Designing Intelligent Agents

EMATM0042 – Intelligent Information Systems

Monday 11 March – Part 2

kevin.mcareavey@bristol.ac.uk

Previous Lecture…

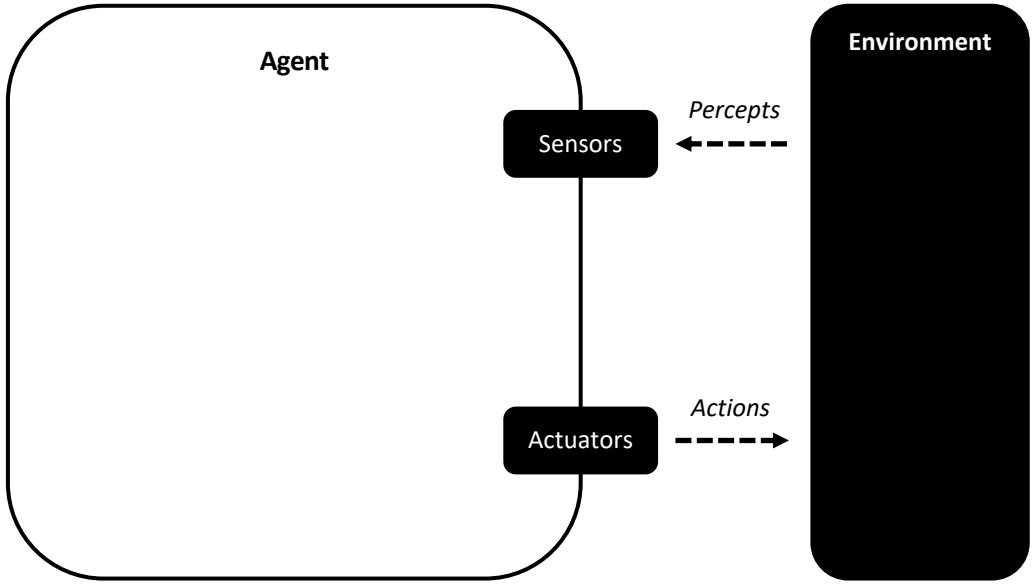
1. Background
   - Trends in computing
2. Informal definition
   - Agent
   - MAS
3. Understanding the problem
   - Micro vs. macro
   - Engineering vs. theory
4. Applications

This Lecture…

1. Abstract agent representations
   - Agent function
   - Agent program
2. Task environments
   - Problem Analysis
   - Properties
3. Types of agents
   - Reactive agents
   - Proactive agents
   - Cognitive agents

Agents

Basic Definition

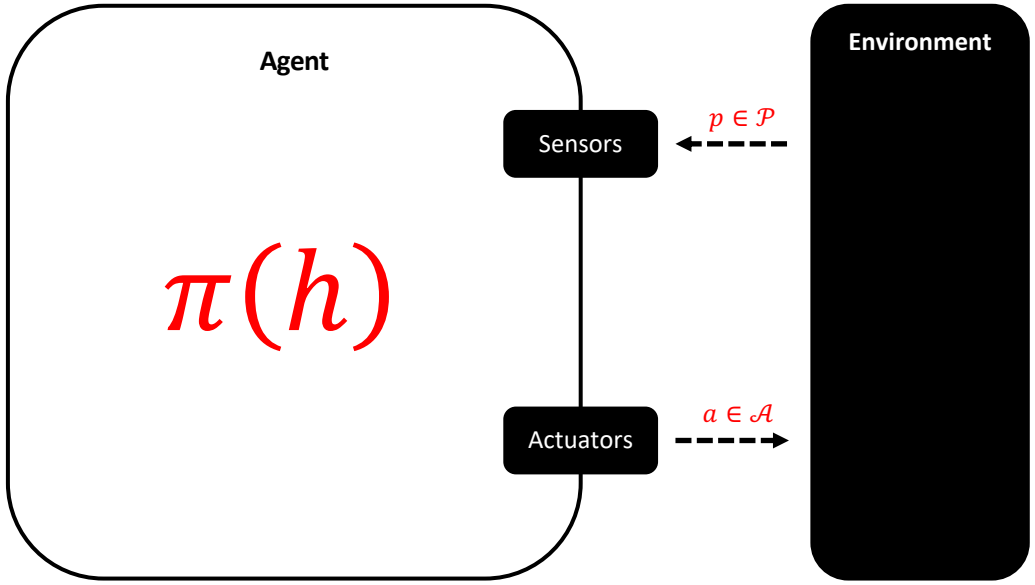


- An agent is an entity that acts
  - Situated in an environment
  - Acts upon its environment through actuators
  - Perceives its environment though sensors

Agent	Environment	Actuators	Sensors
Human?	Real-world	Hands, vocals, etc.	Eyes, ears, etc.
Robot?	Real-world	Motors, etc.	Cameras, etc.
Internet bot?	Internet	HTTP POST, etc.	Text search, etc.
Light switch?	Circuit	Current	Switch

Agents

Basic Definition: Formal



- Set of actions $\mathcal{A} = \{a, a', \dots\}$
- Set of percepts $\mathcal{P} = \{p, p', \dots\}$

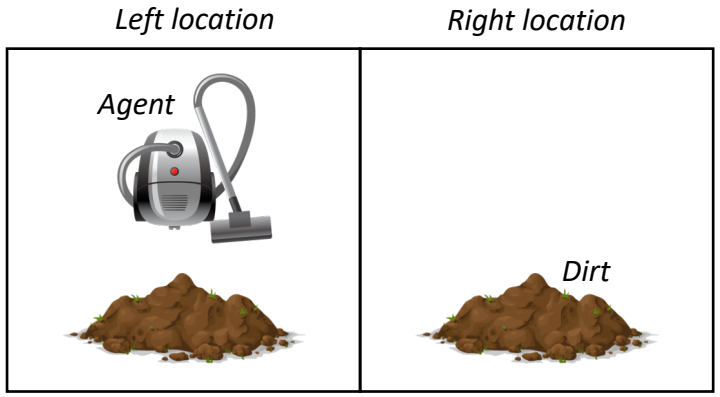
} Atomic symbols

- A history is a finite sequence $h = (a, p, \dots, a', p')$
  - Action-percept sequence starting with an action and ending with a percept
  - Complete record of what has happened in the agent's life
- Set of histories $H = \{h, h', \dots\}$
- An agent function is a function $\pi : H \to \mathcal{A}$
  - A mapping from histories to actions

Agents

Basic Definition: Formal – Example

- Suppose we have a vacuum-cleaner agent…
  - Assume that it persists forever
  - Assume that its sensing is restricted to its current location

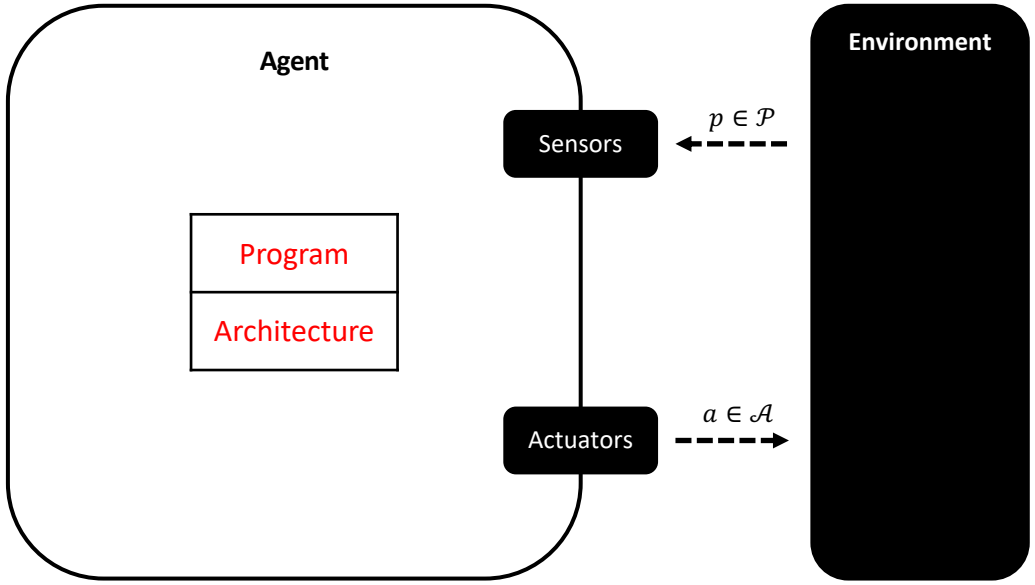


- Actions $a \in \mathcal{A}$
  - $\mathrm{Move(Right)}$: move to right location
  - $\mathrm{Move(Left)}$: move to left location
  - Clean: remove dirt from current location

- Percepts $p \in \mathcal{P}$
  - $\mathrm{Sensed(Left, Dirty)}$: dirt in left location
  - $\mathrm{Sensed(Left, Clean)}$: no dirt in left location
  - $\mathrm{Sensed(Right, Dirty)}$: dirt in right location
  - $\mathrm{Sensed(Right, Clean)}$: no dirt in right location

History $h \in H$	Agent function $\pi(h)$
( )	Clean
$(\mathrm{Clean}, \mathrm{Sensed(Left, Clean)})$	Move(Right)
$(\mathrm{Clean}, \mathrm{Sensed(Right, Clean)})$	Move(Left)
…	…
$(\mathrm{Clean}, \mathrm{Sensed(Left, Clean)}, \mathrm{Move(Right)}, \mathrm{Sensed(Right, Dirty)})$	Clean
$(\mathrm{Clean}, \mathrm{Sensed(Left, Clean)}, \mathrm{Move(Right)}, \mathrm{Sensed(Right, Clean)})$	Move(Left)
…	…

Agents
Implementation



- Agent function is just an abstract mathematical description of an agent
  - Set of histories is very large or infinite
  - Says nothing about how to choose action $\pi(h)$
- Need a way to implicitly define an agent function
  - Concise representation
  - Decide what action to perform
- An agent program implements an agent function
  - This is our main focus
- An agent program runs on an agent architecture
  - Physical robot, virtual machine, agent interpreter, etc.

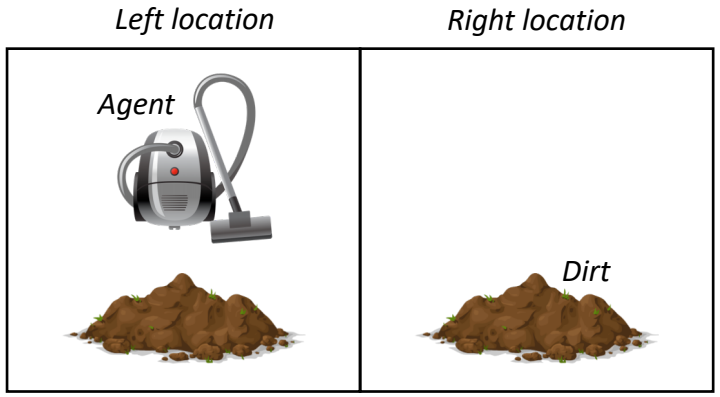
Agents

Implementation – Example

Algorithm: Vacuum-cleaner agent program

```
1 EXECUTE(Clean)
2 while true do
3     Sensed(x, y) ← SENSE()
4     if y = Dirty then
5         EXECUTE(Clean)
6     else if x = Left then
7         EXECUTE(Move(Right))
8     else
9         EXECUTE(Move(Left))
```

Implicitly defines...



History $h \in H$	Agent function $\pi(h)$
( )	Clean
(Clean, Sensed(Left, Clean))	Move(Right)
(Clean, Sensed(Right, Clean))	Move(Left)
...	...
(Clean, Sensed(Left, Clean), Move(Right), Sensed(Right, Dirty))	Clean
(Clean, Sensed(Left, Clean), Move(Right), Sensed(Right, Clean))	Move(Left)
...	...

Agents

Intelligent Agents

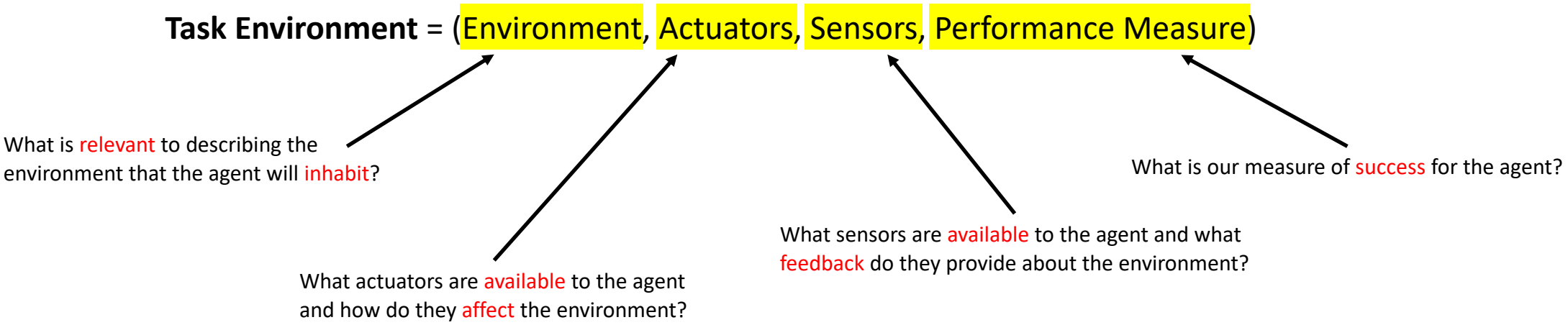
- An agent is autonomous if it does not rely exclusively on prior knowledge
  - The agent should at least account for its own experience
- An agent is rational if it acts towards achieving the “best” outcome
  - Behaviour should fit some criterion for success
- We are typically interested in agents that are autonomous and rational
  - We might call these intelligent agents, but the name is not important…

Agents

Understanding the Problem

- An agent is a complete system that solves a specific problem by integrating a range of (often shallow) competences

- Analysis of the task environment helps us understand what competences are important and what are not



Agents

Understanding the Problem – Example

Agent	Environment	Actuators	Sensors	Performance Measure
Driverless car	Roads, other vehicles, pedestrians, weather	Steering, accelerator, brake, signal, horn, display	Cameras, sonar, speedometer, GPS, odometer, accelerometer, engine sensors, keyboard	Safety, speed, legality, comfort
Medical diagnosis system	Patient, hospital, staff	Display of questions, tests, diagnoses, treatments, referrals	Keyboard entry of symptoms, findings, patient's answers	Healthy patient, reduced costs
Satellite image analysis system	Downlink from orbiting satellite	Display of scene categorization	Color pixel arrays	Correct image categorization
Part-picking robot	Conveyor belt with parts; bins	Jointed arm and hand	Camera, joint angle sensors	Percentage of parts in correct bins
Refinery controller	Refinery, operators	Valves, pumps, heaters, displays	Temperature, pressure, chemical sensors	Purity, yield, safety
Interactive English tutor	Set of students, testing agency	Display of exercises, suggestions, corrections	Keyboard entry	Student's score on test

Task Environment

Properties

- Deciding how to model a task environment is crucial to choosing an appropriate agent design
  - What properties does it exhibit?
  - What design trade-offs are appropriate?
- In practice, we need to impose simplifying assumptions or restrictions in our model
  - Only possible to approximate many real task environments (e.g. robots in the the real-world)
  - Computational considerations
- Allow more complexity for properties that have greater impact on the performance measure, and less complexity for others

#	Less Challenging	More Challenging
1	Discrete	Continuous
2	Deterministic	Stochastic (or non-deterministic)
3	Fully observable	Partially observable (or non-observable)
4	Episodic	Sequential
5	Static	Dynamic
6	Single agent	Multi-agent
7	Known	Unknown

Task Environment

Properties: (1) Discrete vs. Continuous

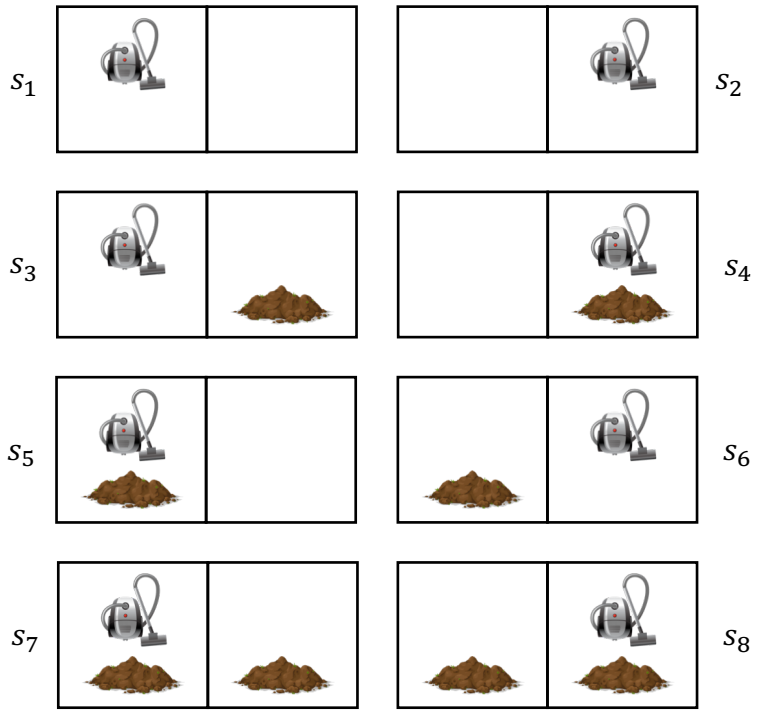
- …discrete if there is a finite number of environment states, actions, and percepts

- …continuous if there is not a finite number of environment states, actions, or percepts
  - Subsumes discrete task environments (e.g. continuous environment states with discrete actions and percepts)

#	Less Challenging	More Challenging
1	Discrete	Continuous
2	Deterministic	Stochastic (or non-deterministic)
3	Fully observable	Partially observable (or non-observable)
4	Episodic	Sequential
5	Static	Dynamic
6	Single agent	Multi-agent
7	Known	Unknown

Task Environment

Properties: (1) Discrete vs. Continuous – Example



- Discrete environment states $\mathcal{S} = \{s_1, s_2, \dots, s_8\}$
  - Suppose we include (real) time in the environment states?
- Discrete actions $\mathcal{A} = \left\{ \begin{matrix} \text{Move(Left)}, \\ \text{Move(Right)}, \\ \text{Clean} \end{matrix} \right\}$
- Discrete percepts $\mathcal{P} = \left\{ \begin{matrix} \text{Sensed(Left, Clean)}, \\ \text{Sensed(Left, Dirty)}, \\ \text{Sensed(Right, Clean)}, \\ \text{Sensed(Right, Dirty)} \end{matrix} \right\}$

…this environment is discrete

Task Environment

Properties: (2) Deterministic vs. Stochastic

- …deterministic if each action (or event) leads from one environment state to a unique successor state

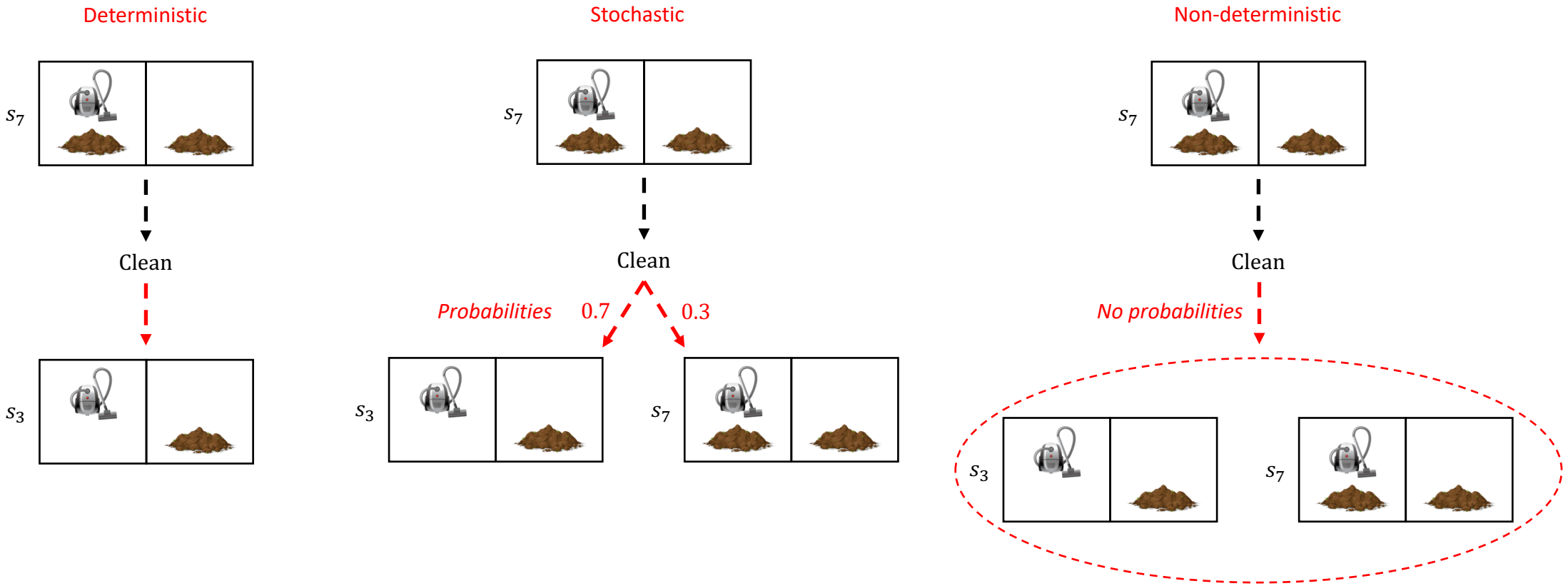
- …non-deterministic if each action (or event) leads to one-of-many possible successor states
  - Subsumes deterministic task environments

- …stochastic if it is non-deterministic and there are probabilities associated with successor states
  - Subsumes non-deterministic task environments, assuming that possible successors states have equal (non-zero) probability of occurring

#	Less Challenging	More Challenging
1	Discrete	Continuous
2	Deterministic	Stochastic (or non-deterministic)
3	Fully observable	Partially observable (or non-observable)
4	Episodic	Sequential
5	Static	Dynamic
6	Single agent	Multi-agent
7	Known	Unknown

Task Environment

Properties: (2) Deterministic vs. Stochastic – Example



Task Environment

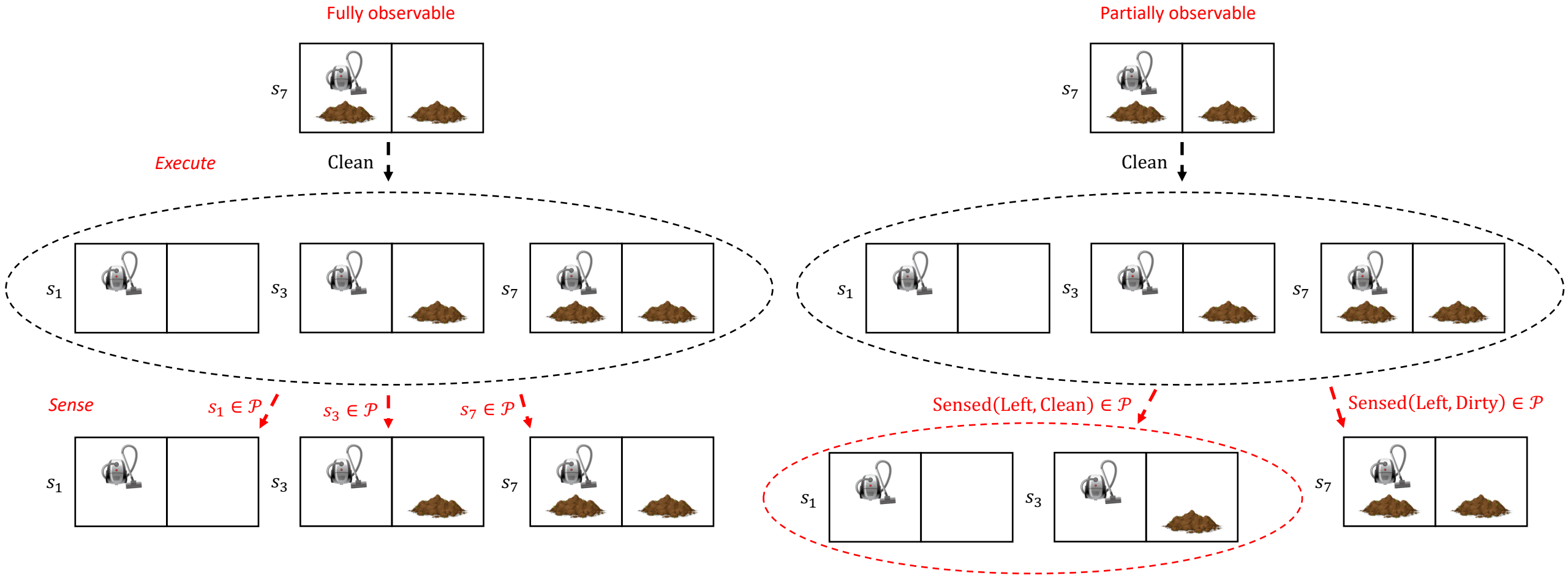
Properties: (3) Fully Observable vs. Partially Observable

- …fully observable if percepts provide complete information about current environment state
  - Agent receives feedback and is always certain about its current environment state when executing actions
- …partially observable if percepts provide incomplete information
  - Agent receives feedback but may be uncertain about its current environment state when executing actions
  - Subsumes fully observable environments
- …non-observable if percepts provide no information
  - Agent receives no feedback and may be uncertain about its current environment state when executing actions
  - Subsumed by partially observable environments

#	Less Challenging	More Challenging
1	Discrete	Continuous
2	Deterministic	Stochastic (or non-deterministic)
3	Fully observable	Partially observable (or non-observable)
4	Episodic	Sequential
5	Static	Dynamic
6	Single agent	Multi-agent
7	Known	Unknown

Task Environment

Properties: (3) Fully Observable vs. Partially Observable – Example



Task Environment

Properties: (4) Episodic vs. Sequential

- …episodic if the agent's decisions occur in discrete episodes with no dependencies on the past or future
  - Agent chooses actions based on its latest percept only, and does not consider long-term consequences

- …sequential if the agent's decisions depend on its past and/or its expectations about the future
  - Agent chooses actions based on its history, and considers the long-term consequences of its actions
  - Subsumes episodic task environments

#	Less Challenging	More Challenging
1	Discrete	Continuous
2	Deterministic	Stochastic (or non-deterministic)
3	Fully observable	Partially observable (or non-observable)
4	Episodic	Sequential
5	Static	Dynamic
6	Single agent	Multi-agent
7	Known	Unknown

Task Environment

Properties: (4) Episodic vs. Sequential – Example

Episodic: part-picking robot

Sequential: chess

Task Environment

Properties: (5) Static vs. Dynamic

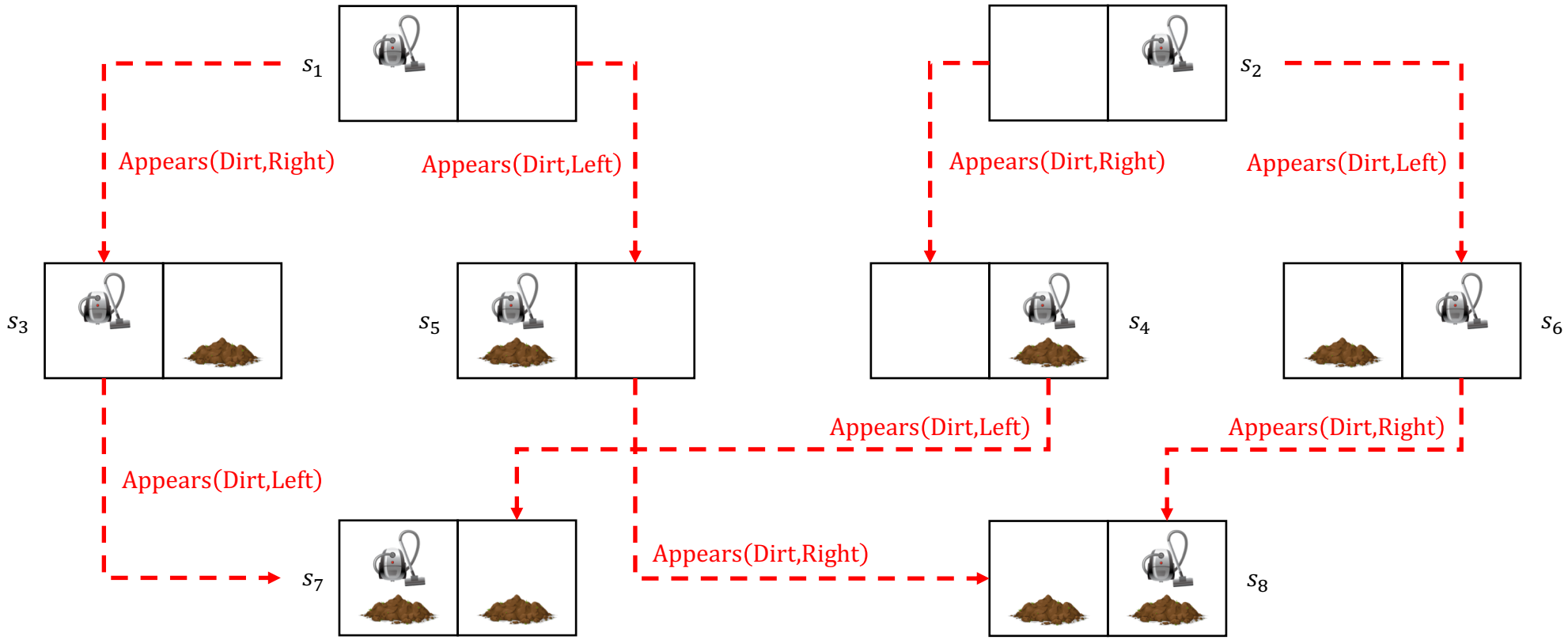
- …static if there are no exogenous events
  - Agent has complete control over the environment
- …dynamic if there may be exogenous events
  - Environment state may change due to external events

#	Less Challenging	More Challenging
1	Discrete	Continuous
2	Deterministic	Stochastic (or non-deterministic)
3	Fully observable	Partially observable (or non-observable)
4	Episodic	Sequential
5	Static	Dynamic
6	Single agent	Multi-agent
7	Known	Unknown

Task Environment

Properties: (5) Static vs. Dynamic – Example

Exogenous events $\mathcal{E} = \{\text{Appears(Dirt,Left)}, \text{Appears(Dirt,Right)}\}$



Task Environment

Properties: (6) Single Agent vs. Multi-Agent

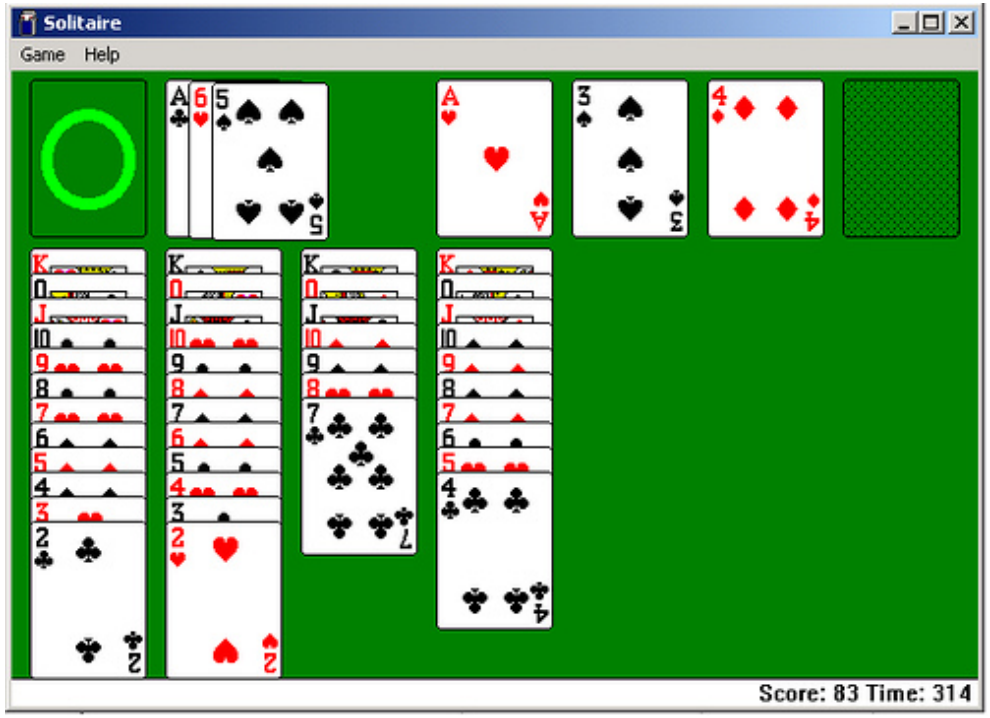
- …single agent if the agent does not account for the behaviour other agents
  - Might be the only agent that inhabits the environment
  - Might treat the actions of other agents as exogenous events (i.e. single agent but dynamic)

- …multi-agent if the agent accounts for the behavior of other agents when choosing its own actions
  - Cooperation, competition, etc.
  - Subsumes single agent task environments

#	Less Challenging	More Challenging
1	Discrete	Continuous
2	Deterministic	Stochastic (or non-deterministic)
3	Fully observable	Partially observable (or non-observable)
4	Episodic	Sequential
5	Static	Dynamic
6	Single agent	Multi-agent
7	Known	Unknown

Task Environment

Properties: (6) Single Agent vs. Multi-Agent – Example



Single agent: solitaire

Multi-agent: poker

Task Environment

Properties: (7) Known vs. Unknown

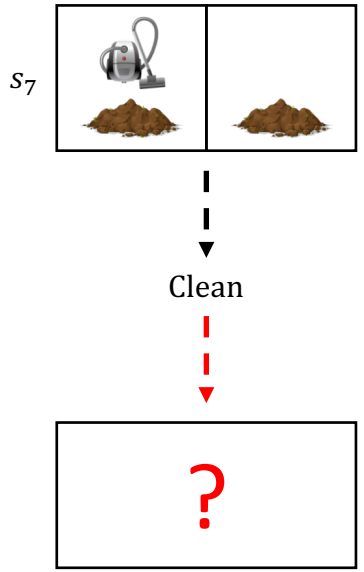
- …known if the agent has a complete model of the task environment
  - Model may be imperfect, but is typically fixed
- …unknown if the agent has an incomplete model (or no model) of the task environment
  - Agent may be required to learn (or improve) the model

#	Less Challenging	More Challenging
1	Discrete	Continuous
2	Deterministic	Stochastic (or non-deterministic)
3	Fully observable	Partially observable (or non-observable)
4	Episodic	Sequential
5	Static	Dynamic
6	Single agent	Multi-agent
7	Known	Unknown

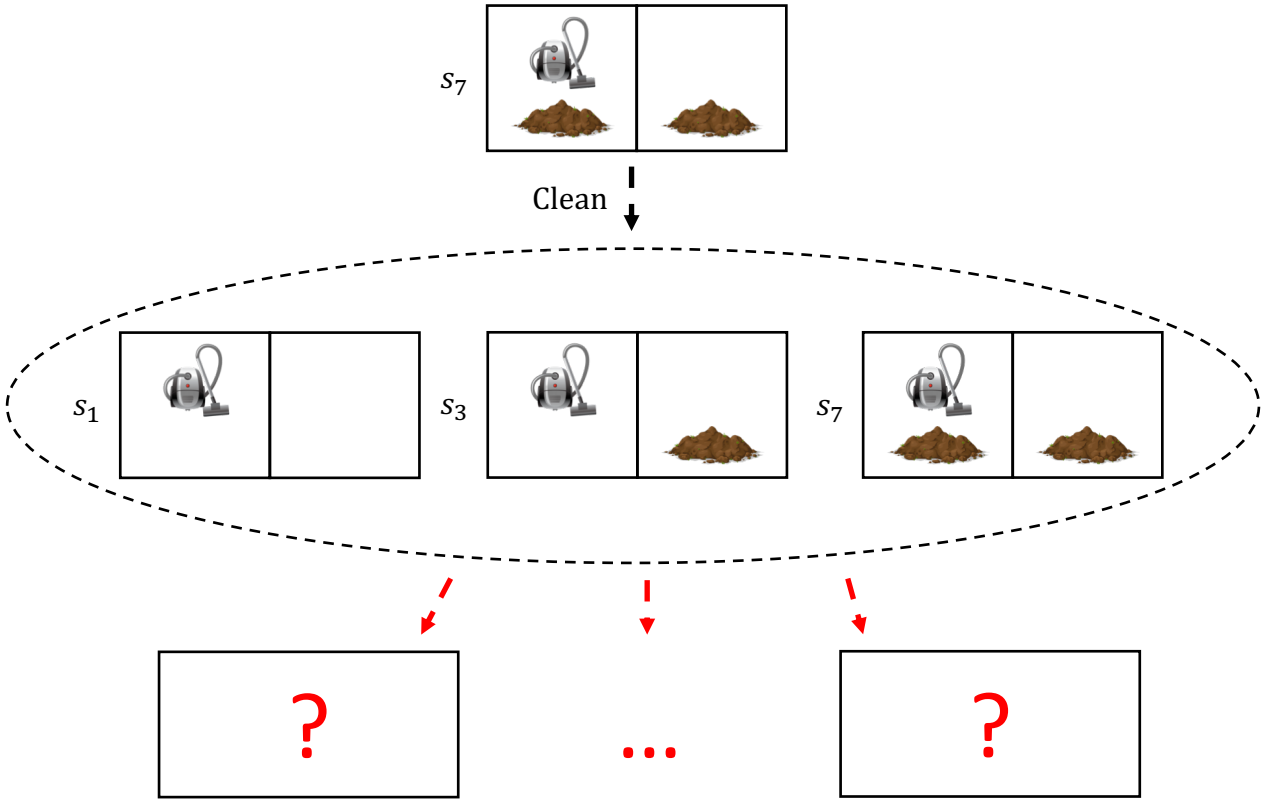
Task Environment

Properties: (7) Known vs. Unknown – Example

Unknown actuator model



Unknown sensor model



Types of Agents

Overview

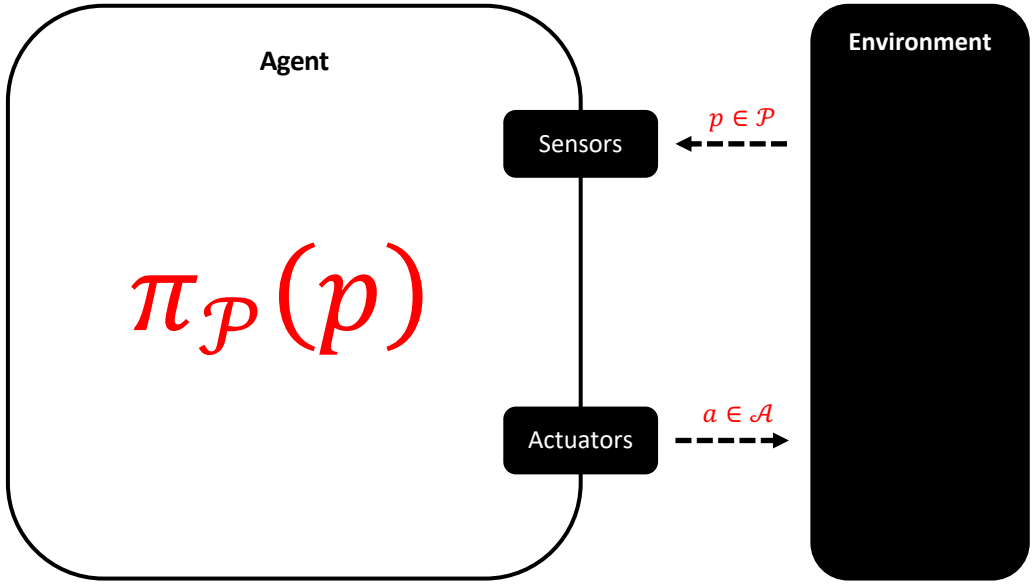
- Reactive agents
  - Memoryless
  - With belief state
- Proactive agents
  - Goal-based agents
  - Utility-based agents
- Cognitive agents
  - Belief-desire-intention agents

We will touch on each of these properties…

#	Less Challenging	More Challenging
1	Discrete	~~Continuous~~
2	Deterministic	Stochastic (or non-deterministic)
3	Fully observable	Partially observable (or non-observable)
4	Episodic	Sequential
5	Static	Dynamic
6	Single agent	~~Multi-agent~~
7	Known	~~Partially known (or unknown)~~

Reactive Agents

Memoryless



- A memoryless agent function is $\pi_{\mathcal{P}} : \mathcal{P} \rightarrow \mathcal{A}$
  - A mapping from percepts to actions
- Decides what to do based only on latest percept
  - Ignores rest of history
- Induces an agent function π
  - $\pi(h) = \pi_{\mathcal{P}}(p)$ for any $h \in H$ such that h ends with p
- Can work well when the agent has full observability
  - Problematic when the agent only has partial observability…

Reactive Agents

Memoryless – Example

Algorithm: Memoryless rule-based agent program

Input: Condition-action rule base K

```
1 while true do
2     p ← SENSE()
3     for each (φ, a) ∈ K do
4         if p ⊨ φ then
5             EXECUTE(a)
6             break
```

#	Condition ϕ	Action $a \in \mathcal{A}$
1	Sensed(x, Dirty)	Clean
2	Sensed(Left, Clean)	Move(Right)
3	Sensed(Right, Clean)	Move(Left)

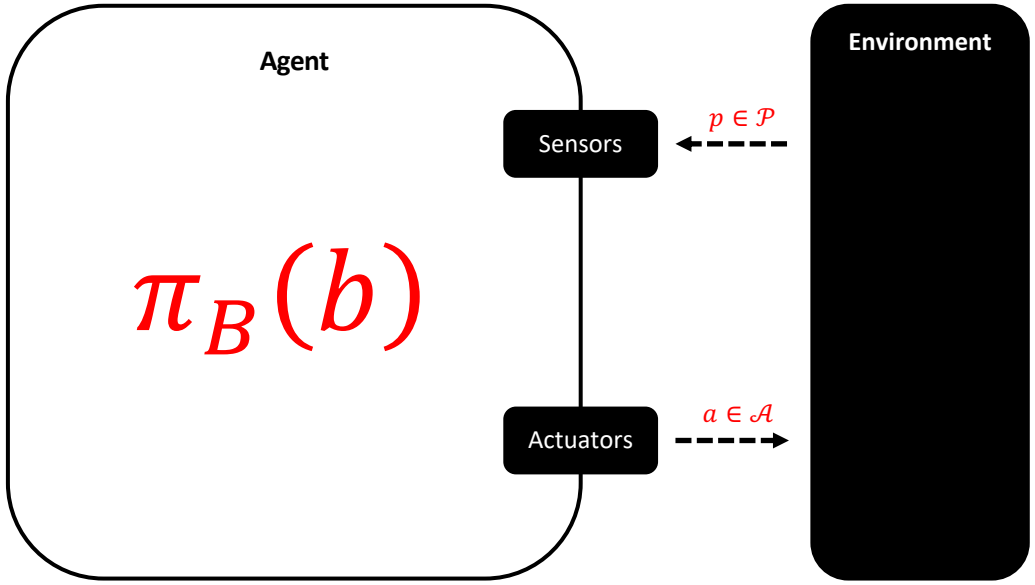
Condition-action rule

Agent function

Percept $p \in \mathcal{P}$	Memoryless agent function $\pi_{\mathcal{P}}(p)$
Sensed(Left, Dirty)	Clean
Sensed(Right, Dirty)	Clean
Sensed(Left, Clean)	Move(Right)
Sensed(Right, Clean)	Move(Left)

Reactive Agents

With Belief State



- Set of belief states B
  - A belief state $b \in B$ represents the agent's (potentially uncertain) view of the current environment state
- A belief-based agent function $\pi_B : B \to \mathcal{A}$
  - A mapping from belief states to actions
- Induces an agent function π
  - $\pi(h) = \pi_B(b)$ for any $h \in H$ such that b is the belief state associated with h
- How to model an agent's belief state?

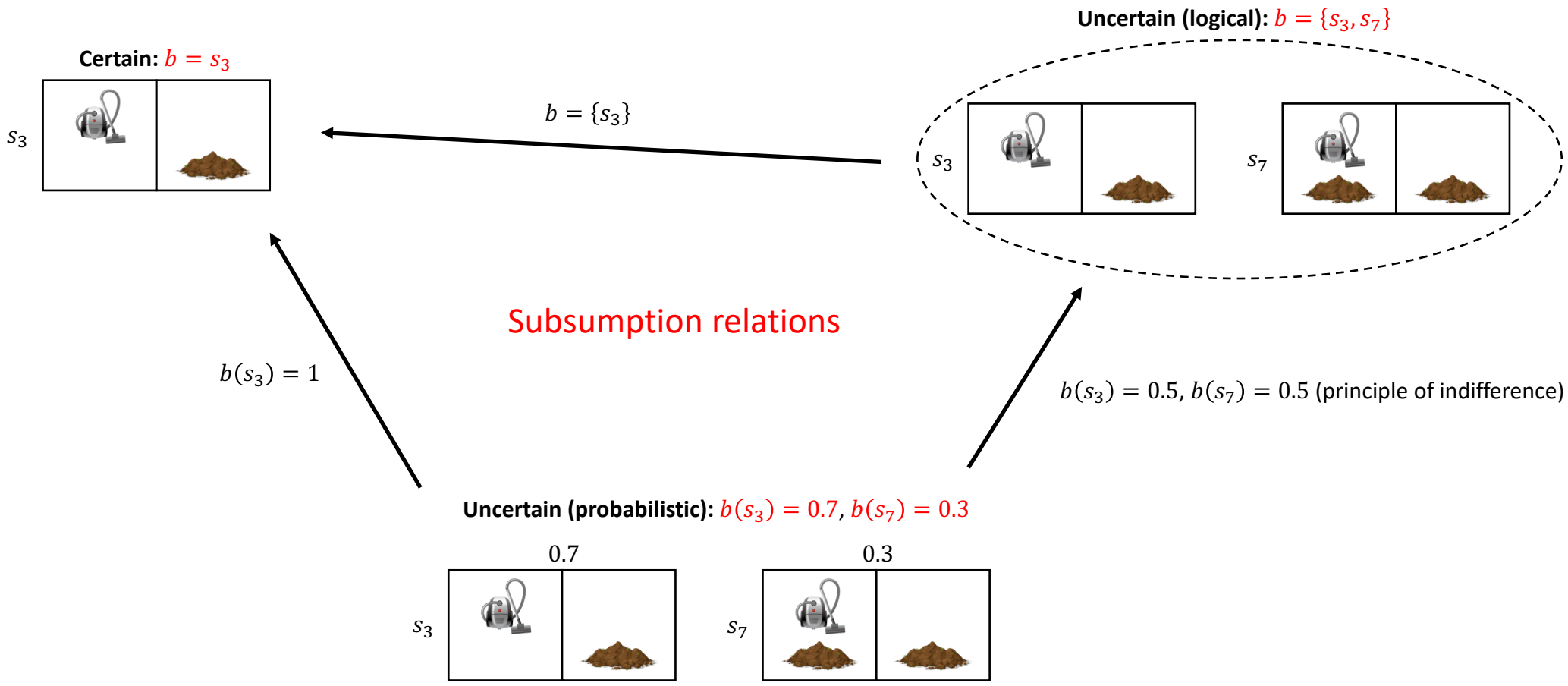
Reactive Agents

With Belief State: Belief State Model

- If the agent has full observability, then $B = S$
  - No uncertainty over the current environment state, since each percept uniquely identifies an environment state
- If the agent has only partial observability, then $B \neq S$, e.g.
  - $B = 2^S$ if we use a logical model of uncertainty
  - $B = \Pi(S)$ if we use a probabilistic model of uncertainty
- With partial observability, it is necessary to track the agent's current belief state
  - With full observability, a memoryless agent is sufficient
- How to update the belief state after each action-percept cycle?
  - Need an actuator model and a sensor model

Reactive Agents

With Belief State: Belief State Model – Example



Reactive Agents

With Belief State: Actuator Model

- Actuator model is typically formalized as a transition function $T : \mathcal{S} \times \mathcal{A} \rightarrow \cdots$
  - Effect of action depends only on current environment state (see e.g. Markov assumption)

- Deterministic transition function $T : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$
  - Executing action $a \in \mathcal{A}$ in environment state $s \in \mathcal{S}$ results in a single successor state $T(s, a) \in \mathcal{S}$

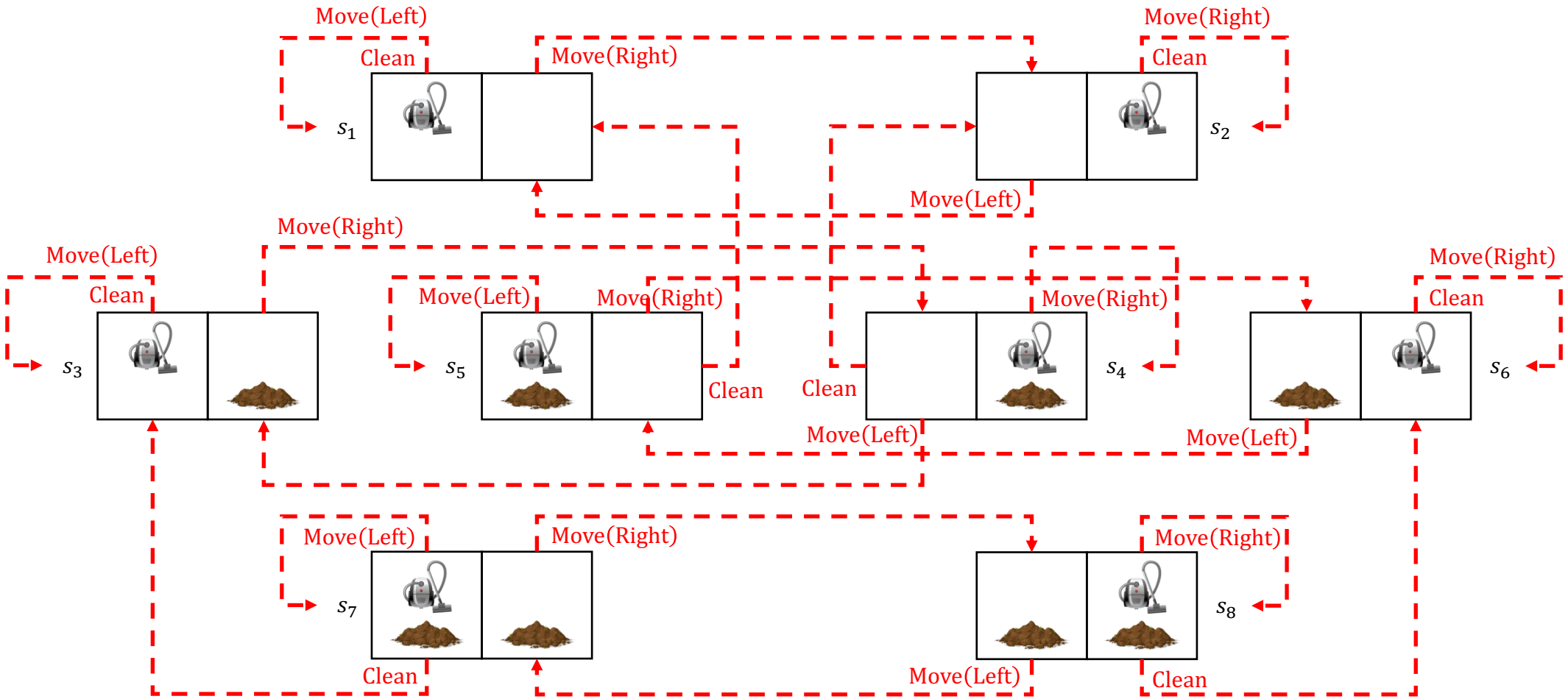
- Non-deterministic transition function $T : \mathcal{S} \times \mathcal{A} \rightarrow 2^{\mathcal{S}}$
  - Executing action $a \in \mathcal{A}$ in environment state $s \in \mathcal{S}$ results in a set of possible successor states $T(s, a) \subseteq \mathcal{S}$

- Stochastic transition function $T : \mathcal{S} \times \mathcal{A} \rightarrow \Pi(\mathcal{S})$
  - Executing action $a \in \mathcal{A}$ in environment state $s \in \mathcal{S}$ results in a probability distribution over successor states $T(s, a) \in \Pi(\mathcal{S})$

Reactive Agents

With Belief State: Actuator Model – Example

…this is a deterministic transition function



Reactive Agents

With Belief State: Sensor Model

- Sensor model is typically formalized as an observation function $O : \mathcal{S} \rightarrow \mathcal{P}$
  - If the agent arrives in environment state $s \in \mathcal{S}$, they will receive a single percept $O(s) \in \mathcal{P}$
  - May also depend on action as $O : \mathcal{A} \times \mathcal{S} \rightarrow \mathcal{P}$ (i.e. arriving in state $s \in \mathcal{S}$ after executing action $a \in \mathcal{A}$…)

- Observability depends on how O is defined
  - Always (weakly) partially observable
  - Fully observable if O is a bijection (i.e. a one-to-one mapping) from $\mathcal{S}$ to $\mathcal{P}$
  - Non-observable if $O(s_1) = \cdots = O(s_n) = p$ (i.e. p is meaningless)
  - Strictly partially observable if neither fully observable or non-observable

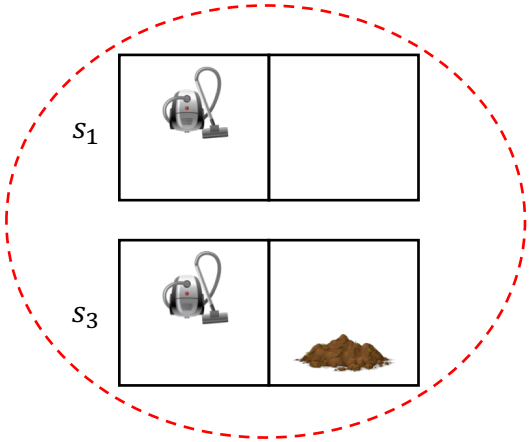
- The above is a deterministic observation function, but it is also possible to define a noisy sensing model
  - Non-deterministic observation function $O : \mathcal{S} \rightarrow 2^{\mathcal{P}}$
  - Stochastic observation function $O : \mathcal{S} \rightarrow \Pi(\mathcal{P})$

Reactive Agents

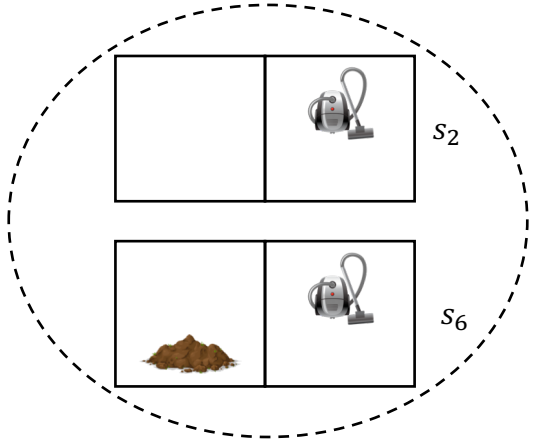
With Belief State: Sensor Model – Example

…this is a deterministic but partially observable observation function

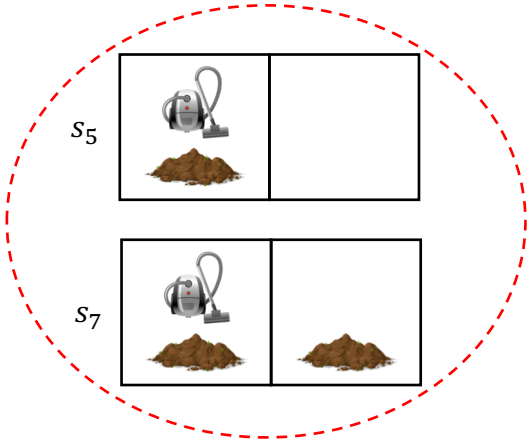
$O(s_1) = O(s_3) = \text{Sensed(Left, Clean)}$



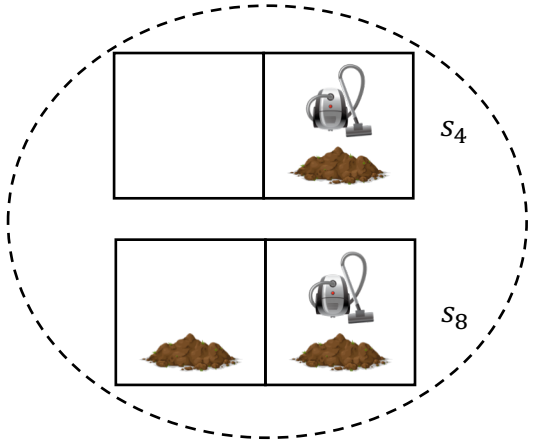
$O(s_2) = O(s_6) = \text{Sensed(Right, Clean)}$



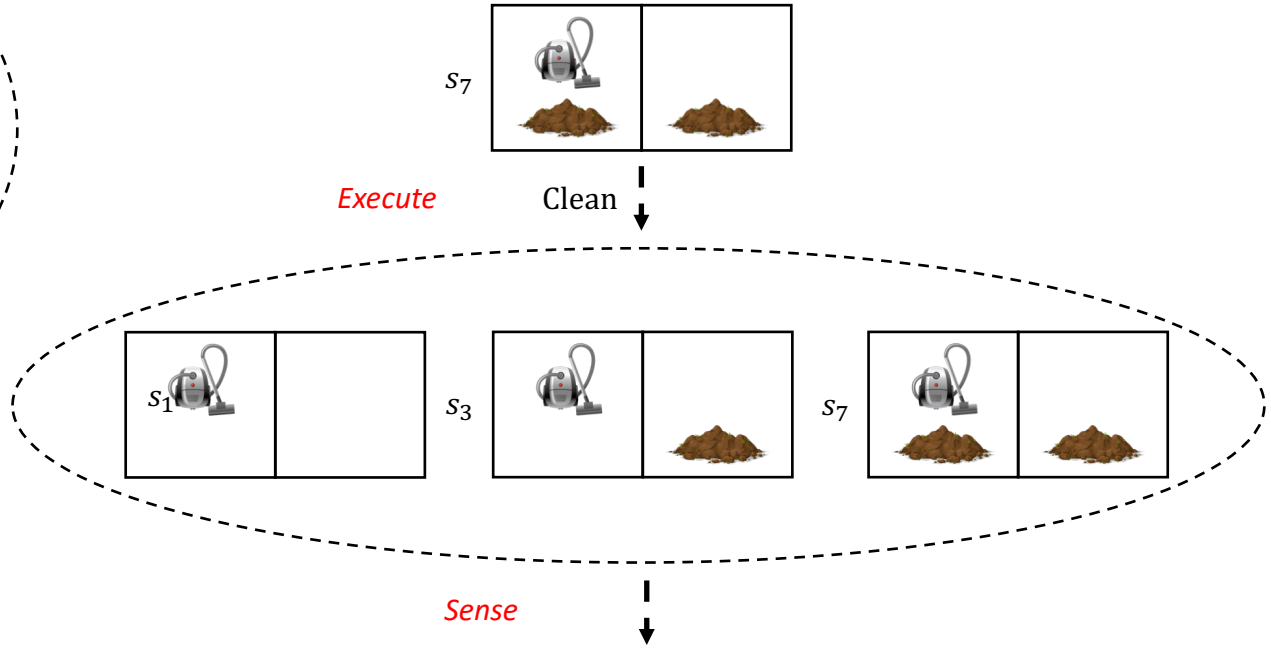
Observation function



$O(s_5) = O(s_7) = \text{Sensed(Left, Dirty)}$



$O(s_4) = O(s_8) = \text{Sensed(Right, Dirty)}$



$p \in \{O(s_1), O(s_3), O(s_7)\} \Leftrightarrow p \in \{\text{Sensed(Left, Clean)}, \text{Sensed(Left, Dirty)}\}$

Reactive Agents

With Belief State: Update

- An update function $U : B \times \mathcal{A} \times \mathcal{P} \rightarrow B$
  - If agent executes action $a \in \mathcal{A}$ in belief state $b \in B$ and observes percept $p \in \mathcal{P}$, then it will be in belief state $U(b, a, p) \in B$

- With a logical model of uncertainty, U would be defined as follows:

$$U(b, a, p) = \{s' \in T(s, a) \mid s \in b, O(s') = p\}$$

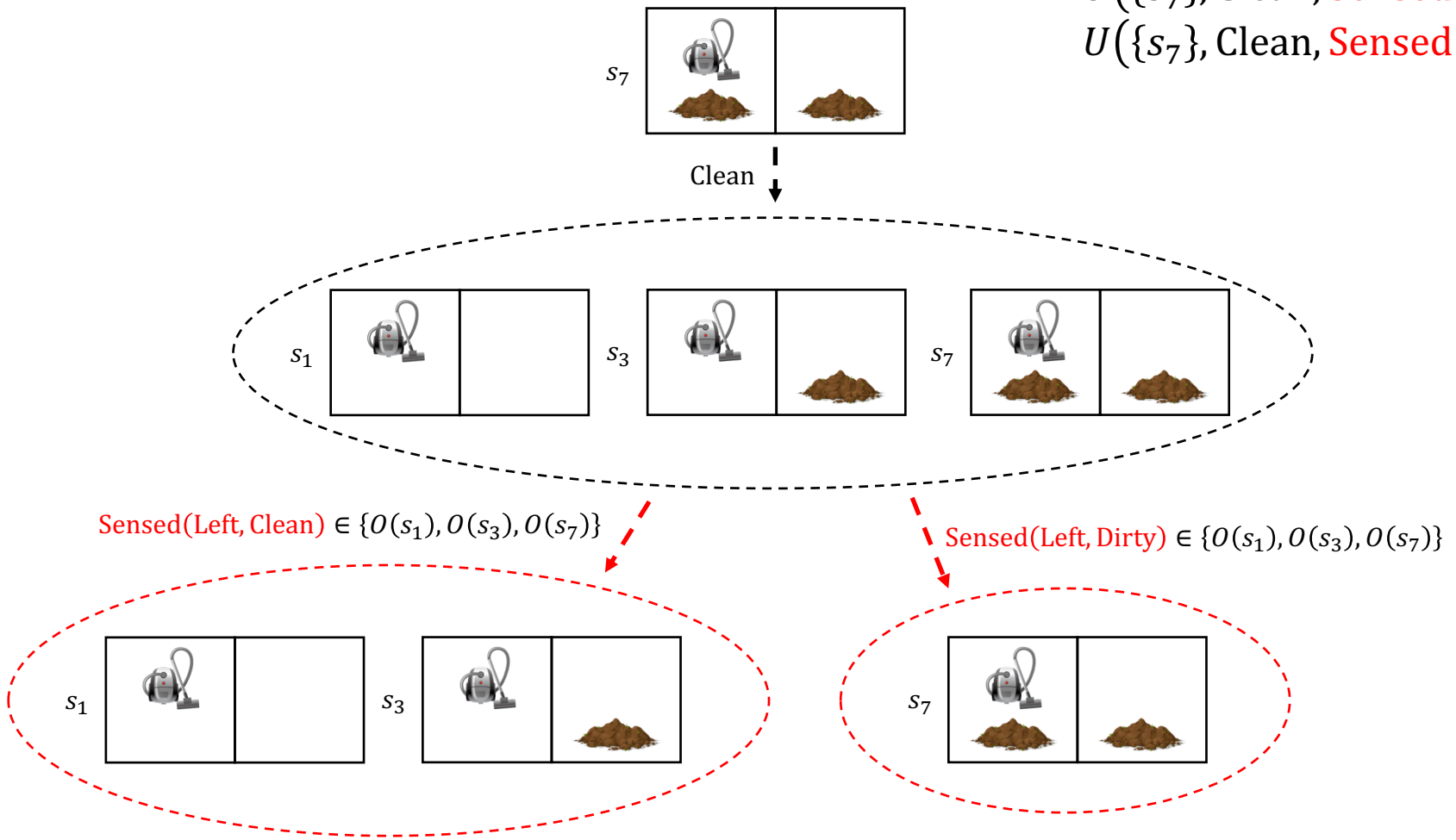
- Similar definitions are possible for probabilistic uncertainty, and for noisy sensing…

Reactive Agents

With Belief State: Update – Example

$U(\{s_7\}, \text{Clean}, \text{Sensed(Left, Clean)}) = \{s_1, s_3\}$

$U(\{s_7\}, \text{Clean}, \text{Sensed(Left, Dirty)}) = \{s_7\}$



Reactive Agents

With Belief State – Example

Algorithm: Rule-based agent program with belief state

Input: 1. Initial belief state b_1
2. Condition-action rule base K

```
1 b ← b_1
2 while true do
3     for each (φ, a') ∈ K do
4         if b ⊨ φ then
5             a ← a'
6             EXECUTE(a)
7             break
8     p ← SENSE()
9     b ← UPDATE(b, a, p)
```

#	Condition ϕ	Action $a \in \mathcal{A}$
1	$\text{Agent}(x) \wedge \text{Dirt}(x)$	Clean
2	$\neg\text{Agent}(x) \wedge \text{Dirt}(x)$	$\text{Move}(x)$
3	$\top$	Null

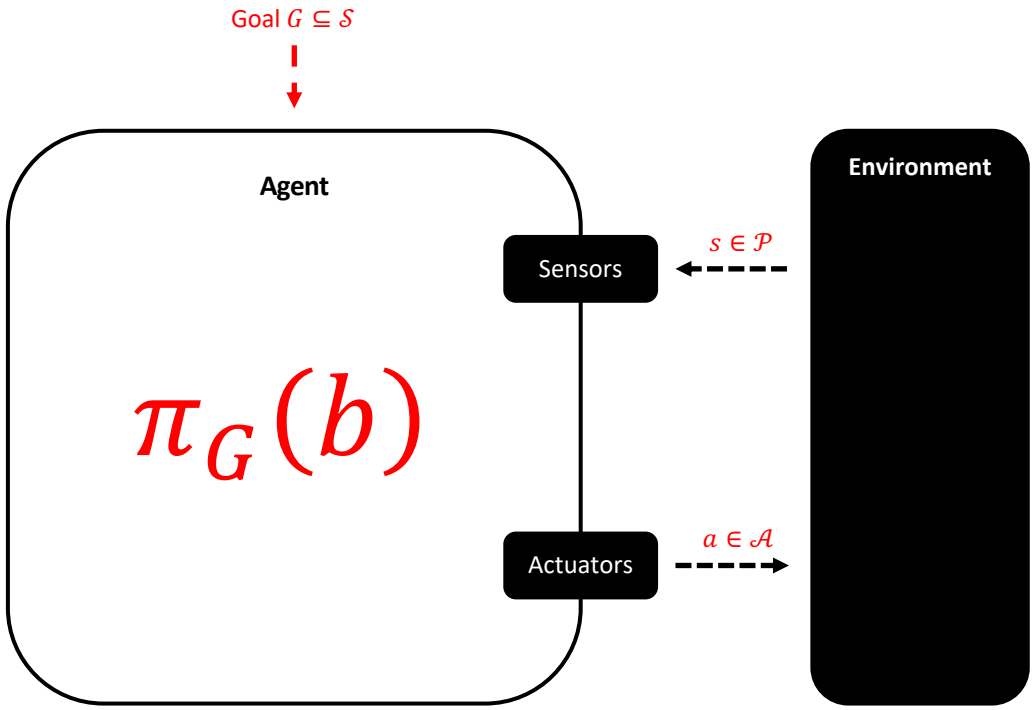
Condition-action rule

Agent function

Belief state $b_i \in B$	Belief-based agent function $\pi_B(b_i)$
$\{s_3, s_4\}$	Move(Right)
$\{s_5, s_6\}$	Move(Left)
$\{s_4, s_5, s_7, s_8\}$	Clean
...	...

Proactive Agents

Goal-Based



- A goal is a set of environment states $G \subseteq \mathcal{S}$
  - Goal is achieved when the agent arrives in some state $s \in G$
- Agent terminates at h if the goal is believed to be achieved in belief state associated with h
  - With a certain belief state, agent terminates when $b \in G$
  - With a logically uncertain belief state, agent terminates when $b \subseteq G$
  - With a probabilistic uncertain belief state, agent terminates when e.g. $\sum_{s \in G} b(s) = 1$ or $\sum_{s \in G} b(s) \geq \theta$ with $0 < \theta \leq 1$
- Agent function π_G assigns actions with the objective of achieving the goal at some point in the future
  - Agent continues to execute actions until termination

Proactive Agents

Goal-Based – Example (1)

Algorithm: Goal-based agent program

Input: Initial belief state $b_1 \in B$

1 $b \leftarrow b_1$
2 $\pi_B \leftarrow \textsc{Plan}(b, T, O, G)$
3 **while** $b \not\models G$ **do**
4 $\quad a \leftarrow \pi_B(b)$
5 $\quad \textsc{Execute}(a)$
6 $\quad p \leftarrow \textsc{Sense}()$
7 $\quad b \leftarrow \textsc{Update}(b, a, p)$

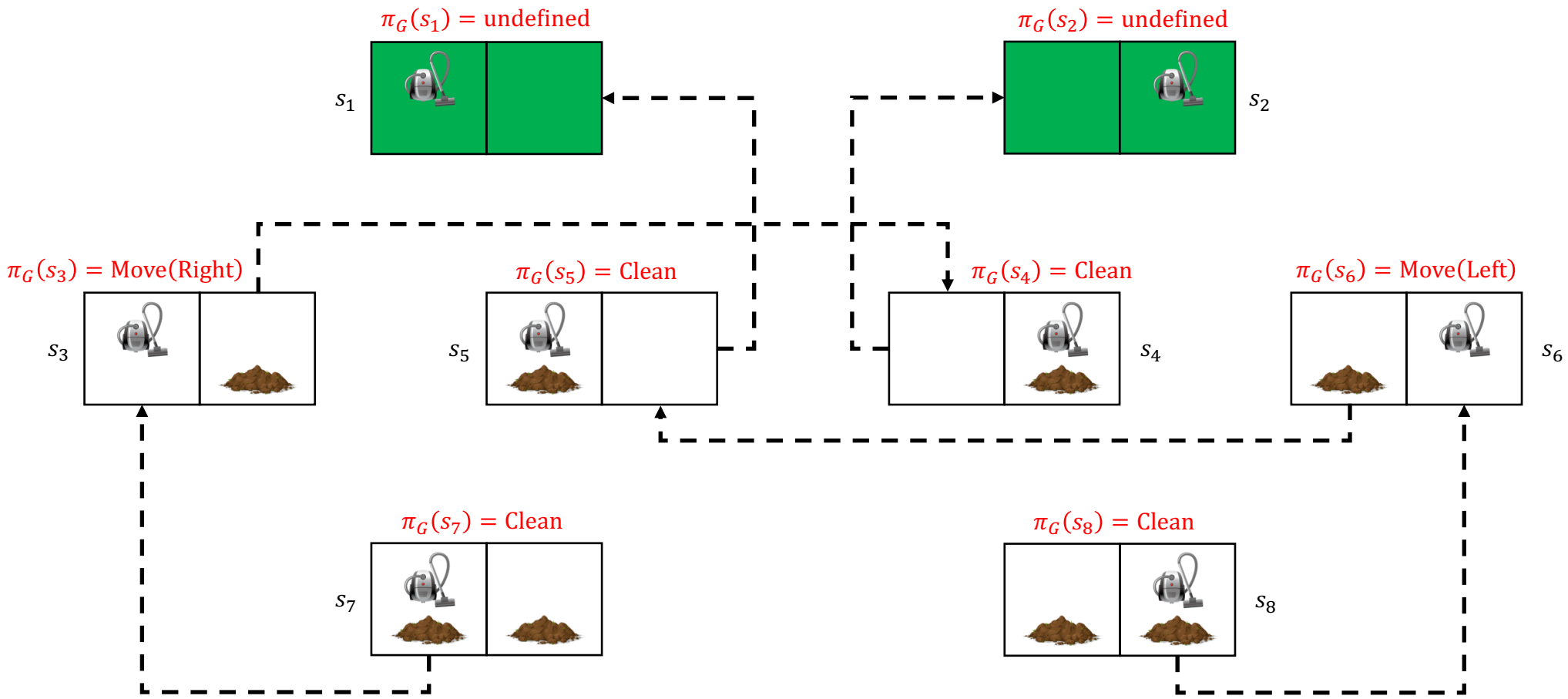
Generate plan for G (subject of next lecture)

Execute until goal is achieved

Proactive Agents

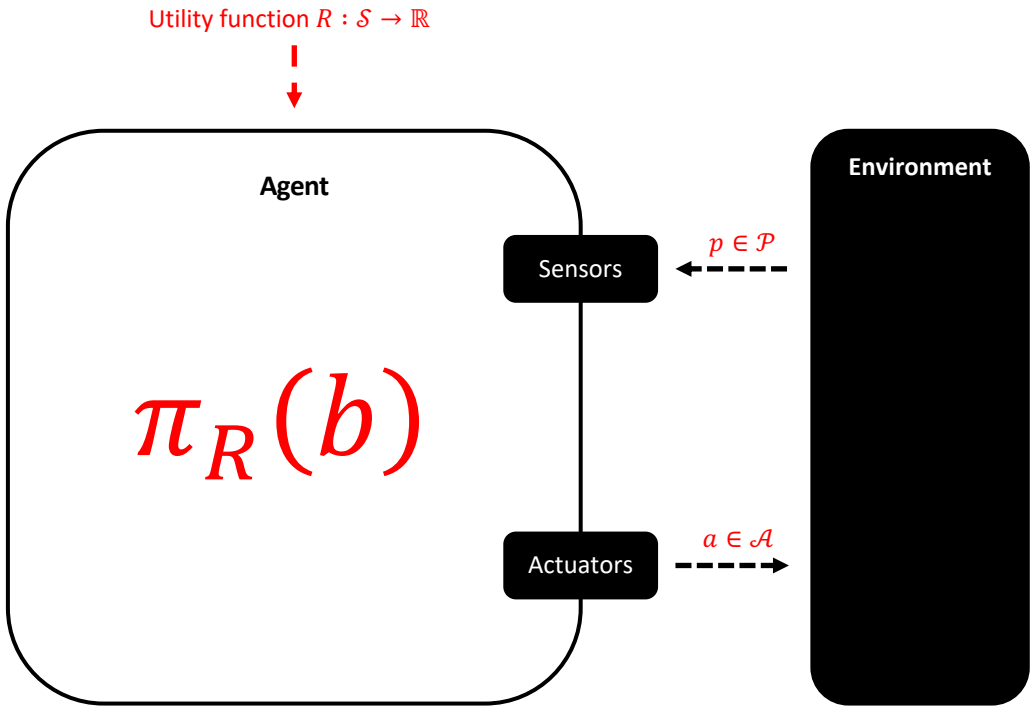
Goal-Based – Example (2)

Goal $G = \{s_1, s_2\}$



Proactive Agents

Utility-Based



- A utility (or reward) function is a function $R : \mathcal{S} \to \mathbb{R}$
  - An environment state $s \in \mathcal{S}$ preferred to state $s' \in \mathcal{S}$ if $R(s) > R(s')$
- Cumulative utility is utility that accumulates over the agent's life
  - For example, sum of utility obtained at all preceding decision-steps
- Agent function π_R assigns actions with the objective of optimising cumulative utility over the agent's life
  - Agent continues to execute π_U forever

Proactive Agents

Utility-Based – Example (1)

Algorithm: Utility-based agent program

Input: Initial belief state $b_1 \in B$

1 $b \leftarrow b_1$

2 $\pi_B \leftarrow \text{PLAN}(b, T, O, R)$

3 **while** *true* **do**

4 $\quad a \leftarrow \pi_B(b)$

5 $\quad \text{EXECUTE}(a)$

6 $\quad p \leftarrow \text{SENSE}()$

7 $\quad b \leftarrow \text{UPDATE}(b, a, p)$

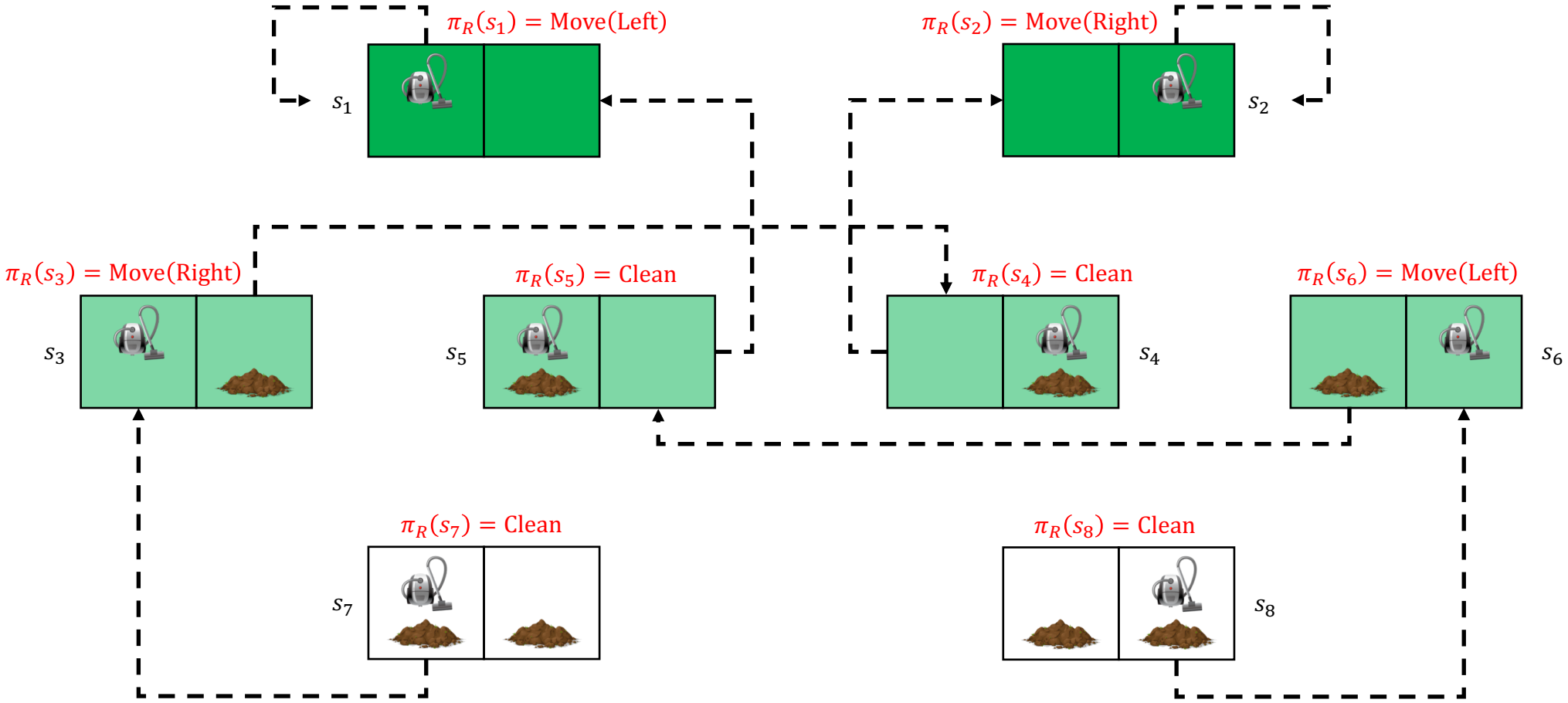
Generate plan for R (subject of next lecture)

Execute plan forever

Proactive Agents

Utility-Based – Example (2)

$$R(s_1) = R(s_2) = 2$$
$$R(s_3) = R(s_4) = R(s_5) = R(s_6) = 1$$
$$R(s_7) = R(s_8) = 0$$



Cognitive Agents

Motivation

- Need to balance reactive and proactive behavior
  - Pure reactive behavior is very limited with regards to intelligence and autonomy
  - Pure proactive behavior can be unnecessary and/or unrealistic (e.g. it is expensive to plan)
- Multi-tasking
  - Like humans, agents need to pursue multiple commitments simultaneously
- Goal formation
  - For true autonomy and intelligence, agents need to form their own goals through complex reasoning
- Behavior reconsideration
  - In dynamic and resource-bounded environments, agents need to revise their behavior in light of new information
  - For example, agents may need to drop goals that are no longer achievable, or may need to replan
- Human-agent interaction
  - Agents need to be able to explain their behavior in a way that is understandable to humans
- Many others…

Cognitive Agents

Intentional Systems

- Cognitive agents mirror theories of human practical reasoning (especially as proposed by Bratman)
  1. Deliberation: deciding what states of the world we want to achieve
  2. Means-end reasoning: deciding how to achieve those states of the world

- Practical reasoning is about figuring out what to do (i.e. reasoning directed towards action)
  - Different from theoretical reasoning (i.e. reasoning directed towards beliefs)
  - *"Practical reasoning is a matter of weighing conflicting considerations for and against competing options, where the relevant considerations are provided by what the agent desires/values/cares about and what the agent believes."* [Bratman]

- Output at each step is a set of intentions
  1. Deliberation: declarative intentions
  2. Means-end reasoning: procedural intentions

- Planning (from first-principles) is a form of means-end reasoning
  - So is hierarchical planning and plan selection

Cognitive Agents

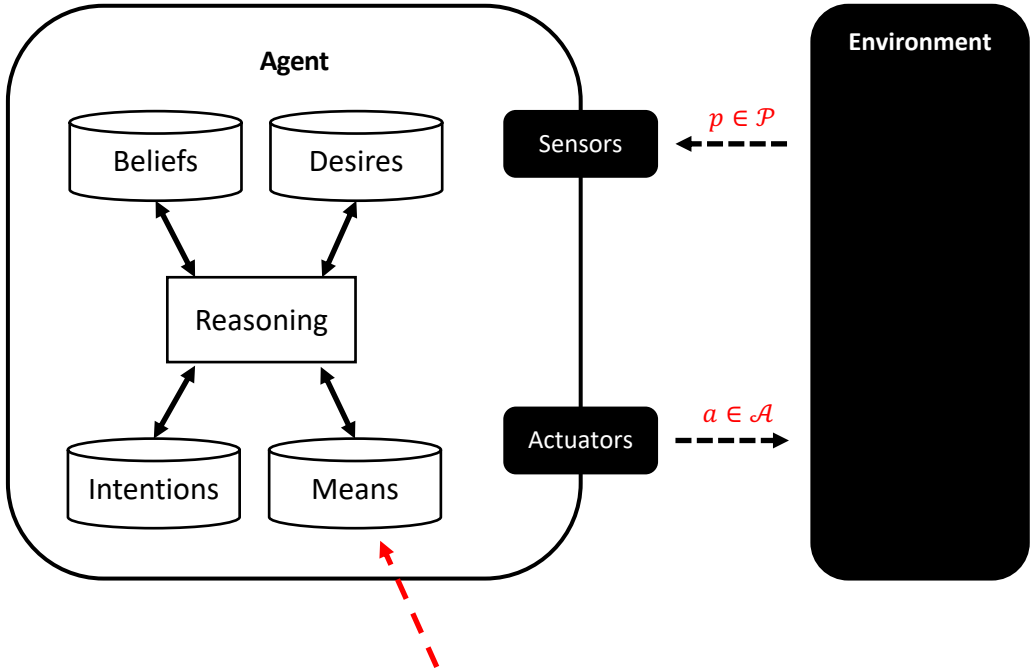
Intentions

Principle	Explanation
Intentions pose problems for agents, who need to determine ways of achieving them	If an agent has an intention φ, you would expect the agent to devote resources to deciding how to bring about φ
Intentions provide a filter for adopting other intentions, which must not conflict	If an agent has an intention φ, you would not expect the agent to adopt an intention ψ that was incompatible with φ
Agents track the success of their intentions, and are inclined to try again if their attempts fail	If an agent's first attempt to achieve φ fails, then all other things being equal, the agent will try an alternative means of achieving φ
Agents believe their intentions are possible	An agent should believe there is at least some way that their intentions could be achieved
Agents do not believe they will not bring about their intentions	It would not be rational for an agent to adopt an intention φ if the agent believed it would fail with φ
Under certain circumstances, agents believe they will bring about their intentions	If the agent intends φ, then the agent believes that under "normal circumstances" it will succeed with φ
Agents need not intend all the expected side effects of their intentions	If the agent believes $\varphi \Rightarrow \psi$ and intends φ, the agent does not necessarily intend ψ also

"My desire to play basketball this afternoon is merely a potential influencer of my conduct this afternoon. It must vie with my other relevant desires […] before it is settled what I will do. In contrast, once I intend to play basketball this afternoon, the matter is settled: I normally need not continue to weigh the pros and cons. When the afternoon arrives, I will normally just proceed to execute my intentions." [Bratman]

Cognitive Agents

Belief-Desire-Intention (BDI)



- Beliefs are the same as belief states
  - Sometimes we use the term belief base instead (but a belief base is usually represented in some logical language)
- Desires are states of the world that the agent wants to bring about
  - Desires are like goals but do not imply commitment
  - Desires may be mutually inconsistent (i.e. it need not be the case that all desires can be achieved simultaneously)
- Intentions are desires that the agent has committed to achieving (and possibly the means to do so)
  - See declarative vs. procedural intentions
  - Declarative intentions are like goals in that they imply commitment
  - Intentions must be mutually consistent and must be consistent with beliefs

Cognitive Agents

Belief-Desire-Intention (BDI) – Example

Algorithm: BDI agent program [Wooldridge, 2009]

Input: 1. Initial belief state $b_1 \in B$
2. Initial intention set I_1

1 $b \leftarrow b_1$
2 $I \leftarrow I_1$
3 **while** *true* **do**
4 $D \leftarrow \text{OPTIONS}(b, I)$
5 $I \leftarrow \text{FILTER}(b, D, I)$
6 $\pi_B \leftarrow \text{PLAN}(b, I)$
7 **while** $\neg\text{SUCCEEDED}(b, I) \wedge \neg\text{IMPOSSIBLE}(b, I)$ **do**
8 $a \leftarrow \pi_B(b)$
9 $\text{EXECUTE}(a)$
10 $p \leftarrow \text{SENSE}()$
11 $b \leftarrow \text{UPDATE}(b, a, p)$
12 **if** $\text{RECONSIDER}(b, I)$ **then**
13 $D \leftarrow \text{OPTIONS}(b, I)$
14 $I \leftarrow \text{FILTER}(b, D, I)$
15 **if** $\neg\text{SOUND}(\pi_B, b, I)$ **then**
16 $\pi_B \leftarrow \text{PLAN}(b, I)$

Deliberation

Generate plan for *I*

Means-end reasoning

While intentions are believed to be achievable

While intentions are not believed to have been achieved

Decide whether to revise intentions

If (possibly revised) intentions cannot be achieved with current plan

Replan

Intelligent Agents

Some Reflections

- Agent functions are mathematically identical to plans (or policies) in automated planning
  - But an agent function is much more than a plan…
- Properties of the task environment have implications for agent design
  - These properties typically refer to designer-imposed restrictions on the model (i.e. approximations) rather than inherent properties
- Intelligent agents should balance reactive and proactive behavior
  - Proactive behavior can be slow/expensive and is not always necessary
  - Humans behave both proactively and reactively
- Intelligent agents should go beyond simple goal- or utility-based agents
  - Agents need to formulate goals, revise commitments, etc.
  - BDI is a just one approach, but an influential one (e.g. see BDI theory, BDI logics, BDI programming languages)

This Lecture…

1. Abstract agent representations
   - Agent function
   - Agent program
2. Task environments
   - Problem Analysis
   - Properties
3. Types of agents
   - Reactive agents
   - Proactive agents
   - Cognitive agents

Next Lecture…

1. Classical planning
   - Planning languages
2. Planning under uncertainty
   - Conformant planning
   - Contingent planning
   - Markov decision processes
3. Online planning
   - Classical replanning

Bibliography

- Michael Wooldridge. *An Introduction to Multiagent Systems*. John Wiley & Sons, 2nd edition, 2009.

Suggested reading (Chapter 2)

- Stuart J. Russell & Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 3rd edition, 2009.

- Hector Geffner & Blai Bonet. *A Concise Introduction to Models and Methods for Automated Planning*. Morgan & Claypool, 2013.

- Anand S. Rao. AgentSpeak(L): BDI agents speak out in a logical computable language. In *Proceedings of the 7th European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW'96)*, pages 42-55, 1996.

- Rafael H. Bordini, Jomi Fred Hübner, & Michael Wooldridge. *Programming Multi-Agent Systems in AgentSpeak Using Jason*. John Wiley & Sons, 2007.