CS420 – Artificial Intelligence

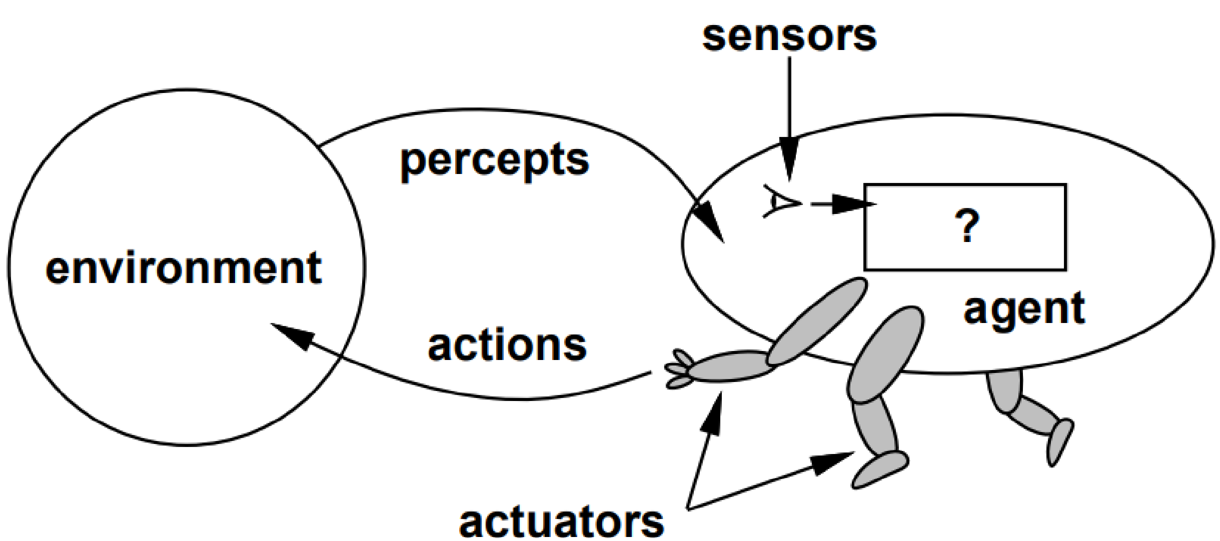
INTELLIGENT AGENTS

Nguyễn Ngọc Thảo
nnthao@fit.hcmus.edu.vn

Outline

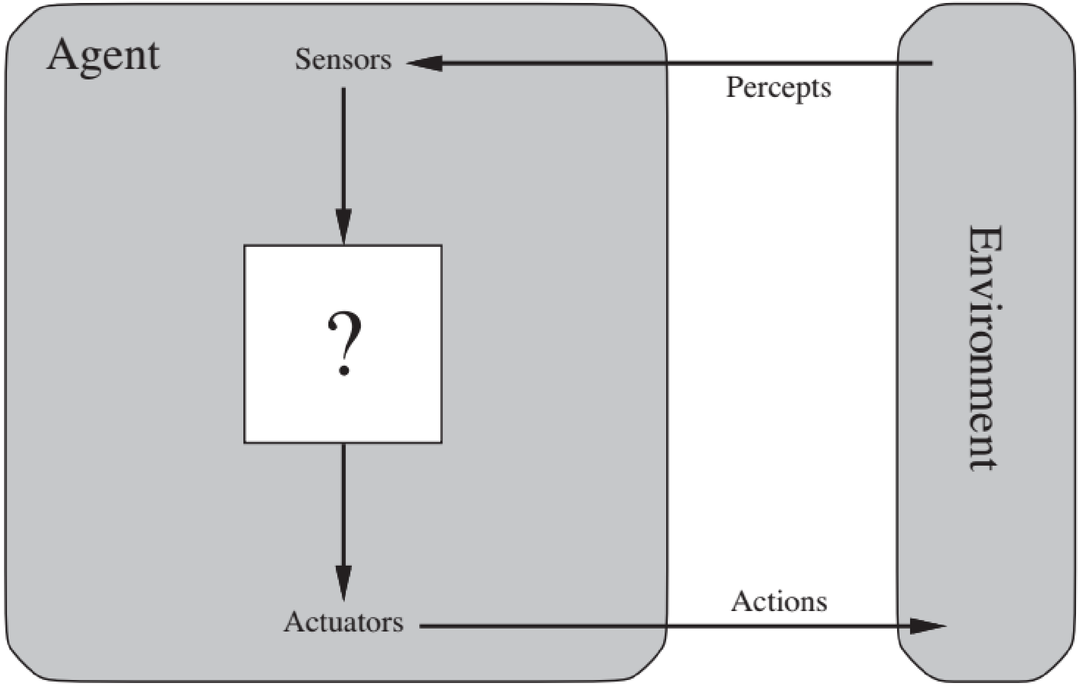
- Agents and Environments
- Good Behavior: The Concept of Rationality
- The Nature of Environments
- The Structure of Agents

Agents and Environments



What is Agent?

- An agent **perceives** its environment through **sensors** and **acts** upon that environment through **actuators**.



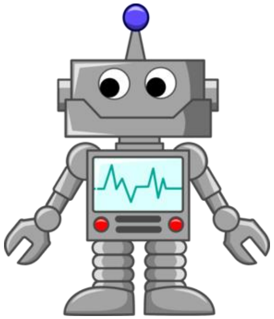
Agents interact with environments through sensors and actuators.

Examples of agent

Human agent

Sensors: eyes, ears, and other organs.

Actuators: hands, legs, vocal tract, etc.



Robotic agent

Sensors: cameras, infrared range finders, etc.

Actuators: levels, motors, etc.

Software agent

Sensors: keystrokes, file contents, network packets, etc.

Actuators: displaying on screen, writing files, sending network packets, etc.

The IA's behavior

- **Percept:** the IA's perceptual inputs at any given instant
- **Percept sequence**: the complete history of everything the IA has ever perceived
- An agent's behavior is described by the **agent function** that maps any given percept sequence to an action.

$$\boldsymbol{f}: \mathcal{P} \rightarrow \mathcal{A}$$

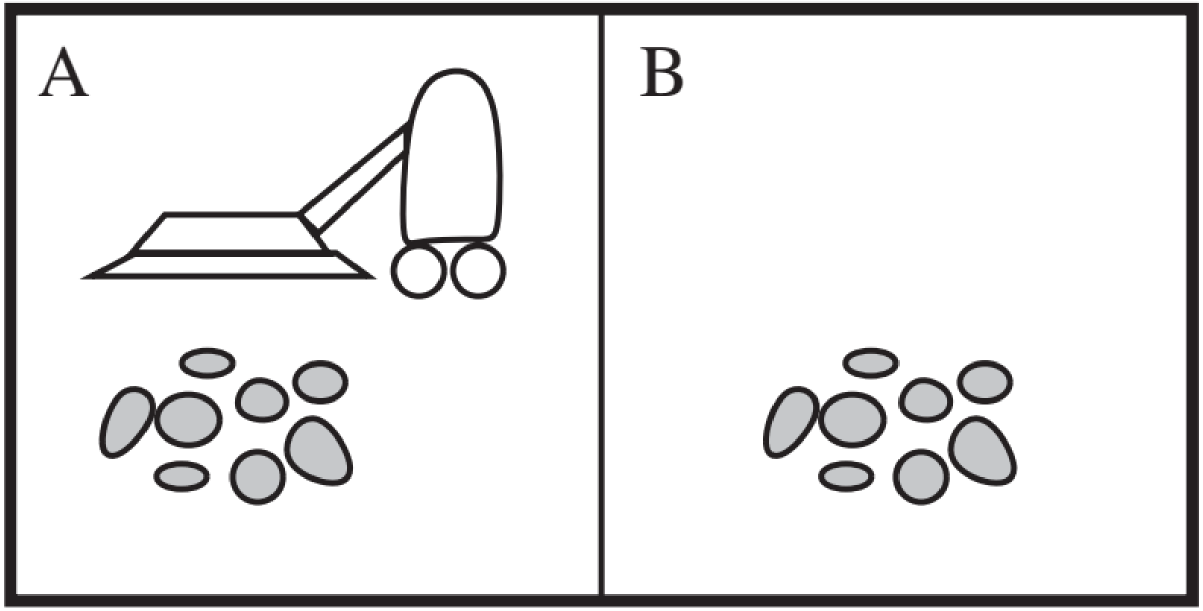
- **Agent program:** the implementation of the agent function

agent = architecture + program

mathematical

practical

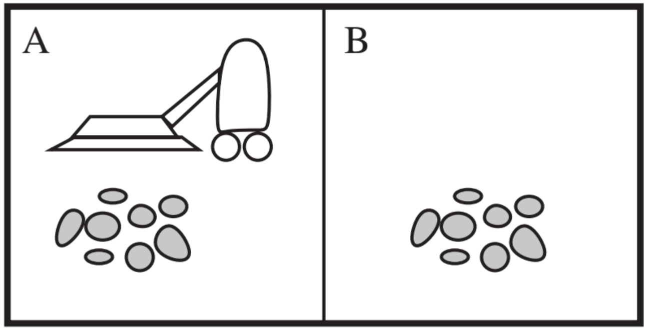
The Vacuum-cleaner world



A vacuum-cleaner world with just two locations

- Percepts: location and contents, e.g., [A,Dirty]
- Actions: Left, Right, Suck, Do Nothing

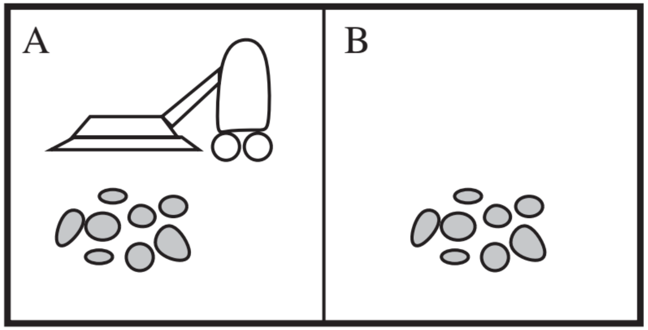
The Vacuum-cleaner world



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

Partial tabulation of a simple agent function for the vacuum-cleaner world

The Vacuum-cleaner world

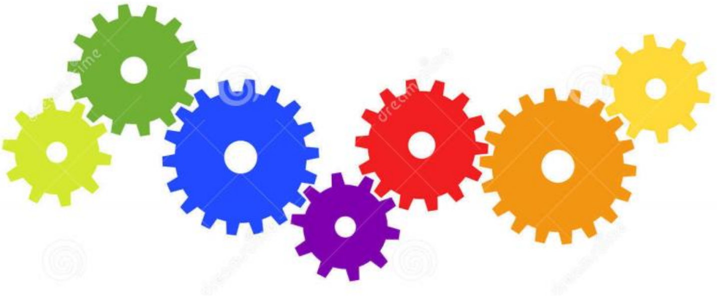


function REFLEX-VACUUM-AGENT([*location*,*status*]) **returns** an action
    if *status* = *Dirty* **then return** *Suck*
    else if *location* = *A* **then return** *Right*
    else if *location* = *B* **then return** *Left*

The agent program for a simple reflex agent in the two-state vacuum environment.

The Concept of Rationality

- *Rationality*
- *Omniscience, Learning, and Autonomy*



Rational agents

- A **rational agent** is one that does the **right thing.**
  - Every entry in the table for the agent function is filled out correctly.
- What is "**right**" thing?
  - The actions that cause the agent to be most successful
- We need ways to **measure success.**

Performance measure

Performance measure

- An agent, based on its percepts $\rightarrow$ generates actions sequence $\rightarrow$ environment goes to sequence of states
  - If this sequence of states is desirable then the agent performed well
- **Performance measure** evaluates any given sequence of **environment states** (remember, **not agent states!!!**).
  - An objective function that determines how the agent does successfully.
  - 90%? 30%?

Design performance measures

- **General rule:** Design performance measures according to

  What one actually **wants** in the environment

  Not how one **thinks** the agent should behave

- For example, in vacuum-cleaner world
  - The amount of dirt cleaned up in a single eight-hour shift, or
  - The floor clean, no matter how the agent behaves
  - *Which one is better?*

Rationality

- What is rational at any given time depends on

Performance measure Define the criterion of success	**Prior knowledge** What the agent knows about the environment
Percept sequence The agent's percept to date	**Actions** What the agent can perform

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.

- For example, in an exam,
  - Maximize marks based on the questions on the paper and your knowledge

The Vacuum-cleaner agent

- Performance measure
  - Awards one point for each clean square at each time step, over 10000 time steps
- Prior knowledge about the environment
  - The geography of the environment (2 squares)
  - The effect of the actions
- Actions that can perform
  - Left, Right, Suck and Do Nothing
- Percept sequences
  - Where is the agent?
  - Whether the location contains dirt?
- Under this circumstance, the agent is **rational.**

Omniscience, learning, and autonomy

Omniscience vs. Rationality

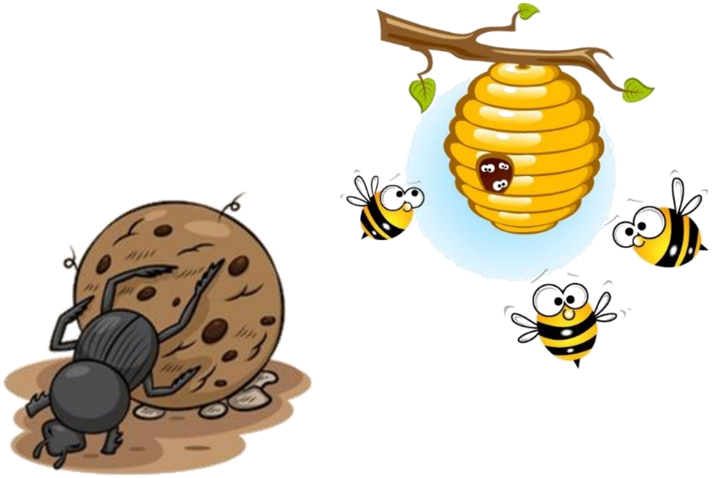
Omniscience	Rationality
• Knows the actual outcome of its actions in advance • No other possible outcomes • However, impossible in real world • Example?	Maximize performance measure given the percepts sequence to date and prior knowledge Rationality is not perfection

Information gathering

- The agent must not engage in unintelligent activities due to **inadvertency**.
- **Information gathering –** Doing actions in order to modify future percepts
  - E.g. exploration
- This is an important part of rationality.

Learning

- A rational agent also has to **learn** as much as possible from what it perceives.
  - The agent's initial configuration may be modified and augmented as it gains experience.
- There are extreme cases in which the environment is completely known *a priori*.

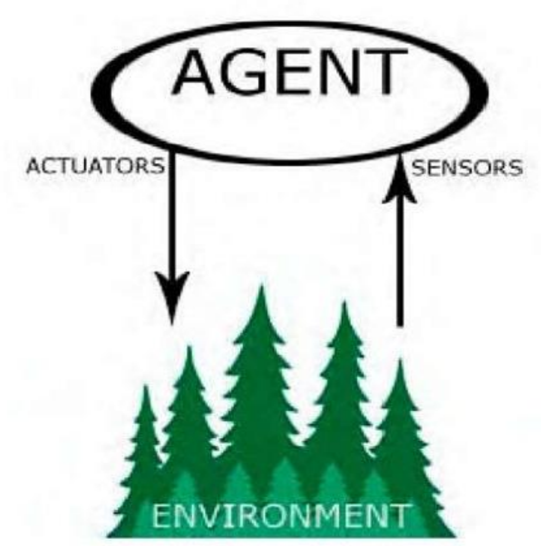


Autonomy

- A rational agent should be **autonomous** – Learn what it can to compensate for partial or incorrect prior knowledge.
  - If an agent just relies on the prior knowledge of its designer rather than its own percepts then the agent lacks **autonomy**.
  - E.g., a clock
    - No input (percepts)
    - Run its own algorithm (prior knowledge)
    - No learning, no experience, etc.

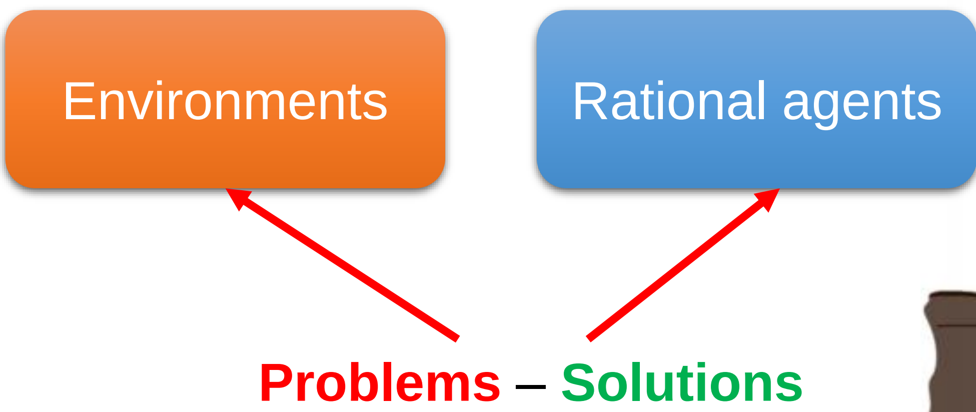
The Nature of Environments

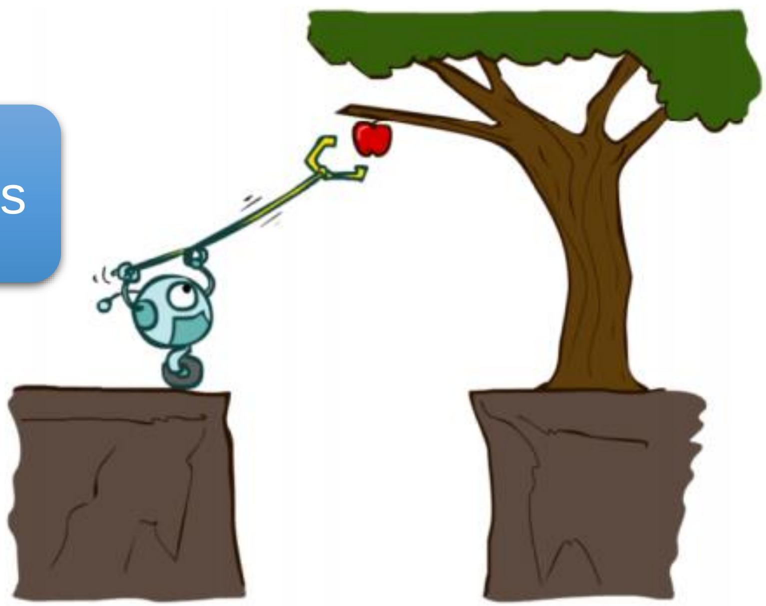
- *Specifying the Task Environment*
- *Properties of Task Environments*



The task environment

- **Task environments** are essentially the "problems" to which rational agents are the "solutions."





- They come in a variety of flavors, which directly affects the appropriate design for the agent program.

The task environment

- The task environment includes

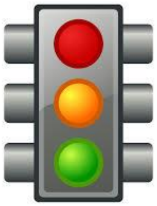
PEAS

- **P**erformance measure
- **E**nvironment
- Agent's **A**ctuators
- Agent's **S**ensors

- In designing an agent, the first step must always be to specify the **task environment (PEAS)** as fully as possible.

An example: Automated taxi driver

- **Performance measure**
  - How can we judge the automated driver?
  - Which factors are considered?
    - getting to the correct destination
    - minimizing fuel consumption
    - minimizing the trip time and/or cost
    - minimizing the violations of traffic laws
    - maximizing the safety and comfort
    - etc.



An example: Automated taxi driver

- Environment
  - A taxi must deal with a variety of roads (rural lane, urban alley, etc.)
  - Traffic lights, other vehicles, pedestrians, stray animals, road works, police cars, puddles, potholes, etc.
  - Interact with the passengers
- Actuators (for outputs)
  - Control over the accelerator, steering, gear, shifting and braking
  - A display to communicate with the customers
- Sensors (for inputs)
  - Controllable cameras for detecting other vehicles, road situations
  - GPS (Global Positioning System) to know where the taxi is
  - Many more devices are necessary: speedometer, accelerometer, etc.

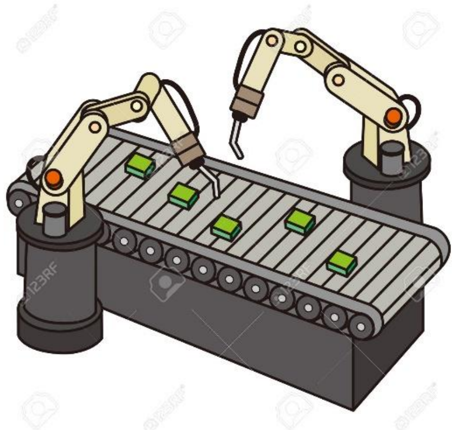
An example: Automated taxi driver

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

PEAS description of the task environment for an automated taxi.

Software agents

- Sometimes, the environment may not be the real world.
  - E.g., flight simulator, video games, Internet
  - They are all artificial but very complex environments



- Those agents working in these environments are called **software agent** (softbots).
  - All parts of the agent are software.

Agents and their PEAS descriptions

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

Examples of agent types and their PEAS description

Quiz: PEAS description

- For each of the following activities, give a PEAS description of the task environment
  - Playing a tennis match
  - Practicing tennis against a wall

Properties of Task environment

Fully observable	Partially observable
Single agent	Multiagent
Deterministic	Stochastic
Episodic	Sequential
Static	Dynamic
Discrete	Continuous
Known	Unknown

- These dimensions determine the appropriate agent design and the applicability of techniques for agent implementation

Fully Observable vs. Partially observable

- **Fully observable:** The agent's sensory gives it access to the complete state of the environment.
  - The agent need not maintain internal state to keep track of the world.
- **Partially observable**
  - Noisy and inaccurate sensors
  - Parts of the state are simply missing from the sensor data, e.g., a vacuum agent with only a local dirt sensor cannot tell whether there is dirt in other squares
- **Unobservable:** The agent has no sensors at all

Single agent vs. Multiagent

- **Single agent:** An agent operates by itself in an environment.
  - E.g., solving crossword → single-agent, playing chess → two-agent
- *Which entities must be viewed as agents?*
  - Whether B's behavior is described as maximizing a performance measure whose value depends on A's behavior.
- **Competitive** vs. **Cooperative** multiagent environment
  - E.g., playing chess → competitive, driving on road → cooperative

Deterministic vs. Stochastic

- **Deterministic:** The next state of the environment is completely determined by the current state and the action executed by the agent.
  - E.g., the vacuum world $\rightarrow$ deterministic, driving on road $\rightarrow$ stochastic
- Most real situations are so complex that they must be treated as **stochastic**.

Episodic vs. Sequential

- **Episodic:** The agent's experience is divided into atomic episodes, in each of which the agent receives a percept and then performs a single action.
  - Quality of action depends just on the episode itself
  - Do not need to think ahead
- **Sequential:** A current decision could affect future decisions.
- For example,
  - Spotting defective parts on an assembly line vs. playing chess

Static vs. Dynamic

- **Static:** The environment is unchanged while an agent is deliberating.
  - E.g., crossword puzzles $\rightarrow$ static, taxi driving $\rightarrow$ dynamic
- **Semidynamic:** The environment itself does not change with the passage of time but the agent's performance score does
  - E.g., chess playing with a clock

Properties of Task environment

- **Discrete vs. continuous**
  - The distinction applies to the state of the environment, to the way time is handled, and to the agent's percepts and actions
  - E.g., the chess has a finite number of distinct states, percepts and actions; while the vehicles' speeds and locations sweep through a range of continuous values smoothly over time.
- **Known vs. unknown**
  - Known environment: the outcomes (or outcome probabilities if the environment is stochastic) for all actions are given.
  - Unknown environment: the agent needs to learn how it works to make good decisions.

Environments and their characteristics

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
Backgammon	Fully	Multi	Stochastic	Sequential	Static	Discrete
Taxi driving	Partially	Multi	Stochastic	Sequential	Dynamic	Continuous
Medical diagnosis	Partially	Single	Stochastic	Sequential	Dynamic	Continuous
Image analysis	Fully	Single	Deterministic	Episodic	Semi	Continuous
Part-picking robot	Partially	Single	Stochastic	Episodic	Dynamic	Continuous
Refinery controller	Partially	Single	Stochastic	Sequential	Dynamic	Continuous
Interactive English tutor	Partially	Multi	Stochastic	Sequential	Dynamic	Discrete

Examples of task environments and their characteristics.

Properties of Task environment

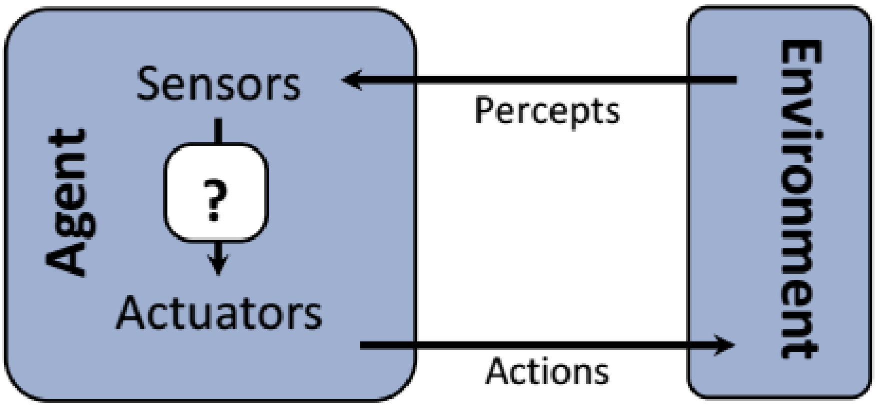
- The simplest environment: Fully observable, deterministic, episodic, static, discrete and single-agent.
- Most real situations: Partially observable, stochastic, sequential, dynamic, continuous and multi-agent.

Quiz: Properties of Task environment

- For each of the following activities, characterize its task environment in term of properties listed.
  - Playing a tennis match
  - Practicing tennis against a wall

The Structure of Agents

- *Agent Programs*
- *Simple Reflex Agents*
- *Model-based Reflex Agents*
- *Goal-based Agents*
- *Utility-based Agents*
- *Learning Agents*



The agent architecture

agent = architecture + program

- **Architecture:** some sort of computing device with physical sensors and actuators that this program will run on.
  - Ordinary PC, robotic car with several onboard computers, cameras, and other sensors, etc.
- The program has to be appropriate for the architecture.
  - Program: Walk action $\rightarrow$ Architecture: legs

The agent programs

- Input for Agent Program
  - Only the current percept
- Input for Agent Function
  - The entire percept sequence
  - The agent must remember all of them
- Implement the agent program as
  - A look up table (agent function)

The agent programs

- A trivial agent program: keep track of the percept sequence and index into a table of actions to decide what to do.

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 TABLE-DRIVEN-AGENT program is invoked for each new percept and returns an action each time. It retains the complete percept sequence in memory.

The agent programs

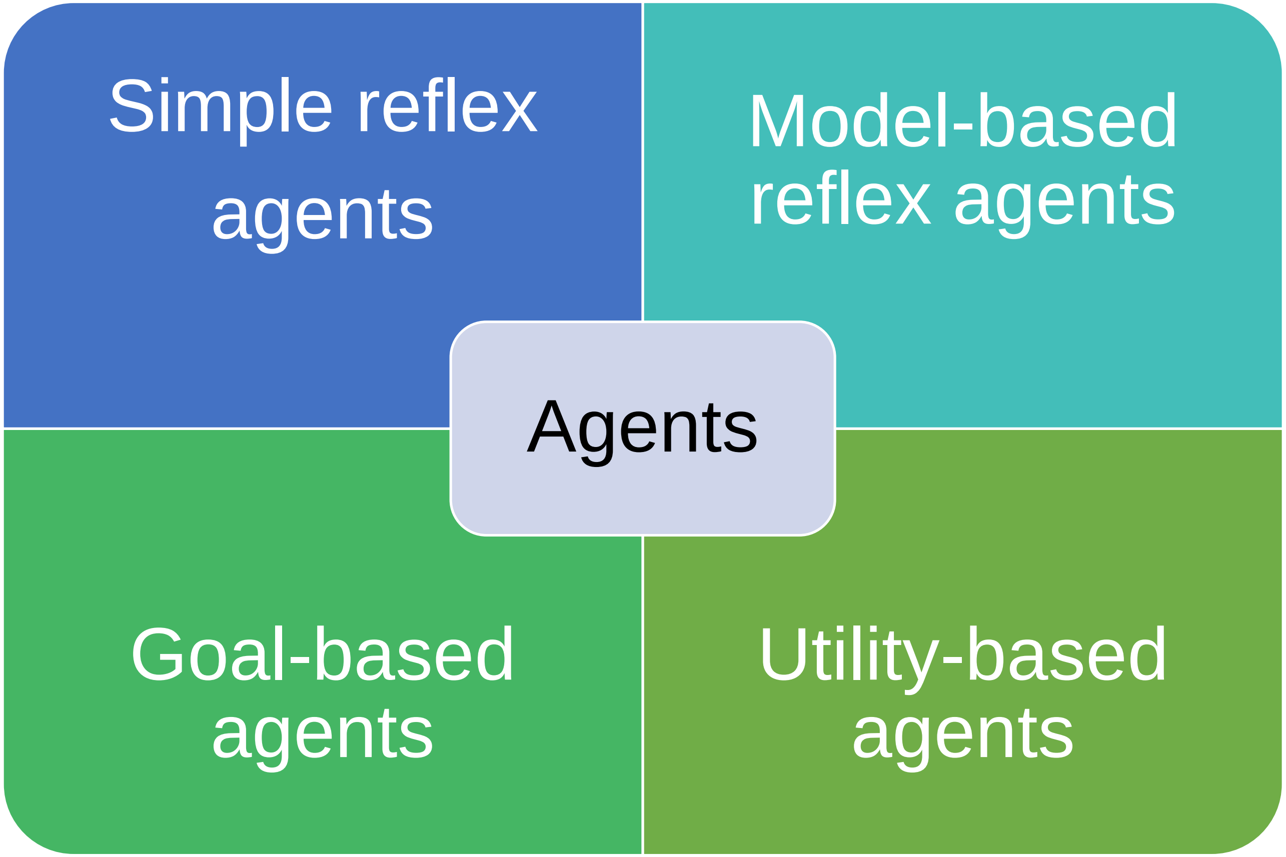
- $P =$ the set of possible percepts
- $T =$ lifetime of the agent
  - I.e. the total number of percepts it receives
- The size of the look up table is $\sum_{t=}^{T} |P|^t$
- For example, consider playing chess
  - $P = 10,\ T = 150 \rightarrow$ A table of at least 10^{150} entries

- Despite of huge size, look up table does what we want

The key challenge of AI

- Find out how to write programs that, to the extent possible, produce **rational behavior** from a small amount of code rather than a large amount of table entries
  - E.g., a five-line program of Newton’s Method vs. huge tables of square roots,…

Types of agent programs

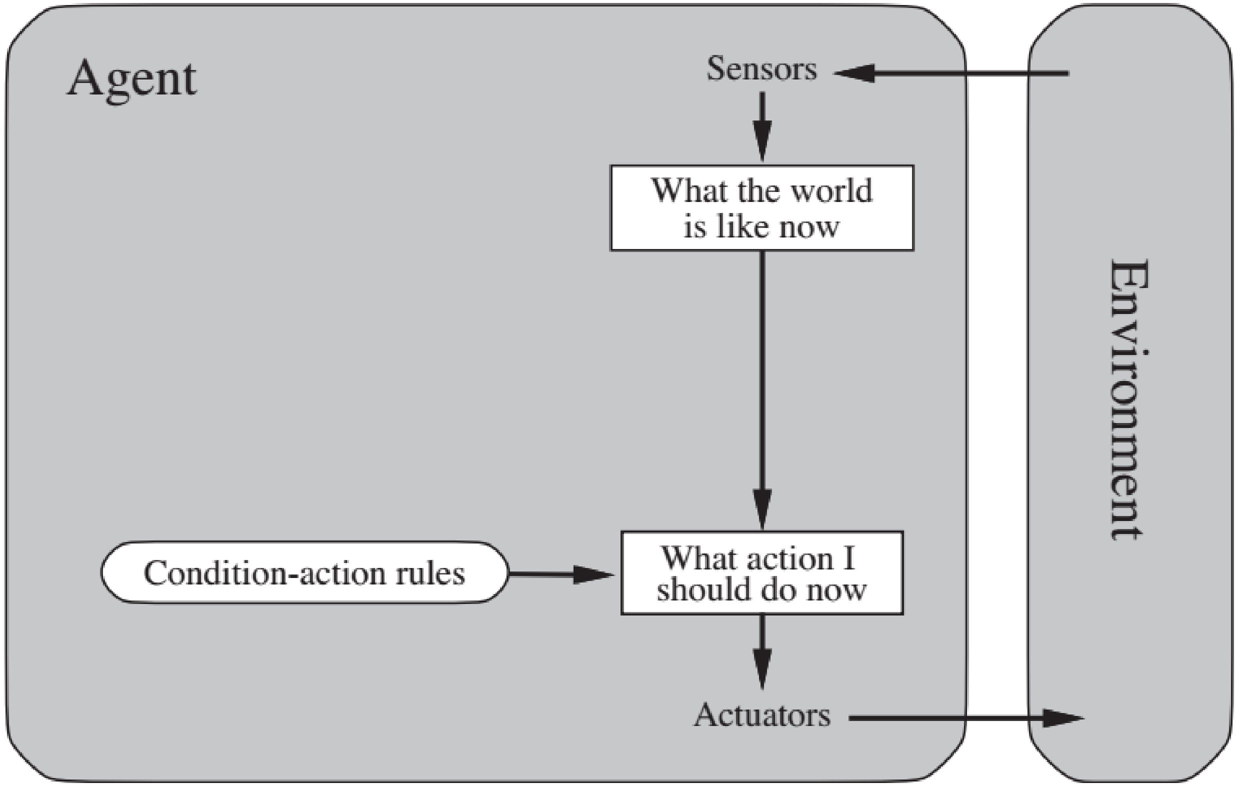


Simple reflex agents

- The simplest kind of agent, but of limited intelligence
- Select actions based on the **current percept**, ignoring the rest of the percept history
- The connection from percept to action is represented by **condition-action rules**.

IF *current percept* **THEN** *action*

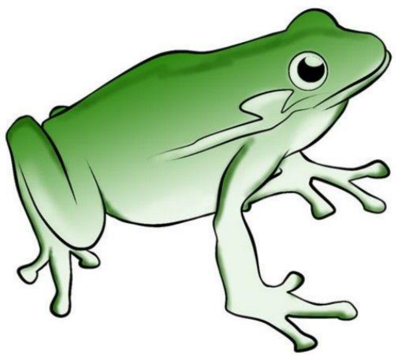
  - E.g., IF *car-in-front-is-braking* THEN *initiate-braking*.
- Limitations
  - Knowledge sometimes cannot be stated explicitly → low applicability
  - *Work only if the environment is fully observable*



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

A Simple reflex agent in nature

percepts
(size, motion)



Action: SNAP or AVOID or NOOP

RULES:

(1) If small moving object, then activate SNAP

(2) If large moving object, then activate AVOID and inhibit SNAP

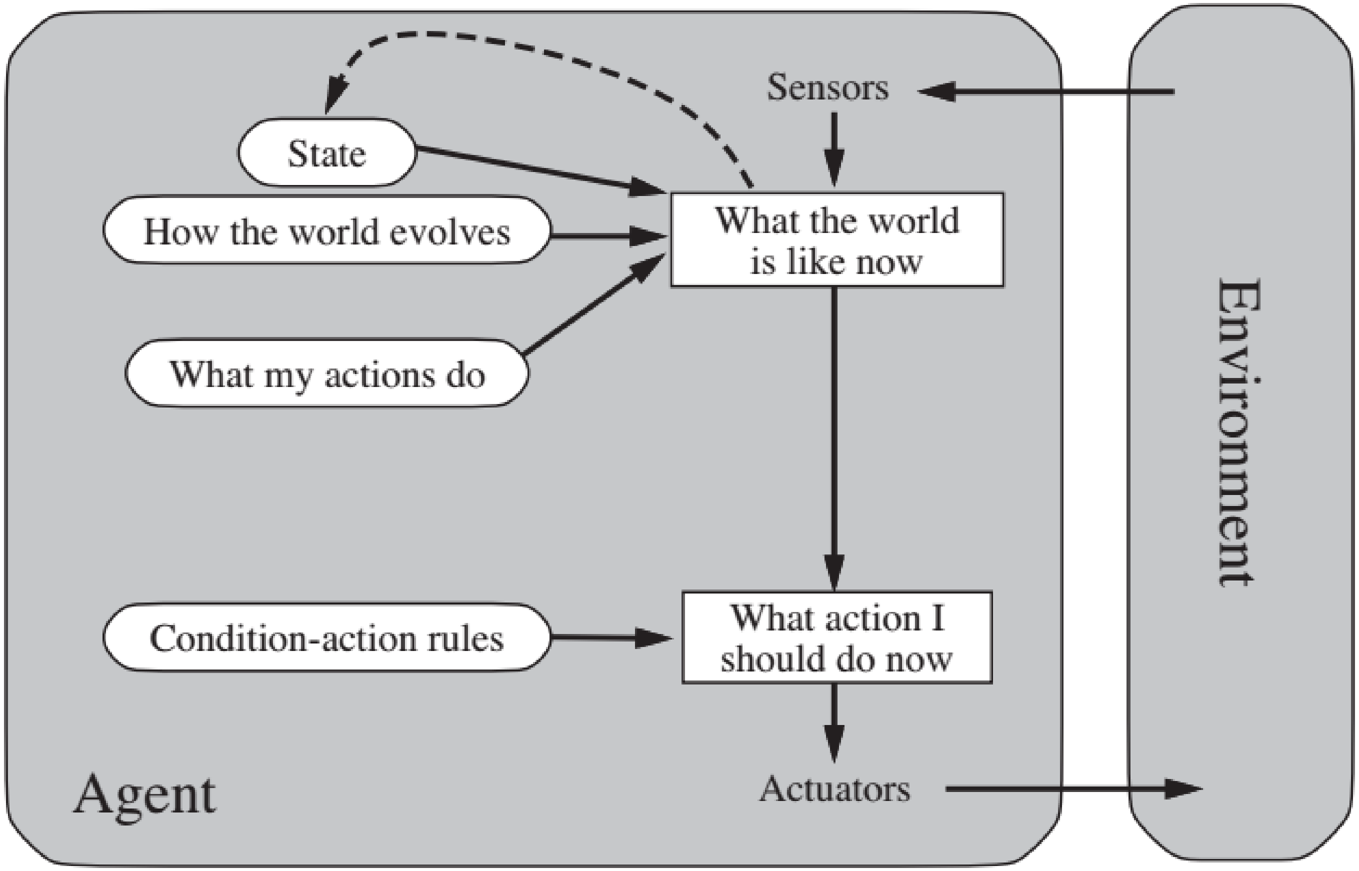
ELSE (not moving) then NOOP

Needed for completeness

Model-based reflex agents

- Partially observability → the agent has to keep track of an internal state
  - Depend on the percept history and reflect some of the unobserved aspects
  - E.g., driving a car and changing lane
- The agent program updates the internal state information as time goes by by encoding two kinds of knowledge
  - How the world evolves independently of the agent
  - How the agent's actions affect the world

model of the world



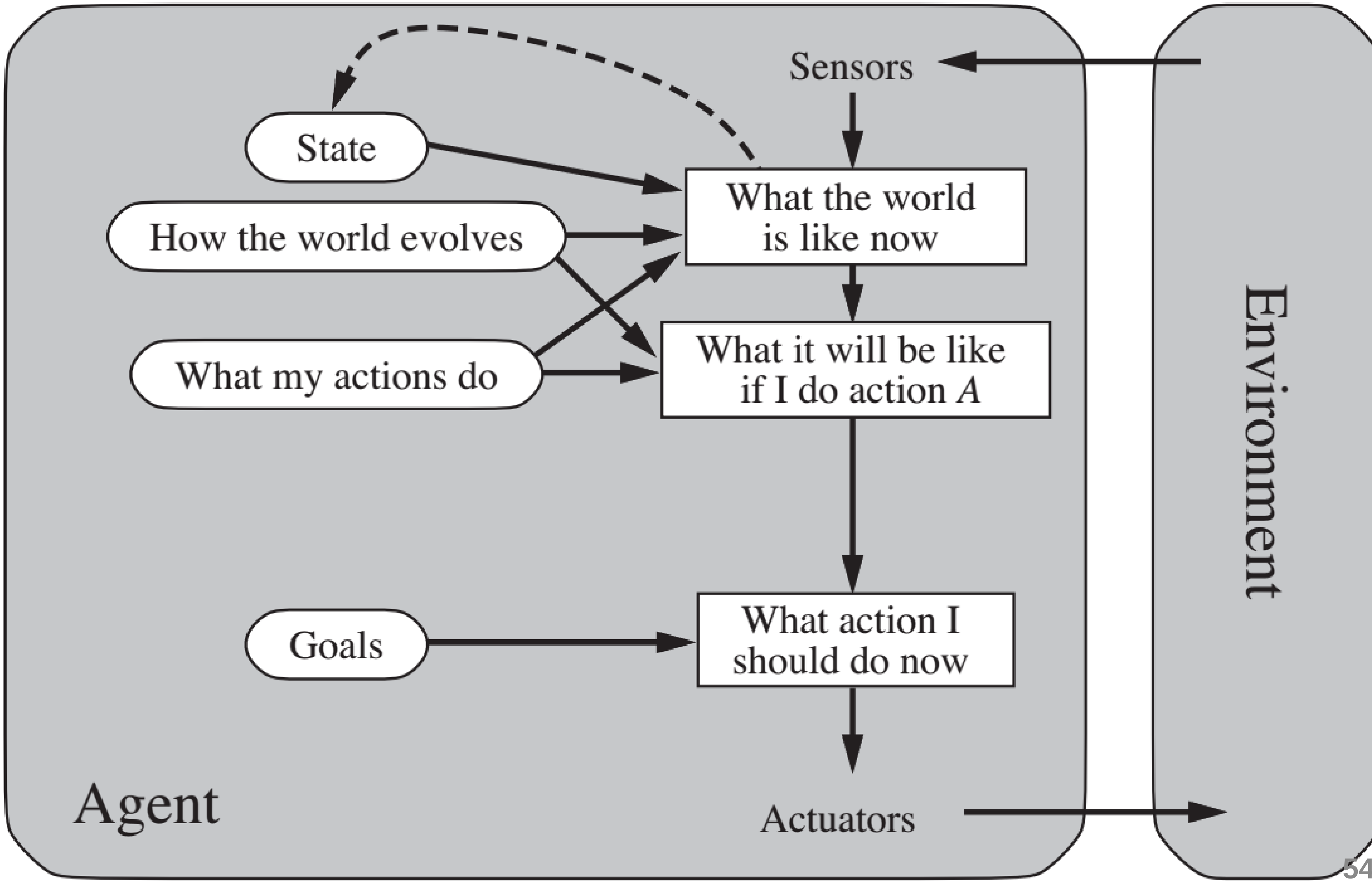
IF	THEN
Saw an object ahead and turned right, and it’s now clear ahead	Go straight
Saw an object ahead and turned right, and object ahead again	Halt
See no object ahead	Go straight
See an object ahead	Turn randomly

Example table agent with internal state

Goal-based agents

- Current state of the environment is always not enough
- The agent further needs some sort of goal information that describes situations that are desirable.
  - E.g., at a road junction, the taxi can turn left, turn right, or go straight on, depending on where the taxi is trying to get to.
- Less efficient but more flexible
  - Knowledge that supports the decisions is represented explicitly and can be modified

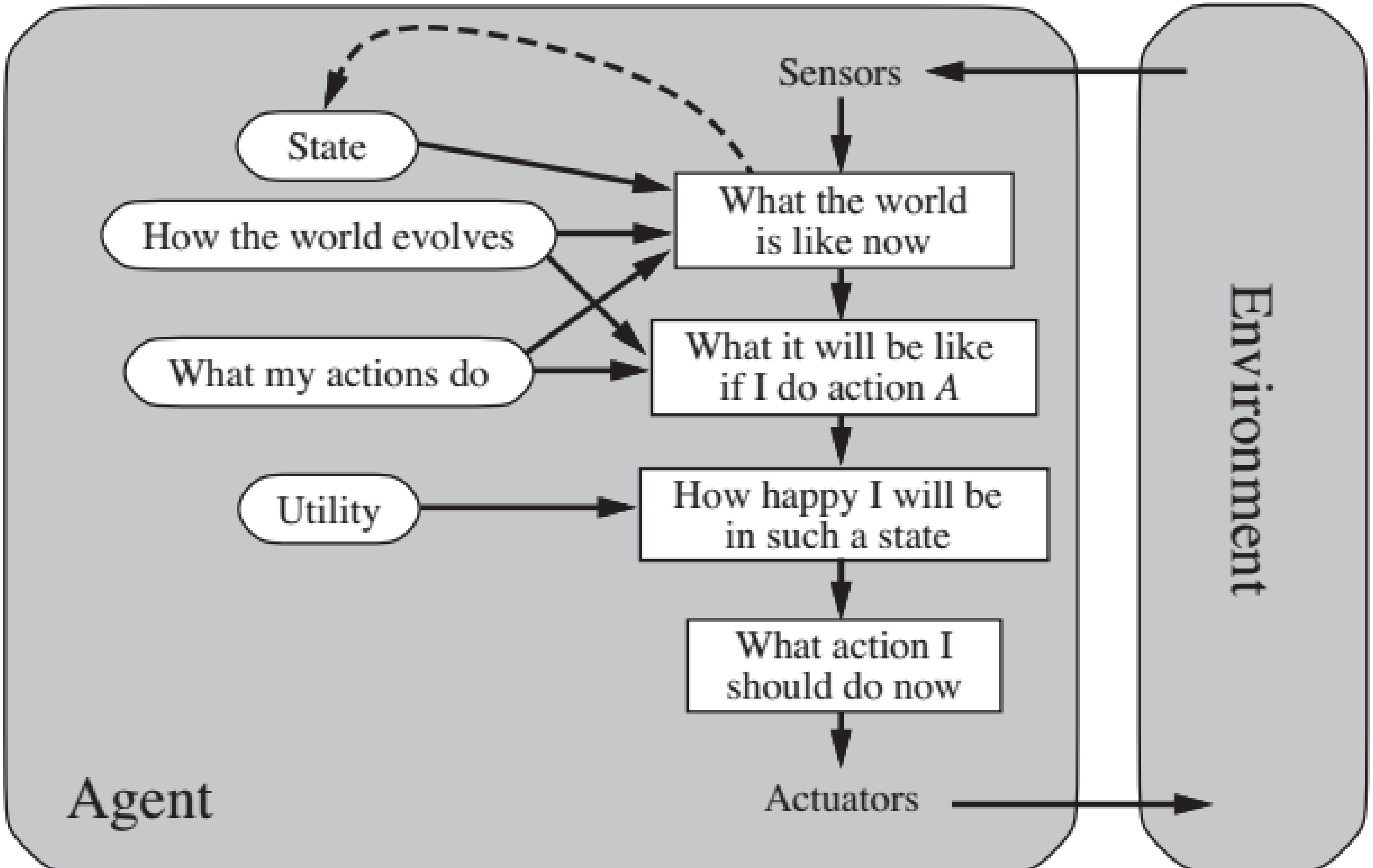
Goal-based agents



Utility-based agent

- Goals alone are not enough to generate high-quality behavior in most environments
  - Many action sequences to get the goals, some are better and some worse
  - E.g., go home: Vinasun taxi or Grab car?
- An agent's **utility function** is essentially an internalization of the performance measure.
  - Goal $\rightarrow$ success, utility $\rightarrow$ degree of success (how successful it is)
  - If state A is more preferred than others then A has higher utility

Utility-based agent



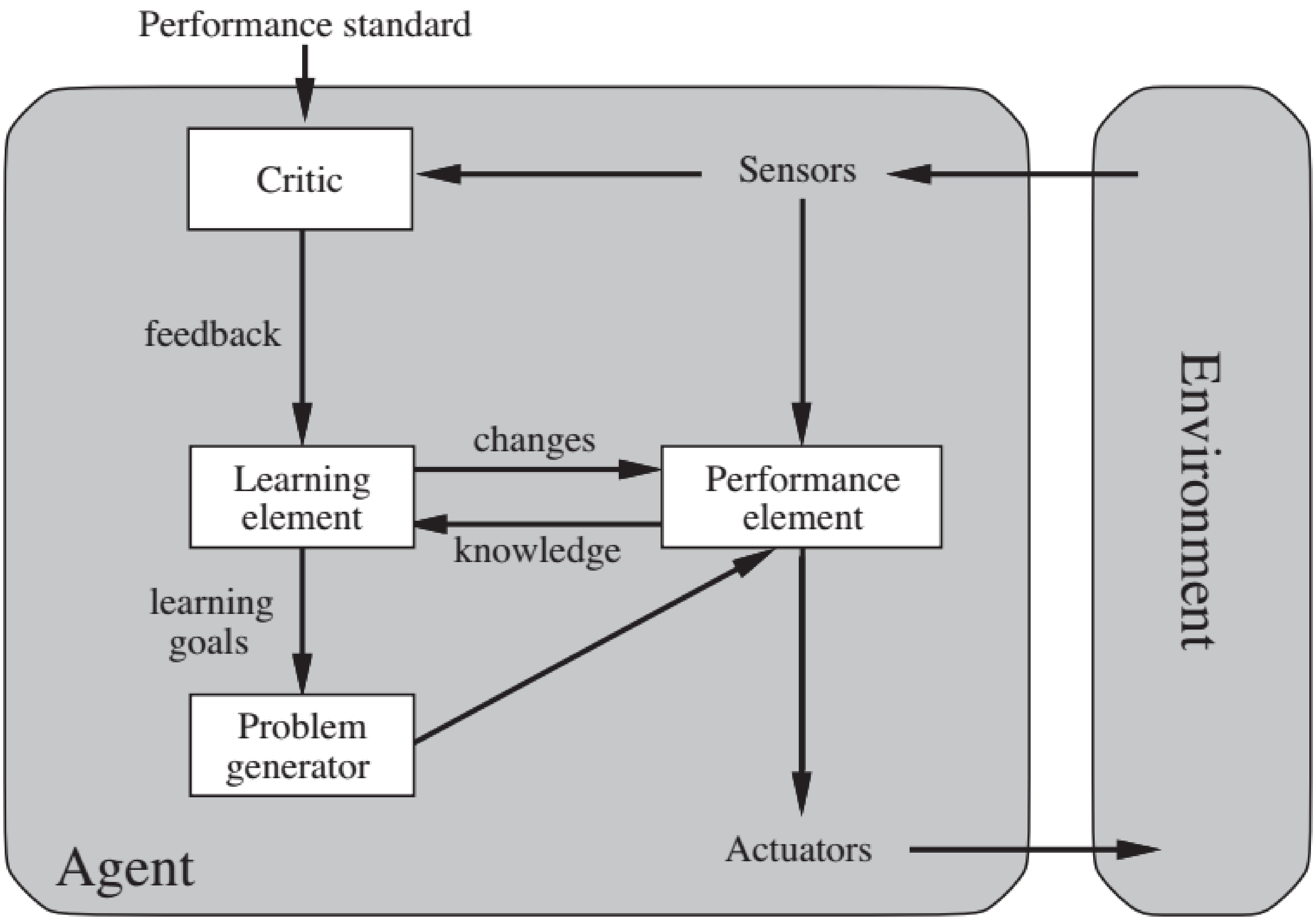
Utility-based agent: Advantages

- When there are conflicting goals
  - Only some of which can be achieved, e.g., speed and safety
  - The utility function specifies the appropriate tradeoff.
- When there are several goals that the agent can aim for
  - None of which can be achieved with certainty
  - The utility weights the likelihood of success against the importance of the goals.
- The rational utility-based agent chooses the action that maximizes the **expected utility** of the action outcomes

Learning agents

- After an agent is programmed, can it work immediately?
  - No, it still need teaching
- Once an agent is done, what can we do next?
  - Teach it by giving it a set of examples
  - Test it by using another set of examples
- We then say the agent **learns** → **learning agents**

Learning agents



Learning agents

- A learning agent is divided into four conceptual components
  1. Learning element → Making improvement
  2. Performance element → Selecting external actions
  3. Critic → Tells the Learning element how well the agent is doing with respect to fixed performance standard. (Feedback from user or examples, good or not?)
  4. Problem generator → Suggest actions that will lead to new and informative experiences

Learning in intelligent agents is a process of ***modification of each component*** *of the agent to bring the components into* ***closer agreement*** *with the available feedback information, thereby improving the overall performance of the agent.*

Learning agents: An example

- Performance element
  - Whatever collection of knowledge and procedures the taxi has for selecting its driving actions (may be further modified)
- Critic
  - Observe the world and pass information to the learning element
  - E.g., quick left turn across three lanes of traffic $\rightarrow$ shocking language used by other drivers observed $\rightarrow$ bad action
- Learning element
  - Formulate new rules from the experience told by the critic
  - E.g., a new rule for the above bad action
- Problem generator
  - Identify certain areas of behavior in need of improvement and suggest experiments, e.g., trying out the brakes on different road surfaces under different conditions

Quiz: Learning agents

- Give an example of learning rational agent following four conceptual elements

THE END