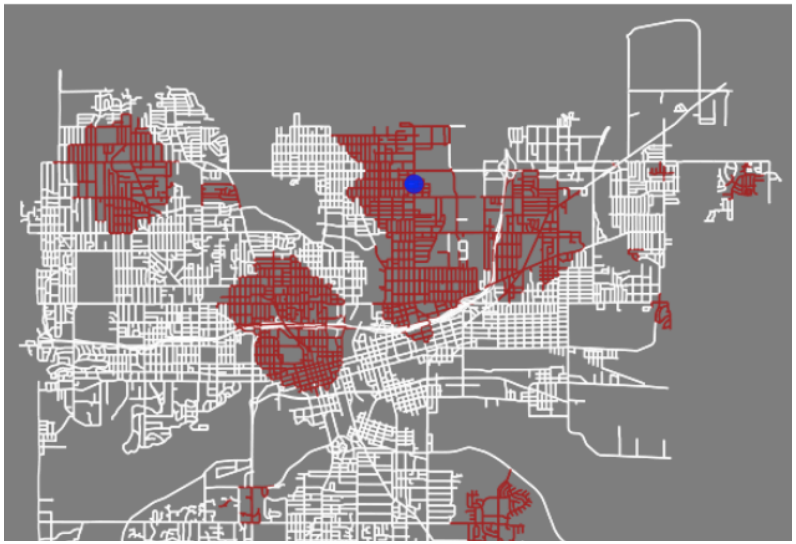
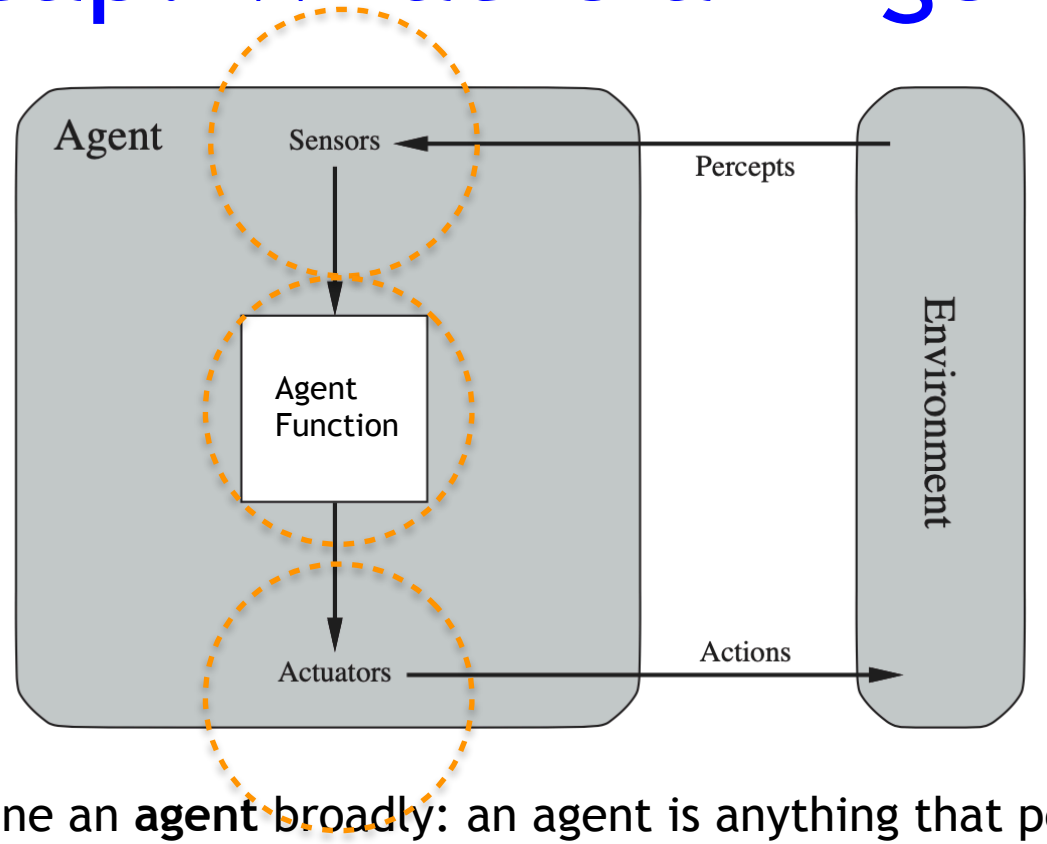
CS143: Artificial Intelligence



Structure of Intelligent Agent
Implementing a Simple Agent on a 2D map

Drake UNIVERSITY

Recap: What is an Agent?



- We will define an **agent** broadly: an agent is anything that perceives its environment via **sensors** and acts upon the environment with **actuators**.
- **Sensors** receive percepts (inputs), while **actuators** yield actions (outputs)
- This characterization suggests that our agent is a robot (as it uses the terms sensors and actuators), but it is much broader than that.

Recap: Example Agents

- Example 1: Humans
  - Sensors: Eyes, ears, other sensory organs
  - Actuators: Hands, legs, vocal chords, etc
- Example 2: Robots
  - Sensors: Camera, distances sensors, microphones
  - Actuators: mechanical arms, motors, lights
- Example 3: Software agents
  - Sensors: Receiving keystrokes, file contents, network packets
  - Actuators: Displaying on the screen, writing files, sending network packets

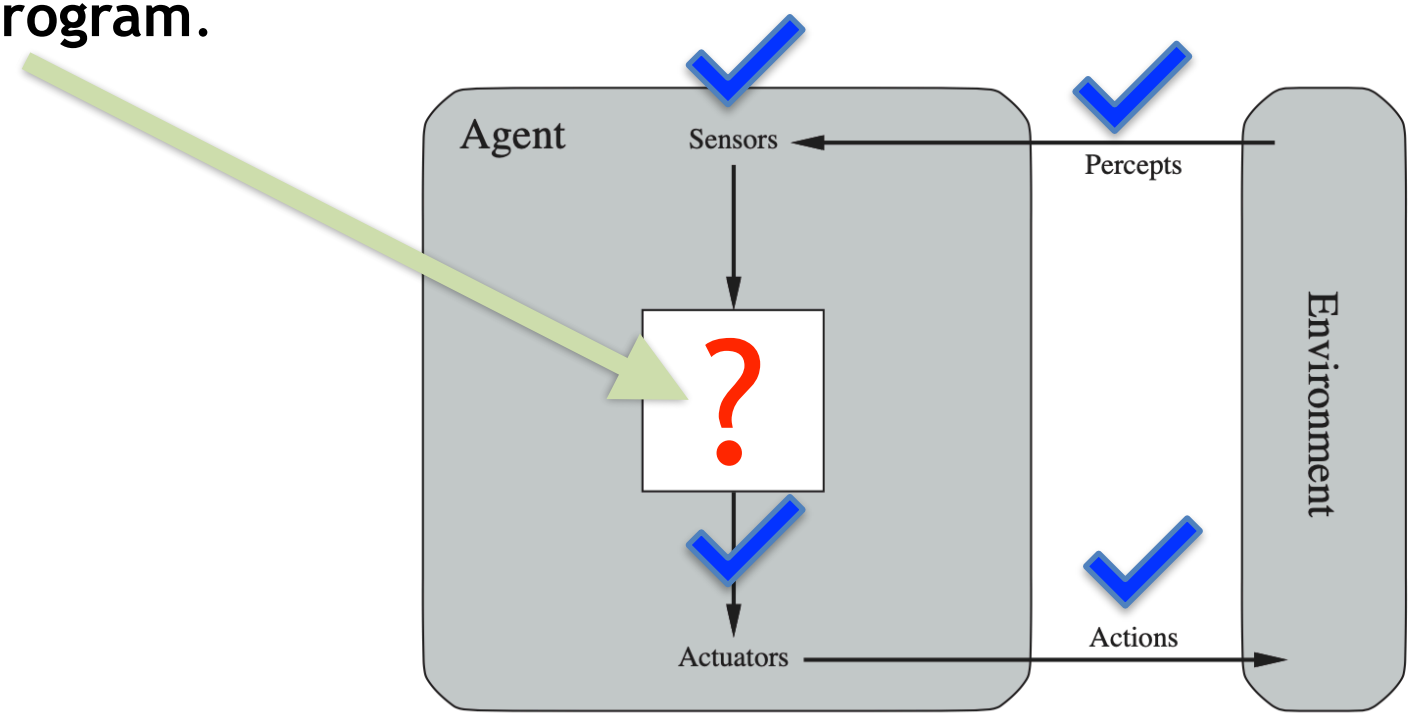
Recap: Percepts

- A **percept** of an agent refers to its perceptual inputs at any given instant.
- If we take the entire history of percepts of an agent up to the present moment, this gives us the agent's **percept sequence**.
- In general, an agent's choice of action may depend on its current percept sequence, but this should not depend on anything the agent has not perceived.

Recap: Agent function

- An agent's behavior is described by an **agent function**, which is a function that maps every possible percept sequence to an action.

- we can *internally* characterize an agent by means of a program that implements the **agent function**, which we call an **agent program**.



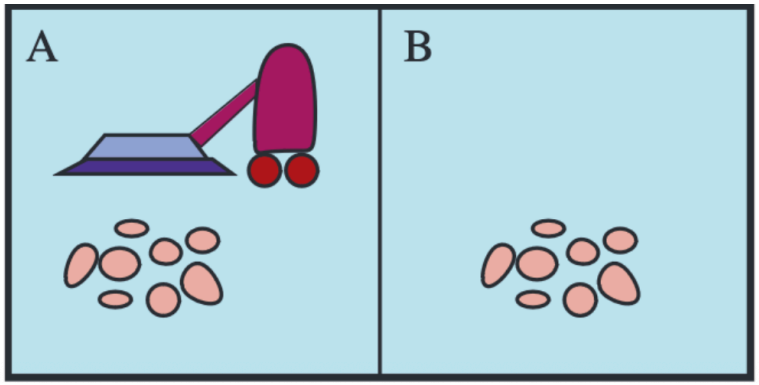
Recap: what is a rational agent?

- An **agent** is an entity that perceives and acts.

- Our course is about designing a **rational agent**; that is, we seek to design an agent that can make the **best decision** under a given set of resources/ circumstances.
    - Let's first a define agent
    - Then define a rational agent

Recap: performance measure for a vacuum-cleaner agent example

- Fixed **performance measure** evaluates the environment's sequence of states. If the sequence is desirable, the agent has performed well:
  - **Measure#1:** Measure performance by the amount of dirt cleaned in an eight hour shift.
    - One point per square cleaned up in time T?
  - **Measure#2:** Measure performance by rewarding the agent for having a clean floor
    - One point per clean square per time step, minus one per move?



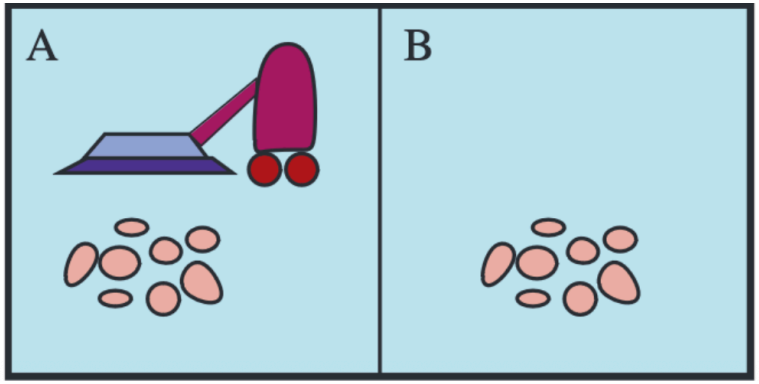
Recap: precise definition of rationality

- We can now define a rational agent as follows:
  - A rational agent is one that selects an action that maximizes its expected (or average) performance measure given the percept sequence to date.

Recap: rational vacuum agent

- Is the vacuum agent defined by the following agent function rational?

```
function REFLEX-VACUUM-AGENT([location,status]) returns an action
    if status = Dirty then return Suck
    else if location = A then return Right
    else if location = B then return Left
```



Recap: what is task environment?

- Task environments can be thought of as the *problems* to which the rational agents are the *solutions*.

- We define a task environment by specifying four factors:
  - **P**erformance Measure
  - **E**nvironment
  - **A**ctuators
  - **S**ensors

- We call this the PEAS description of a task environment

Recap: PEAS example for automated taxi driver

Agent Type	Performance Measure	Environment	Actuators	Sensors
Taxi driver	Safe, fast, legal, comfortable trip, maximize profits	Roads, other traffic, pedestrians, customers	Steering, accelerator, brake, signal, horn, display	Cameras, sonar, speedometer, GPS, odometer, accelerometer, engine sensors, keyboard

Recap: PEAS more examples

Agent Type	Performance Measure	Environment	Actuators	Sensors
Medical diagnosis system	Healthy patient, reduced costs	Patient, hospital, staff	Display of questions, tests, diagnoses, treatments, referrals	Keyboard entry of symptoms, findings, patient's answers
Satellite image analysis system	Correct image categorization	Downlink from orbiting satellite	Display of scene categorization	Color pixel arrays
Part-picking robot	Percentage of parts in correct bins	Conveyor belt with parts; bins	Jointed arm and hand	Camera, joint angle sensors
Refinery controller	Purity, yield, safety	Refinery, operators	Valves, pumps, heaters, displays	Temperature, pressure, chemical sensors
Interactive English tutor	Student's score on test	Set of students, testing agency	Display of exercises, suggestions, corrections	Keyboard entry

Recap: classifying task environments

- A wide range of task environments might arise in AI.
- We can categorize task environments along several dimensions, which are conducive to appropriately designing various agents.
- We will consider six such general categories.

Task Environment	Observable	Agents	Deterministic	Episodic	Static	Discrete
Crossword puzzle	Fully	Single	Deterministic	Sequential	Static	Discrete
Chess with a clock	Fully	Multi	Deterministic	Sequential	Semi	Discrete
Poker	Partially	Multi	Stochastic	Sequential	Static	Discrete
Taxi driving	Partially	Multi	Stochastic	Sequential	Dynamic	Continuous

Agenda

- Small assignment#1 (write-up) is due by tonight
- Discussion of structure of an intelligent agent
- Exploring a practical implementation of a simple agent on a 2D map in Python

Structure of intelligent agents

Structure of Agents

- An agent's behavior is described by an **agent function**, which is a function that maps every possible percept sequence to an action.

- we can *internally* characterize an agent by means of a program that implements the **agent function**, which we call an **agent program**.

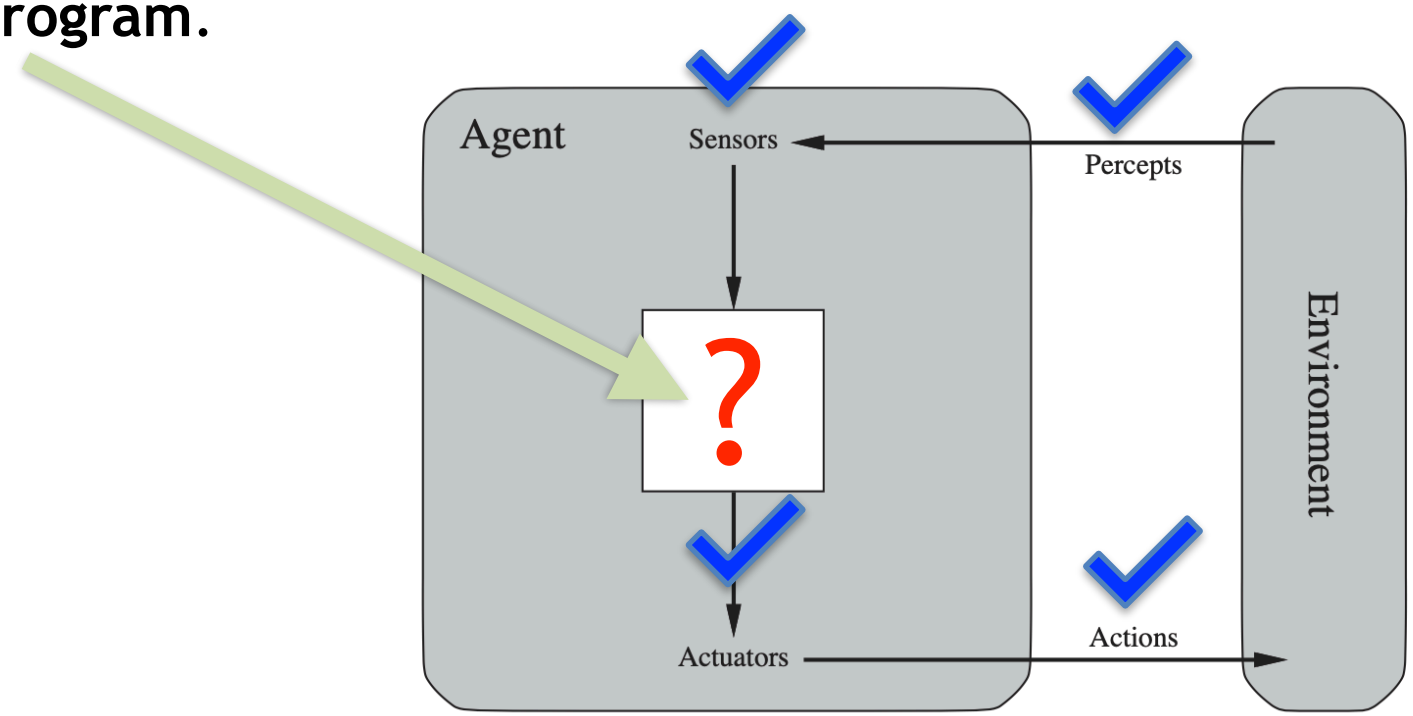


Table-driven Agent

- The following agent does what we want:

```
function TABLE-DRIVEN-AGENT(percept) returns an action
  persistent: percepts, a sequence, initially empty
              table, a table of actions, indexed by percept sequences, initially fully specified

  append percept to the end of percepts
  action ← LOOKUP(percepts, table)
  return action
```

- The problem is that lookup tables can be utterly massive.
  - A full lookup table for chess should have at least 10^{150} entries

Table-driven Agent

Percept sequence	Action
$[A, Clean]$	$Right$
$[A, Dirty]$	$Suck$
$[B, Clean]$	$Left$
$[B, Dirty]$	$Suck$
$[A, Clean], [A, Clean]$	$Right$
$[A, Clean], [A, Dirty]$	$Suck$
⋮	⋮
$[A, Clean], [A, Clean], [A, Clean]$	$Right$
$[A, Clean], [A, Clean], [A, Dirty]$	$Suck$
⋮	⋮

- The problem is that lookup tables can be utterly massive
  - A full lookup table for chess should have at least 10^{150} entries
- Our goal is to design agents that behave rationally using programs that are of a manageable size rather than from a table.
  - we formerly used massive tables to calculate square roots, but now calculators can use simple and accurate approximations of the square root function in a few lines of code.

Agent Types

- There are four main types of agent that embody the behavior of almost all intelligent behavior:
  - Simple reflex agents
  - Model-based reflex agents
  - Goal-based agents
  - Utility-based agents
- Each kind of agent gives rise to actions via different ways of combining different components.

Simple reflex agent

Simple Reflex Agent

- Simple reflex agents select actions based on the current percept.
- In particular, they ignore any percept history.
- We've already seen an example of a simple reflex agent:

```
function REFLEX-VACUUM-AGENT( [location,status]) returns an action

    if status = Dirty then return Suck
    else if location = A then return Right
    else if location = B then return Left
```

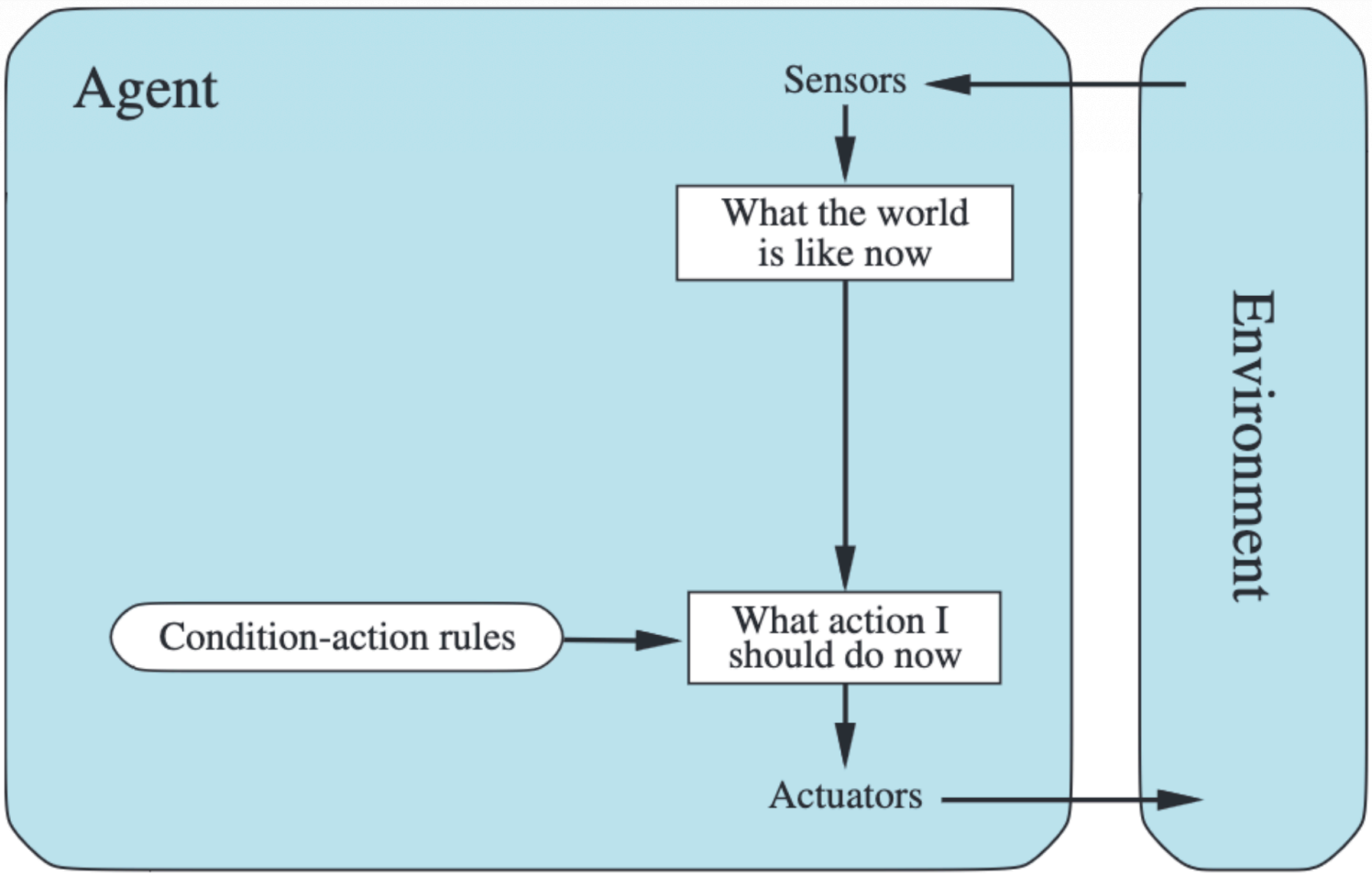
Condition-Action Rule

- An established connection between a percept given by a sensor and an actuator is called a **condition-action rule**.
  - Eg: When I see a brake light, I hit the brakes.

    ```
    if car-in-front-is-braking then initiate-braking
    ```

  - Eg: If the vacuum is in a dirty square, the vacuum sucks up the dirt.
- A condition-action statement is essentially an if-then statement.

Simple Reflex Agent



Advantages vs. Disadvantages

- **Advantages**
  - Simple, easy to implement
  - Limited memory needed

- **Disadvantages**
  - Works only if the correct decision can be made on the current percepts
  - Requiring the environment to be fully observable; any uncertainty (e.g. partially observable) can causes issues

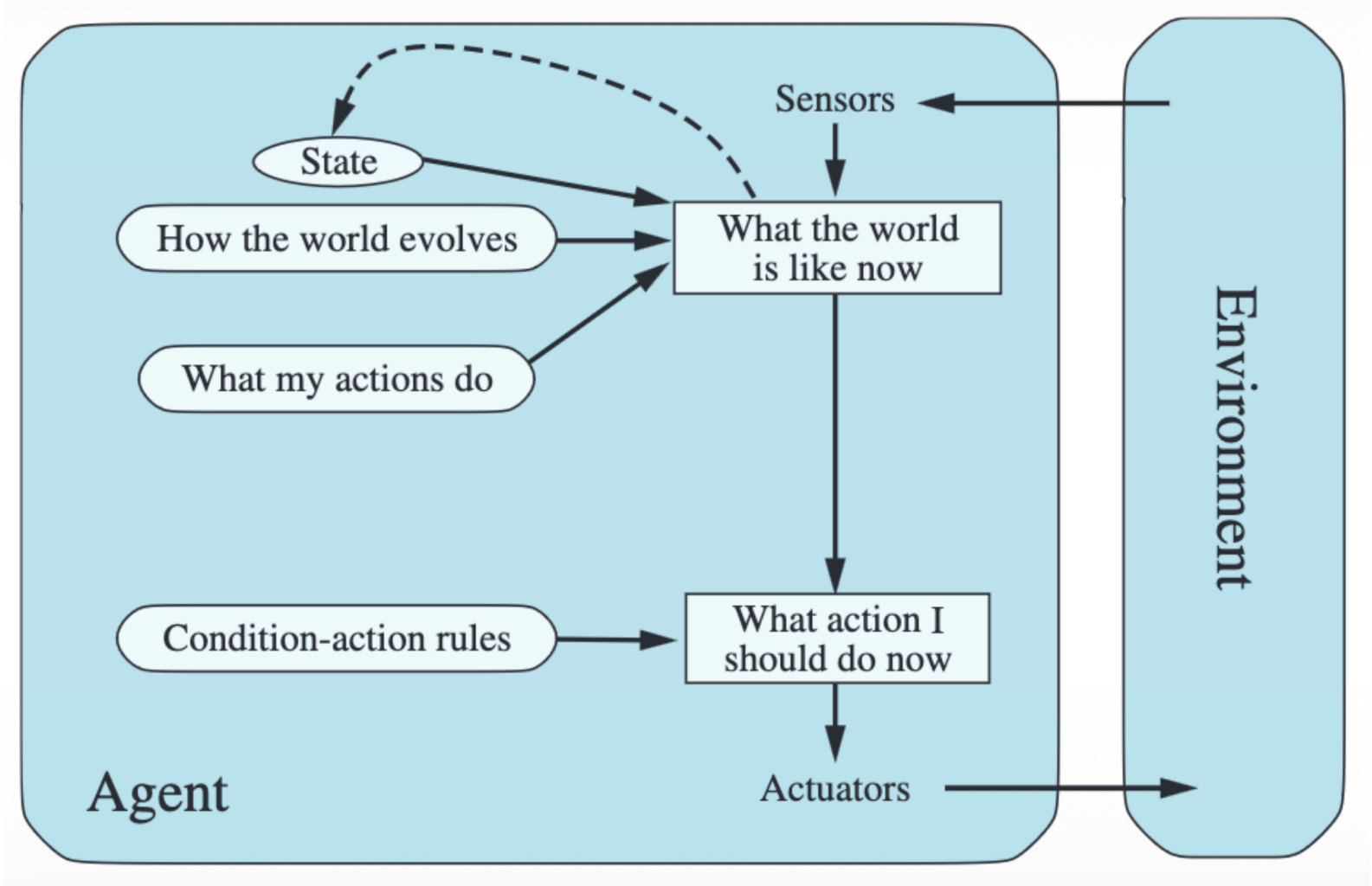
    Eg, when the location sensor is missing for the vacuum agent?

Model-based Reflex Agent

Model-based Reflex Agent

- The most effective way to handle partial observability is for the agent to keep track of the part of the world it can't see.
- That is, the agent should use internal states that depend on the percept history to build a model of the environment.
- The agent needs to incorporate two kinds of information:
  - How the world evolves (called a model of the world).
  - How the agent's actions affect the world.

Model-based Reflex Agent



Example

last_A: time since *A* was last cleaned

```
function MODEL-BASED-REFLEX-VACUUM-AGENT( [location,status])
returns an action
static: last_A, last_B, numbers, initially ∞

    last_A++, last_B++
    if location = A and last_A > 3 and status = Dirty
        then  last_A  ← 0; return Suck
    if location = A and last_B > 3 then return Right
    ...
```

Advantages vs. Disadvantages

- **Advantage**
  - Works better than simple reflex agents with partially observable environments

- **Disadvantages**
  - Requires memory
  - Requires a model to be developed

Goal-based reflex agent

Goal-based Agents

- **Setting a goal:** knowing something about the current environment may not be enough to decide what to do.
  - Example: Simple vacuum agent cleans cells ‘A’ and ‘B’. Should it stop or keep moving?
  - Example: A taxi reaches a fork in the road: Should it turn left or right?
- Implementing a **goal** may help an agent make a decision.

Goal-based Agents

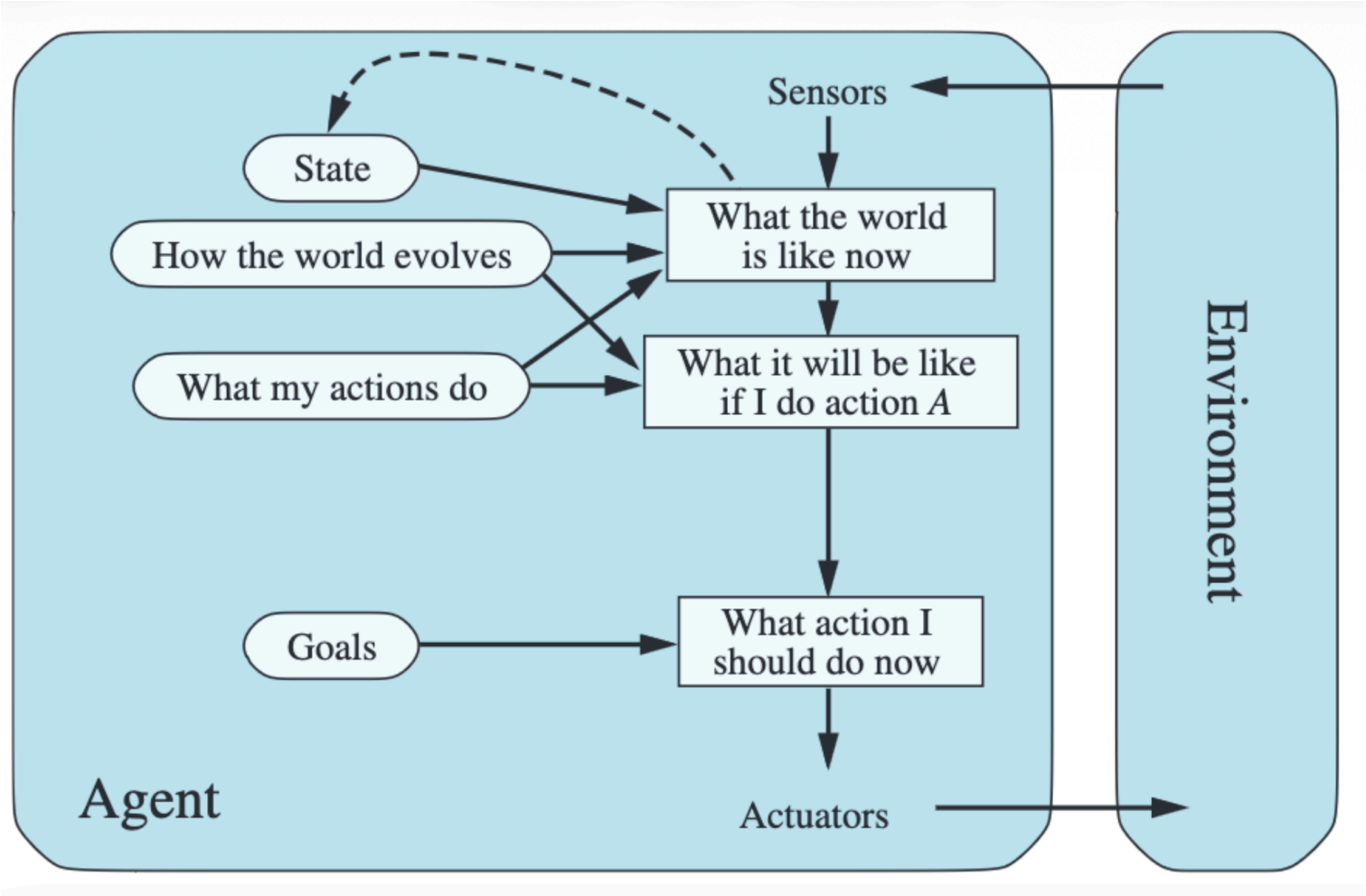
- Goal-based agents are different than simple reflexive agents.

- Goal-based agents involve the consideration of future states.
  - Will move X allow me to achieve the goal *in the future*? In reflex agents, built-in rules determine actions, while in goal-based agents, the defined goal determines actions.
  - Example: Taxi braking because it sees a brake light (reflex) vs. taxi braking because it doesn't want to hit car (goal of safety).

Goal-based Agents

- We can combine both the goal-based and model-based approaches.
- A goal-based agent keeps track of world state as well as the set of goals it is trying to achieve.
- The agent then chooses action that will lead to the goal.

Goal-based Agents



Advantages vs. Disadvantages

- **Advantages**

  Flexible: If environment changes, the goal remains same

  Example: If it rains, the agent can update its knowledge of how effective its brakes will be, while for a reflexive agent we may have to rewrite all of its condition-action rules.

- **Disadvantages**
  - Less efficient, since it requires the processing of goals and possible outcomes to make a decision.
  - It is not simply following condition-action rules, which are much simpler to execute.

Utility-based agent

Utility-based Agents

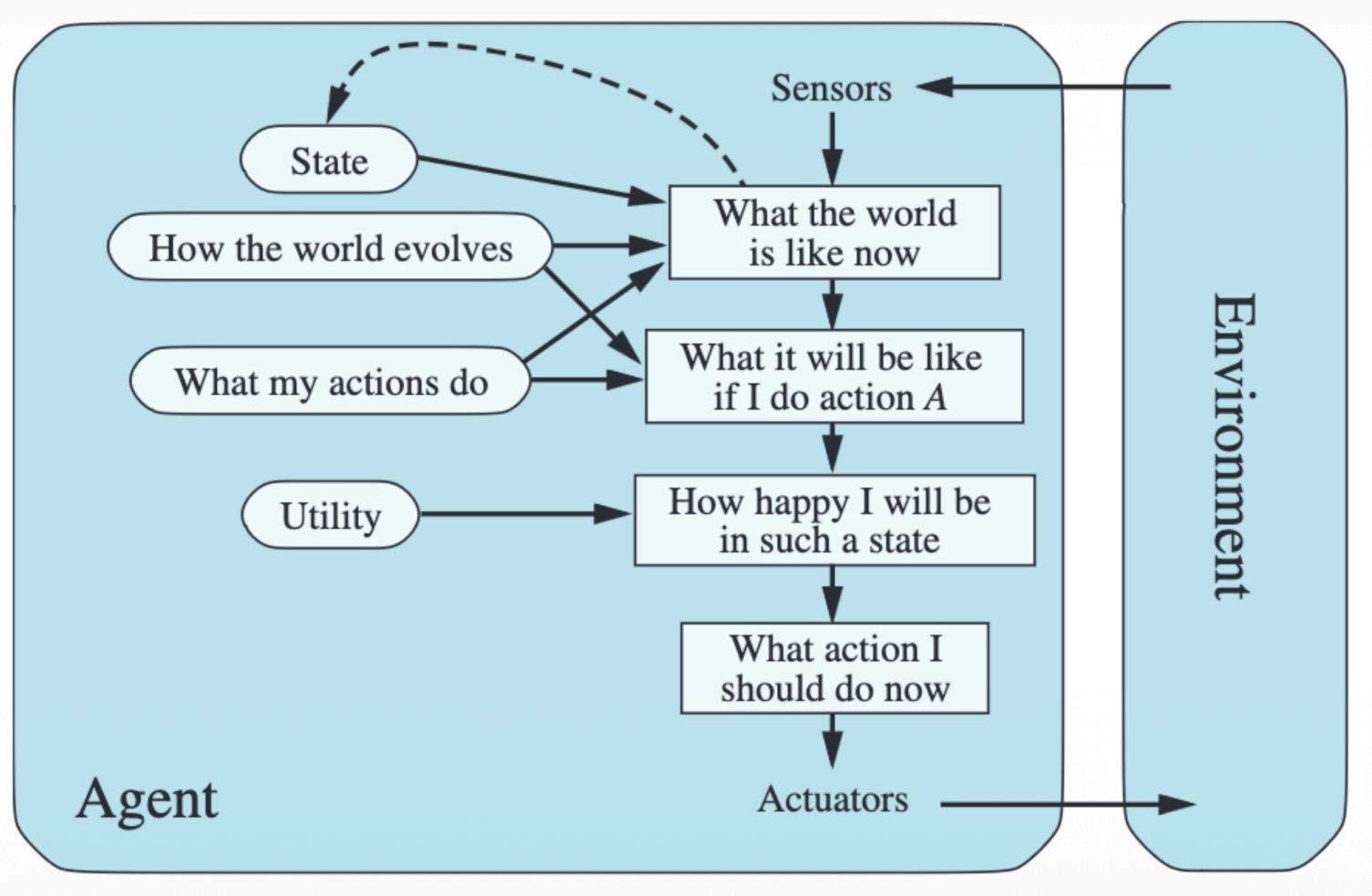
- Goals alone are not enough to generate high-quality behavior in most environments.
- Goals only distinguish between “happy” and “unhappy” states.
- We want to take into account a more refined range of happiness or **utility**.

Utility-based Agents

- A utility function is a agent's internalization of a performance measure.
  - Think of the utility function as quantitative measure of the happiness of the agent.

- An agent's goal is to maximize the expected utility of the action outcomes.

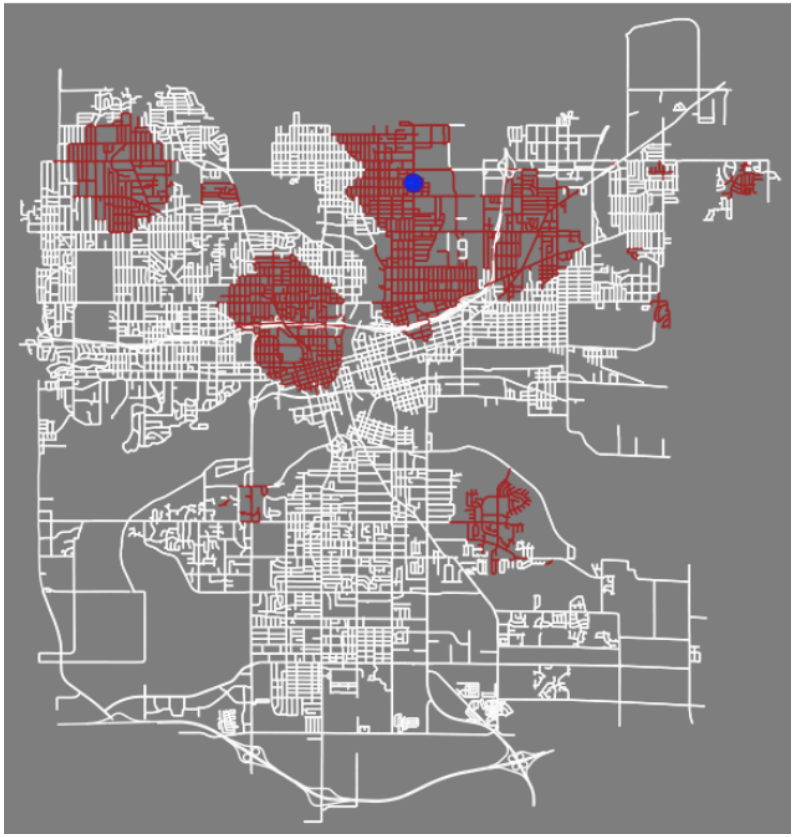
Utility-based Agents



Implementation: a simple vacuum cleaning agent on a 2D map

Simple street-cleaning agent

- The robot agent moves across the 2D map shown above. It starts at a random location on the map (indicated by a blue circle)



- Download the Python notebook for the dirt-cleaning agent