Search with No/Partial Percepts

Outline

I. Searching with no observation

II. Searching in partially observable environments

* Figures are from the textbook site (or by the instructor) unless the source is specifically cited.

Sensorless Situation

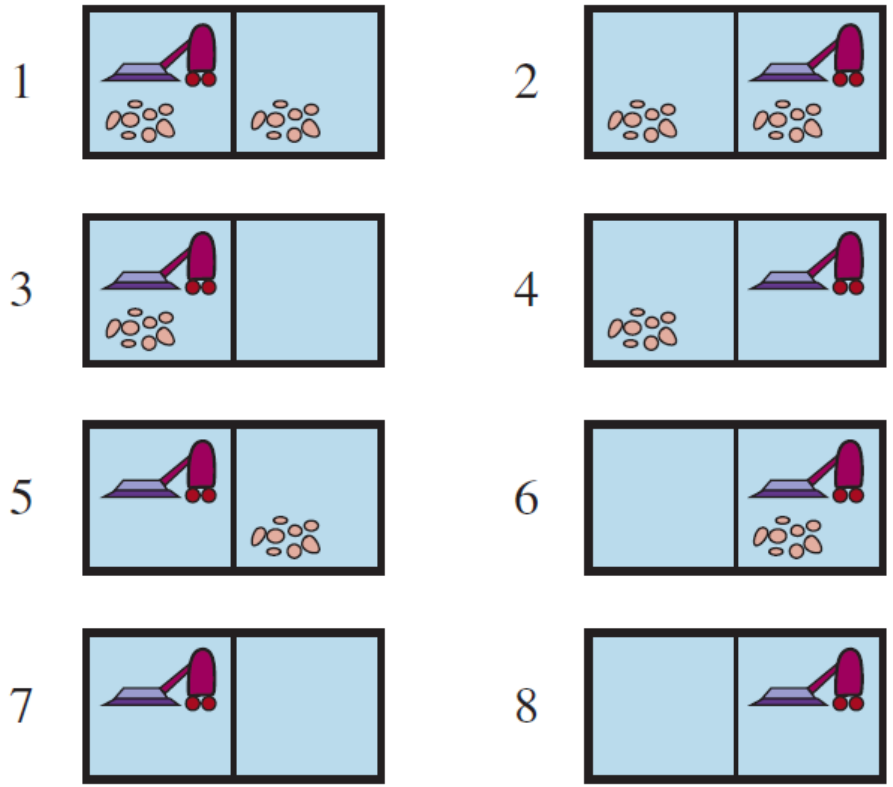
Agent's precepts provide no information at all.

- Sensorless solutions are surprisingly common.
- They can be *robust* due to zero dependence on sensors working properly.

Sensorless Situation

Agent's precepts provide no information at all.

- Sensorless solutions are surprisingly common.
- They can be *robust* due to zero dependence on sensors working properly.

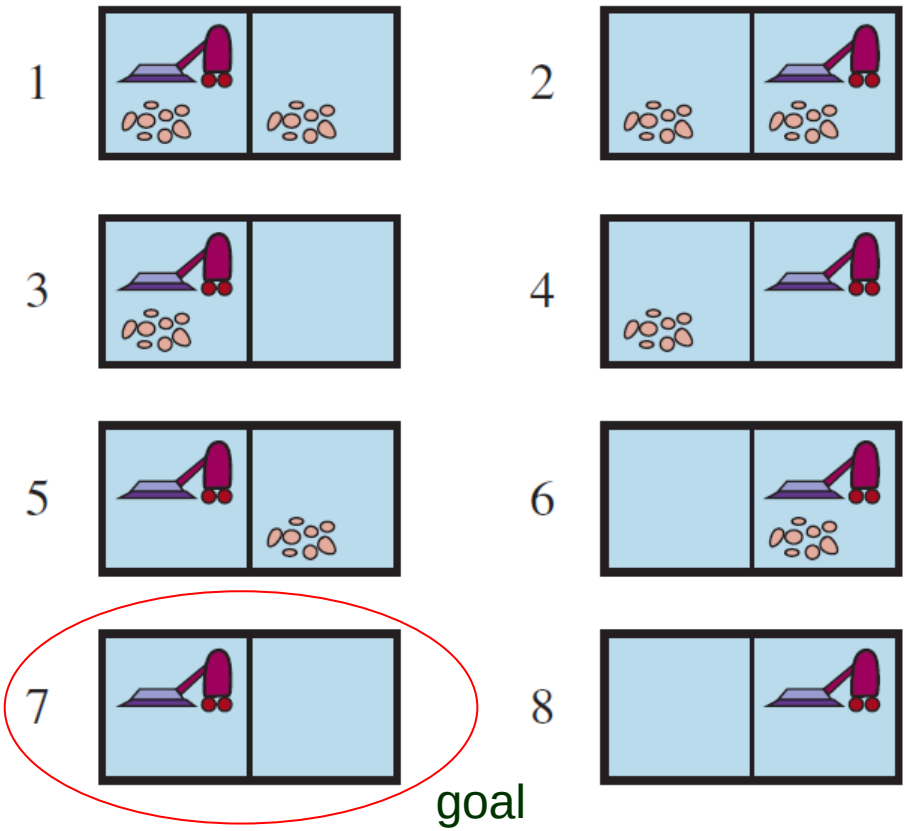


- Knowledge of geography
- No knowledge about location and dirt distribution

Sensorless Situation

Agent's precepts provide no information at all.

- Sensorless solutions are surprisingly common.
- They can be *robust* due to zero dependence on sensors working properly.



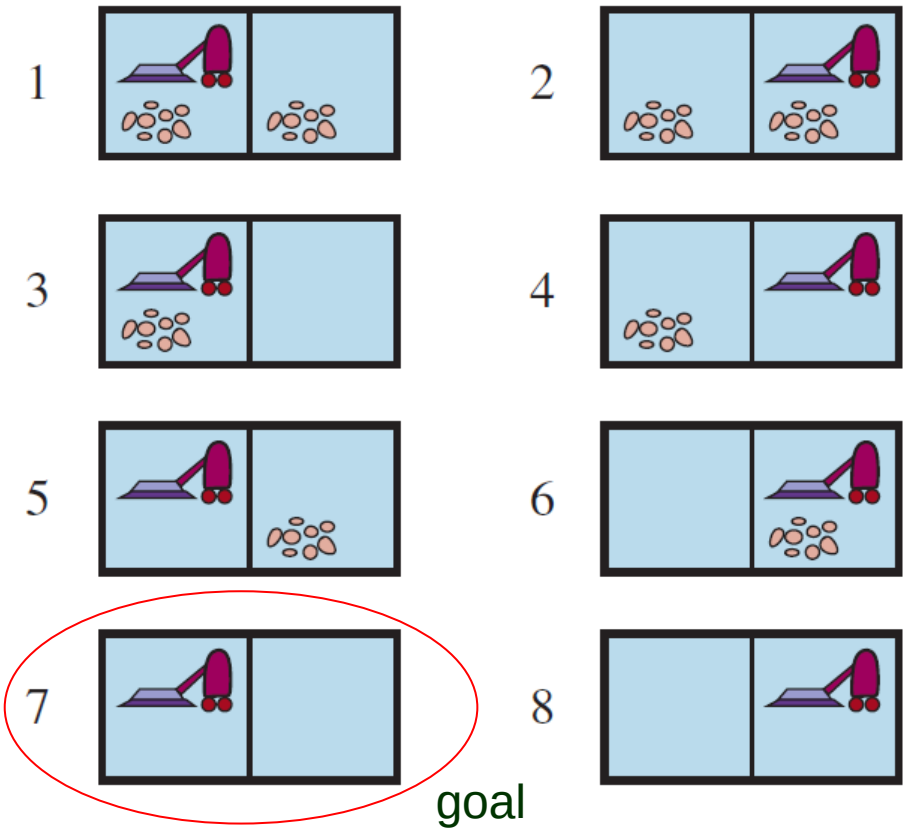
- Knowledge of geography
- No knowledge about location and dirt distribution

$$\{1, 2, 3, 4, 5, 6, 7, 8\}$$

Sensorless Situation

Agent's precepts provide no information at all.

- Sensorless solutions are surprisingly common.
- They can be *robust* due to zero dependence on sensors working properly.



- Knowledge of geography
- No knowledge about location and dirt distribution

$$\{1, 2, 3, 4, 5, 6, 7, 8\}$$

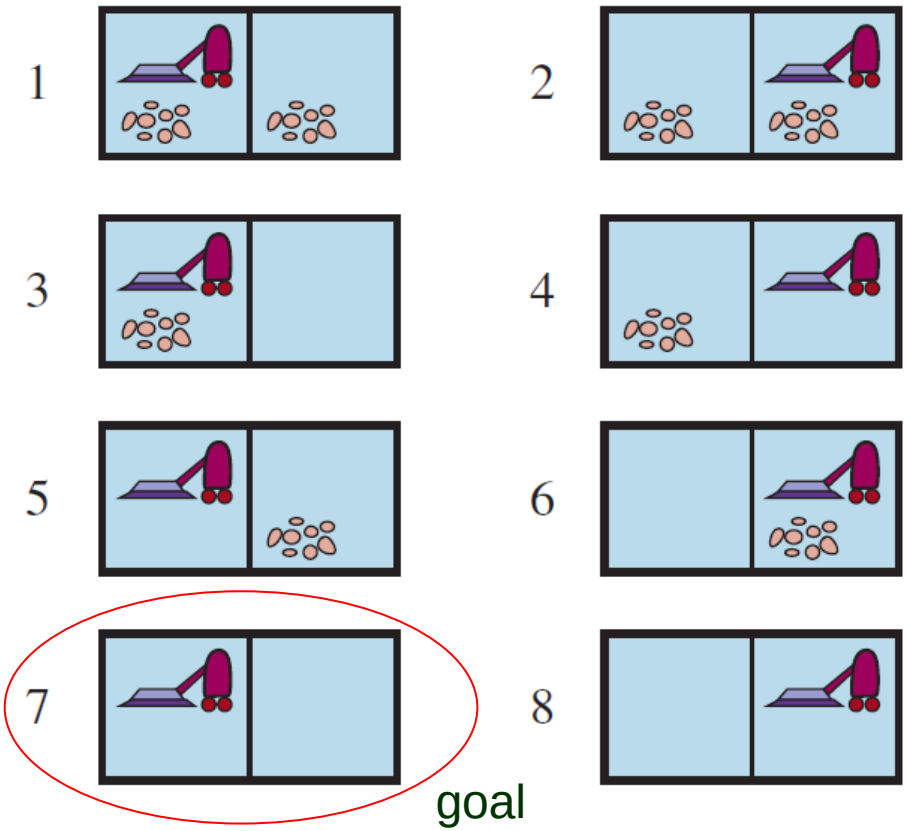
$\Downarrow$ *Right*

$$\{2, 4, 6, 8\}$$

Sensorless Situation

Agent's precepts provide no information at all.

- Sensorless solutions are surprisingly common.
- They can be *robust* due to zero dependence on sensors working properly.



- Knowledge of geography
- No knowledge about location and dirt distribution

$$\{1, 2, 3, 4, 5, 6, 7, 8\}$$

$\Downarrow$ *Right*

$$\{2, 4, 6, 8\}$$

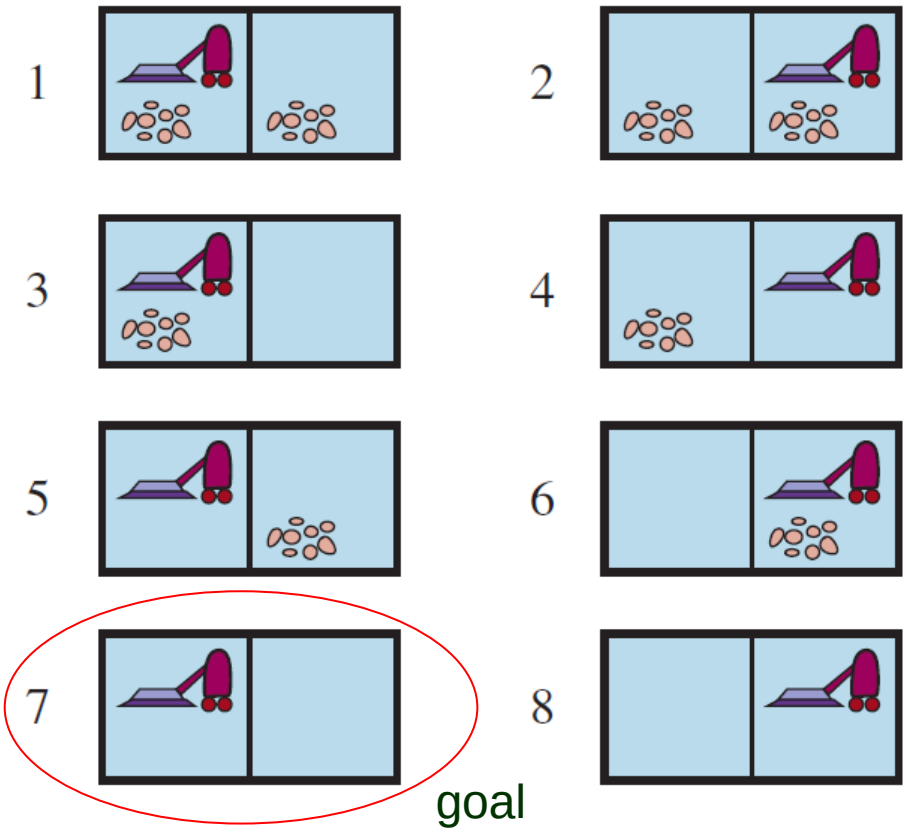
$\Downarrow$ *Suck*

$$\{4, 8\}$$

Sensorless Situation

Agent's precepts provide no information at all.

- Sensorless solutions are surprisingly common.
- They can be *robust* due to zero dependence on sensors working properly.



- Knowledge of geography
- No knowledge about location and dirt distribution

$$\{1, 2, 3, 4, 5, 6, 7, 8\}$$

$\Downarrow$ *Right*

$$\{2, 4, 6, 8\}$$

$\Downarrow$ *Suck*

$$\{4, 8\}$$

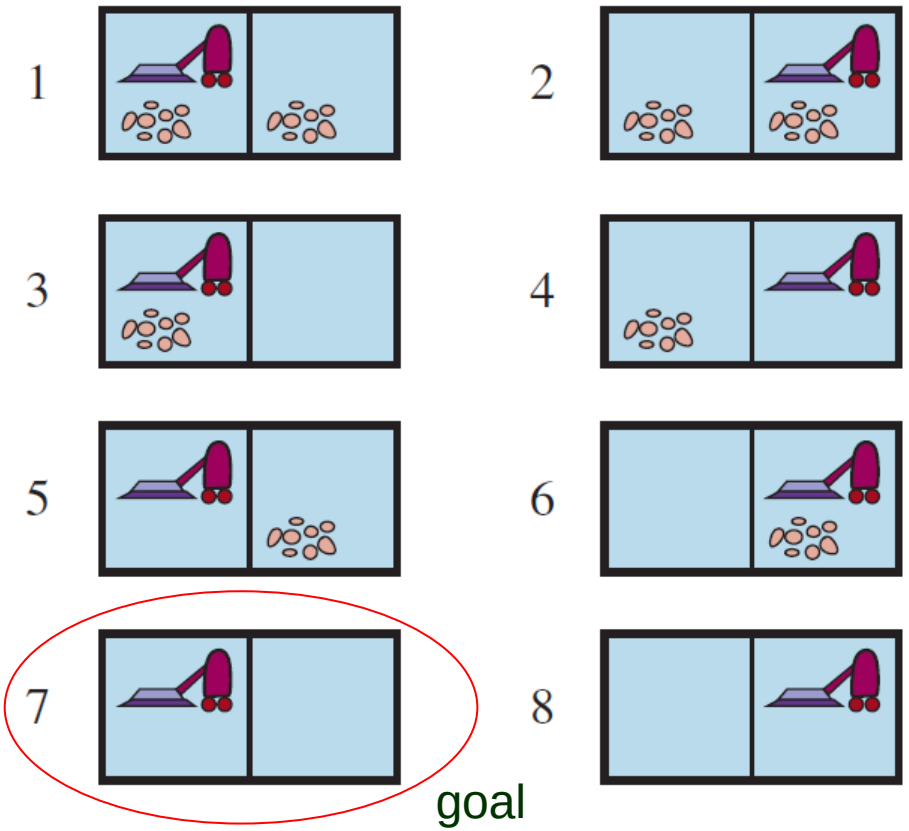
$\Downarrow$ *Left*

$$\{3, 7\}$$

Sensorless Situation

Agent's precepts provide no information at all.

- Sensorless solutions are surprisingly common.
- They can be *robust* due to zero dependence on sensors working properly.



- Knowledge of geography
- No knowledge about location and dirt distribution

$$\{1,2,3,4,5,6,7,8\}$$

$\Downarrow$ *Right*

$$\{2,4,6,8\}$$

$\Downarrow$ *Suck*

$$\{4,8\}$$

$\Downarrow$ *Left*

$$\{3,7\} \xrightarrow{\textit{Suck}} \{7\}$$

Sensorless Manipulation - Tray Tilting

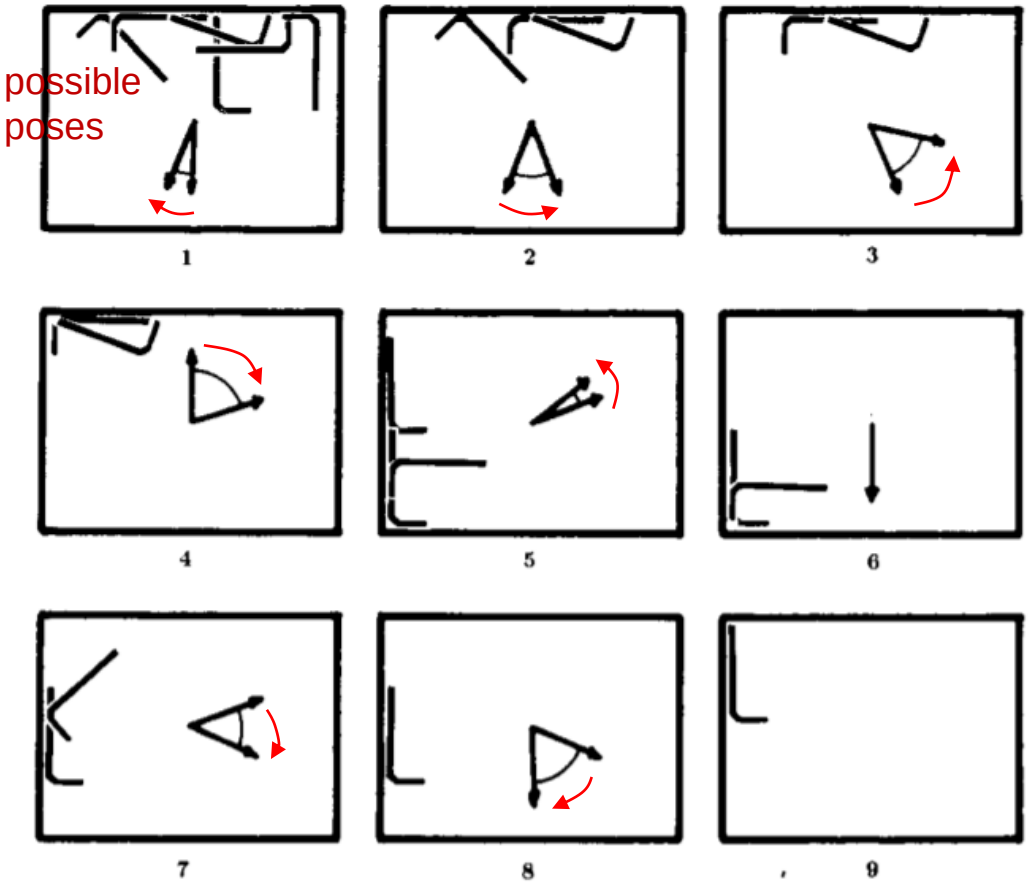


Fig. 2. Beginning at the upper left and moving from left to right, we can trace an automatically generated program that orients the wrench. Each frame shows the set of possible wrench contacts, and the operation to be applied. Each operation is represented by an interval of azimuths. The azimuth arrows indicate the tray's direction of steepest ascent; gravity acts in the opposite direction.

Erdmann & Mason (1988):

Orients the Allen wrench with eight tilts

- Slow planar motion
- Coulomb friction

IEEE Journal of Robotics and Automation, vol. 4, no. 4, pp. 369-379, 1988.

Solution to a Sensorless Problem

A sequence of actions from search in the space B of *belief states (b-states)*.

- Full observability in B: the agent always knows its belief state.

Solution to a Sensorless Problem

A sequence of actions from search in the space B of *belief states (b-states)*.

- Full observability in B: the agent always knows its belief state.

1. Transform the physical problem into a belief state problem.
2. Apply existing search algorithms.

Solution to a Sensorless Problem

A sequence of actions from search in the space B of *belief states (b-states)*.

- ♦ Full observability in B: the agent always knows its belief state.

1. Transform the physical problem into a belief state problem.
2. Apply existing search algorithms.

- States: 2^N possible subsets of the set Σ of N physical states.

Solution to a Sensorless Problem

A sequence of actions from search in the space B of *belief states (b-states)*.

- ♦ Full observability in B: the agent always knows its belief state.

1. Transform the physical problem into a belief state problem.
2. Apply existing search algorithms.

- States: 2^N possible subsets of the set Σ of N physical states.
- Initial state: S.

Solution to a Sensorless Problem

A sequence of actions from search in the space B of *belief states (b-states)*.

- Full observability in B: the agent always knows its belief state.

1. Transform the physical problem into a belief state problem.
2. Apply existing search algorithms.

- States: 2^N possible subsets of the set Σ of N physical states.
- Initial state: S.
- Actions: At the belief state b,

$$\text{ACTIONS}(b) = \cup_{s \in b} \text{ACTIONS}_P(s)$$

Solution to a Sensorless Problem

A sequence of actions from search in the space B of *belief states (b-states)*.

- ♦ Full observability in B: the agent always knows its belief state.

1. Transform the physical problem into a belief state problem.
2. Apply existing search algorithms.

- States: 2^N possible subsets of the set Σ of N physical states.
- Initial state: S.
- Actions: At the belief state b,

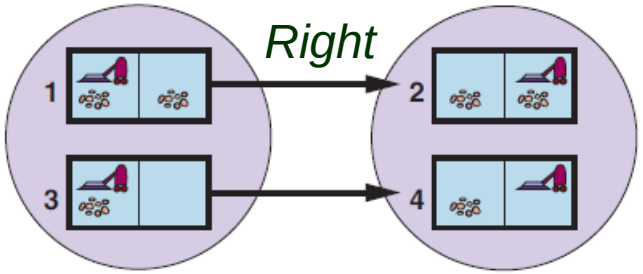
$$\text{ACTIONS}(b) = \bigcup_{s \in b} \text{ACTIONS}_P(s)$$

Action on a physical state

Transition Model

- Transition model: The new belief state is, deterministically,

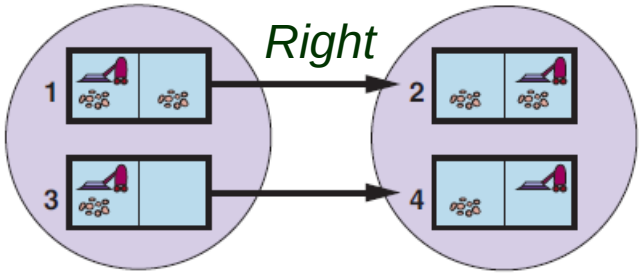
$$b' = RESULT(b, a) = \{s' \mid s' = RESULT_P(s, a) \text{ and } s \in b\}$$



Transition Model

- Transition model: The new belief state is, deterministically,

$$b' = RESULT(b, a) = \{s' \mid s' = RESULT_P(s, a) \text{ and } s \in b\}$$

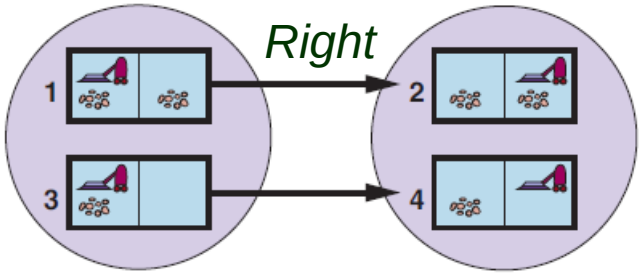


or, nondeterministically,

Transition Model

- Transition model: The new belief state is, deterministically,

$$b' = RESULT(b, a) = \{s' \mid s' = RESULT_P(s, a) \text{ and } s \in b\}$$



or, nondeterministically,

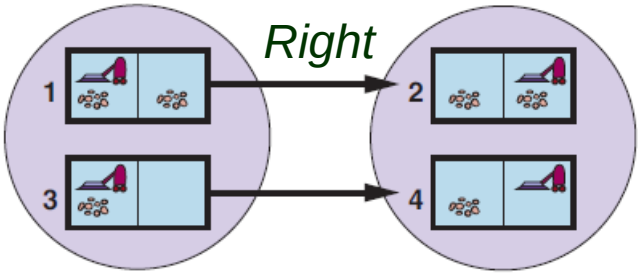
$$b' = RESULT(b, a) = \{s' \mid s' \in RESULT_P(s, a) \text{ and } s \in b\}$$

$$= \bigcup_{s \in b} RESULT_P(s, a)$$

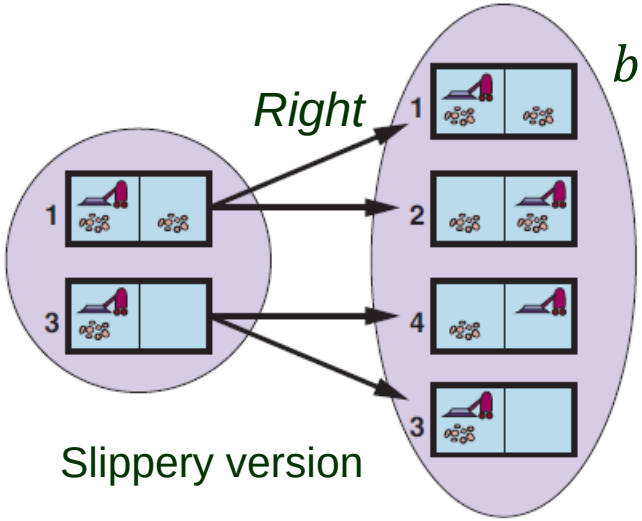
Transition Model

- Transition model: The new belief state is, deterministically,

$$b' = RESULT(b, a) = \{s' \mid s' = RESULT_P(s, a) \text{ and } s \in b\}$$



or, nondeterministically,



$$b' = RESULT(b, a) = \{s' \mid s' \in RESULT_P(s, a) \text{ and } s \in b\}$$

$$= \bigcup_{s \in b} RESULT_P(s, a)$$

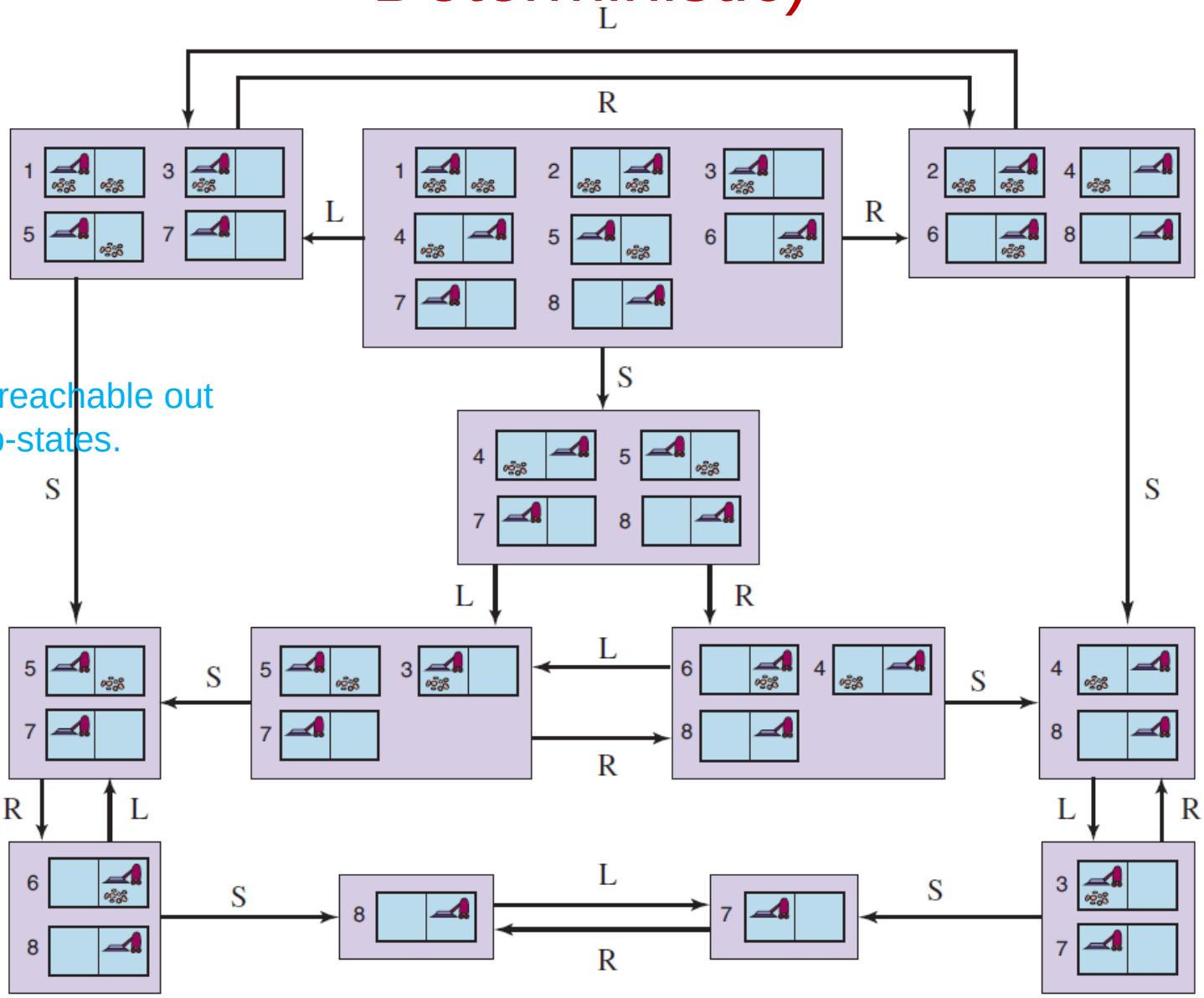
Goal & Cost of Action

- Goal test: The goal is achieved
  - ♣ *possibly* if one of the states $s \in b$ passes the test;
  - ♣ *necessarily* if every state $s \in b$ passes the test.

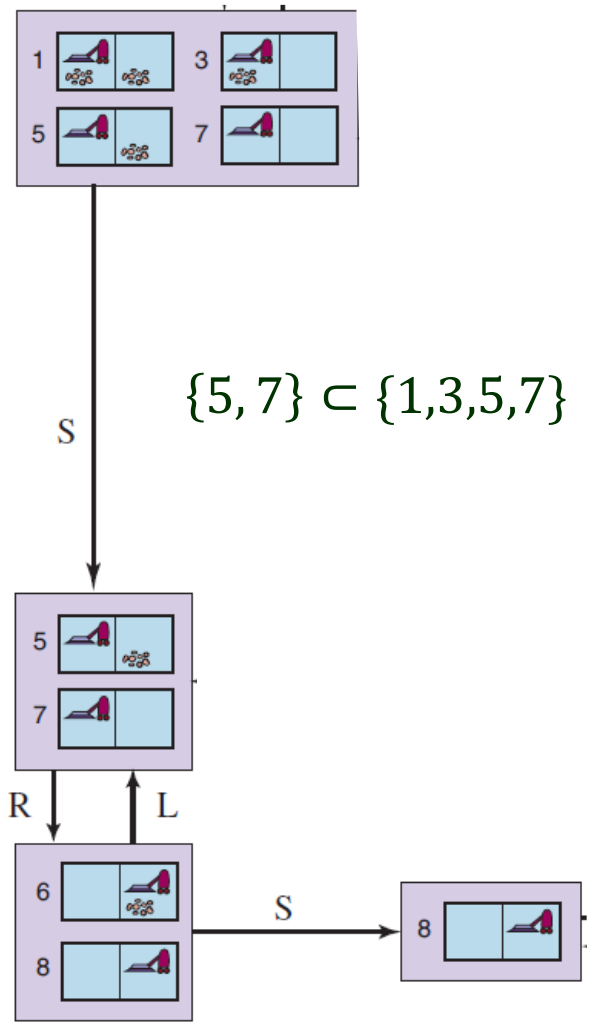
- Action cost: Could be one of several values if the same action has different costs in different states.

Reachable Belief States (Sensorless & Deterministic)

Only 12 are reachable out of 256= 2^8 b-states.



Subset and Superset States



$\{5, 7\} \subset \{1,3,5,7\}$

- Prune a superset b-state to concentrate on solving the easier subset b-state.
- Prune a subset b-state if a superset b-state has been found to be solvable already.

Handling Search Space Complexity

N physical states $\Longrightarrow 2^N$ belief states!

- Use of compact description, e.g., mathematical logic (Chapter 7)

  e.g., Use “nothing” to represent the initial state.

Handling Search Space Complexity

N physical states $\Longrightarrow$ 2^N belief states!

- Use of compact description, e.g., mathematical logic (Chapter 7)

  e.g., Use “nothing” to represent the initial state.

- Incremental belief-state search to build up solution one physical state at a time.

Handling Search Space Complexity

N physical states $\Longrightarrow 2^N$ belief states!

- Use of compact description, e.g., mathematical logic (Chapter 7)

  e.g., Use “nothing” to represent the initial state.

- Incremental belief-state search to build up solution one physical state at a time.

  - Initial state: $\{1, 2, 3, 4, 5, 6, 7, 8\}$

Handling Search Space Complexity

N physical states $\Longrightarrow$ 2^N belief states!

- Use of compact description, e.g., mathematical logic (Chapter 7)

  e.g., Use “nothing” to represent the initial state.

- Incremental belief-state search to build up solution one physical state at a time.

  - Initial state: $\{1, 2, 3, 4, 5, 6, 7, 8\}$
  - Find an action sequence that works for state 1.

Handling Search Space Complexity

N physical states $\Longrightarrow$ 2^N belief states!

- Use of compact description, e.g., mathematical logic (Chapter 7)

  e.g., Use “nothing” to represent the initial state.

- Incremental belief-state search to build up solution one physical state at a time.

  - Initial state: $\{1, 2, 3, 4, 5, 6, 7, 8\}$
  - Find an action sequence that works for state 1.
  - Check if it works for state 2.

Handling Search Space Complexity

N physical states $\Longrightarrow 2^N$ belief states!

- Use of compact description, e.g., mathematical logic (Chapter 7)

  e.g., Use "nothing" to represent the initial state.

- Incremental belief-state search to build up solution one physical state at a time.

  - Initial state: $\{1, 2, 3, 4, 5, 6, 7, 8\}$
  - Find an action sequence that works for state 1.
  - Check if it works for state 2.
    - If yes, check if it works for state 3, and so on.

Handling Search Space Complexity

N physical states $\Longrightarrow$ 2^N belief states!

- Use of compact description, e.g., mathematical logic (Chapter 7)

  e.g., Use “nothing” to represent the initial state.

- Incremental belief-state search to build up solution one physical state at a time.

  - Initial state: $\{1, 2, 3, 4, 5, 6, 7, 8\}$
  - Find an action sequence that works for state 1.
  - Check if it works for state 2.
    - If yes, check if it works for state 3, and so on.
    - If no, find a different solution for state 1, and so on.

Handling Search Space Complexity

N physical states $\Longrightarrow 2^N$ belief states!

- Use of compact description, e.g., mathematical logic (Chapter 7)

  e.g., Use "nothing" to represent the initial state.

- Incremental belief-state search to build up solution one physical state at a time.

  - Initial state: $\{1, 2, 3, 4, 5, 6, 7, 8\}$
  - Find an action sequence that works for state 1.
  - Check if it works for state 2.
    - If yes, check if it works for state 3, and so on.
    - If no, find a different solution for state 1, and so on.

Partially Observable Environments

Many problems (e.g., the 8-puzzle) are unsolvable without sensing.

A little sensing (e.g., only one visible square in the 8-puzzle) can go a long way.

$$\text{PERCEPTS}(s) = \{\text{percept received in the state } s\}$$

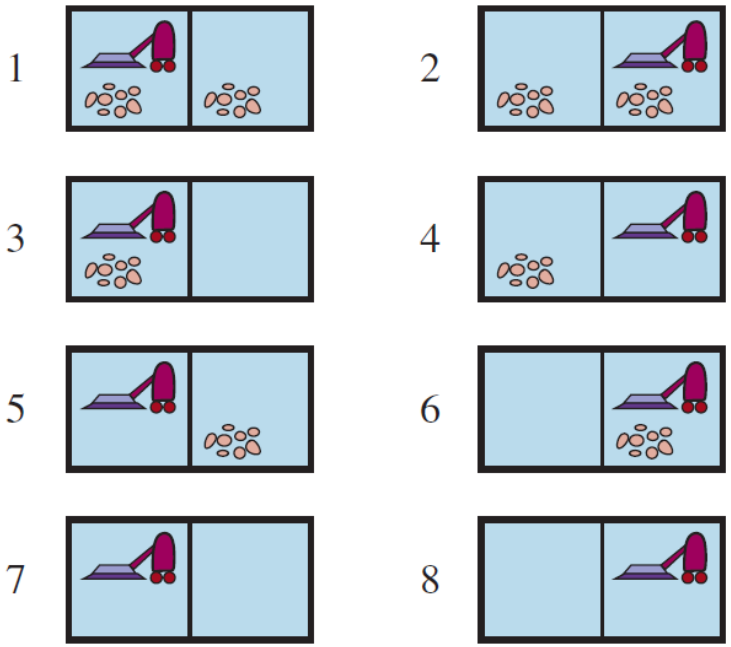
- Several states may yield the same percept.

Partially Observable Environments

Many problems (e.g., the 8-puzzle) are unsolvable without sensing.

A little sensing (e.g., only one visible square in the 8-puzzle) can go a long way.

$$\text{PERCEPTS}(s) = \{\text{percept received in the state } s\}$$



Local-sensing vacuum world

- Several states may yield the same percept.

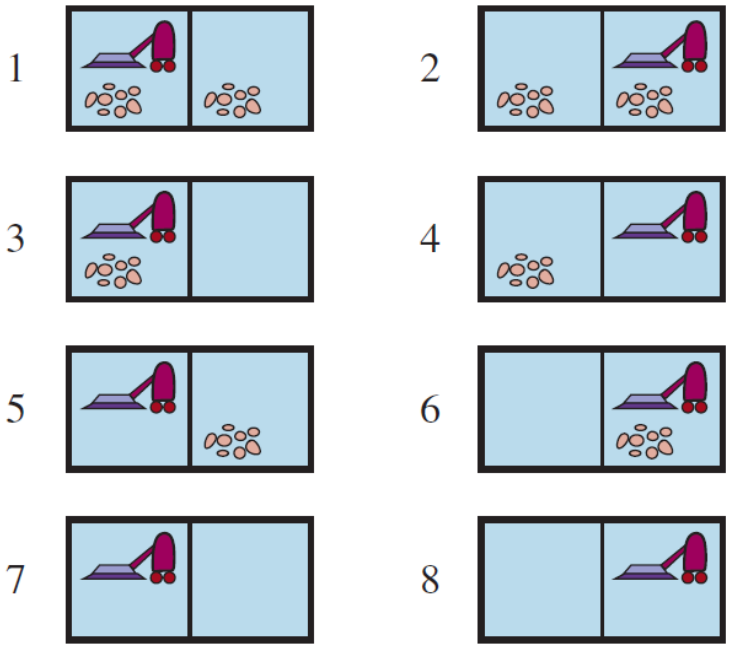
Local-sensing vacuum world: The vacuum cleaner cannot sense the state of the adjacent square.

Partially Observable Environments

Many problems (e.g., the 8-puzzle) are unsolvable without sensing.

A little sensing (e.g., only one visible square in the 8-puzzle) can go a long way.

$$\text{PERCEPTS}(s) = \{\text{percept received in the state } s\}$$



Local-sensing vacuum world

- Several states may yield the same percept.

Local-sensing vacuum world: The vacuum cleaner cannot sense the state of the adjacent square.

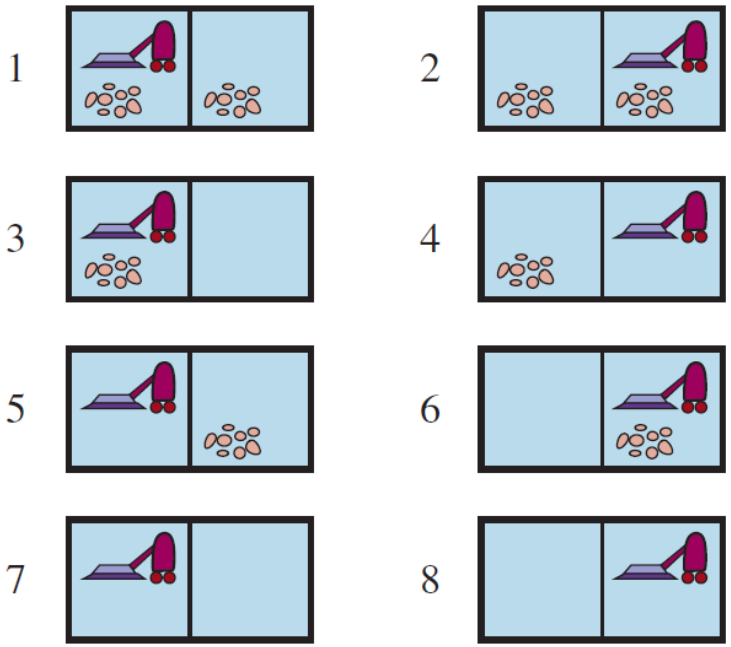
L: in the left square
R: in the right square
Dirty: current square dirty

Partially Observable Environments

Many problems (e.g., the 8-puzzle) are unsolvable without sensing.

A little sensing (e.g., only one visible square in the 8-puzzle) can go a long way.

$$\text{PERCEPTS}(s) = \{\text{percept received in the state } s\}$$



Local-sensing vacuum world

- Several states may yield the same percept.

Local-sensing vacuum world: The vacuum cleaner cannot sense the state of the adjacent square.

L: in the left square
R: in the right square
Dirty: current square dirty

$\{L, Dirty\} \longrightarrow$ b-state {1, 3}

Transition Model Between B-states

- Prediction: computes the b-state from an action a.

estimate →

$\hat{b}$ =PREDICT(b, a)

Transition Model Between B-states

- Prediction: computes the b-state from an action a.

estimate →

$\hat{b}$ =PREDICT(b, a)

- Possible percepts: computes the set of percepts that could be observed in the predicted b-state.

POSSIBLE-PERCEPTS$(\hat{b}) = \{o \mid o =$ PERCEPTS(s) and $s \in \hat{b}\}$

Transition Model Between B-states

- Prediction: computes the b-state from an action a.

estimate

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

- Possible percepts: computes the set of percepts that could be observed in the predicted b-state.

$$\text{POSSIBLE-PERCEPTS}(\hat{b}) = \{o \mid o = \text{PERCEPTS}(s) \text{ and } s \in \hat{b}\}$$

- Update: computes the set of states in $\hat{b}$ that could have produced the percept.

$$b_o = \text{UPDATE}(\hat{b}, o) = \{s \mid o = \text{PERCEPTS}(s) \text{ and } s \in \hat{b}\}$$

Possible B-states from an Action

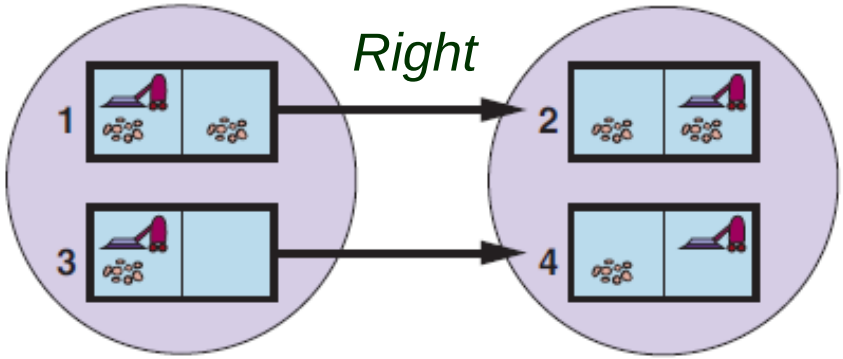
$$\text{RESULTS}(b, a) = \{b_o \mid b_o = \text{UPDATE}(\text{PREDICT}(b, a), o) \text{ where}$$

$$o \in \text{POSSIBLE-PERCEPTS}(\text{PREDICT}(b, a))\}$$

Possible B-states from an Action

$$\text{RESULTS}(b,a) = \{b_o \mid b_o = \text{UPDATE}(\text{PREDICT}(b,a), o) \text{ where}$$

$$o \in \text{POSSIBLE-PERCEPTS}(\text{PREDICT}(b,a))\}$$

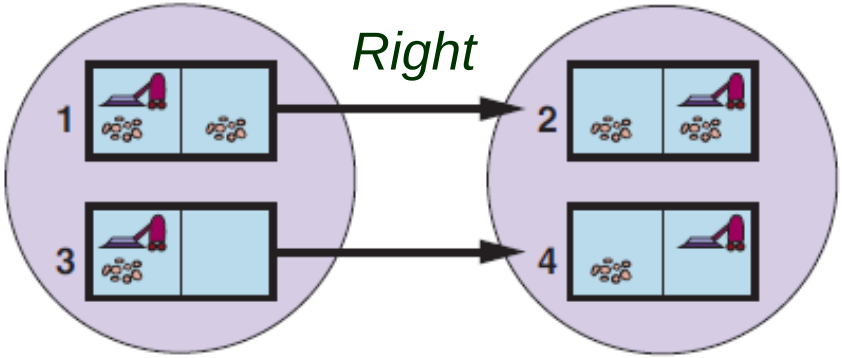


Deterministic
(local sensing)

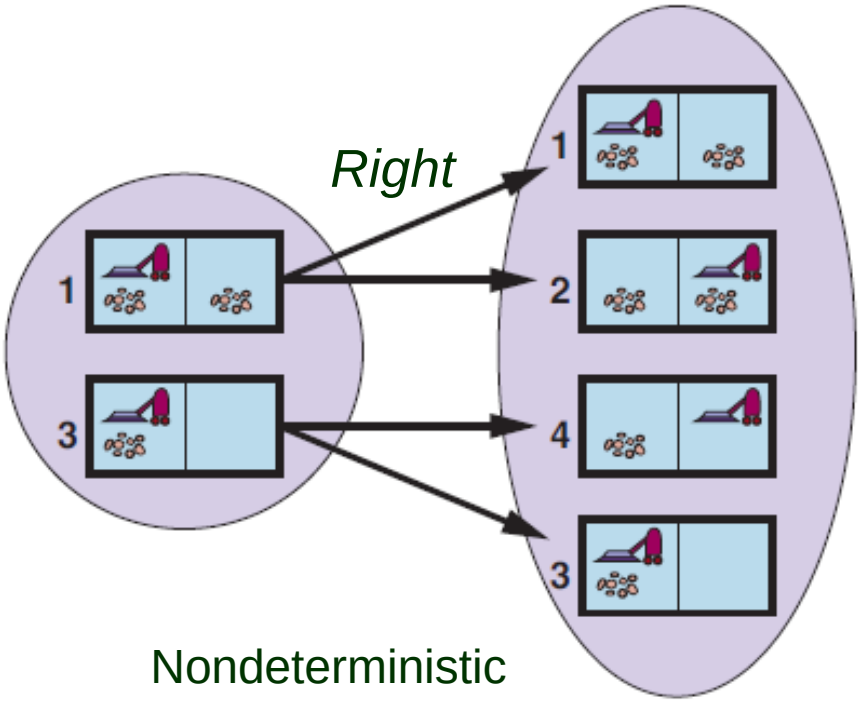
Possible B-states from an Action

$$\text{RESULTS}(b,a) = \{b_o \mid b_o = \text{UPDATE}(\text{PREDICT}(b,a), o) \text{ where}$$

$$o \in \text{POSSIBLE-PERCEPTS}(\text{PREDICT}(b,a))\}$$



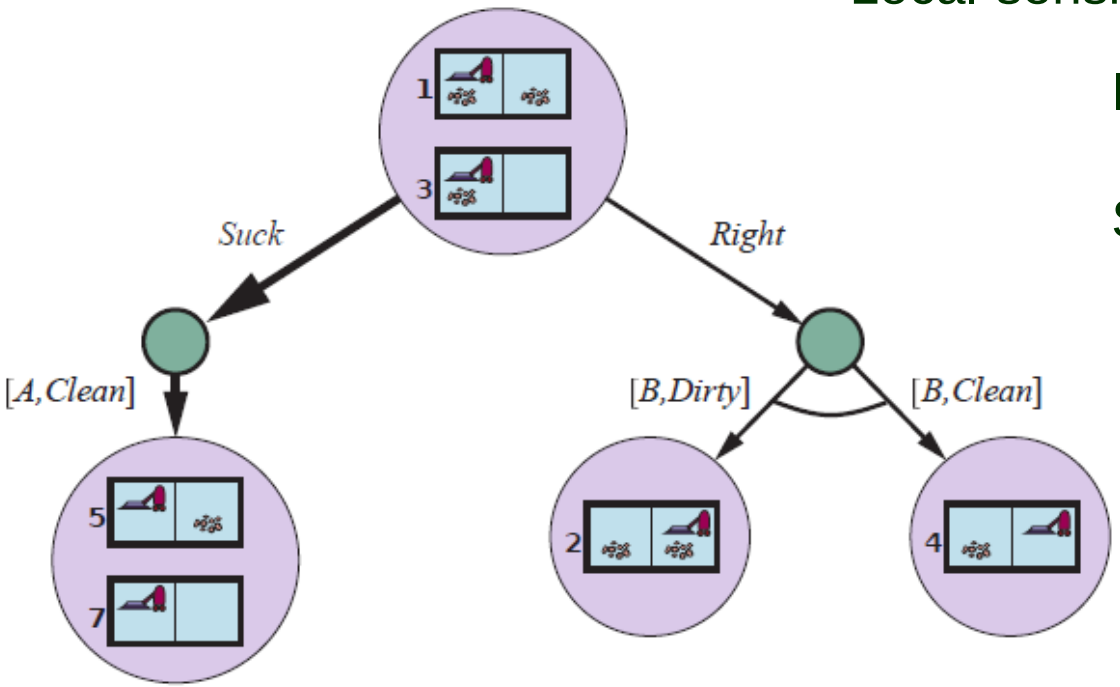
Deterministic
(local sensing)



Nondeterministic
(local sensing + slippery world)

Solution with Partial Observation

Apply the AND-OR search algorithm.



...

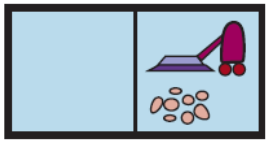
Local-sensing vacuum world

Initial percept: [*A*, *Dirty*]

Solution:

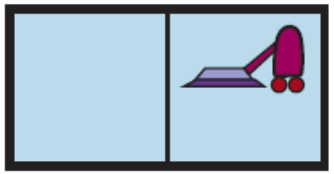
Suck, *Right*,
if b-state={6}
then *Suck*
else []

6



Goal:

8



Prediction-Update Cycles

- Test the condition and execute the appropriate branch.
- Maintain its belief state as it performs actions and receives percepts.

$$b' = \text{UPDATE}(\text{PREDICT}(b, a), o)$$

Prediction-Update Cycles

- Test the condition and execute the appropriate branch.
- Maintain its belief state as it performs actions and receives percepts.

$$b' = \text{UPDATE}(\text{PREDICT}(b, a), o)$$

Consider the local-sensing vacuum world in which

any square may become dirty *any time* unless being actively cleaned.

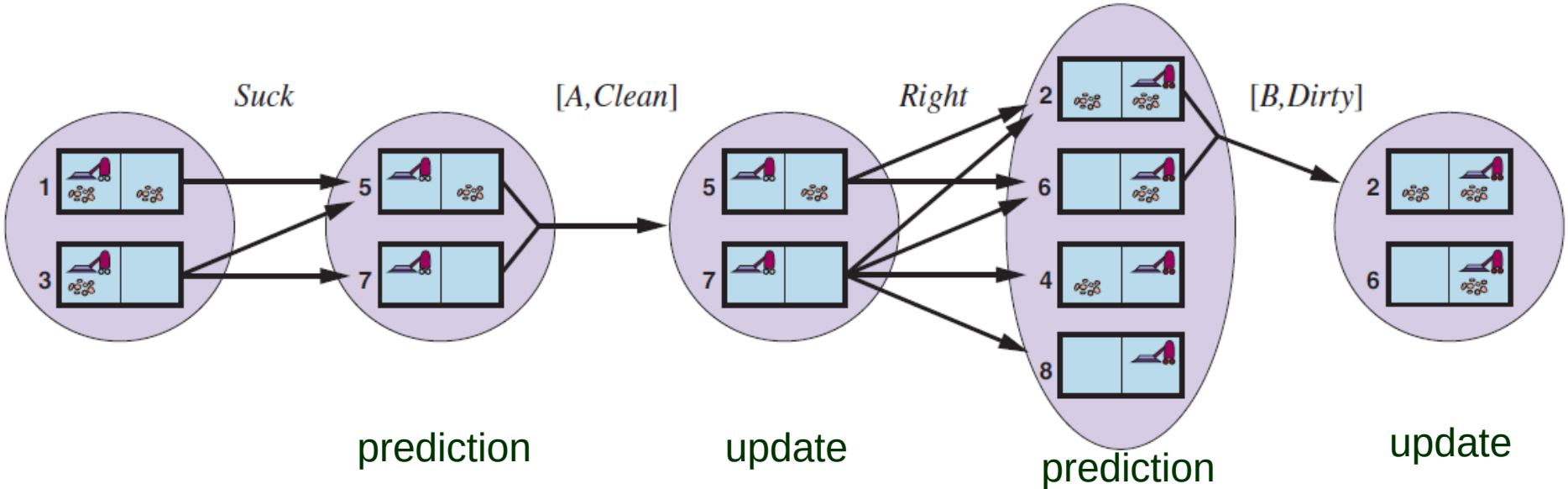
Prediction-Update Cycles

- Test the condition and execute the appropriate branch.
- Maintain its belief state as it performs actions and receives percepts.

$$b' = \text{UPDATE}(\text{PREDICT}(b, a), o)$$

Consider the local-sensing vacuum world in which

any square may become dirty *any time* unless being actively cleaned.



Robot Localization

Problem Use a map to determine where a robot is located with respect to its environment.

Robot Localization

Problem Use a map to determine where a robot is located with respect to its environment.

- Perfect sensing (by four sonar sensors)

  Percept: 4 bits *NESW* (obstacle in the north, east, south, and west?)

Robot Localization

Problem Use a map to determine where a robot is located with respect to its environment.

- Perfect sensing (by four sonar sensors)

  Percept: 4 bits *NESW* (obstacle in the north, east, south, and west?)

b: set of all locations

Robot Localization

Problem Use a map to determine where a robot is located with respect to its environment.

- Perfect sensing (by four sonar sensors)

Percept: 4 bits *NESW* (obstacle in the north, east, south, and west?)

b: set of all locations

$\Downarrow$ *NESW* $= 1011$ (obstacles in *N*, *S*, *W*)

$b_0 =$Update$(b, 1011)$

Robot Localization

Problem Use a map to determine where a robot is located with respect to its environment.

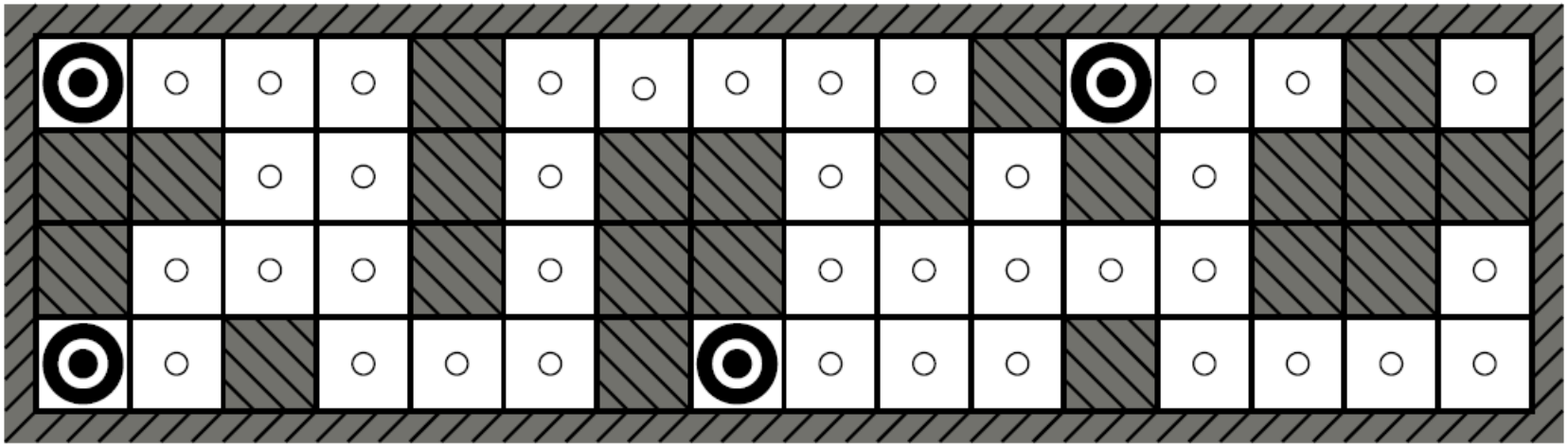
- Perfect sensing (by four sonar sensors)

Percept: 4 bits *NESW* (obstacle in the north, east, south, and west?)

b: set of all locations

⇩ *NESW* = 1011 (obstacles in *N*, *S*, *W*)

b_0 =Update$(b, 1011)$



Robot Localization

Problem Use a map to determine where a robot is located with respect to its environment.

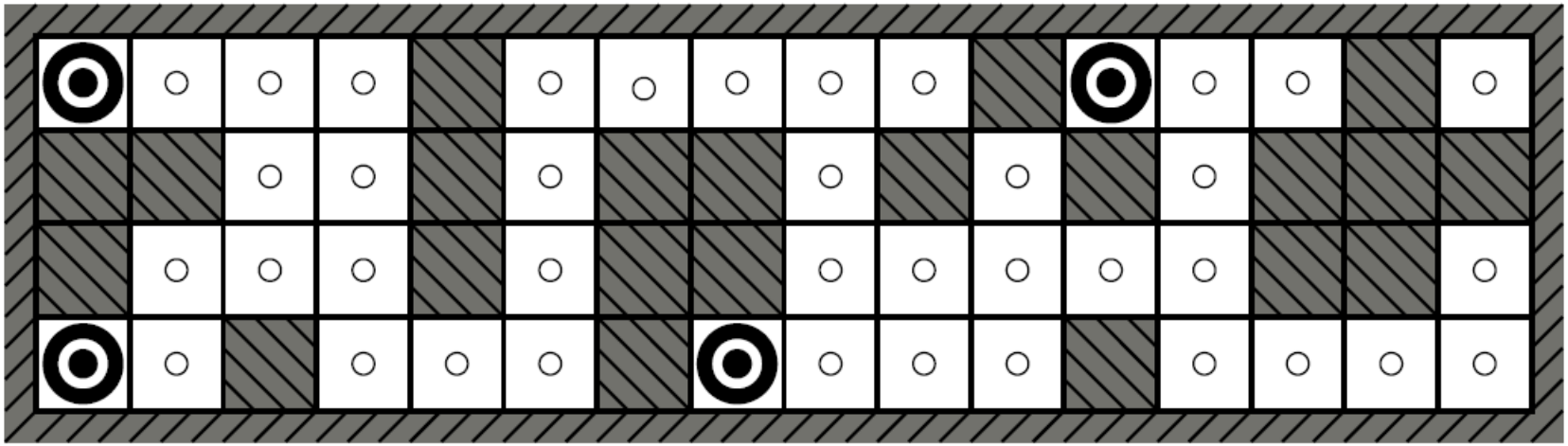
- Perfect sensing (by four sonar sensors)

Percept: 4 bits *NESW* (obstacle in the north, east, south, and west?)

b: set of all locations

⇩ *NESW* = 1011 (obstacles in *N*, *S*, *W*)

b_0 =Update$(b, 1011)$ // only four possible locations



Robot Localization

- Broken navigation: Action *Right* takes the robot randomly to one of the four adjacent squares.

Robot Localization

- Broken navigation: Action *Right* takes the robot randomly to one of the four adjacent squares.

Next, the robot executes a *Right* action.

b_a= PREDICT$(b_0, Right)$

Robot Localization

- Broken navigation: Action *Right* takes the robot randomly to one of the four adjacent squares.

Next, the robot executes a *Right* action.

b_a= PREDICT$(b_0, Right)$

$\Downarrow$ *NESW* $= 1010$

b_1 =Update$(b_a, 1010)$

Robot Localization

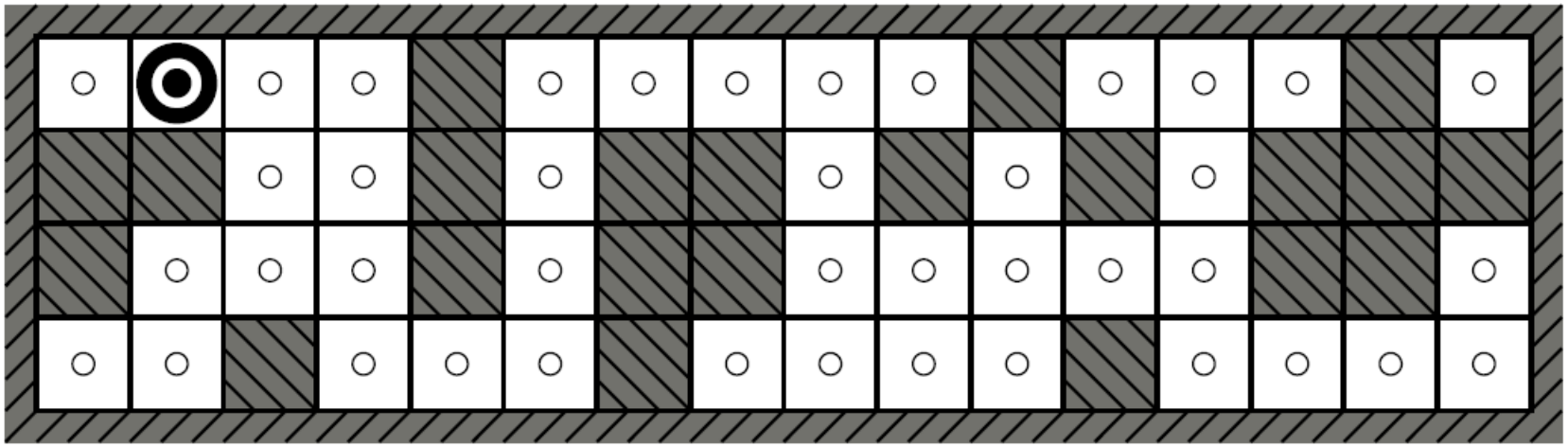
- Broken navigation: Action *Right* takes the robot randomly to one of the four adjacent squares.

Next, the robot executes a *Right* action.

b_a= PREDICT$(b_0, Right)$

$\Downarrow$ $NESW = 1010$

b_1 =Update$(b_a, 1010)$ // One possible location!



Robot Localization

- Broken navigation: Action *Right* takes the robot randomly to one of the four adjacent squares.

Next, the robot executes a *Right* action.

b_a= PREDICT$(b_0, Right)$

$\Downarrow$ $NESW = 1010$

b_1 =Update$(b_a, 1010)$ // One possible location!