CS480 – Artificial Intelligence Fall 2015

Topic: Intelligent Agents
Chapter: 2
Date: 8/26

Mustafa Bilgic

http://www.cs.iit.edu/~mbilgic

CS480 – Artificial Intelligence – Please do not distribute.

Outline

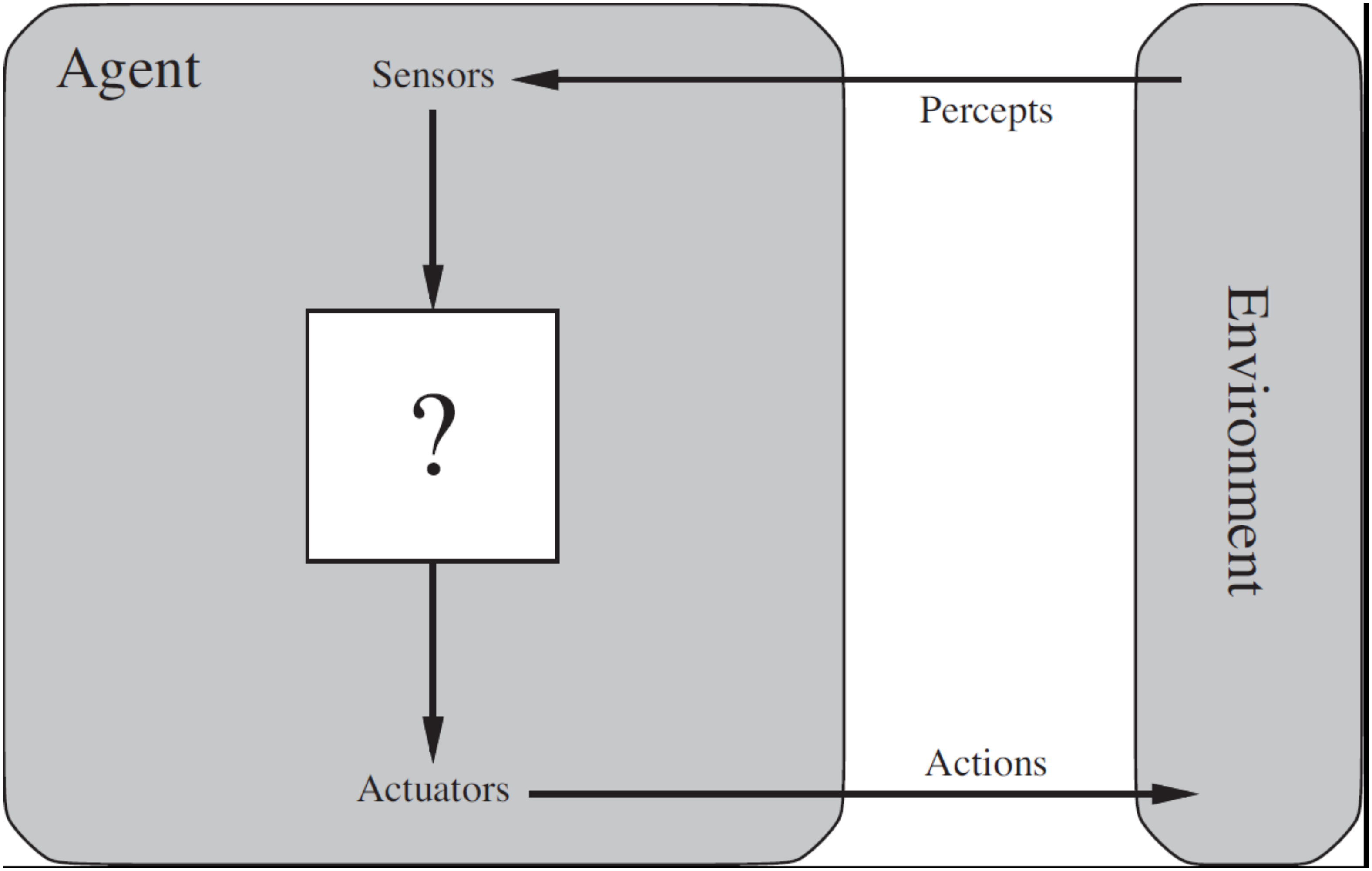
- Definition of an agent
- The concept of rationality
- Characteristics of the task environments
- Agent programs

Agent

An **agent** is anything that can be viewed as perceiving its **environment** thorough **sensors** and acting upon that environment through **actuators**.

Agent

What are the sensors and actuators for humans? For robots?

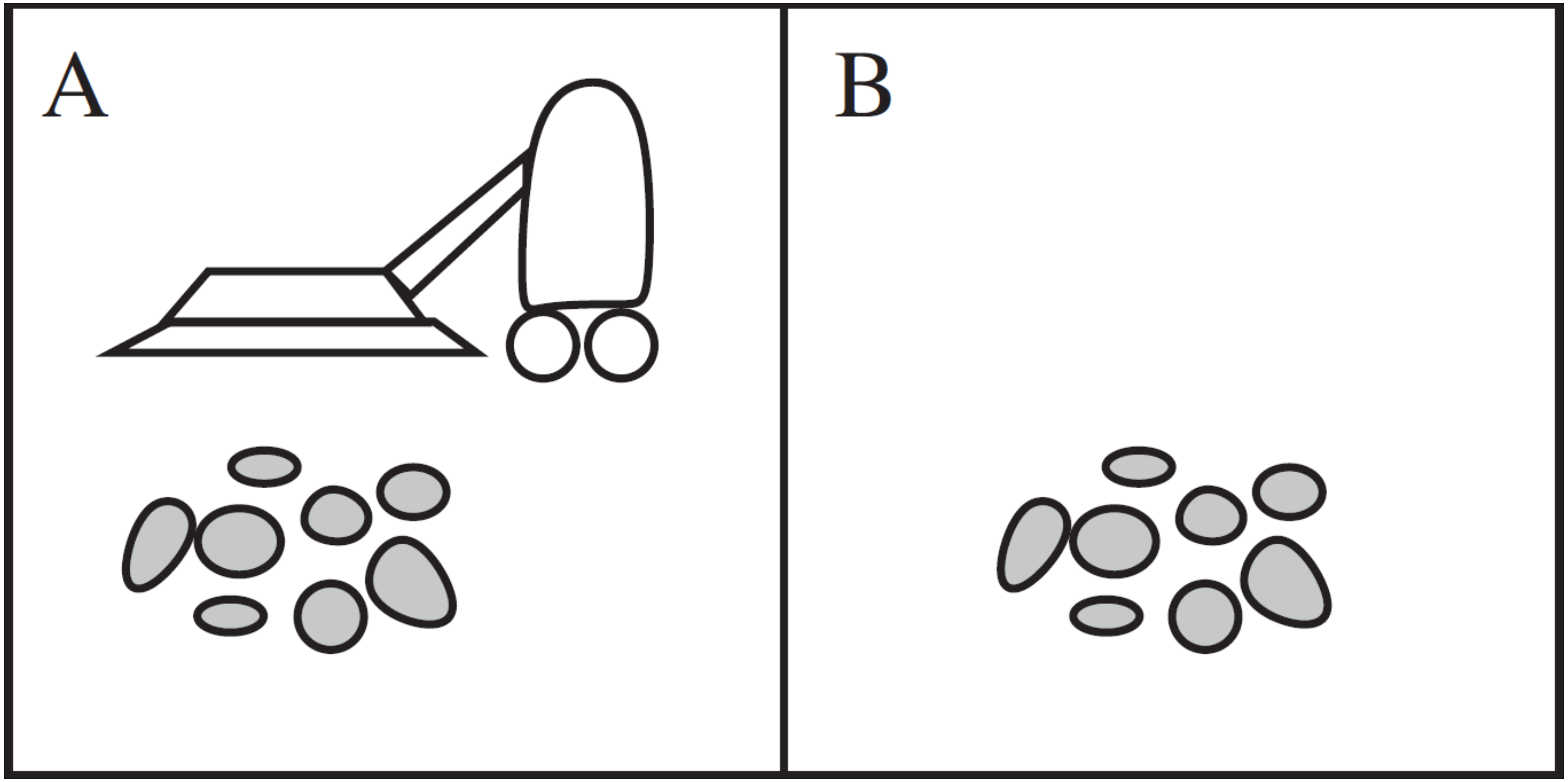


Perceive and Act

- **Percept:** An agent's perceptual inputs at any given instant
- **Percept sequence**: The complete history of everything the agent has perceived
- **Agent function**: *perceptSequence* → *action*
- **Agent program**: The implementation of the agent function

The agent function is a mathematical abstraction and the agent program is a concrete implementation of it

The Vacuum Environment



A Possible Agent function for VE

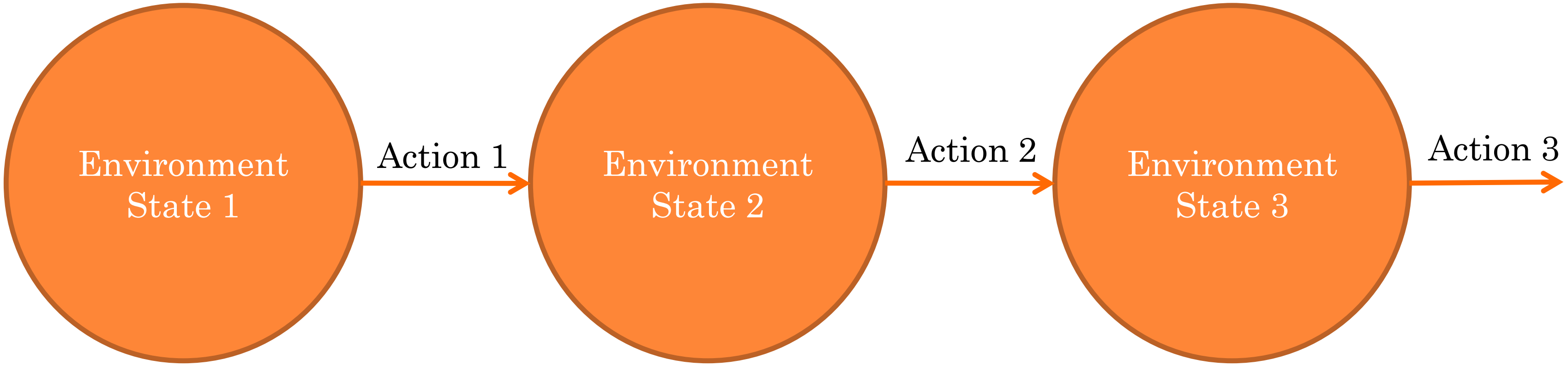
Percept Sequence	Action
[A, Clean]	Right
[A, Dirty]	Suck
[B, Clean]	Left
[B, Dirty]	Suck
[A, Clean], [A, Clean]	Right
[A, Clean], [A, Dirty]	Suck
[A, Clean], [B, Clean]	Left
…	
[A, Clean], [A, Clean], [A, Clean]	Right
…	

What is the size of this table?

What is the **right** action for a given percept sequence?

The Concept of Rationality

- **Rational agent:** the agent that does the *right* thing; i.e., the right hand side of the function table is filled *correctly*.
- How do we define the *right* thing?



Performance Measure

- **Performance measure** evaluates the sequence of *environment states*
  - Note the emphasis on environment states, not agent states
  - If we defined performance measure in terms of agent states, the agent can delude itself
- Example performance measures for the VE?
  - Amount of dirt the agent sucked
  - Number of clean cells per unit time

$\Rightarrow$ Performance measures should be designed according to what is desired in the environment rather than according to how one thinks the agent should behave

CS480 – Artificial Intelligence – Please do not distribute.

Coming Back to Rationality

- Is the simple agent that acts based on the agent function "if dirty, suck; if clean, move to the other square" rational?
- It depends!
- Rationality depends on
  1. The performance measure
  2. Agent's prior knowledge of the environment
  3. Actions it can perform
  4. Agent's percept sequence to date

Definition of a Rational Agent

- For each possible percept sequence, a **rational agent** should select an action that is
  - expected to maximize its performance measure, given
    - the evidence provided by the percept sequence and
    - whatever built-in knowledge the agent has.

Simple Agent is Rational if

1. The performance measure awards a point for each clean square at each time step
2. The geography of the environment is known but the dirt distribution and the initial location is unknown
3. Clean squares stay clean and sucking cleans the current square
4. The only available actions are Left, Right, and Suck
5. The agent correctly perceives its location and its dirt status

What if

- We also allow the action “Stop.” Is the simple agent (“if dirty, suck; if clean, move to the other square”) still rational?
- What if, in addition, we change the performance measure as “reward for clean squares; penalize for electricity consumption” ?

Rationality and perfection

- The rationality is not the same as perfection
- Perfection maximizes *actual* performance
- We cannot predict everything that'll happen in the future, and thus cannot maximize actual performance
- Rationality maximizes *expected* performance
  - Given what it knows, what information it can gather, what it can learn, what actions it can perform, and what the performance measure is, a rational agent is the one that can maximize expected performance

Autonomy and Learning

- Gathering information is not just enough; the agents should also learn from past experience
- For e.g., if we look at a more realistic scenario
  - A clean square can become dirty again
  - The dirt distribution is not even; some squares become dirty more often than others

The Task Environments

- **PEAS**
  - **P**erformance
  - **E**nvironment
  - **A**ctuators
  - **S**ensors

CS480 – Artificial Intelligence – Please do not distribute.

Automated Taxi

- Performance
  - Safe, fast, legal, comfortable, maximum profits
  - Side note: It may not be (and often is not) possible to maximize all performance measures; trade-offs have to be defined.
- Environment
  - Roads, other traffic, pedestrians, customers, weather
- Actuators
  - Steering, accelerator, brake, signal, horn, display
- Sensors
  - Cameras, GPS, speedometer, engine sensors, keyboard, microphone

More examples in Figure 2.5 in the book

Environment Properties

- **Fully observable vs. partially observable:**
  - If the sensors detect all aspects that are *relevant* to the choice of action
  - Fully observable $\Rightarrow$ No need to keep track of an internal representation of the world
  - Partially observable environments can be due to noise, inaccurate sensors, or the information is simply not available
    - Chess: FO, Poker: PO
    - Vacuum environment where the cleaner has a local sensor?
    - Taxi environment?

Environment Properties

- **Single agent vs. multi-agent**
  - Crossword puzzle: SA
  - Chess: MA
  - MA: Can be competitive, cooperative, a mix
  - Taxi environment?

Environment Properties

- **Deterministic vs. Stochastic:**
  - Deterministic if the next state is *completely* determined by the current state and the action executed by the agent
  - A multi-agent system can be deterministic; for e.g., chess is deterministic
  - Uncertain ≠ Stochastic
  - Uncertain if partially observable *or* stochastic

ENVIRONMENT PROPERTIES

- **Episodic vs. Sequential:**
  - In each episode, the agent receives a percept and then performs a single action. The next episode does not depend on the actions taken in the previous episodes
  - Classification/categorization of objects: E
  - Chess: S

Environment Properties

- **Static vs. Dynamic:**
  - Dynamic if the environment can change while the agent is thinking
  - In dynamic environments, “no decision” is equal to “deciding not to act”
  - Poker: S
  - Driving: D

Environment Properties

- **Discrete vs. Continuous:**
  - Discrete if states, percepts, and actions are discrete
  - Chess : state of the board and moving pieces are D
  - Driving: the location, steering, speeding are C

Environment Properties

- **Known vs. Unknown**:
  - Known if the agent knows the "laws of the physics" of the environment
    - That is, the outcomes in a deterministic environment and the probabilities of the outcomes in a stochastic environment are known
- K vs. U is not the same as FO vs. PO
  - An environment can be PO and K: e.g., card games where the rules are known but the state is partially observable

The Simplest vs. Hardest

- Simplest: Fully-observable, single-agent, deterministic, episodic, static, discrete, and known
  - Many toy examples
- Hardest: Partially-observable, multi-agent, stochastic, sequential, dynamic, continuous, and unkown
  - Driving in a foreign country using a rental car

Characteristics of Environments

- Most of the characteristics depend on how you define the task
- Often, the real-world is the hardest
- However, simplifications are made to make progress
- For e.g., often playing chess is a known environment, but the first-time player might not fully know the rules of the game, yet.

CS480 – Artificial Intelligence – Please do not distribute.

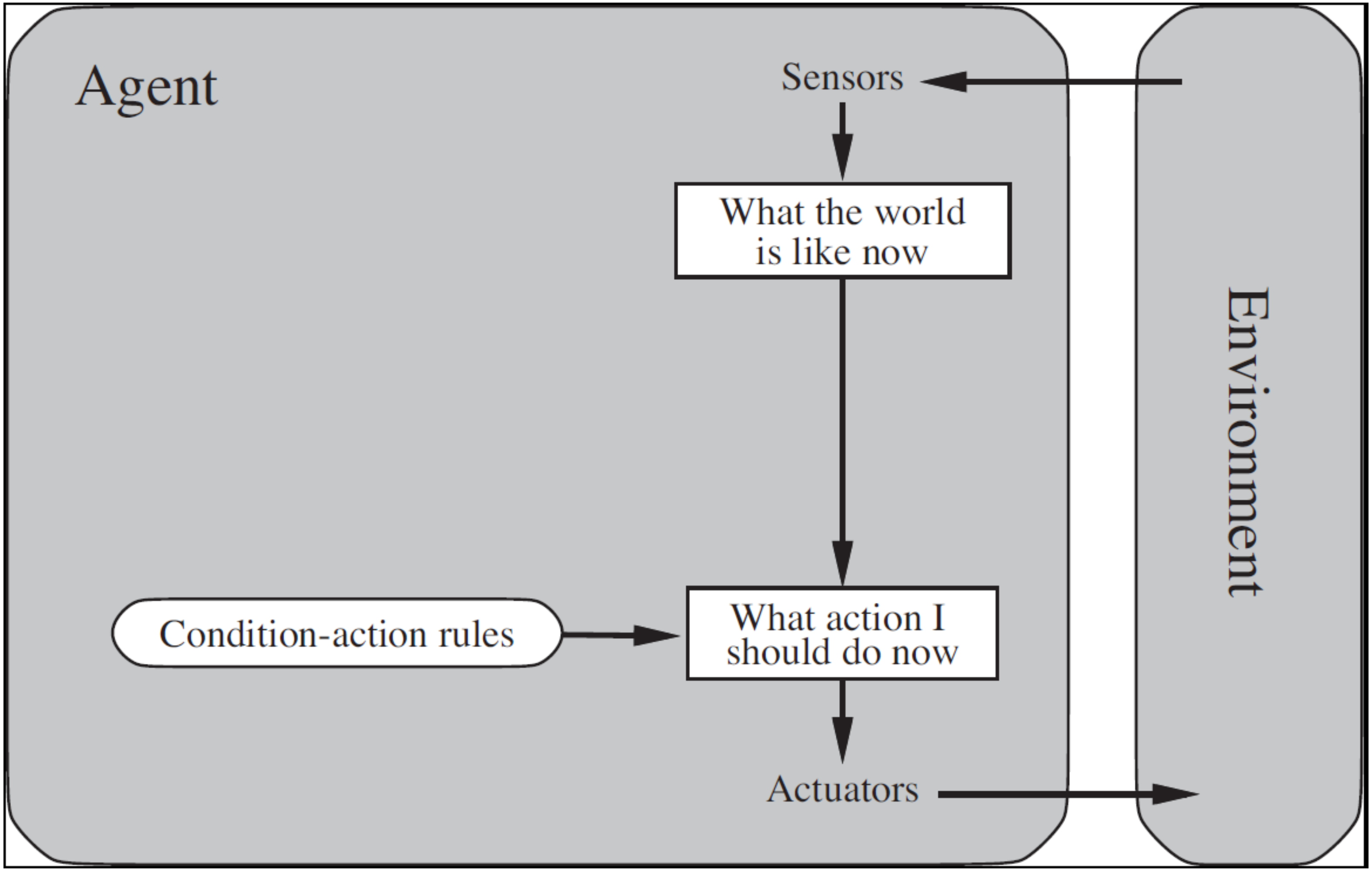
Agent Program Types

1. Simple reflex agents
2. Model-based reflex agents
3. Goal-based agents
4. Utility-based agents

Simple Reflex Agents

- Select actions based on *only* the current percept ignoring the past
- Works if the environment is fully-observable and episodic

Simple Reflex Agents



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*

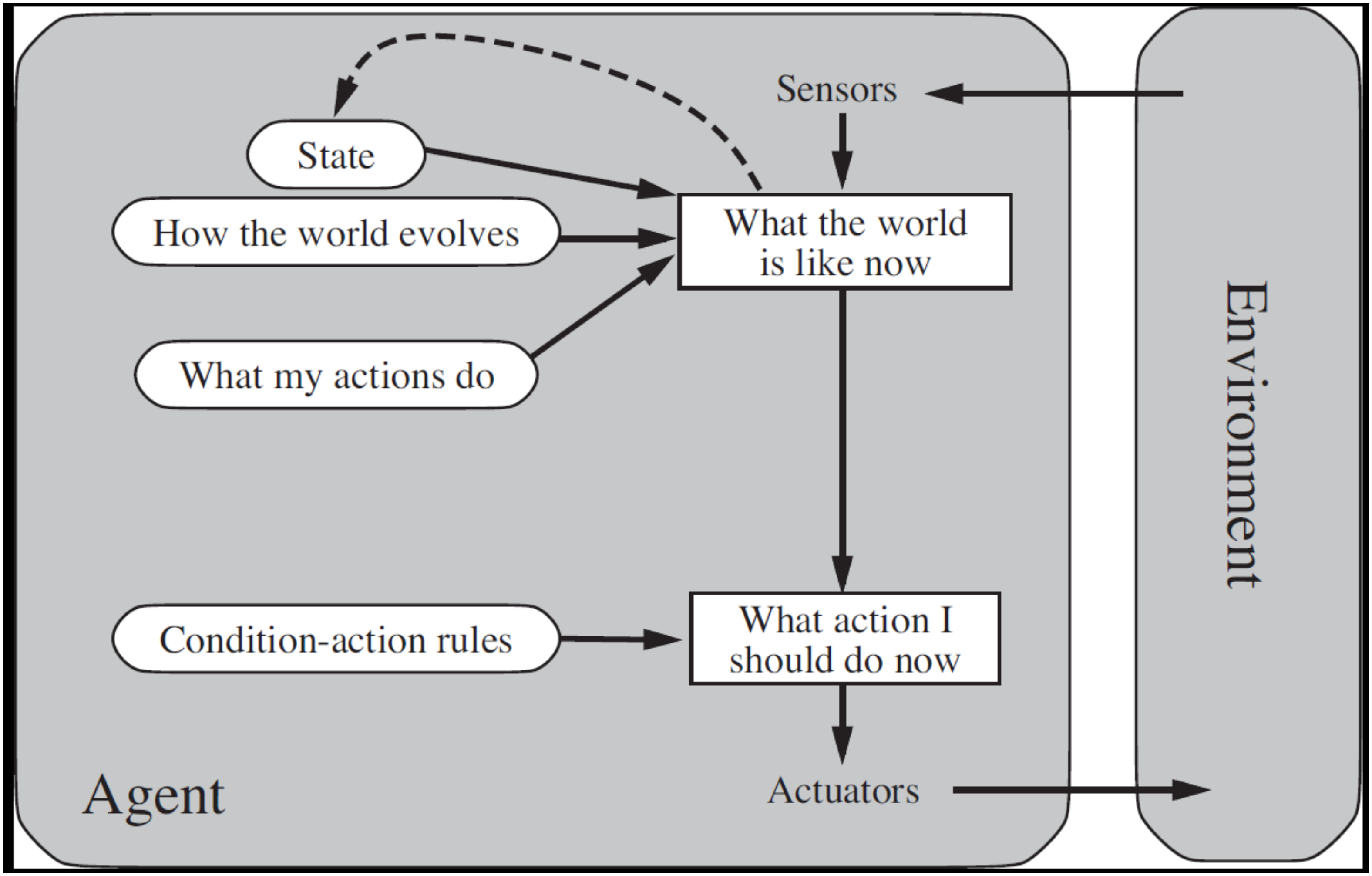
function SIMPLE-REFLEX-AGENT(*percept*) **returns** an action
  persistent: *rules*, a set of condition–action rules

  state ← INTERPRET-INPUT(*percept*)
  rule ← RULE-MATCH(*state*, *rules*)
  action ← *rule*.ACTION
  return *action*

Model-Based Agent

- Handle partial-observability → keep track of what is not observed
  - Keep an internal state based on the percept history
- The agent needs the knowledge of how the world works → a model of the world

Model-Based Agent



Model-Based Agent

function SIMPLE-REFLEX-AGENT(*percept*) **returns** an action
  persistent: *rules*, a set of condition–action rules

  state ← INTERPRET-INPUT(*percept*)
  rule ← RULE-MATCH(*state*, *rules*)
  action ← *rule*.ACTION
  return *action*

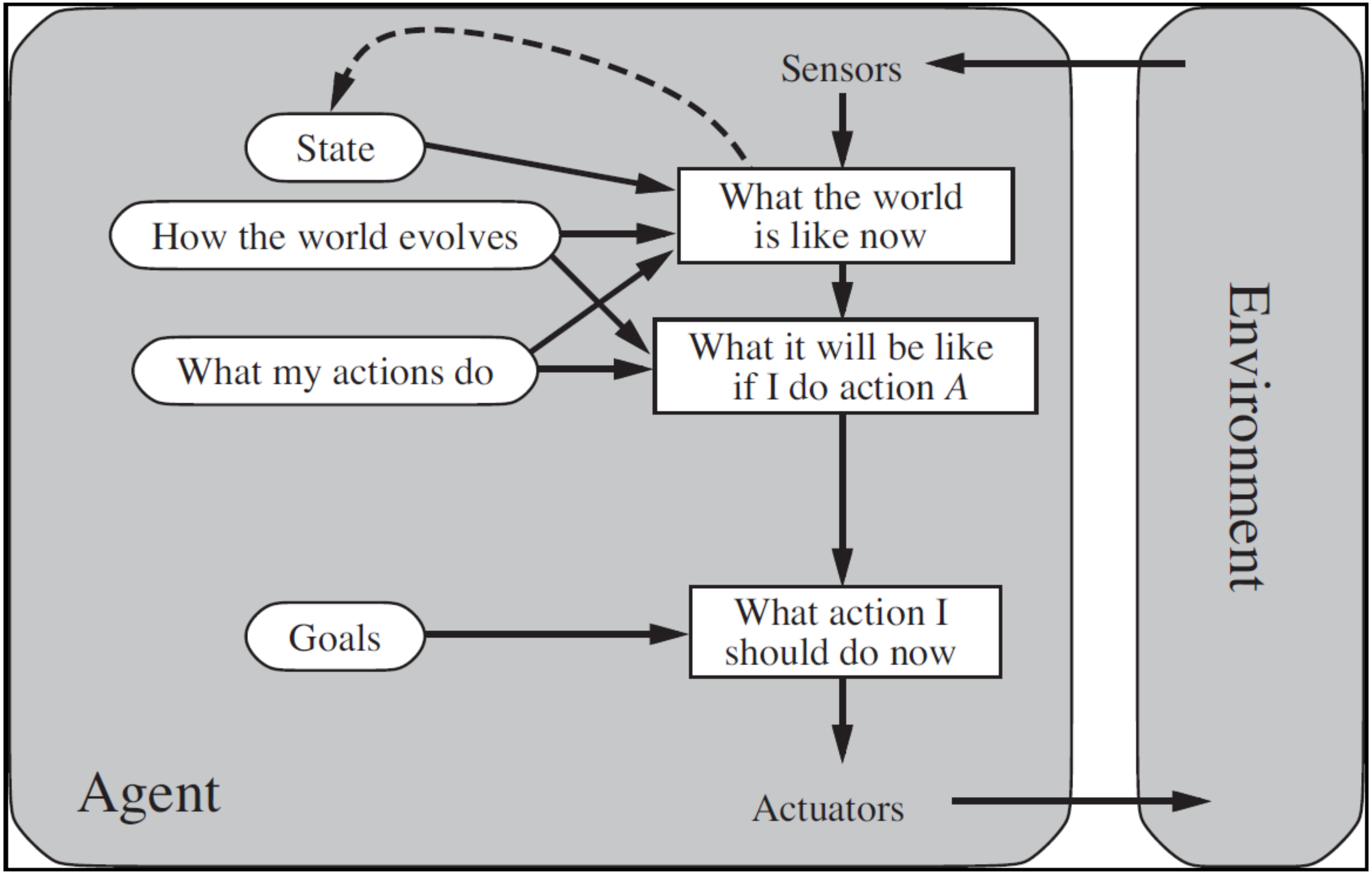
function MODEL-BASED-REFLEX-AGENT(*percept*) **returns** an action
  persistent: *state*, the agent's current conception of the world state
    model, a description of how the next state depends on current state and action
    rules, a set of condition–action rules
    action, the most recent action, initially none

  state ← UPDATE-STATE(*state*, *action*, *percept*, *model*)
  rule ← RULE-MATCH(*state*, *rules*)
  action ← *rule*.ACTION
  return *action*

Goal-Based Agent

- The current state is not always enough to determine what to do next
- Need goal states that are desirable
- Goal-based action is
  - Easier if a single action can take you there
  - Need **searching** and **planning** if more actions are required
- Goal-based is much more flexible and general than reflex-based agents
  - New goals can be defined easily without changing much of the agent program
  - In the reflex-based agent, the rules will have to be re-written

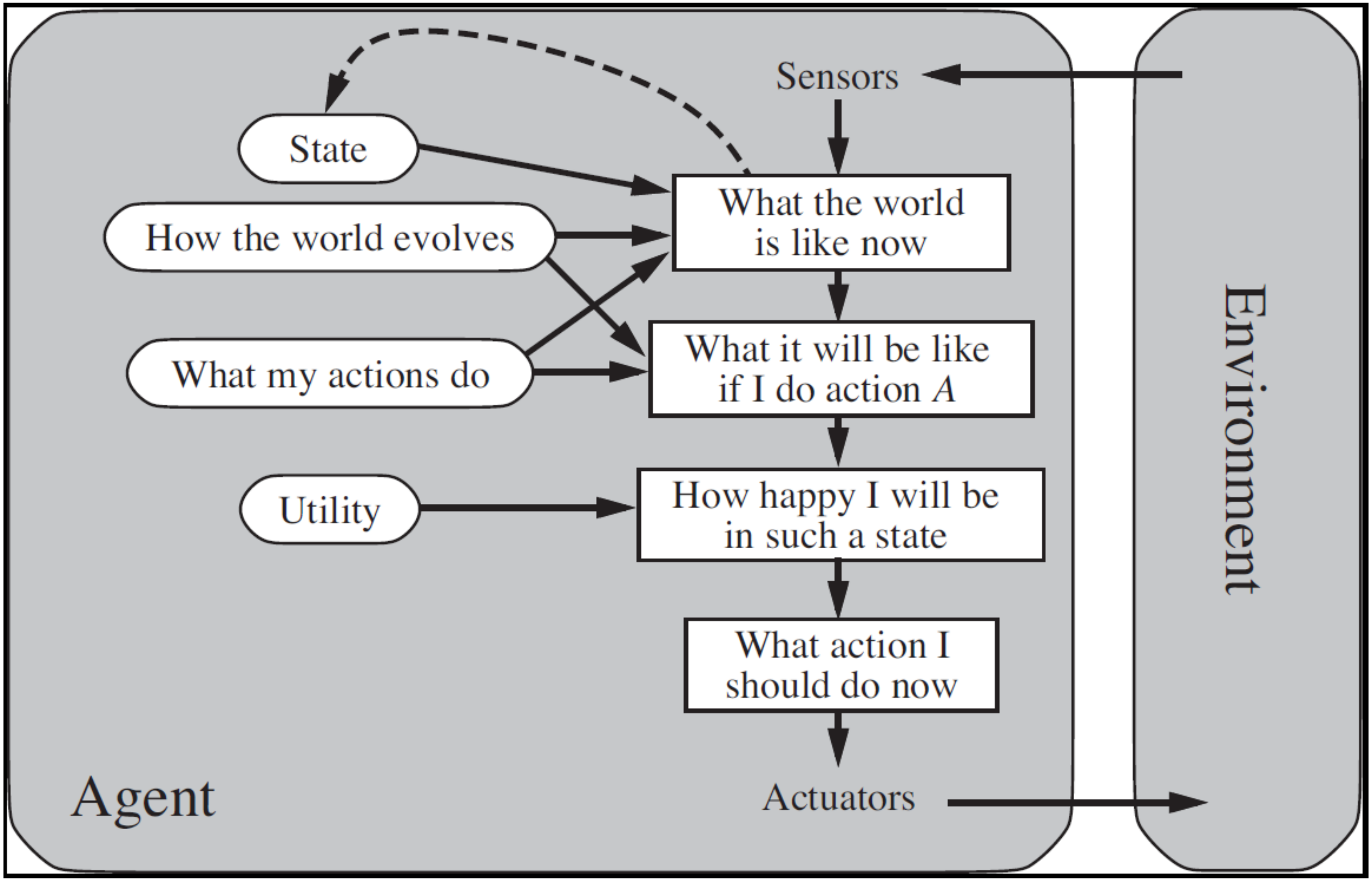
Goal-Based Agent



Utility-based Agent

- Goals define “happy” vs. “not-happy”
- Utility defines “how happy” the agent will be in a given state
- Utility functions are internalizations of the performance measures
- Again, utility-based agents are more flexible than
  - reflexive agents, and
  - goal-based agents
- Utility-based agents can handle stochasticity, multiple conflicting goals, etc.

Utility-based Agent



Learning Agent

- The rules that map a percept / percept sequence to an action are learned
- The learning agent tries to maximize the performance measure
- A critic provides feedback to the agent on how well it has done
- The agent needs to explore different possibilities to be able to improve

LEARNING AGENT