

CS 5/7320
Artificial Intelligence

Search with Uncertainty

AIMA Chapters 4.3-4.5

Slides by Michael Hahsler
with figures from the AIMA textbook



This work is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/).



Types of uncertainty we consider for now*



Nondeterministic Actions:

Outcome of an action in a state is uncertain.



No observations:

Sensorless problem



Partially observable environments:

The agent does not know in what state it is.



Unknown environments:

Online search

* we will quantify uncertainty with probabilities later.

Consequence of Uncertainty

- **Remember:** The solution for the known maze was a fixed **sequence of actions** from start to goal.
- **With uncertainty:** Solution is not a precomputed sequence, but a
conditional plan (a strategy or policy)
that depends on percepts.

A dynamic background image showing a bright yellow powder or smoke explosion against a solid black background. The particles are concentrated on the right side and spread out towards the left, creating a sense of motion and energy.

Nondeterministic Actions

Nondeterministic Actions

Outcome of actions in the environment is nondeterministic = **transition model need to describe uncertainty**

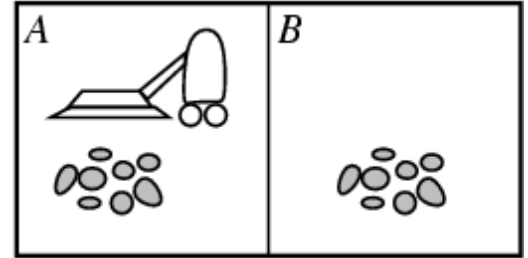
Example transition:

$$Results(s_1, a) = \{s_2, s_4, s_5\}$$

i.e., action a in s_1 can lead to one of several states.

The set of possible states $b = \{s_2, s_4, s_5\}$ is called a **belief state** of the agent.

Example: Erratic Vacuum World

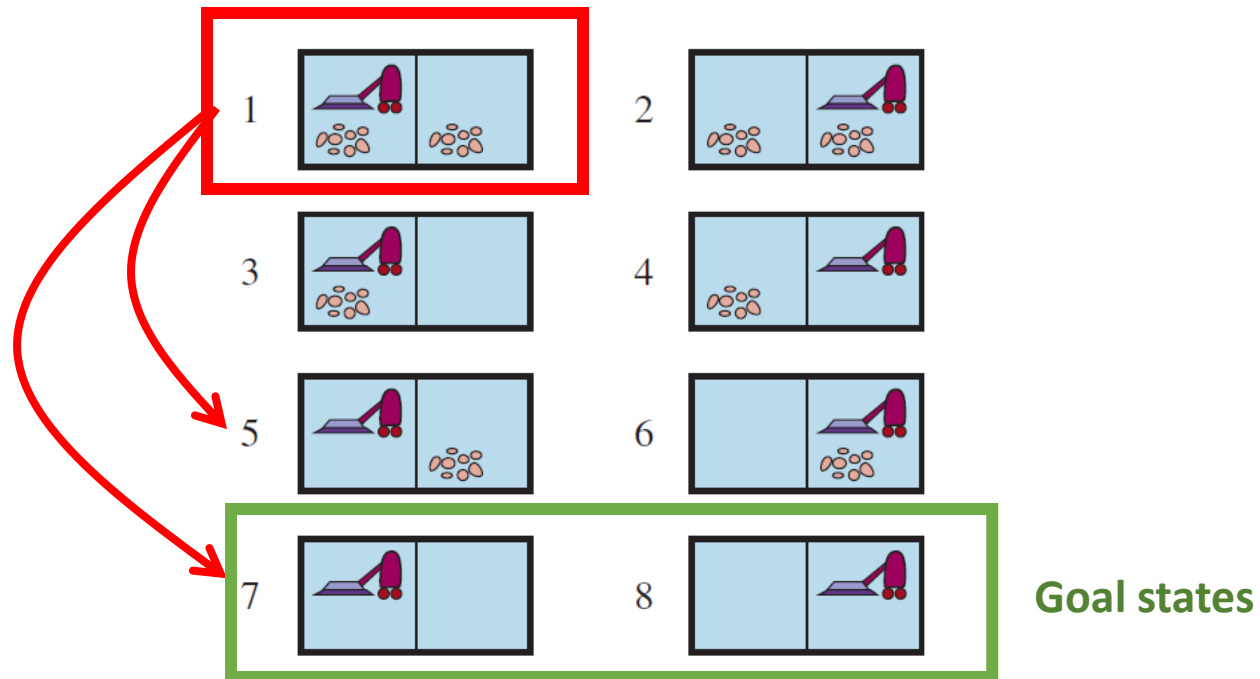


Regular vacuum world, but the action '**suck**' is more powerful and nondeterministic:

- a) **On a dirty square:** cleans the square and sometimes cleans dirt on adjacent squares as well.
- b) **On a clean square:** sometimes deposits some dirt on the square.

Example: Erratic Vacuum World

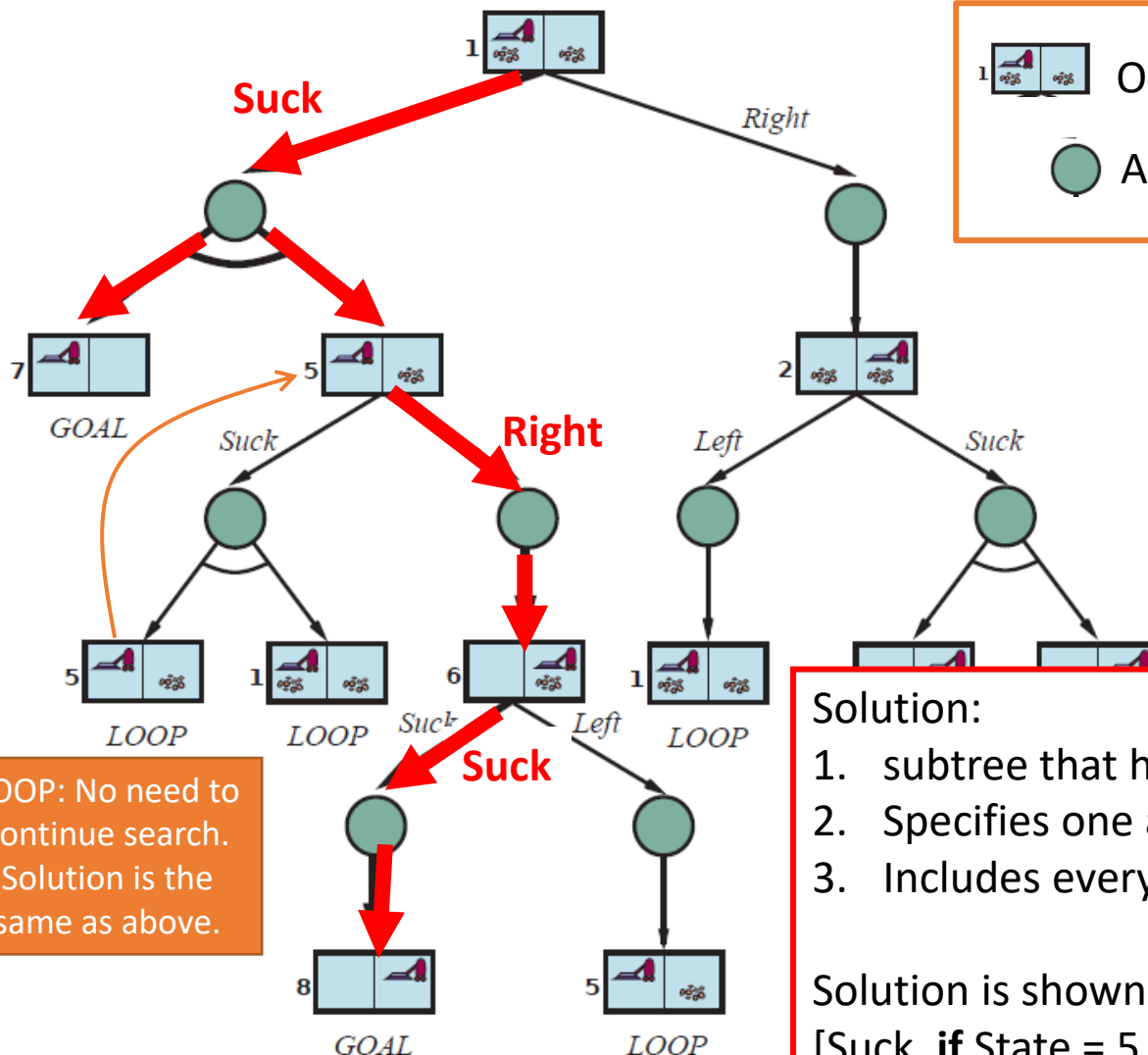
$Results(1, Suck) = \{5, 7\}$





We need a conditional plan

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

AND-OR Search Tree



1  OR node (choose action)

 AND node (possible outcomes)

Solution:

1. subtree that has only GOAL leaf nodes
2. Specifies one action at each OR node (state)
3. Includes every outcome of AND nodes

Solution is shown with **bold** arrows:

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

AND-OR Recursive DFS Algorithm

= nested If-then-else statements

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

path is only maintained for cycle checking!

```
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 // don't follow loops using path.  
  for each action in problem.ACTIONS(state) do // try all possible actions  
    plan  $\leftarrow$  AND-SEARCH(problem, RESULTS(state, action), [state] + path)  
    if plan  $\neq$  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 // try all possible outcomes, none can fail!  
     $plan_i \leftarrow$  OR-SEARCH(problem,  $s_i$ , path) // (= belief state)  
    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  $plan_n$ ]
```

Notes:

- The DFS search tree is implicitly created using the call stack.
- BFS and A* search can also be used to search an AND-OR tree.

Search with no
Observations



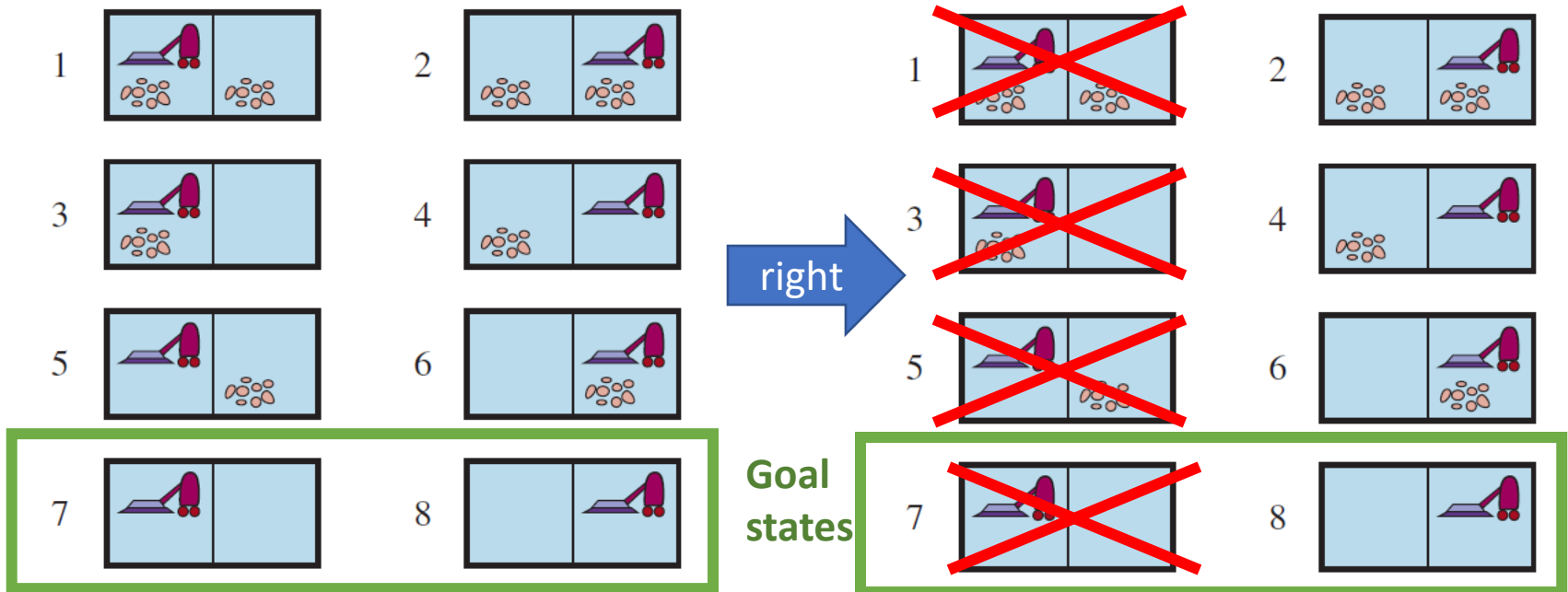
No Observations

- Sensorless problem = conformant problem
- **Example:** Doctor prescribes a broad-band antibiotic instead of performing time-consuming blood work for a more specific antibiotic. This saves time and money.
- **Basic idea:** Find a solution (a sequence of actions) that works from any state.

Actions to Coerce the World into States

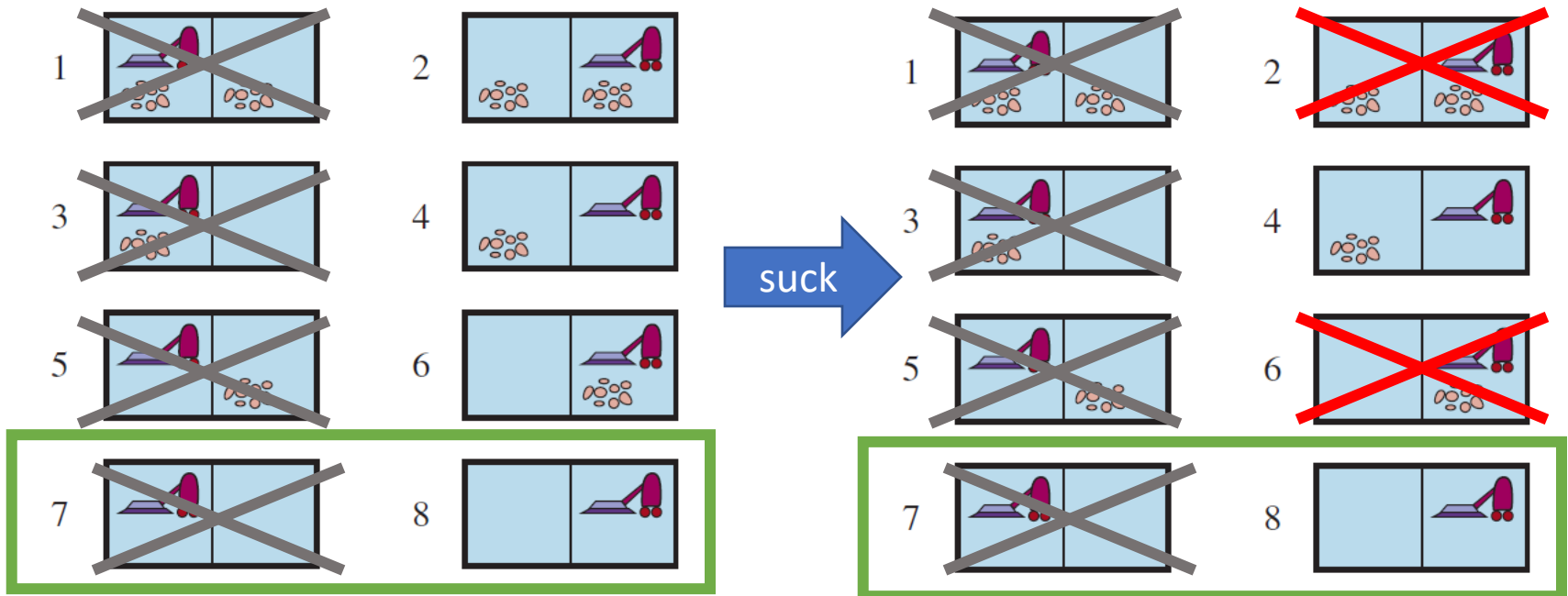
- Actions can reduce the number of possible states.
- **Example:** Deterministic vacuum world. Agent does not know its position and the dirt distribution.

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



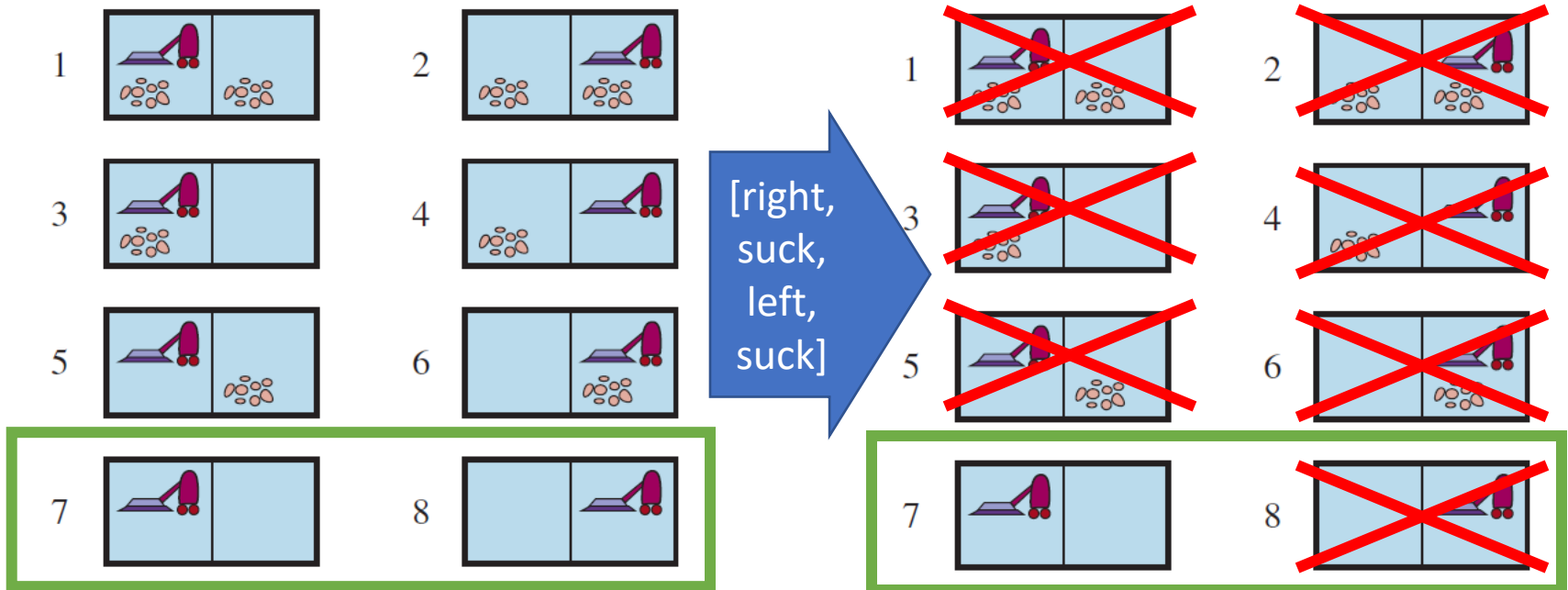
Actions to Coerce the World into States

- Actions can reduce the number of possible states.
- **Example:** Deterministic vacuum world. Agent does not know its position and the dirt distribution.



Actions to Coerce the World into States

- The action sequence [right, suck, left, suck] coerces the world into the goal state 7.
- This works from any initial state and there is no need for a conditional plan!



Find the Solution Sequence

Note: State space size makes this impractical for larger problems!

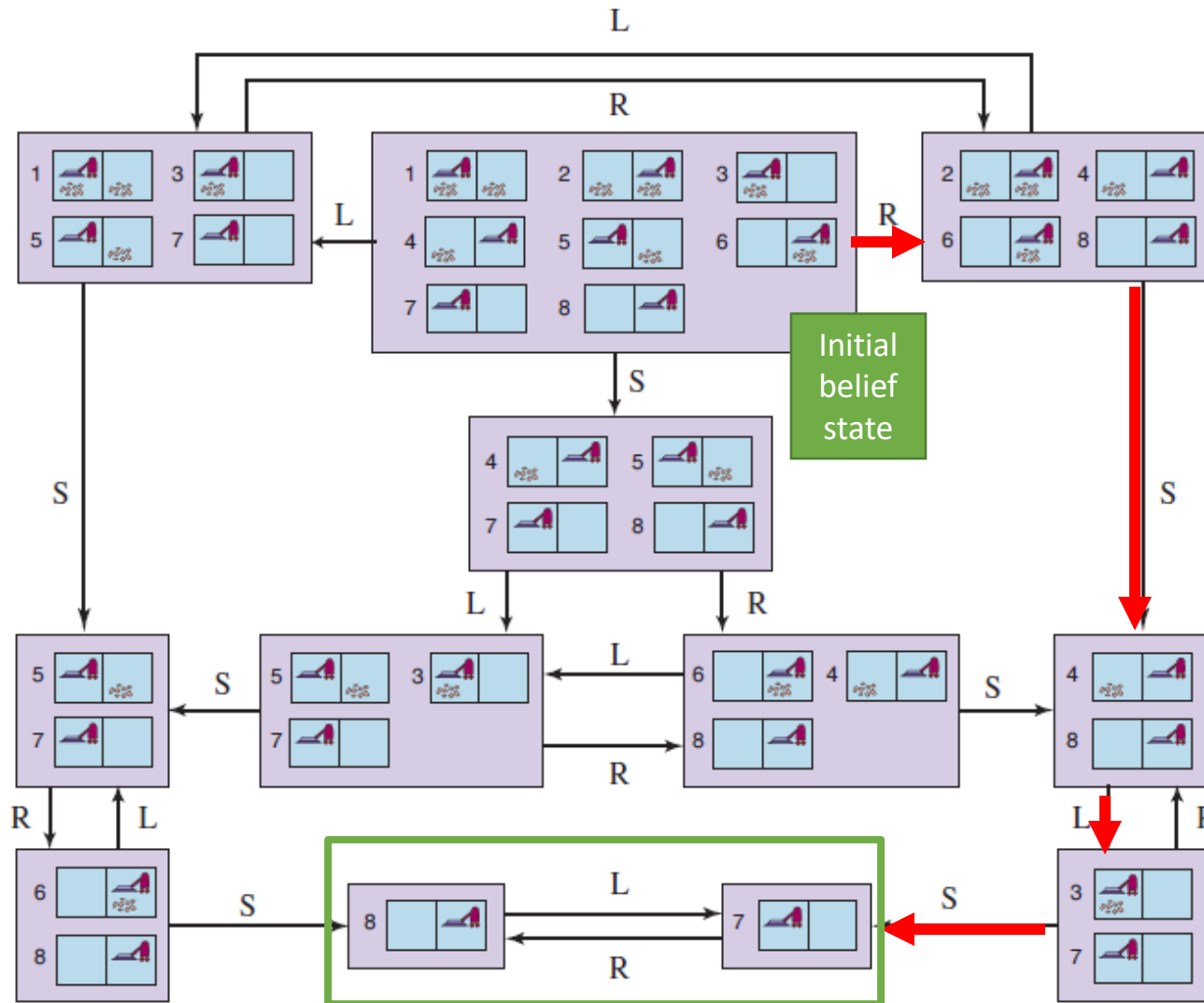
Formulate as a regular search and solve with DFS, BFS or A*:

- **States:** All belief states (=powerset \mathcal{P}_s of states of size 2^N for N states)
- **Initial state:** Often the belief state consisting of all states.
- **Actions:** Actions of a belief state are the union of the possible actions for all the states it contains.
- **Transition model:** $b' = Results(b, a) = \{s' : s' = Result(s, a) \text{ and } s \in b\}$
- **Goal test:** Are all states in the belief state goal states?
- **Simplifying property:** If a belief state (e.g., $b_1 = \{1,2,3,4,5\}$) is solvable (i.e., there is a sequence of actions that coerce all states to only goal states), then belief states that are subsets (e.g., $b_2 = \{2,5\}$) are also solved using the same action sequence. Used to prune the search tree.

Other approach:

- **Incremental belief-state search.** Generate a solution that works for one state and check if it also works for all other states.

Reachable belief-state space for the deterministic, sensorless vacuum world



Solution Sequence:
[right, suck, left, suck]

Size of the belief state space:

$$\mathcal{P}_s = 2^N = 2^8 = 256$$

Only as small fraction is reachable.

A close-up photograph of various colorful, 3D-printed geometric shapes, including letters and numbers, scattered on a blue surface. The shapes are in shades of red, yellow, blue, and green. Some shapes are in sharp focus, while others are blurred in the background, creating a sense of depth. The text "Partially Observable Environments" is overlaid in the center in a white, sans-serif font.

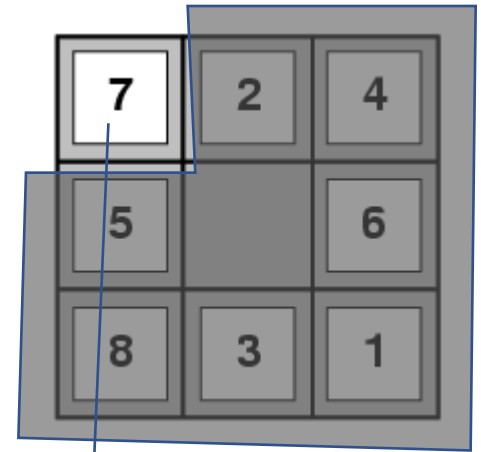
Partially Observable Environments

Percepts

- Many problems cannot be solved efficiently without sensing (e.g., 8-puzzle).
- We need to be able to at least see one square.

Percept function: $Percept(s)$

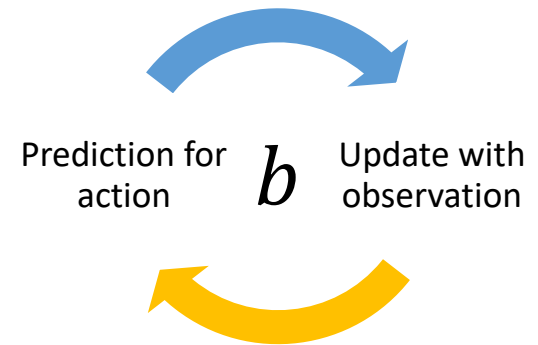
- **Fully observable:** $Percept(s) = s$
- **Sensorless:** $Percept(s) = null$
- **Partially observable:** $Percept(s) = o$



$Percept(s) = Tile7$

Problem: Many states can produce the same percept!

Prediction and Update



Assume we have a belief state b (i.e., the set of states we could be in)

Prediction: Compute new belief state that results from action a .

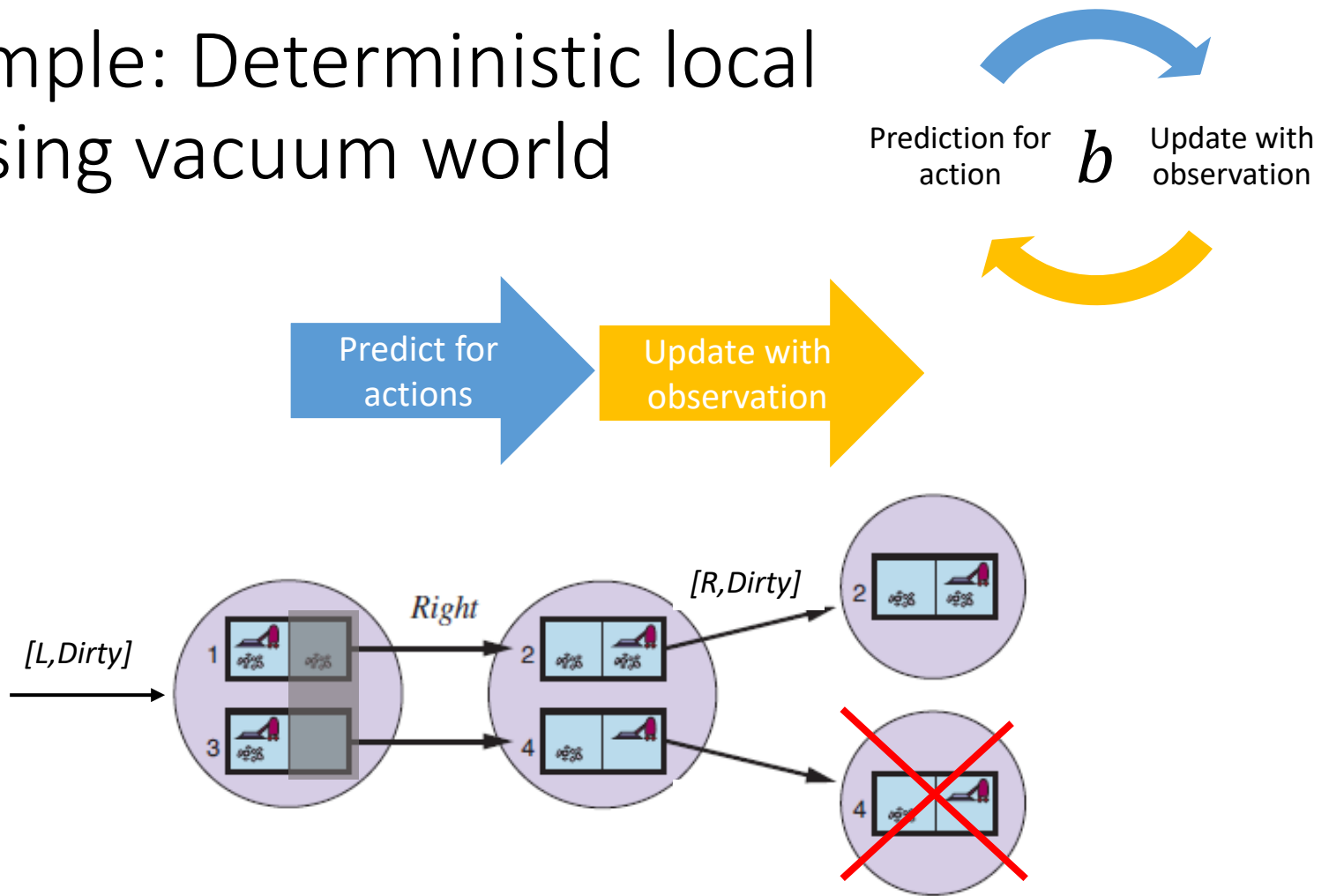
$$\hat{b} = \text{Predict}(b, a)$$

Update: Filter states that are consistent with new observation o .

$$b_o = \text{Update}(\hat{b}, o) = \{s : o = \text{Percept}(s) \text{ and } s \in \hat{b}\}$$

Both steps in one: $b \leftarrow \text{Update}(\text{Predict}(b, a), o)$

Example: Deterministic local sensing vacuum world

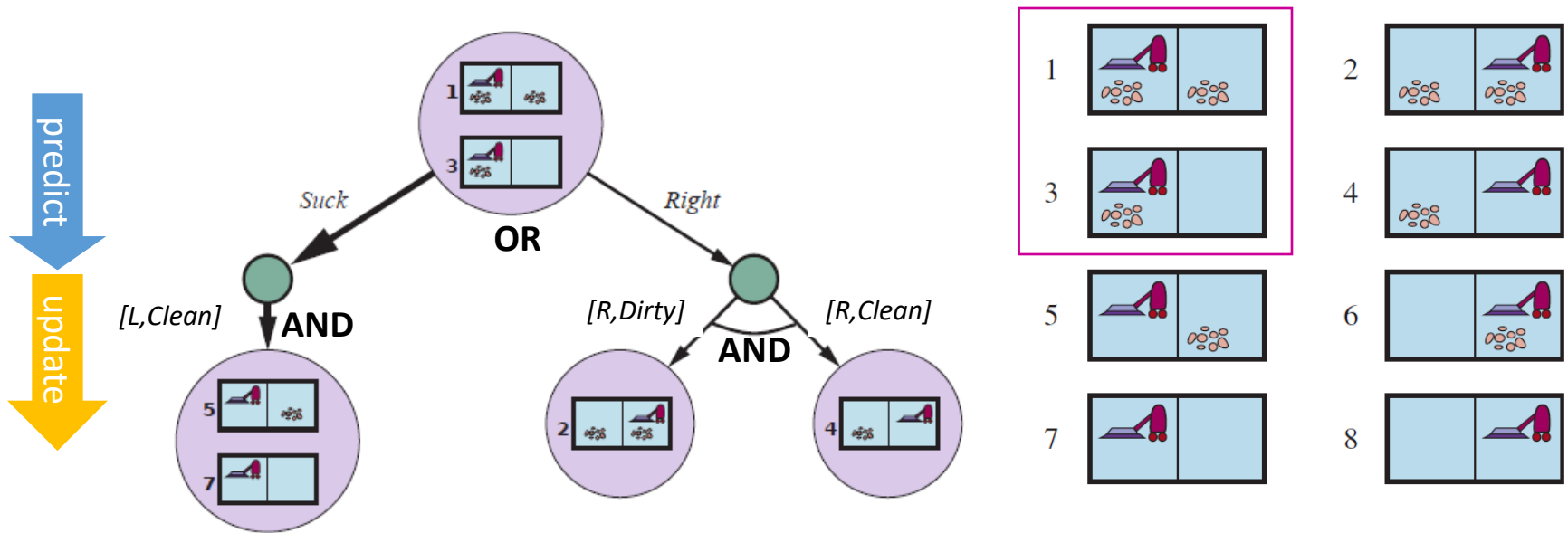


$$b \leftarrow \text{Update}(\text{Predict}(b, a), o)$$

$$\text{Update}(\text{Predict}(\{1, 3\}, \text{Right}), [R, \text{Dirty}]) = \{2\}$$

Solving Partially Observable Problems

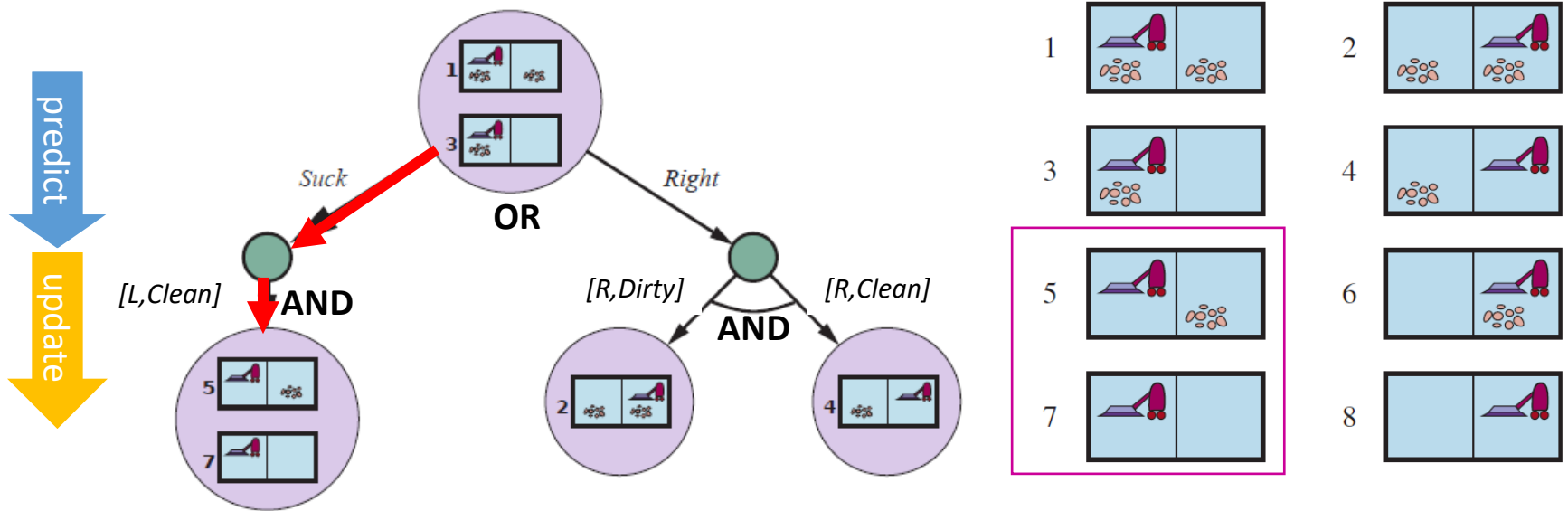
Use an AND-OR tree to create a conditional plan



Solution: *[Suck, Right, if $b = \{6\}$ then Suck else []]*

Solving Partially Observable Problems

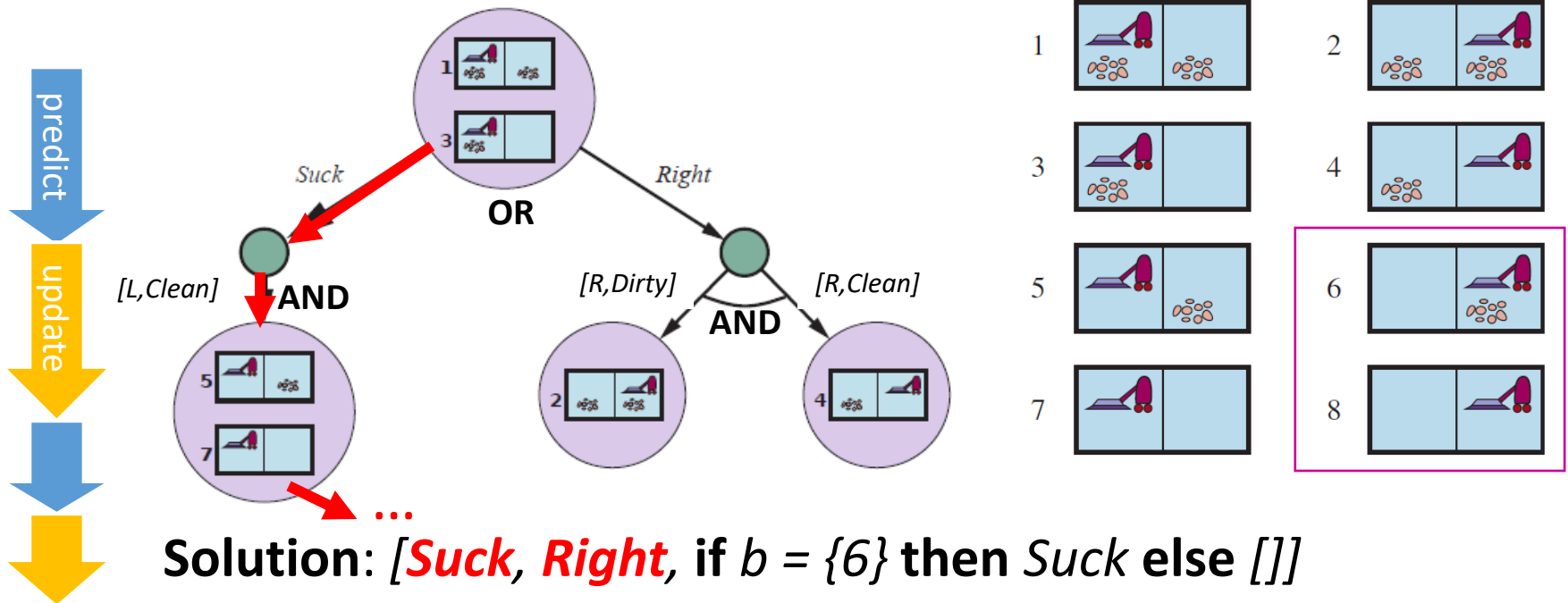
Use an AND-OR tree to create a conditional plan



Solution: [*Suck*, Right, if $b = \{6\}$ then *Suck* else []]

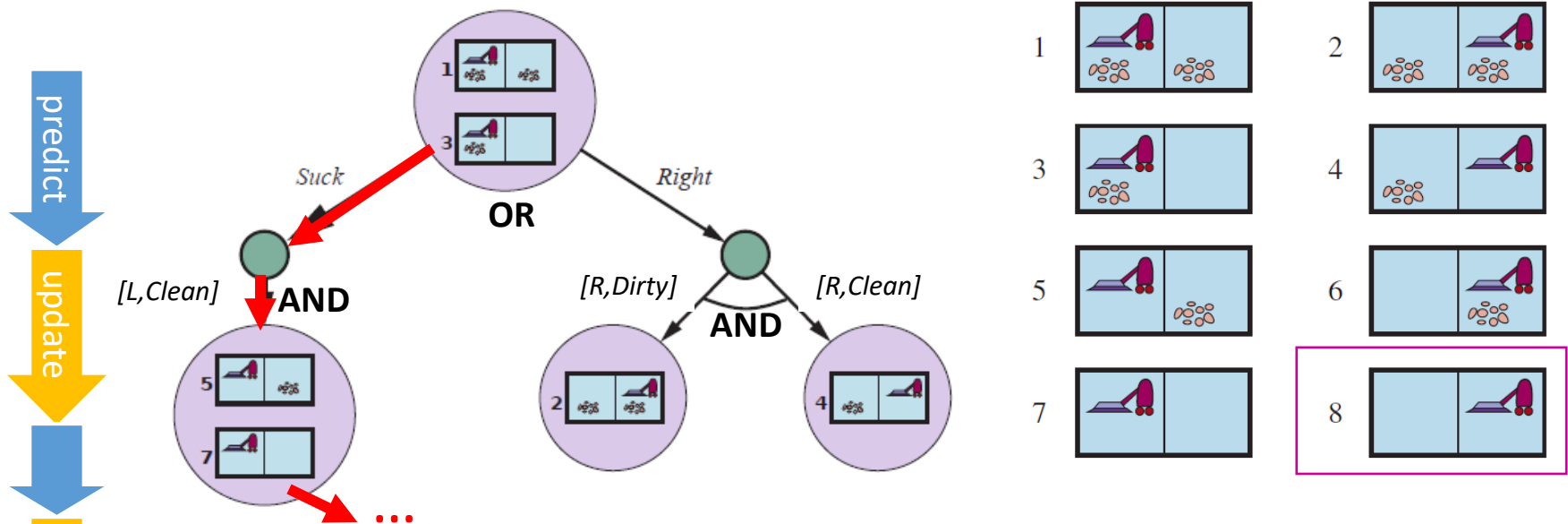
Solving Partially Observable Problems

Use an AND-OR tree to create a conditional plan



Solving Partially Observable Problems

Use an AND-OR tree to create a conditional plan



Solution: [*Suck, Right, if $b = \{6\}$ then Suck else []*]

Note: Belief states that are subsets can be used for pruning!

State Estimation and Approximate Belief States

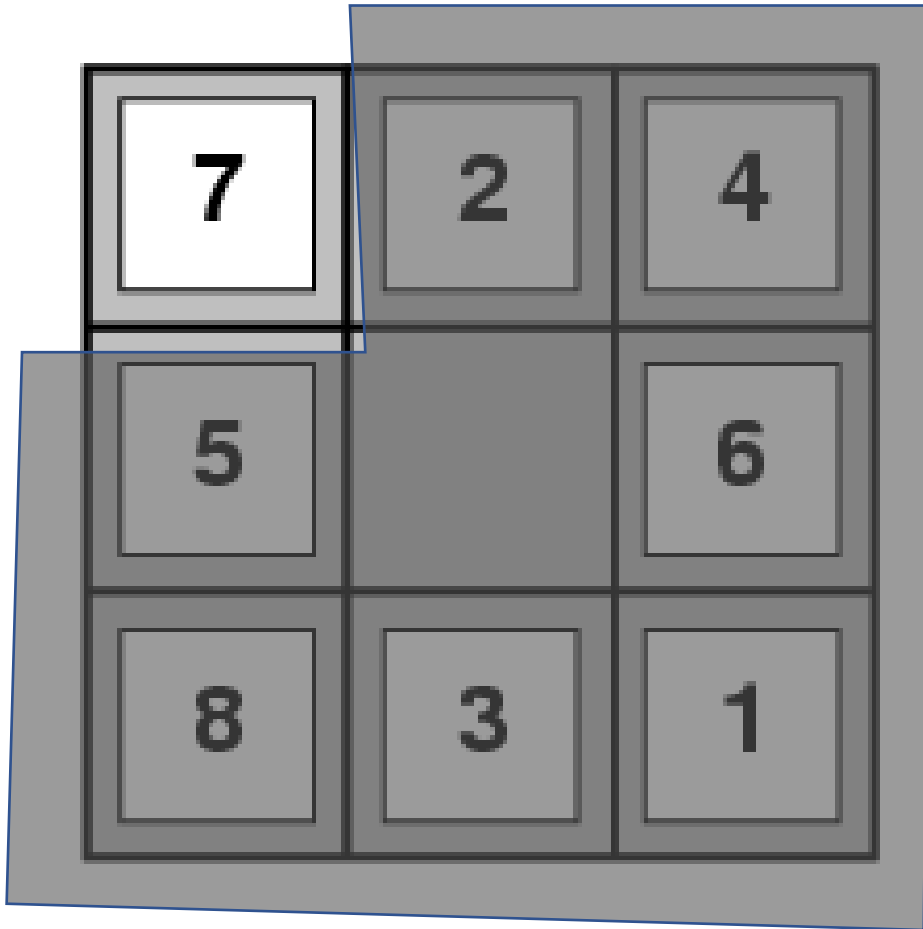
- Agents choose an **action** and then receive an **observation** from the environment.
- The agent keep track of its belief state using the following update:

$$b \leftarrow \text{Update}(\text{Predict}(b, a), o)$$

- This process is often called
 - **monitoring**,
 - **filtering**, or
 - **state estimation**.
- The agent needs to be able to update its belief state following observations in **real time**! For many practical application, there is only time to compute an **approximate belief state**! These approximate methods are outside the scope of this introductory course.

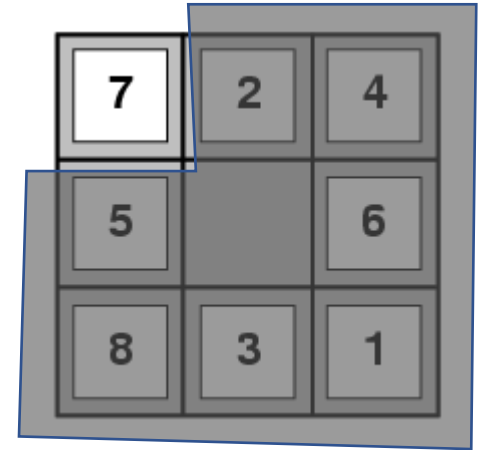
Case Study

Partially Observable 8-Puzzle



Partially Observable 8-Puzzle

- How do we solve this problem? What are the main steps?
- Give a problem description for each step.
 - States:
 - Initial state:
 - Actions:
 - Transition model:
 - Goal test:
 - Percept function:
- What algorithms can be used?





Unknown Environment

Online Search

Online Search

- **Recall offline search:** Create plan using the state space as a model before taking any action. The plan can be a sequence of actions or a conditional plan (for all possible observations).
- **Online search** explores the real world one action at a time. Prediction is replaced by “act” and update by “observe.”



- Useful for
 - **Real-time problems:** When offline computation takes too long and there is a penalty for sitting around and thinking.
 - **Nondeterministic domain:** Deal with what actually happens instead of planning for everything!
 - **Unknown environment:** The agent needs to explore to map an unknown area (state space) and/or what actions do.
The map is the transition model $f : S \times A \rightarrow S$

Issues With Online Search

- **Knowledge:** What does the agent know about the outcome of actions? E.g., does go north and then south lead to the same location?
- **Safely explorable state space/world:** There are **no irreversible actions** that cannot be undone (e.g., traps, cliffs). At least the agent does not execute these actions.
- Expanding nodes in **local order** is more efficient:
Depth-first search with backtracking

Online Search: Agent Program for Unknown Transition model

Environment is deterministic and completely observable ($percept(s) = s'$ after action a), but the transition model (function $result$) and the state space are unknown.

Approach: The algorithm builds the map $result(s, a) \rightarrow s'$ by trying all actions and backtracks when all actions in a state have been explored.

```
function ONLINE-DFS-AGENT(problem,  $s'$ ) returns an action
     $s$ ,  $a$ , the previous state and action, initially null
    persistent: result, a table mapping  $(s, a)$  to  $s'$ , initially empty
                 untried, a table mapping  $s$  to a list of untried actions
                 unbacktracked, a table mapping  $s$  to a list of states never backtracked to

    if problem.IS-GOAL( $s'$ ) then return stop
    if  $s'$  is a new state (not in untried) then untried[ $s'$ ]  $\leftarrow$  problem.ACTIONS( $s'$ )
    if  $s$  is not null then
        result[ $s, a$ ]  $\leftarrow s'$ 
        add  $s$  to the front of unbacktracked[ $s'$ ] // unbacktracked leaves a "breadcrumb trail"
    if untried[ $s'$ ] is empty then
        if unbacktracked[ $s'$ ] is empty then return stop
        else  $a \leftarrow$  an action  $b$  such that result[ $s', b$ ] = POP(unbacktracked[ $s'$ ])
    else  $a \leftarrow$  POP(untried[ $s'$ ])
     $s \leftarrow s'$ 
    return  $a$ 
```

Learns results and
uses it for
backtracking



Important concepts that you should be able to explain and use now...

- Difference between the solution types:
 - a fixed actions sequence, and
 - a conditional plan (also called strategy or policy).
- What are belief states?
- How actions can be used to coerce the world into states.
- State estimation with repeated predict and update steps.
- The use of AND-OR trees.