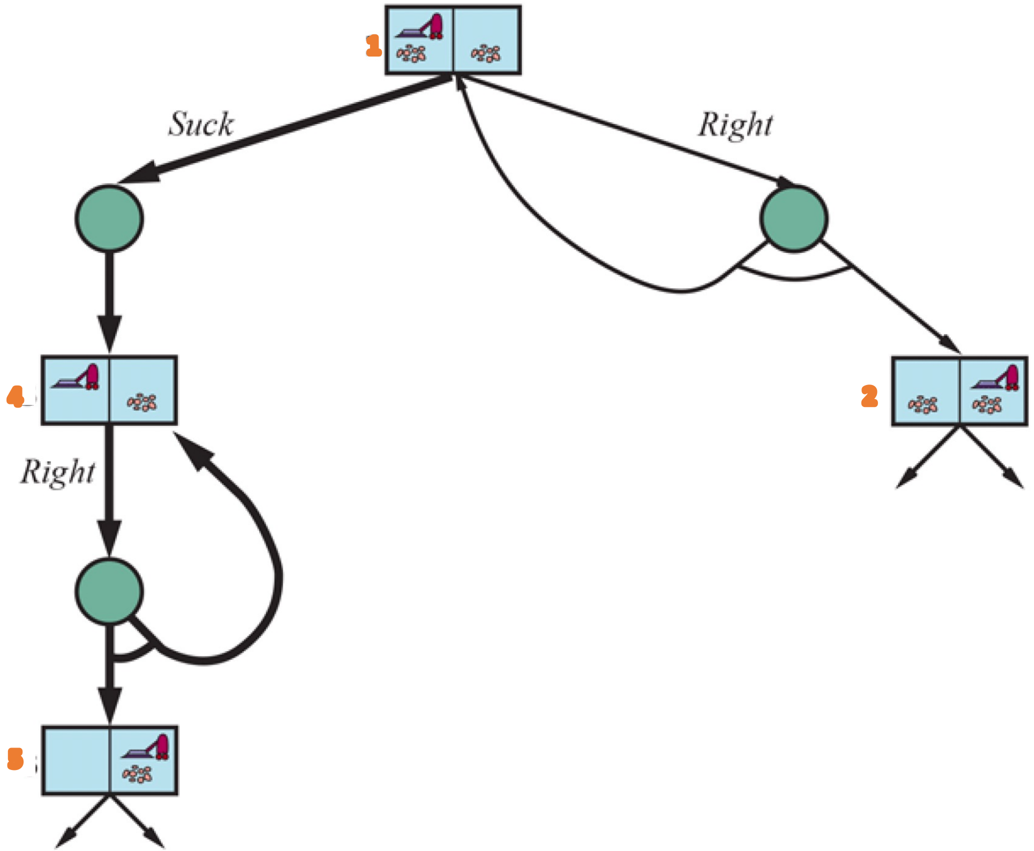


A slippery vacuum world

- It 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}.

A slippery vacuum world



- There are no longer any acyclic solutions from state 1,
- AND-OR search would return with failure.
- There is, however, a cyclic solution, which is to keep trying Right until it works.

How to implement?

While

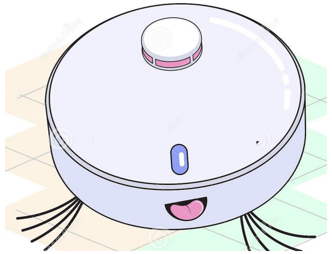
[Suck, **while** State=4 **do** Right, Suck]

When is a cyclic plan a solution?

- A minimum condition is that every leaf is a goal state.
- A leaf is reachable from every point in the plan.
- In addition to that, we need to consider the cause of the nondeterminism.
  - If the robot will never move, then the repeating action will not help.

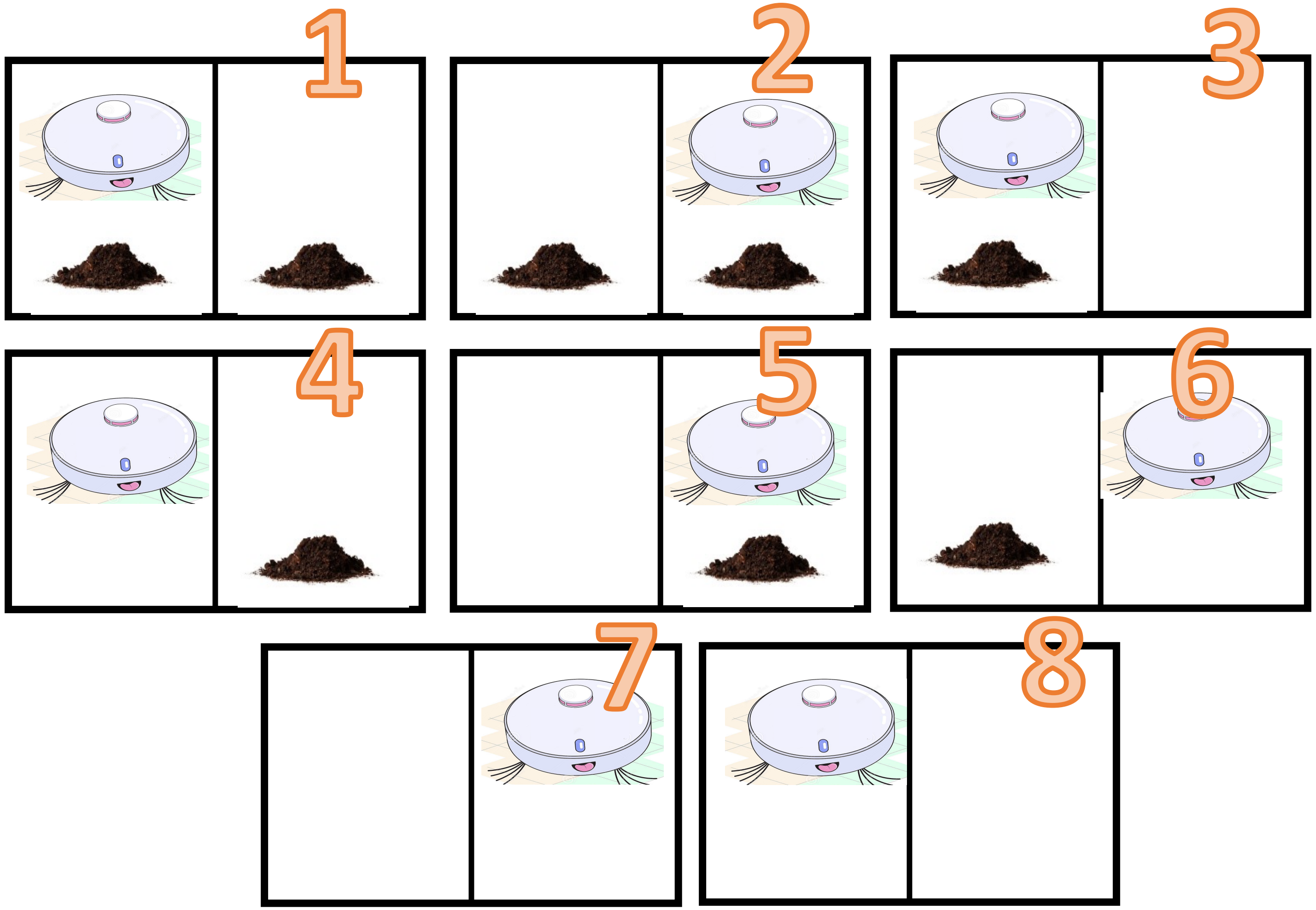
Sensorless vacuum world

- When the agent’s percepts provide *no information at all*, we have what is called a sensorless problem.



- The agent knows the geography of its world, but not its own location or distribution of dirt.
- Initial belief state?
  {1,2,3,4,5,6,7,8}

Sensorless vacuum world



- Initial belief state?
  {1,2,3,4,5,6,7,8}

What if the agent moves right?

Updated belief state: {2,5,6,7}

The agent has gained information without perceiving anything!

The action of moving right aimed to reduce uncertainty about the current state.

[Right, Suck] -> {6,7}
[Right, Suck, Left, Suck] -> {8}

Sensorless problems

- The solution to a sensorless problem is a sequence of actions, not a conditional plan (because there is no perceiving).
- We search in the space of belief state rather than physical states.
- The solution (if any) for a sensorless problem is always a sequence of actions.

Why?

- The percepts received after each action are completely predictable—they're always empty! So there are no contingencies to plan for. <u>This is true even if the environment is nondeterministic</u>.

Solution to a sensorless problem

- We can use the existing algorithms from Chapter 3 if we <u>transform the underlying physical problem into a belief-state problem</u>, in which we search over belief states rather than physical states.

- How to define the search problem?

State space
Initial state
Goal state(s)
Actions
Transition model
Action cost function

Solution to a sensorless problem

Assuming the original problem *P*, has components $Actions_P$, $Results_P$, etc., the belief-state problem has the following components:

- **States:** The belief-state space contains every possible subset of the physical states. If P has N states, then the belief-state. Problem has 2^N belief states, although many of those may be unreachable from the initial state.
- **Initial state:** Typically the belief state consisting of all states in *P*, although in some cases the agent will have more knowledge than this.

Solution to a sensorless problem

- **Goal state:** The agent possibly achieves the goal if any state in the belief state satisfies the goal test of the underlying problem, Is-Goal$_P$(s). The agent necessarily achieves the goal if every state satisfies Is-Goal$_P$(s).
- **Actions:**

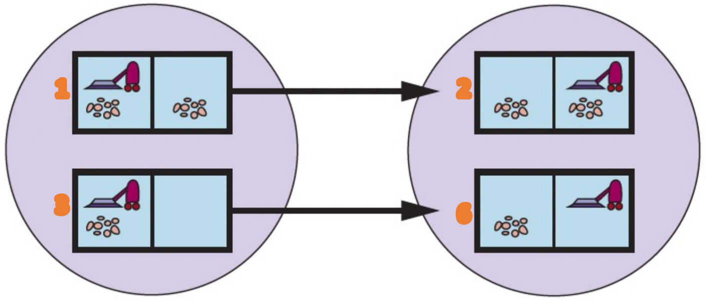
  Suppose the agent is in belief state b={s_1,s_2}, but Actions$_P$(s_1)$\neq$ Actions$_P$(s_2) *then the agent is unsure of which actions are legal.*

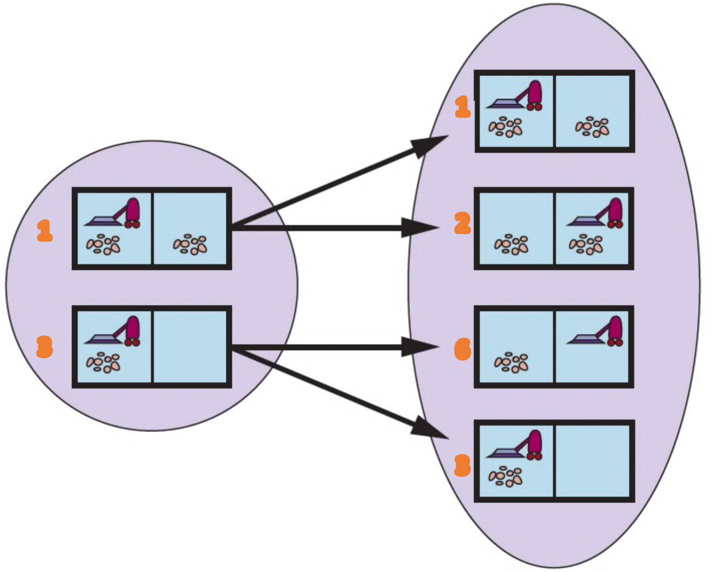
Illegal actions have no effect	Illegal actions might lead to catastrophe
Actions(b) = $\bigcup_{s \in b} Actions_P(s)$	Actions(b) = $\bigcap_{s \in b} Actions_P(s)$

Solution to a sensorless problem

- **Transition model:**

Deterministic	Nondeterministic
$b' = Result(b,a) = \{s' : s' = Result_P(s,a) \text{ and } s \in b\}$	$b' = Result(b,a) = \{s' : s' = Result_P(s,a) \text{ and } s \in b\} = \bigcup_{s \in b} Result_P(s,a)$





Solution to a sensorless problem

- **Action cost:** For now, we assume that the cost of an action is the same in all states and so can be transferred directly from the underlying physical problem.

- Sensorless problem-solving is seldom feasible in practice. Often a search space of size N is too large, and now we have search space of size 2^N.

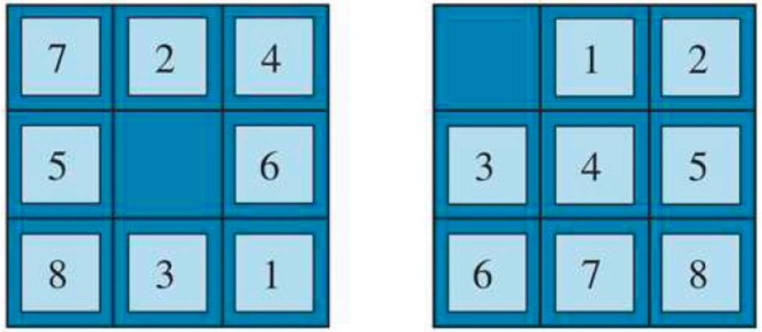
Solution to a sensorless problem

- What are the solutions?
  - One solution is to represent the belief state by some more compact description. For example, *"Not in the rightmost column".*

  - Instead of treating belief states as black boxes, we can look inside the belief states and develop incremental belief-state search.
    - Find a solution for state 1;
    - Then we check if it works for state 2;
    - If not, go back and find a different solution for state 1, and so on.
  - Just as an AND-OR search has to find a solution for every branch at AND node, this algorithm has to find a solution for every state in the belief state.
  - The difference is that AND-OR search can find a different solution for each branch, whereas an incremental belief-state search has to find one solution that works for all states.

Searching in partially observable environments

- Many problems cannot be solved without sensing. For example, the sensorless 8-puzzle is impossible.



Start State Goal State

- On the other hand, a little bit of sensing can go a long way: we can solve 8-puzzles if we can see just the upper-left corner square.

- The solution involves moving each tile in turn into the observable square and keeping track of its location from then on .

Searching in partially observable environments

- For a partially observable problem, the problem specification will specify a PERCEPT(s) function that returns the percept received by the agent in a given state.

- If sensing is nondeterministic, then we can use a PERCEPTS function that returns a set of possible percepts.

Fully Observable Problems	Sensorless problems
Percept(s)= s	Percept(s)= null

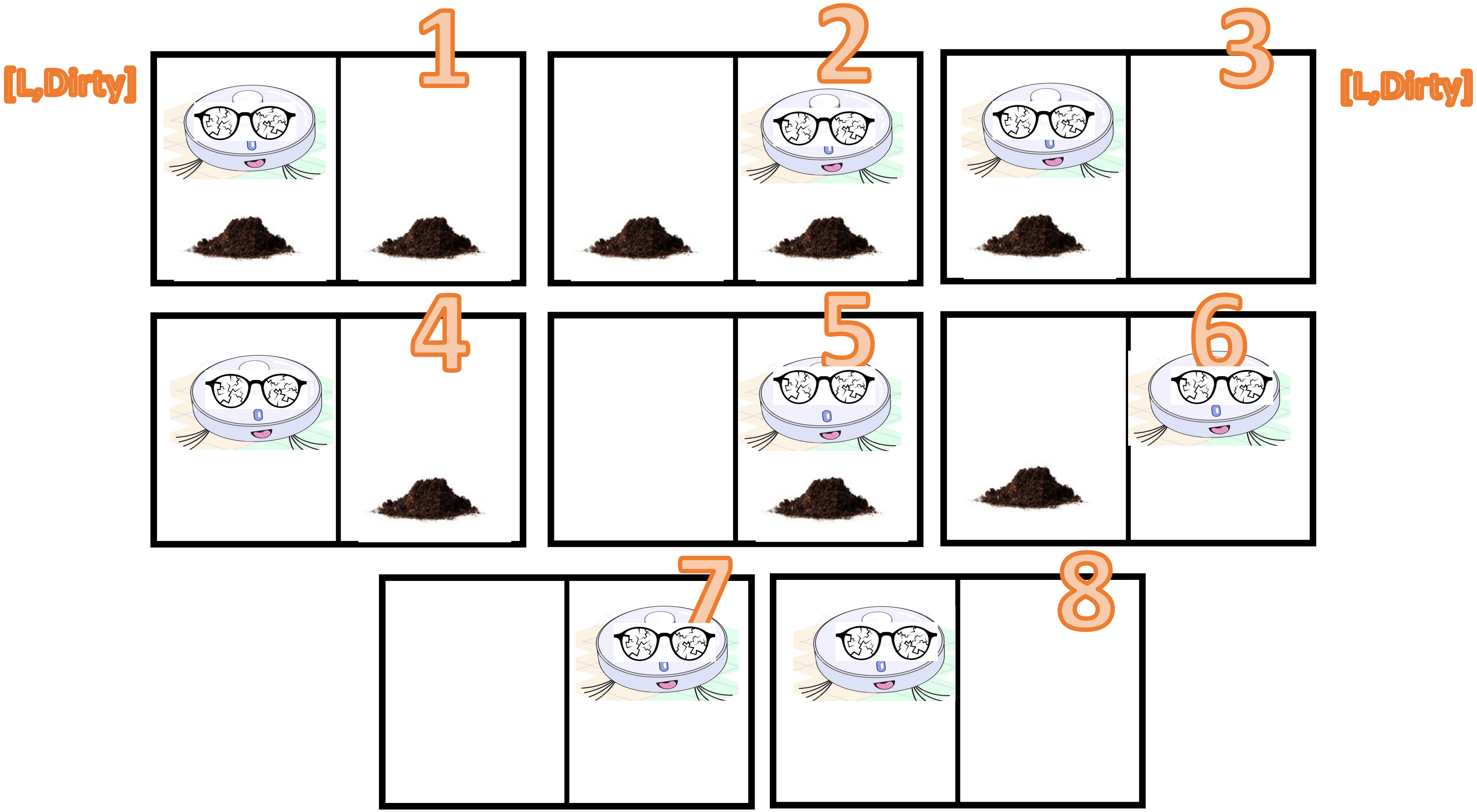
Local-sensing vacuum world

- With partial observability, it will usually be the case that <u>several states produce the same percept</u>;

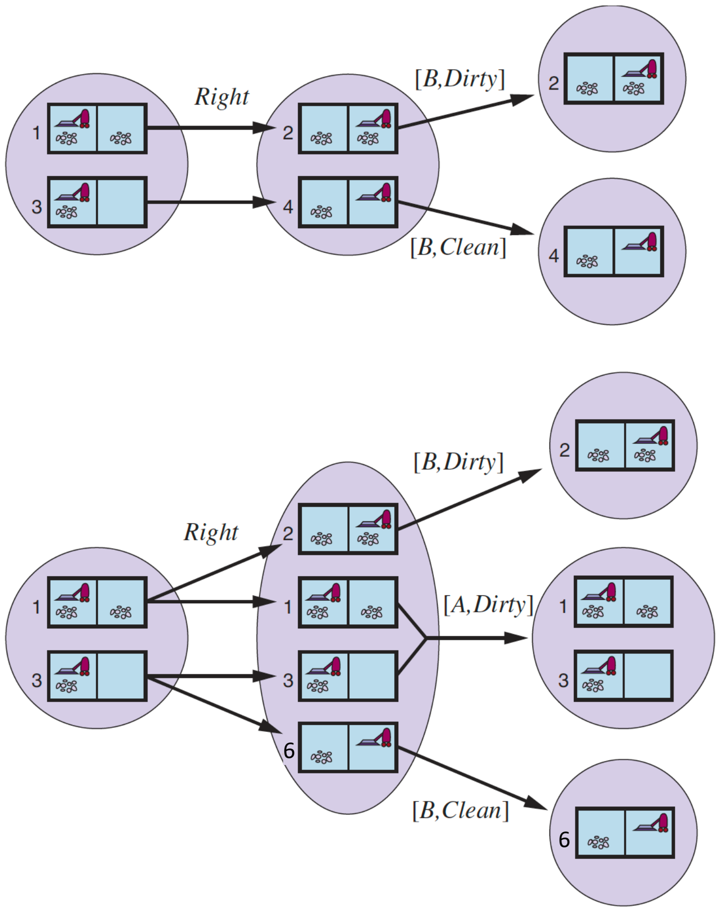
- Consider a local-sensing vacuum world, in which the agent has a position sensor that yields the percept *L* in the left square, and *R* in the right square, and

- A dirt sensor that yields *Dirty* when the current square is dirty and *Clean* when it is clean.

Local-sensing vacuum world



Local-sensing vacuum world



We can think of the transition model between belief states for partially observable problems as occurring in three stages:

1. The prediction stage computes the belief state resulting from the action: $\hat{b} = PREDICT$(b,a).

2. The possible percepts stage computes the set of percepts that could be observed in the predicted belief state:
   $POSSIBLE - PERCEPT(\hat{b})$ = {o: o = $PERCEPT(s)$ and $s \in \hat{b}$}

3. The update stage computes, for each possible percept, the belief state that would result from percept:
   $b_o = UPDATE(\hat{b}, o) = \{s: o = PERCEPT(s)\ and\ s \in \hat{b}\}$

Local-sensing vacuum world

- The agent needs to deal with possible percepts at planning time, because it won't know the actual percepts until it executes the plan.

- Possible belief states resulting from a given action and the subsequent possible percepts:

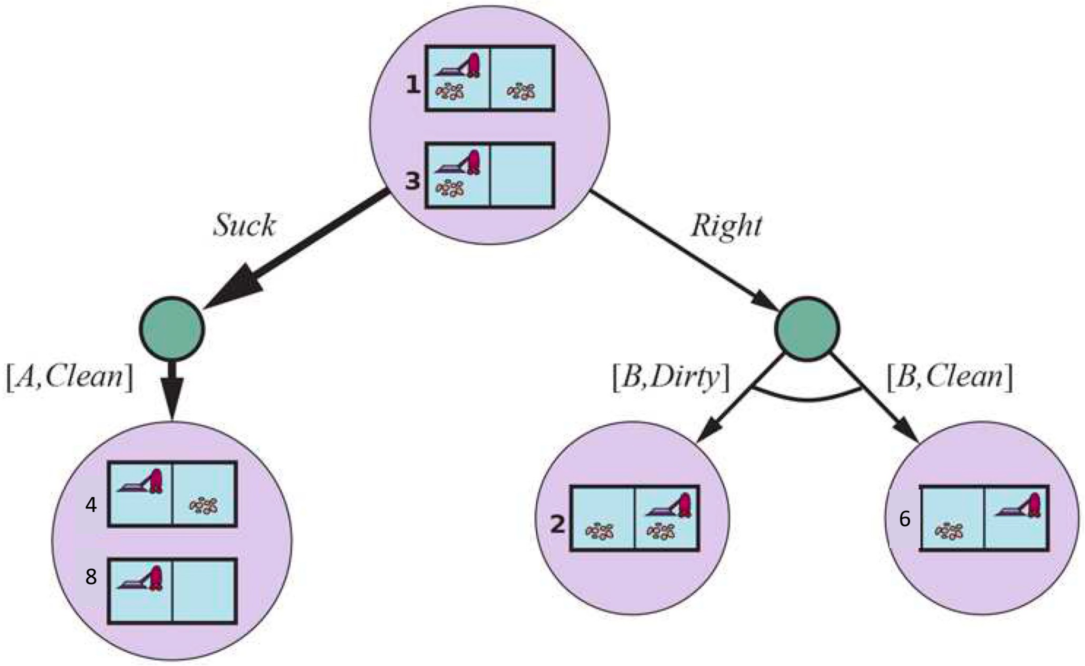
$$Results(b,a) = \{b_o : b_o = UPDATE(PREDICT(b,a),o) \; and \; o \in POSSIBLE - PERCEPTS(PREDICT(b,o))\}$$

UAA College of Engineering
UNIVERSITY of ALASKA ANCHORAGE

Solving partially observable problems

- AND-OR search algorithm can be applied directly to drive a solution.

- Because we supplied a belief-state problem to the AND–OR search algorithm, it returns a conditional plan that tests the belief state rather than the actual state.



An agent for partially observable environments

- An agent for partially observable environments formulates a problem, calls a search algorithm (such as AND-OR-SEARCH) to solve it, and executes the solution.

- There are two main differences between this agent and the one for fully observable deterministic environments.
    1. The solution will be a conditional plan rather than a sequence.
    2. The agent will need to maintain its belief state as it performs actions and receives percepts.