Artificial Intelligence
(IT3160E)

Than Quang Khoat

khoattq@soict.hust.edu.vn

School of Information and Communication Technology
Hanoi University of Science and Technology

2025

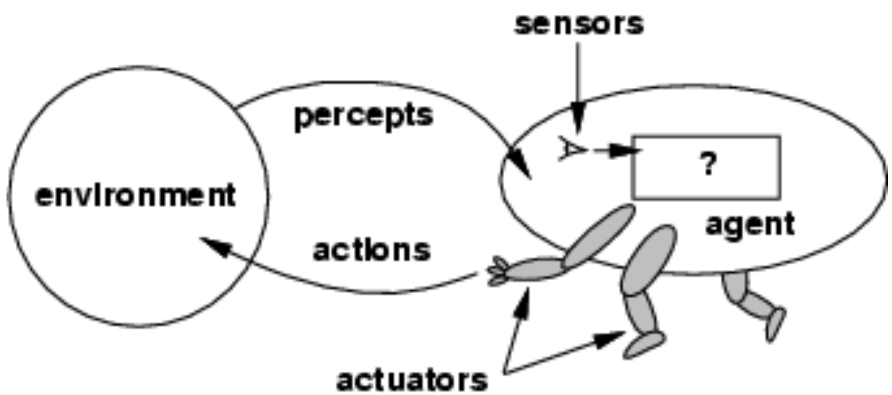
Content:

- Introduction of Artificial Intelligence
- **Intelligent agent**
  - **Definition of agent**
  - **Work environment**
  - **Environment types**
  - **Agent types**
- Problem solving: Search, Constraint satisfaction
- Logic and reasoning
- Knowledge representation
- Machine learning

Definition of Agent

- An *agent* (tác tử) is anything (e.g., humans, robots, thermostats, etc.) which can *perceive* (cảm nhận) its surrounding environment through *sensors* and *act* (hành động) accordingly to that environment through *actuators*
- Human agent
  - Sensors: eyes, ears and other body parts
  - Actuators: hands, legs, mouth and other body parts
- Robot agent
  - Sensors: cameras, infrared signal detectors
  - Actuators: motors

Agent and Environment

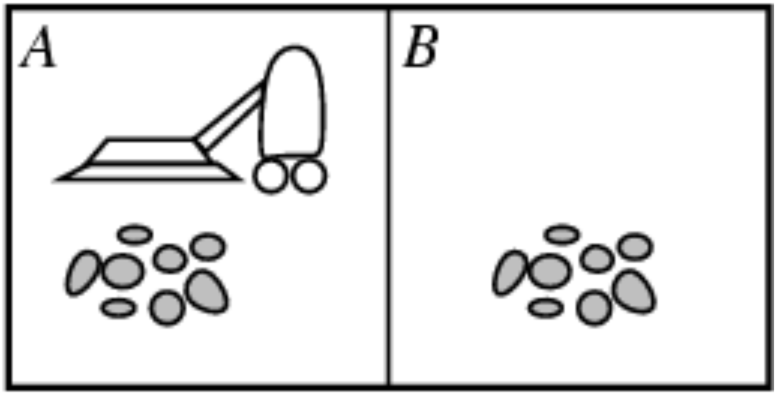


- Agent function: maps the history of perception to actions

$$f: \mathcal{P}^{\star} \rightarrow \mathcal{A}$$

- Agent program: operates based on the actual architecture of the function *f*
- Agent = Architecture + Program

Example: Vacuum cleaner agent

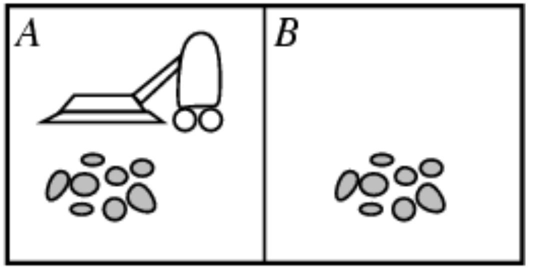


- Perceptions
  - Vacuum cleaner's location and cleanliness level
  - Example: [*A*, *Dirty*], [*B*, *Dirty*]
- Actions
  - The vacuum cleaner moves *left*, *right*, or *sucks*

Vacuum cleaner agent

Table of actions of vacuum cleaner agent

Sequence of perceptions	Action
[A, Clean]	Move right
[A, Dirty]	Suck
[B, Clean]	Move left
[B, Dirty]	Suck
[A, Clean], [A, Clean]	Move right
[A, Clean], [A, Dirty]	Suck
. . .	



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*

Rational agent (1)

- The agent should strive to "do the right thing to do", based on **what it perceives (i.e., knows)** and **the actions it can perform**
- A **right (rational) action** is the one that helps the agent achieve the highest success to the given target
- **Performance evaluation**: The criteria to evaluate the level of success in the performance of an agent
    - Example: Criteria to evaluate the performance of a vacuum cleaner agent can be: *cleanness level*, *vacuuming time*, *power consumption*, *noise levels*, etc.

Rational agent (2)

- **Rational agent**
  - Given *a sequence of perceptions*,
  - A rational agent needs to *choose an action* that maximizes that agent's performance evaluation criteria,
  - Based on the *information* provided by the sequence of perceptions and the *knowledge* possessed by that agent

Rational agent (3)

- Rationale ≠ The understanding of everything
  - The understanding of everything = Know everything, with infinite knowledge
  - Perceptions may not provide all the relevant information
- Agents can perform actions to change perceptions in the future, for the purpose of obtaining useful information (e.g., information gathering, knowledge discovery)
- *Autonomous* agent is one whose actions are determined by its own experience (along with the ability to *learn* and *adapt*)

Work environment – PEAS (1)

- In order to design an intelligent (i.e., rational) agent, it is first necessary to define the values of the PEAS components
- PEAS
  - *Performance measure*: Performance evaluation criteria
  - *Environment*: Surrounding environment
  - *Actuators*: Those parts that allow the agent to do the actions
  - *Sensors*: Those parts that allow the agent to perceive the surrounding environment

Work environment – PEAS (2)

- Example: Design a taxi driving agent
  - Performance measure (P): safe, fast, in compliance with traffic laws, customer satisfaction, optimal profit, etc.
  - Environment (E): roads (streets), other vehicles in traffic, pedestrians, customers, etc.
  - Actuators (A): steering wheel, accelerator, brake, signal lights, horn, etc.
  - Sensors (S): cameras, speedometer, GPS, distance meter, motor sensors, etc.

Work environment – PEAS (3)

- Example: Design a medical diagnostic agent
  - Performance measure (P): the patient's health level, minimizing costs, lawsuits, etc.
  - Environment (E): patients, the hospital, medical staffs, etc.
  - Actuators (A): screen to display the questions, tests, diagnoses, treatments, instructions, etc.
  - Sensors (S): keyboard to enter the symptom information, patient responses to questions, etc.

Work environment – PEAS (4)

- Example: Design an object pick-up agent
  - Performance measure (P): percentage of the items placed in the correct boxes (i.e., containers)
  - Environment (E): Conveyor on that there are objects, boxes (i.e., containers)
  - Actuators (A): arms and connected hands
  - Sensors (S): camera, angle/direction sensors

Work environment – PEAS (5)

- Example: Design an interactive English-teaching agent
  - Performance measure (P): maximizing students' English test scores
  - Environment (E): a group of students
  - Actuators (A): screen to display exercises, suggestions, assignments' corrections
  - Sensors (S): keyboard

Work environment – PEAS (6)

- Example: Design a spam email filtering agent
  - Performance measure (P): the number of errors (e.g., false positives, false negatives)
  - Environment (E): email server and clients
  - Actuators (A): spam email marker, notification sender
  - Sensors (S): the module that receives and analyzes the emails’ content

Environment types (1)

- **Fully observable** (vs. partially observable)?
  - The agent's sensors give it access to the *full state* of the environment at a time
- **Deterministic** (vs. stochastic)?
  - The next state of the environment is determined exactly by the current state and the agent's action (at this current state)
  - If an environment is deterministic, except for the actions of other agents, it is called the *strategic environment*

Environment types (2)

- **Episodic** (vs. sequential)?
  - The agent's experience is divided into atomic "*episodes*"
  - Each episode consists of the agent's perceiving and then performing a single action
  - The choice of action in each episode depends only on the episode itself (i.e., not on the other ones)
- **Static** (vs. dynamic)?
  - The environment is unchanged while the agent is deliberating
  - The environment is *semi-dynamic* if the environment itself does not change with the passage of time but the agent's performance score does
    - Example: Timed game programs

Environment types (3)

- **Discrete** (vs. continuous)?
  - A limited number of distinct, clearly defined percepts and actions
- **Single agent** (vs. multi-agent)?
  - An agent operating by itself (i.e., not dependent on/relating to any others) in an environment

Environment types: Examples

	Chess with a clock	Chess without a clock	Taxi driving
Fully observable?	Yes	Yes	No
Deterministic?	Strategic	Strategic	No
Episodic?	No	No	No
Static?	Semi-dyna.	Yes	No
Discrete?	Yes	Yes	No
Single agent?	No	No	No

- The environment type largely determines the agent design
- A real-world environment is often: *partially observable*, *stochastic*, *sequential*, *dynamic*, *continuous*, *multi-agent*

Agent types

- Four basic agent types:
  - Simple reflex agents
  - Model-based reflex agents
  - Goal-based agents
  - Utility-based agents

Simple reflex agents (1)

→ Act according to a rule that has its conditions consistent with the current state of the environment

function SIMPLE-REFLEX-AGENT(*percept*)

static: *rules* (a set of rules in format of <conditions> - <action>)

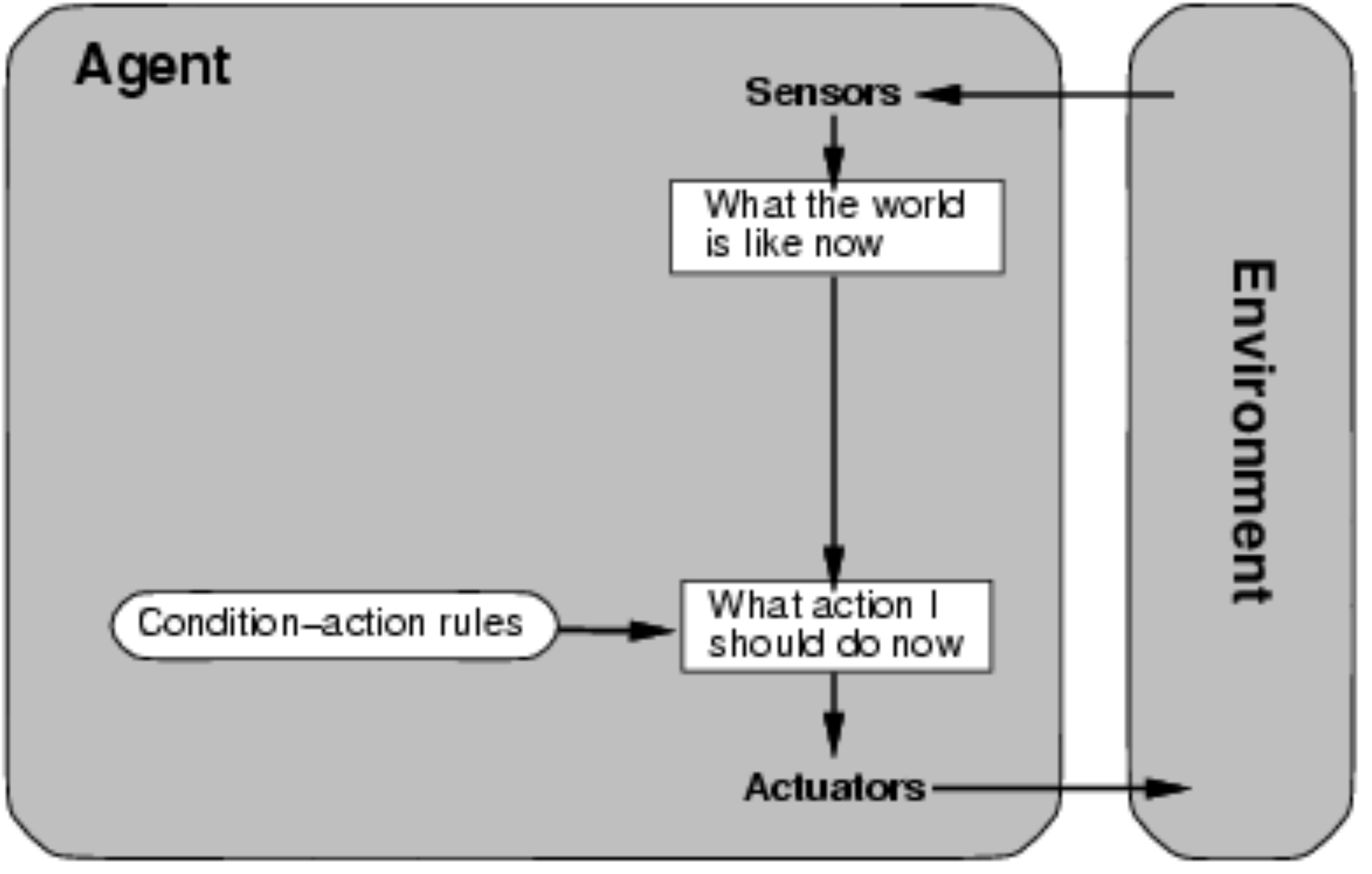
state ← INTERPRET-INPUT(*percept*)

rule ← RULE-MATCH(*state*, *rules*)

action ← RULE-ACTION[*rule*]

return *action*

Simple reflex agents (2)



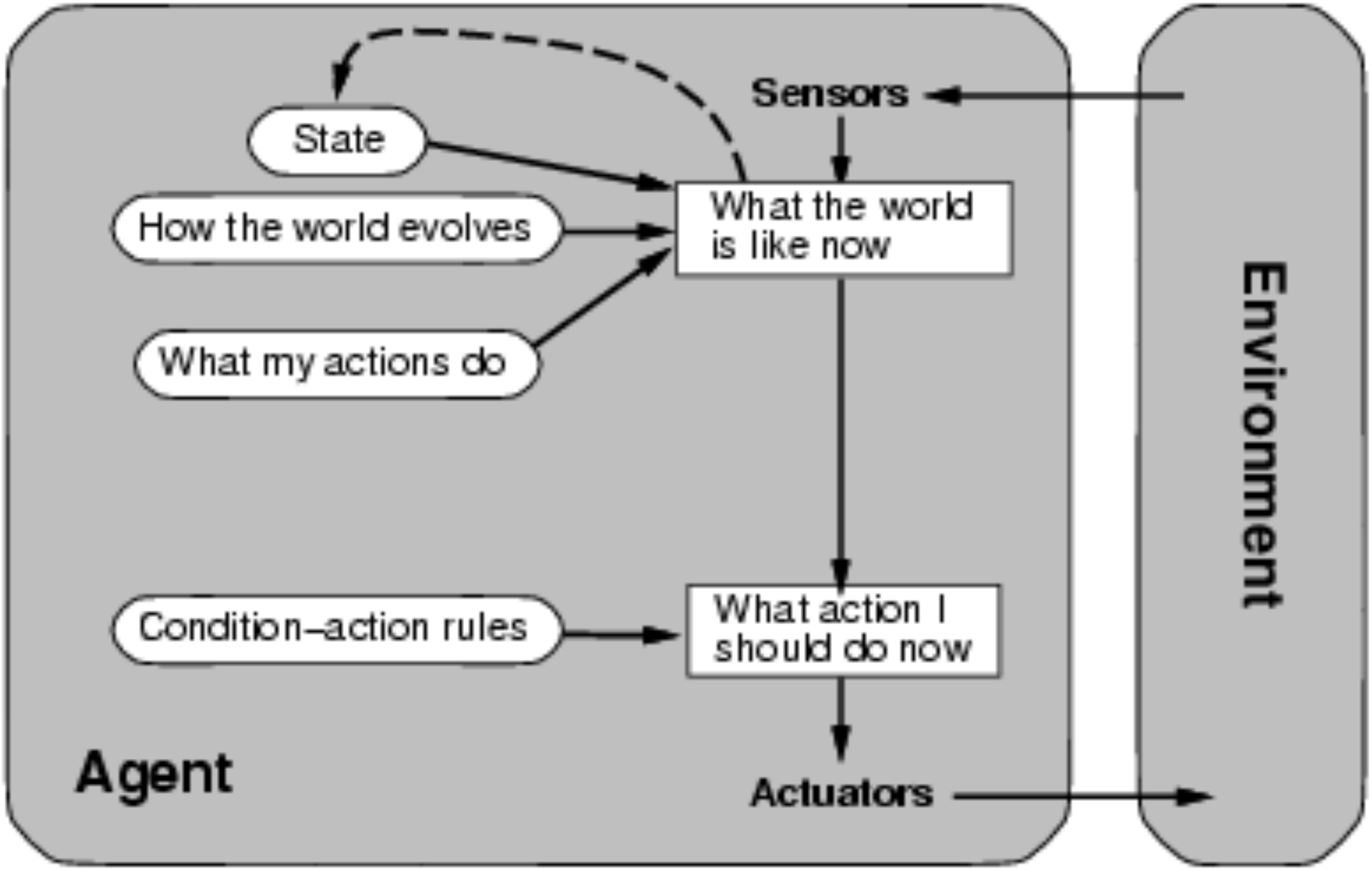
Model-based reflex agents (1)

- Use an internal model to monitor the current state of the environment
- Choose the action: The same as for simple reflex agents

function REFLEX-AGENT-WITH-STATE(*percept*)
static: *state* (representation of the current state of the environment)
rules (a set of rules in format of <conditions> - <action>)
action (the previous/latest action)

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

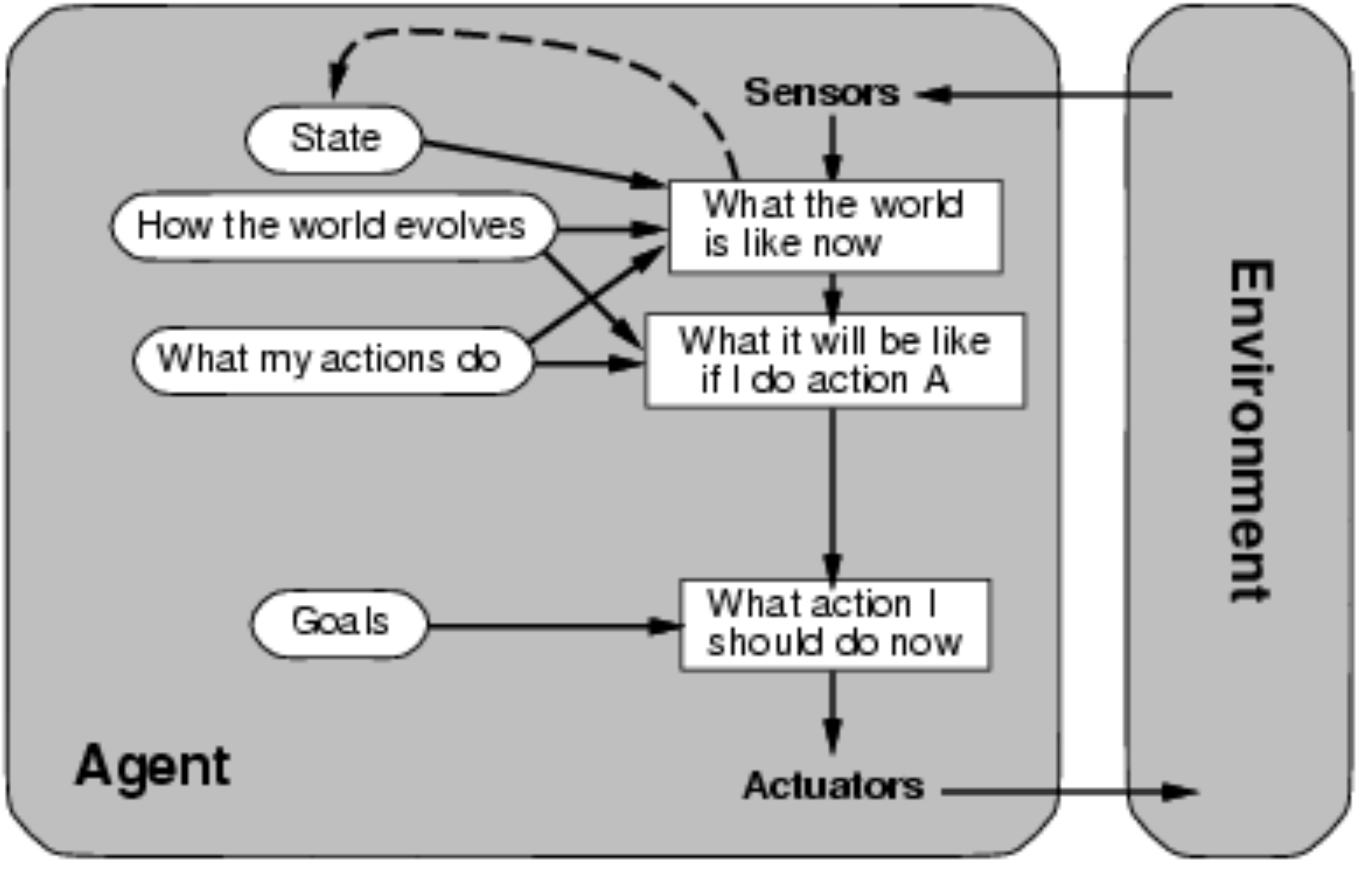
Model-based reflex agents (2)



Goal-based agents (1)

- Know the current state of the environment: Not enough $\rightarrow$ Need information of the goal
  - The current state of the environment: At an intersection, a taxi can turn left, turn right, or go straight
  - Goal information: The taxi needs to reach the passenger's destination
- Goal-based agent
  - Keep track of the current state of the environment
  - Keep a set of goals (to be achieved)
  - Choose the action that allows to (finally) achieve the goals

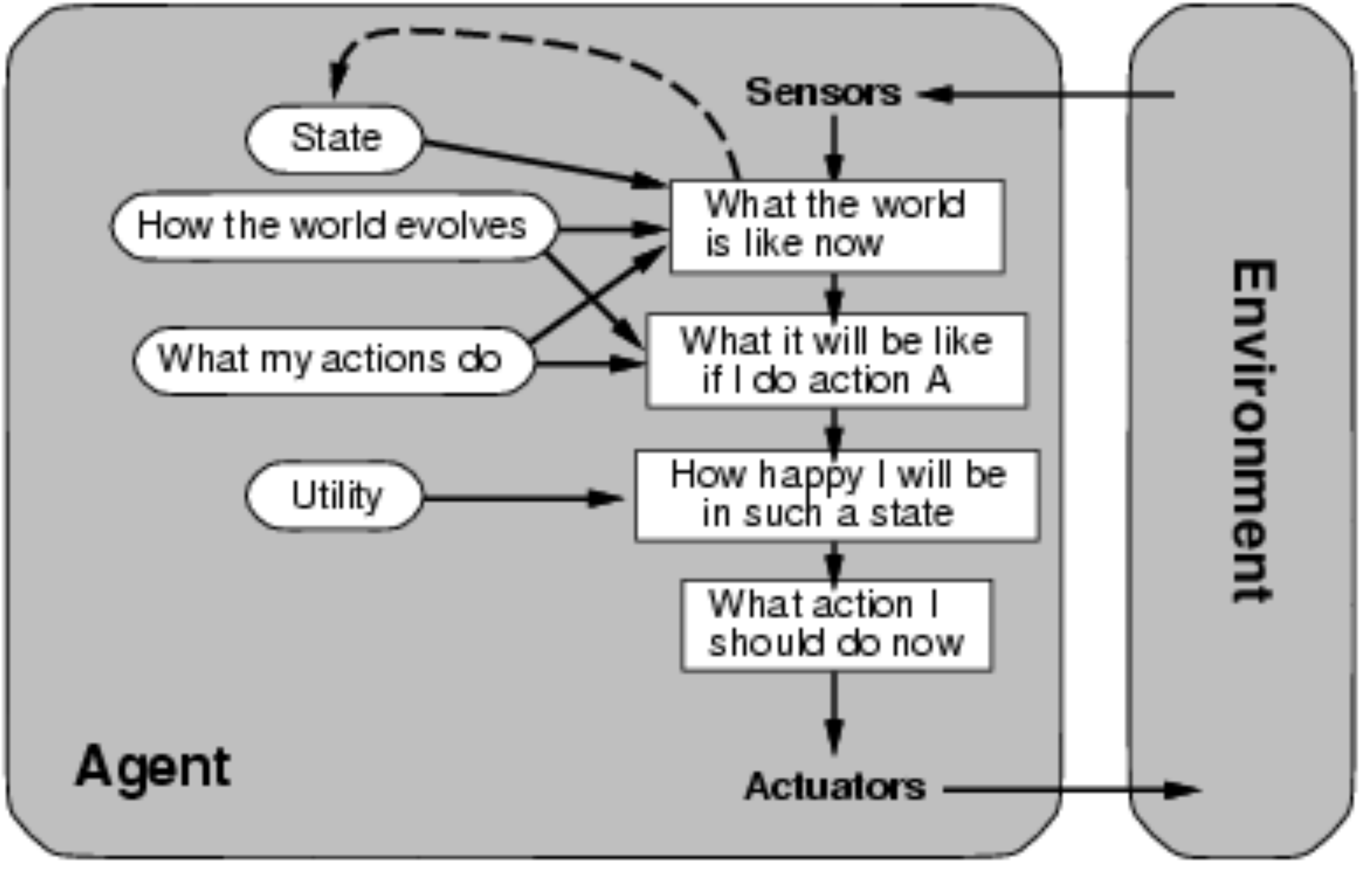
Goal-based agents (2)



Utility-based agents (1)

- In many environments, the information of the goals is not sufficient to assess the effectiveness of actions
  - There are several (or many) sequences of actions to allow a taxi to reach its destination (i.e., achieve the goal)
  - But: Which sequence of actions is faster, safer, more reliable, lower cost?
- Need an assessment of the utility (i.e., benefit) to the agent
- Utility function
  - Mapping the **sequence** of environmental states to a real number (i.e., the level of utility/benefit to the agent)

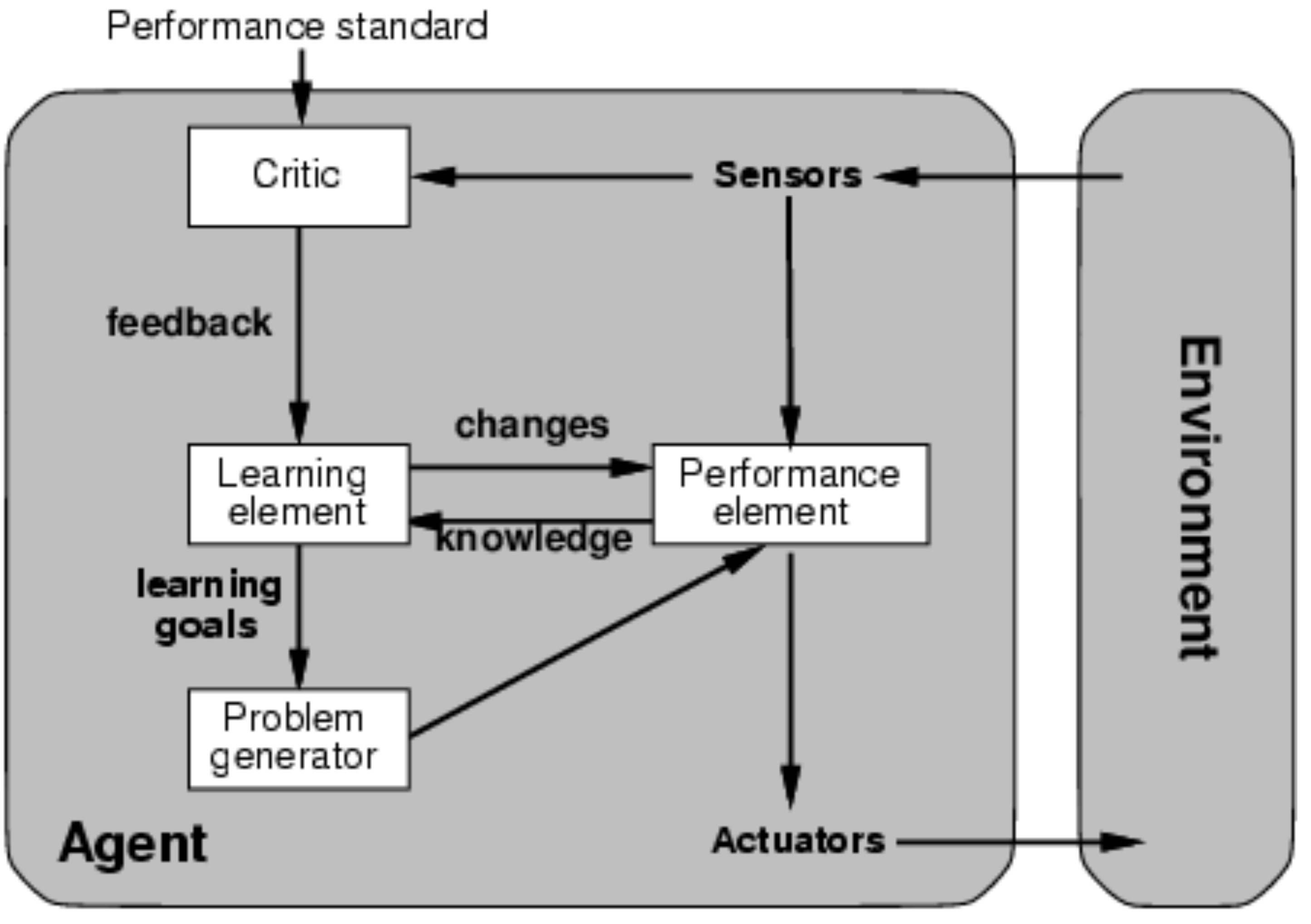
Utility-based agents (2)



Learning agents (1)

- The ability to learn allows the agent to improve its performance
- The 4 elements make up a learning agent:
  - *Performance*: undertakes the choice of action
  - *Critic*: evaluates the performance
  - *Learning*: helps to improve the performance - based on critics, to change (improve) the Performance element
  - *Problem generator*: helps to generate new experiences

Learning agents (2)



Multi-agent (1)

- Work environment: **Collaborative** or **Competitive**?
- In many practical problems, the work environment is always changing → the agent needs to get updated
- Need a model to represent the plans of other agents
- **Collaborative agents**
  - Share goals or plans together
  - Example: Planning (for group activities) in a double's tennis game
  - Collaboration mechanisms: Separate and distribute tasks for each agent

Multi-agent (2)

- **Competitive agents**
  - Example: Chess game
  - Each agent must be aware of the existence (and activity) of the other agents
  - Each agent computes (i.e., predicts) the plans of (some) other actors
  - Each agent computes (i.e., predicts) the effect of the others' plans on its own
  - Each agent determines the optimal action against this predicted effect

Agent: Summary

- An agent interacts with the environment through its sensors and actuators
- A rational agent maximizes its performance
- The agent function determines the actions an agent performs in situations
- Agent programs implement (i.e., execute) the agent functions
- PEAS descriptions define the work environment
- The environments are classified according to the criteria: Fully observable? Deterministic? Episodic? Statistic? Discrete? Single agent?
- Basic agent types: Simple reflex, Model-based reflex, Goal-based, Utility-based