IT3160E
Introduction to Artificial Intelligence

Chapter 2 – Intelligent agents

Lecturer:
Muriel VISANI
Acknowledgements:
Le Thanh Huong
Tran Duc Khanh
Department of Information Systems
School of Information and Communication Technology - HUST

Content of the course

- Chapter 1: Introduction
- Chapter 2: Intelligent agents
- Chapter 3: Problem Solving
  - Search algorithms, adversarial search
  - Constraint Satisfaction Problems
- Chapter 4: Knowledge and Inference
  - Knowledge representation
  - Propositional and first-order logic
- Chapter 5: Uncertain knowledge and reasoning
- Chapter 6: Advanced topics
  - Machine learning
  - Computer Vision

Outline

- Chapter 2: Intelligent agents
  - Purpose of intelligent agents
  - Definitions: agents and environments
  - Exercises
  - PEAS
  - Environment types
  - Agent types
  - Knowledge Bases for agents
  - A few words about multi-agent planning
  - Summary

Goal of this Chapter

Goal	Description of the goal or output requirement	Output division/ Level (I/T/U)
M1	Understand basic concepts and techniques of AI	1.2

Chapter 2: Intelligent agents

Purpose of intelligent agents

Purpose of intelligent agents

- The purpose of intelligent agents is to **act rationally**

	Mimic humans	*Reach an ideal / rationality**
Reasoning	Think like humans	Thinking rationally
Behaviours	Act like humans	Acting rationally

** Different from mimicking humans, as humans sometimes make mistakes*

Chapter 2: Intelligent agents

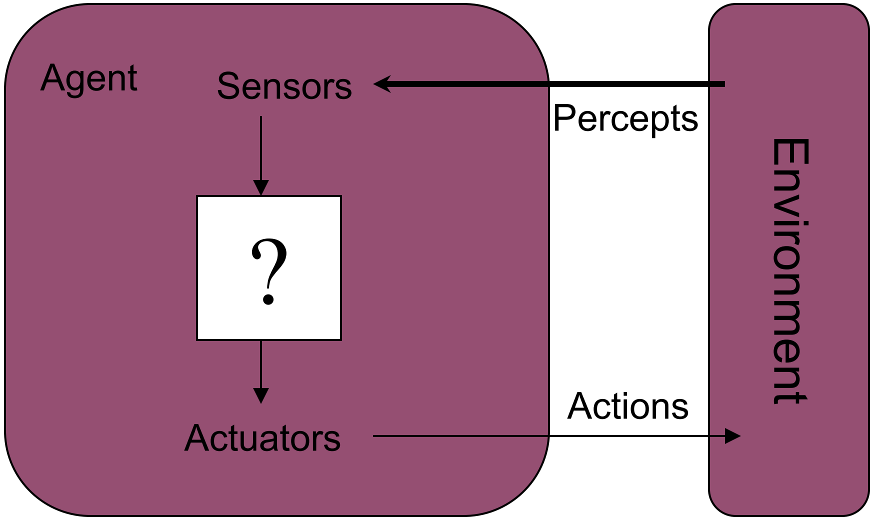
Definitions: agents and environments

Definitions: agents and environments

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

- The agent function maps from percept histories to actions:

  $$[\, f: \mathcal{P}^* \rightarrow \mathcal{A} \,]$$



Definitions: agents and environments

- Example 1: human agent
  - Sensors: eyes, ears, …
  - Actuators: hands, legs, mouth, …

- Example 2: robotic agent (e.g., Aishimo)
  - Sensors: camera, infrared range finders
  - Actuators: various motors

- Example 3: software agent
  - Sensors: keypad, file uploader, network packet receiver…
  - Actuators: screen, file writer, network packet sender…

Definitions: agents and environments

- The agent's percept sequence is the complete history of percepts received by the agent

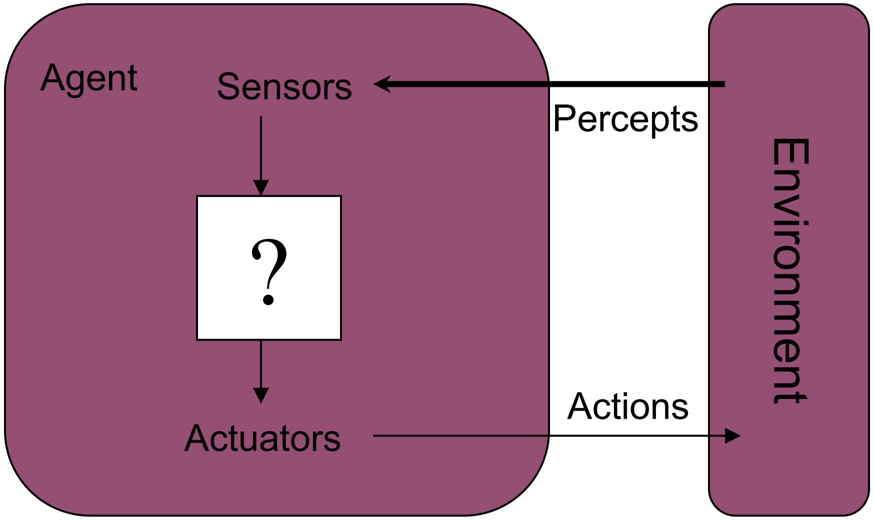
- In general, an agent acts based on its percept sequence
  - But, of course, it cannot act on something it did not perceive

- The "rationality" of an agent's behavior depends a lot on its environment, and on its percept sequence
  - Some environments are more difficult than others
  - Percepts in a percept sequence might be "contradictory"

Definitions: agents and environments

- The agent program runs on the physical architecture to produce the agent function

agent = architecture + program



Recall: agent function

Agent function based on conditional table

- One (basic) way to implement the agent function is to map any given percept sequence to an action

Function TABLE-DRIVEN-AGENT(*input_percept*)

returns an action

static: *current_percept_sequence:* initially empty

table: a table of actions, indexed by all possible percept sequences, initially fully specified

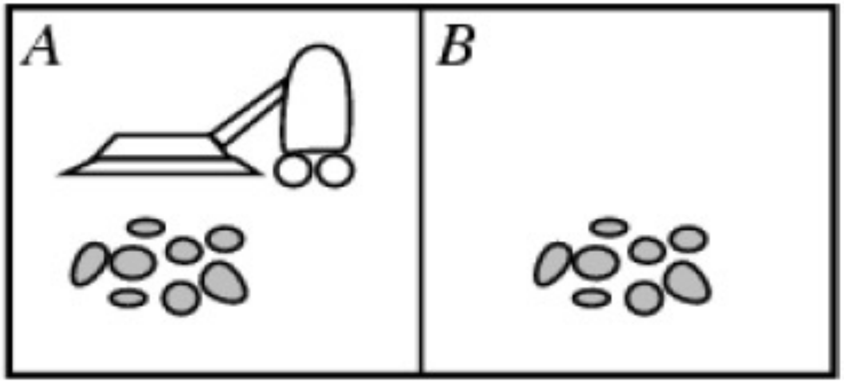
Append *input_percept* to the end of *current_percept_sequence*

action ← LOOKUP(*current_percept_sequence, table*)

Return *action*

Drawback: huge table!

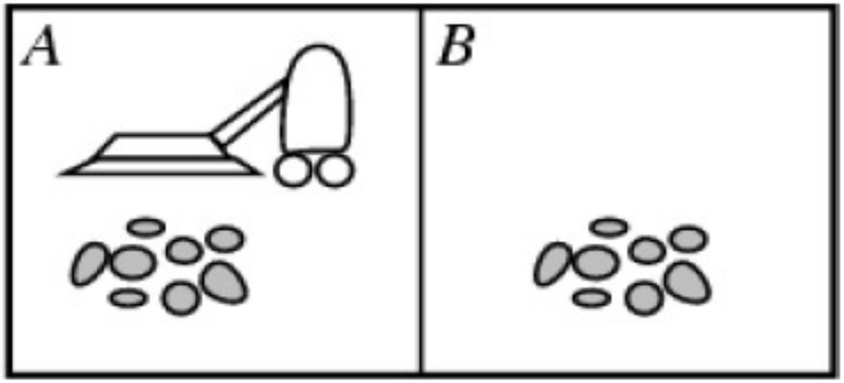
Vacuum-cleaner world



- Percepts:
  - Location (A or B)
  - Cleanliness (clean or dirty)
- Actions:
  - Left
  - Right
  - Suck
  - NoOp

Percept sequence	Action
[A, clean]	Right
[A, dirty]	Suck
[B, clean]	Left
[B, dirty]	Suck
[A, dirty][A, clean]	Right
[B, clean][A, dirty]	Suck

Vacuum-cleaner world



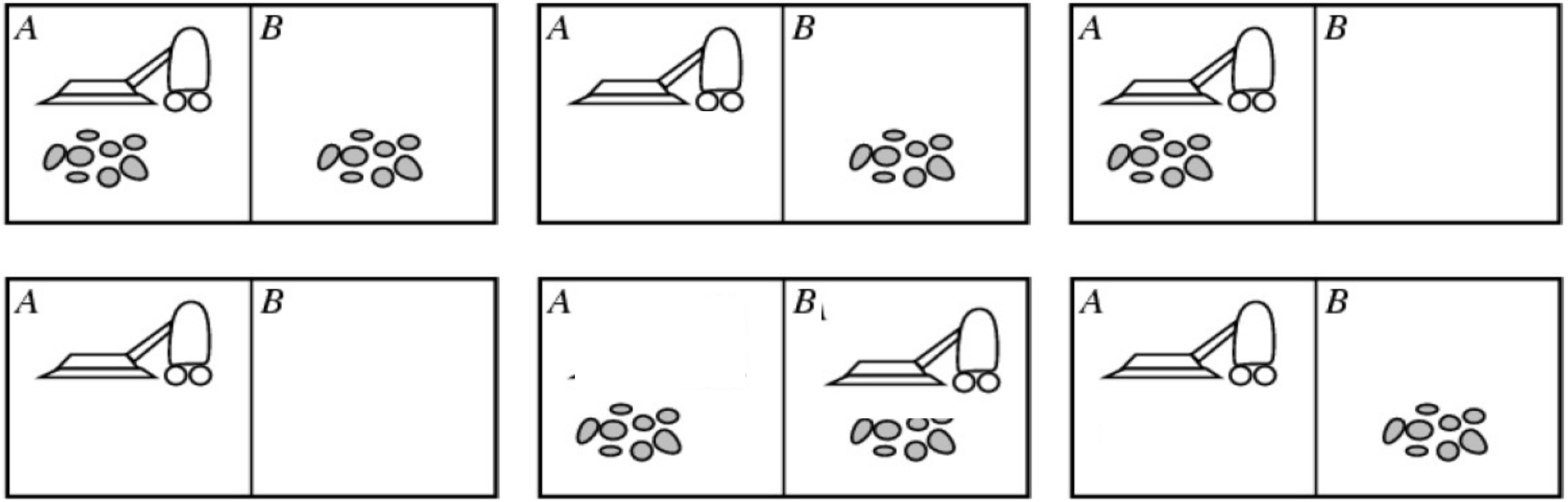
- **Quiz:**
  - What were the actions before Right and Suck in the two last rows of the table?

  Answer: ……

Percept sequence	Action
[A, clean]	Right
[A, dirty]	Suck
[B, clean]	Left
[B, dirty]	Suck
[A, dirty][A, clean]	Right
[B, clean][A, dirty]	Suck

Notion of state

- More generally, many things are often not directly completely observable but can be "guessed" from the percept history
  - It is the case of the **state:**
    - Set of propositions / diagram that completely describes the environment
    - Examples of possible states for the vacuum-cleaner world:



etc…

Vacuum-cleaner world

- Let us consider the following agent function

Funtion Reflex-Vacuum-Agent([*position*, *cleanliness*]) **returns** action

If *cleanliness* = “Dirty” **then return** *Suck*

Else if *position* = “A” **then return** *Right*

Else if *position* = “B” **then return** *Left*

End Function

- **Question**: Does the agent act reasonably?
    - Answer: ….

Rational agent

- A **rational agent** is one that does the *right* thing = the action that makes the agent most successful

- Rationality actually depends on:
  - Performance measure
  - The agent's prior knowledge of the environment
    - An agent cannot be judged irrational if it did not have the correct information to make the best decision
  - The agent's percept sequence to date
    - An agent cannot be judged irrational if it did not have the correct information to make the best decision
  - The actions that the agent can perform
    - An agent cannot be judged irrational if it is not enabled to take the best action

Rationality ≠ omniscience ≠ perfection

Rational agent

- The Performance measure is the criterion for judging of the success of an agent’s behavior
  - *E.g.* performance measures for a vacuum-cleaner agent
    - Amount of dirt cleaned up
    - Amount of time needed to clean
    - Amount of electricity consumed
    - Amount of noise generated
    - A combination of all the above

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 given the built-in knowledge that the agent has
    - *e.g.* the knowledge about its environment

- Information gathering is an important part of rationality
  - *e.g.* checking if the square is clean before (possibly) sucking

Chapter 2: Intelligent agents

Exercises

Vacuum-cleaner exercise

- Let us assume that:
  - The performance measure awards 1 point for each clean square at each time step, over 1000 time steps
  - The "geography" of the environment is known (A&B), but the dirt distribution and the initial agent's location are not
  - Clean squares stay clean
    - No dirt is added during cleaning
  - Sucking a dirty square definitely cleans it
  - If a Left or Right action would move the agent outside the environment, then the agent remains where it is (NoOp)
- Prove that, under these circumstances, the agent is rational

Vacuum-cleaner exercise

- **Proof:**
  - Trivial because of the basic environment
  - Tip: all you need to do, is to:
    - Consider every possible environment **state** (def. on slide 15)
    - Show that, for this environment state, no other agent can do better

  →If no agent can do better => the agent is rational

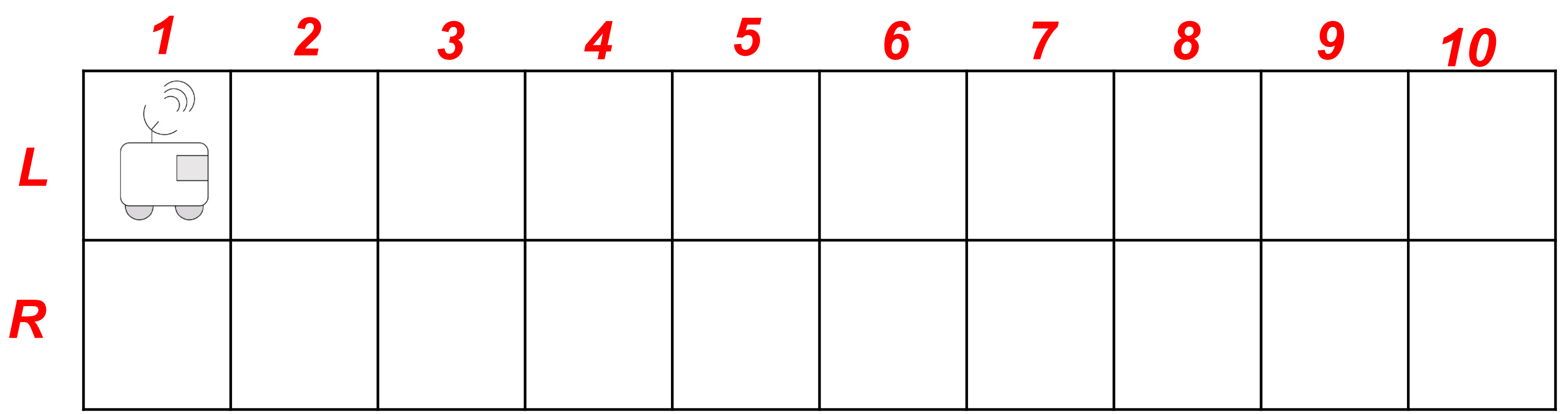
- Now, prove it by yourself!

Vacuum-cleaner exercise

- **Proof**:

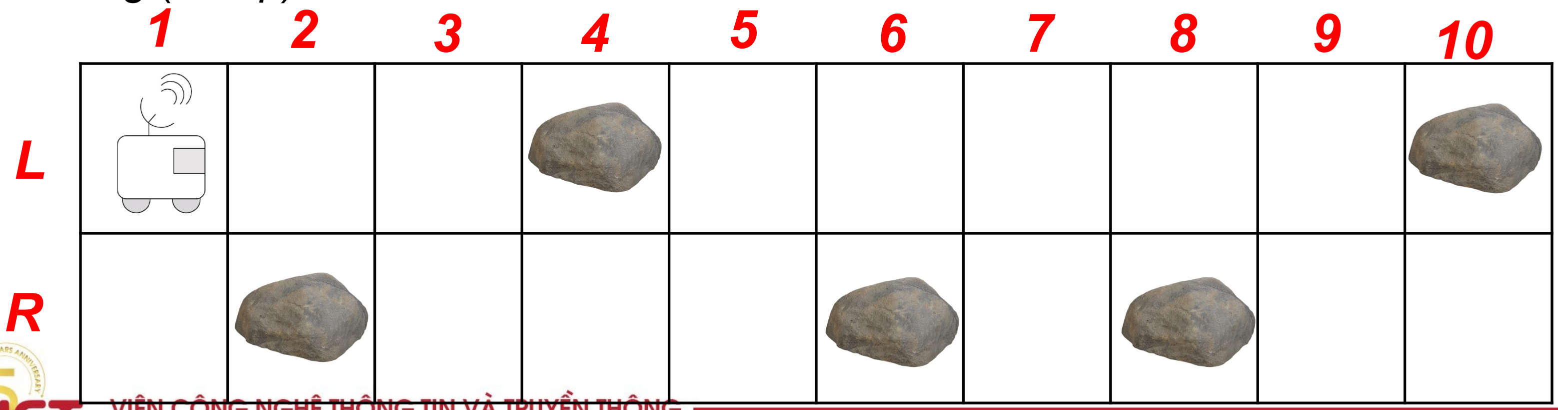
Toy exercise

- A vehicle agent is supposed to move forward, one cell at a time, by staying on its lane
- A lane is composed of 2 cells (in width: L and R), and 10 cells (in length)
- The agent starts on the first left cell of the lane (L1)



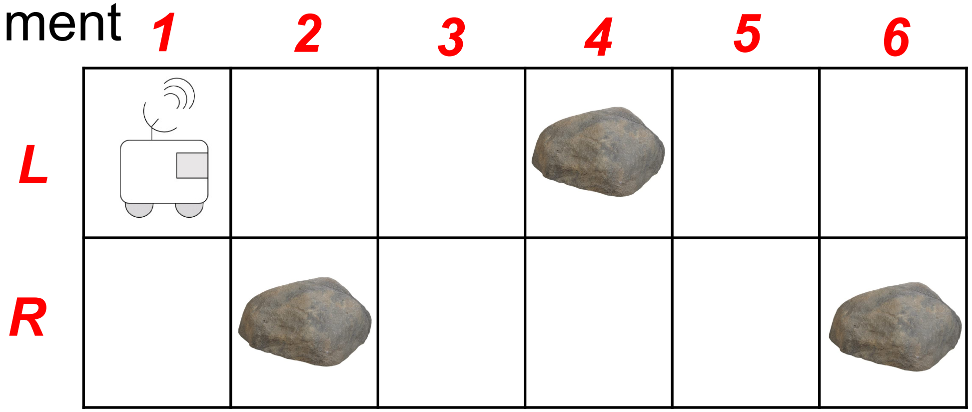
Toy exercise

- There are obstacles on the way; the agent must avoid them
  - The environment is set so that there cannot be 2 obstacles blocking the road in diagonal
    - For instance, obstacles in L3 and R4
- The agent perceives its current location (*e.g. location* = “L1”)
  - The agent keeps a history of its successive locations
- The agent only perceives obstacles that are in the cell in front of it
  - If there is an obstacle in front: obstacle=1 ; otherwise obstacle=0
- The agent can only perform 4 actions:
  - Move forward (*Forward)*
  - Move to the left, if it is in a right cell (*Left)*
  - Move to the right, if it is in a left cell (*Right)*
  - Do nothing (*NoOp)*



Toy exercise

- Give the most rational, **complete** percept sequence / actions of the agent in this environment



Percept sequence	Action

Toy exercise

- **Question 1**: depending on its environment, can the agent always reach the arrival line (last column)?
  - Answer:

Toy exercise

- **Question 2**: In the following table, put some possible percept sequences / actions (invent your own percept sequences), with:
  - A variable size of the percept sequence (sometimes one percept, sometimes two, etc)
    - According to what is required in the table
  - In any case, in the table, only 1 action should be shown (the current action)
    - Already pre-filled in the table
    - Any possible action is represented twice in the table
  - N.B. Your table would normally be different from your neighbours' table

Percept sequence	**Action**
<Possible sequence of length 1>	*Right*
<Possible sequence of length 3>	*Right*
<Possible sequence of length 1>	*Forward*
<Possible sequence of length 2>	*Forward*
<Possible sequence of length 1>	*Left*
<Possible sequence of length 3>	*Left*
<Possible sequence of length 2>	*NoOp*
<Possible sequence of length 4>	*NoOp*

Toy exercise

- **Question 2**: In the following table, put some possible percept sequences / actions (invent your own percept sequences), with:
  - A variable size of the percept sequence (sometimes one percept, sometimes two, etc)
    - According to what is required in the table
  - In any case, in the table, only 1 action should be shown (the current action)
    - Already pre-filled in the table
    - Any possible action is represented twice in the table
  - N.B. Your table would normally be different from your neighbours' table

Percept sequence	Action
	Right
	Right
	Forward
	Forward
	Left
	Left
	NoOp
	NoOp

Toy exercise

- Write an adapted agent function in pseudo-code
    - Try to make the agent as rational as possible
        - By taking into account its sequence of positions, not only its current position
    - Tips:
        - Take as an example the agent function given in slide 16
        - Let’s call *position* the current position perceived by the agent, as a string of characters (*e.g.* position = “R2” => position[0]=‘R’; position[1]=‘2’)
        - Let’s call *position_sequence* the sequence of positions of the agent, as a table of strings of characters (*e.g.* position_sequence[0,:] = “L1” if the agent starts at the top-left corner)
        - Let’s call *obstacle* a Boolean variable stating if there is an obstacle in front of the agent, or not.
        - Answer:

Toy exercise

- Propose at least 3 adapted performance measures for this problem
  - Answer:
    - …

Chapter 2: Intelligent agents

PEAS

PEAS

- 4 factors should be considered when designing an automated agent:
  - **P**erformance measure
  - **E**nvironment
  - **A**ctuators
  - **S**ensors

PEAS - automated taxi driver

- Performance measure: Safe, fast, legal, comfortable trip, maximize profits, …
- Environment: Roads, other traffic, pedestrians, weather, …
- Actuators: Steering wheel, accelerator, brake, signal, horn, …
- Sensors: Cameras, sonar, speedometer, GPS, odometer, engine sensors, keyboard, …

PEAS - automated taxi driver

- STANLEY driverless robotic car won the DARPA Grand Challenge (2005)

PEAS - Spam Filtering Agent

- Performance measure: spam block, false positives, false negatives
- Environment: email client or server
- Actuators: mark as spam, place messages in the inbox
- Sensors: emails (possibly across users), traffic, etc.

PEAS – Mushroom picking agent

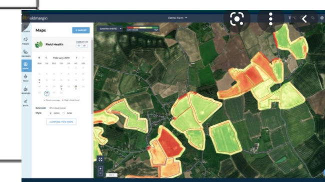
- This agent is automatically given baskets of mushrooms, picks mushrooms one by one, and puts them in the selling package

Performance Measure	Environment	Actuators	Sensors
Percentage of good mushrooms in correct bins processing time	Conveyor belt with mushrooms, bins	Jointed arm and hand	camera, joint angle sensors

Other PEAS examples

[From the reference book from Russel and Norvig]

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 processing time	Downlink from orbiting satellite	Display of scene categorization	Color pixel arrays
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



Chapter 2: Intelligent agents

Environment types

Environment types

- **Fully observable** (vs. partially observable): An agent's sensors give it access to the complete state of the environment at each point in time.
  - The environment is effectively and fully observable if the sensors detect all aspects that are relevant to the choice of action
- **Deterministic** (vs. stochastic): The next state of the environment is completely determined by:
  - the current state
  - the action executed by the agent
  - If the next state of the environment also depends on other factors/agents that are not totally predictable, then the environment is stochastic
- **Single agent** (vs. multi-agent): An agent operating by itself in an environment.
  - For multi-agents: competitive *vs.* cooperative

Do not confuse between an agent (another taxi agent) and an object in the environment (obstacle)

  - Agents are the ones adapting their behavior to what is going on in the environment

Environment types

- Static (vs. dynamic): The environment is unchanged during the time where the agent is making its next decision.
  - If the environment does not change with time, but the agent's performance score does *(e.g.* cases where the agent has time limits), then we say the environment is **semi-dynamic**
- Discrete (vs. continuous): A limited number of distinct, clearly defined percepts and actions
  - For example, the vacuum cleaner environment has a finite number of distinct percepts and actions.
  - Another example of discrete percepts: the gears of a semi-automatic motorbike: N, 1, 2, 3, 4
  - On the other hand, taxi driving is a continuous environment problem (possibly infinite number of percepts and actions)
- Episodic (vs. sequential): The agent's experience is divided into atomic "episodes" (percept + action).
  - In episodic environments, the next episode does not depend on the actions taken in previous episodes
  - Episodic environments are simpler: the agent does not need to think ahead

Quiz

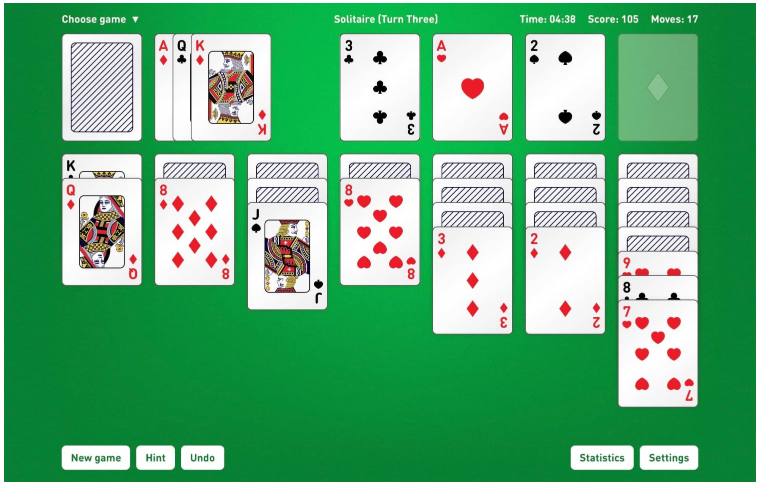
More characteristics of the environment

- Known (vs. unknown): refers to the knowledge of the agent regarding its environment
  - Strictly speaking, not an environment type…
  - In a known environment, the agent knows the outcomes (or outcome probabilities) for all of its actions
    - For instance, a card playing agent knows the rules of the game
  - If the environment is unknown, the agent will have to learn how it works in order to make good decisions.

More characteristics of the environment

Known ≠ observable

- For instance, a solitaire card playing agent:
  - knows the rules of the game -> environment is **known**
  - But cannot see all the cards in the deck -> environment is **partially observable**



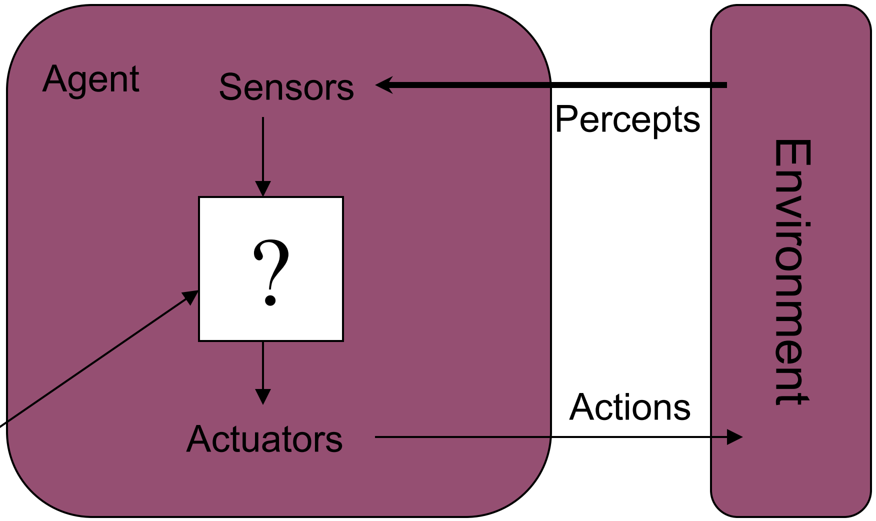
- Another example where the environment is **fully observable**, yet **unknown**:
  - Let’s say I’m the agent and I’m playing a new video game without having read the instructions. I see the entire game state, but I don’t know what the buttons do until I try them…

Chapter 2: Intelligent agents

Agent types

Agent types

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

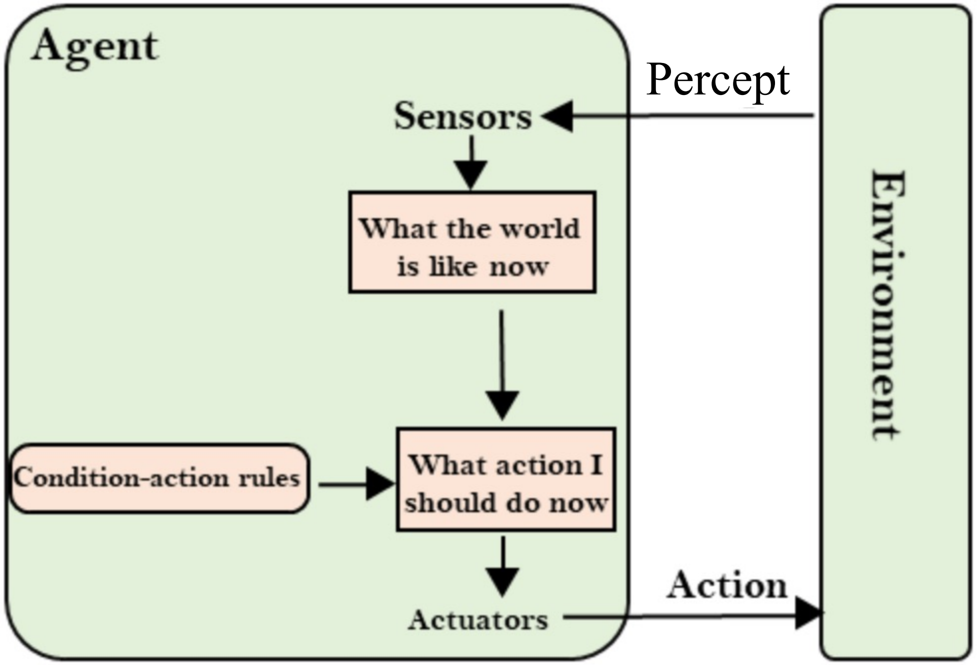


What is inside defines the type of agent

Recall: agent function

Simple reflex agent

- These agents select actions on the basis of the *current* percept ONLY
- They **ignore** the rest of the **percept history**
  - e.g. If I am hungry, then I eat.
  - They are of limited intelligence
- Example: simple vacuum
  - Recall from slide 16



Funtion Reflex-Vacuum-Agent([*position*, *cleanliness*]) **returns** action

If *cleanliness* = “Dirty” **then return** *Suck*

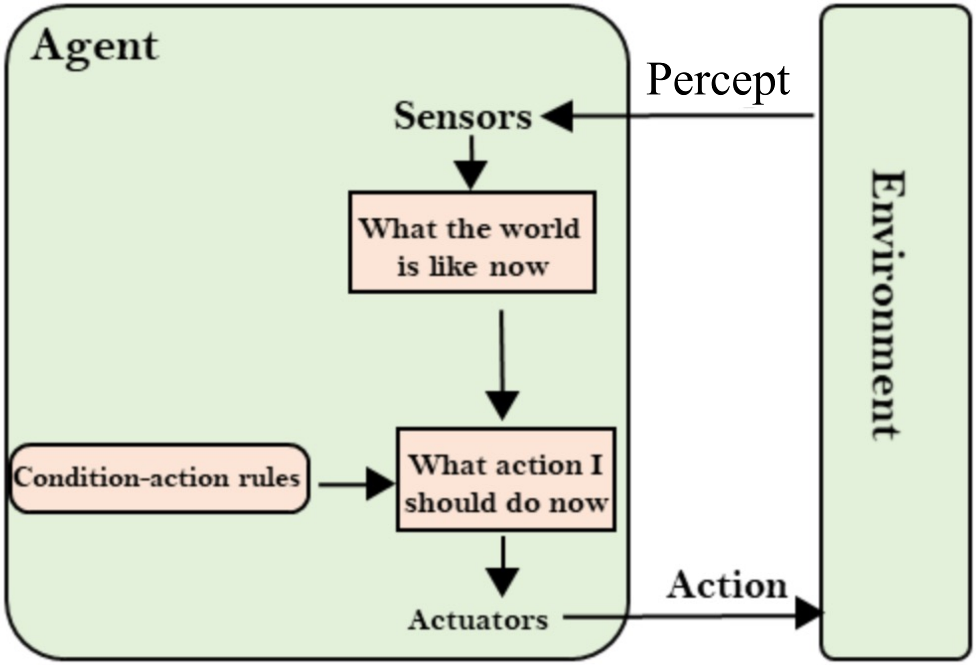
Else if *position* = “A” **then return** *Right*

Else if *position* = “B” **then return** *Left*

End Function

Simple reflex agent

- Often
  - The rule matches the agent’s state with the action
  - The current state is ”guessed” (interpreted) from the current percept



- General pseudo-code:

Function SIMPLE-REFLEX- AGENT(percept) **returns** an action

  static: rules, a set of condition-action rules

  state ← INTERPRET-INPUT(percept)

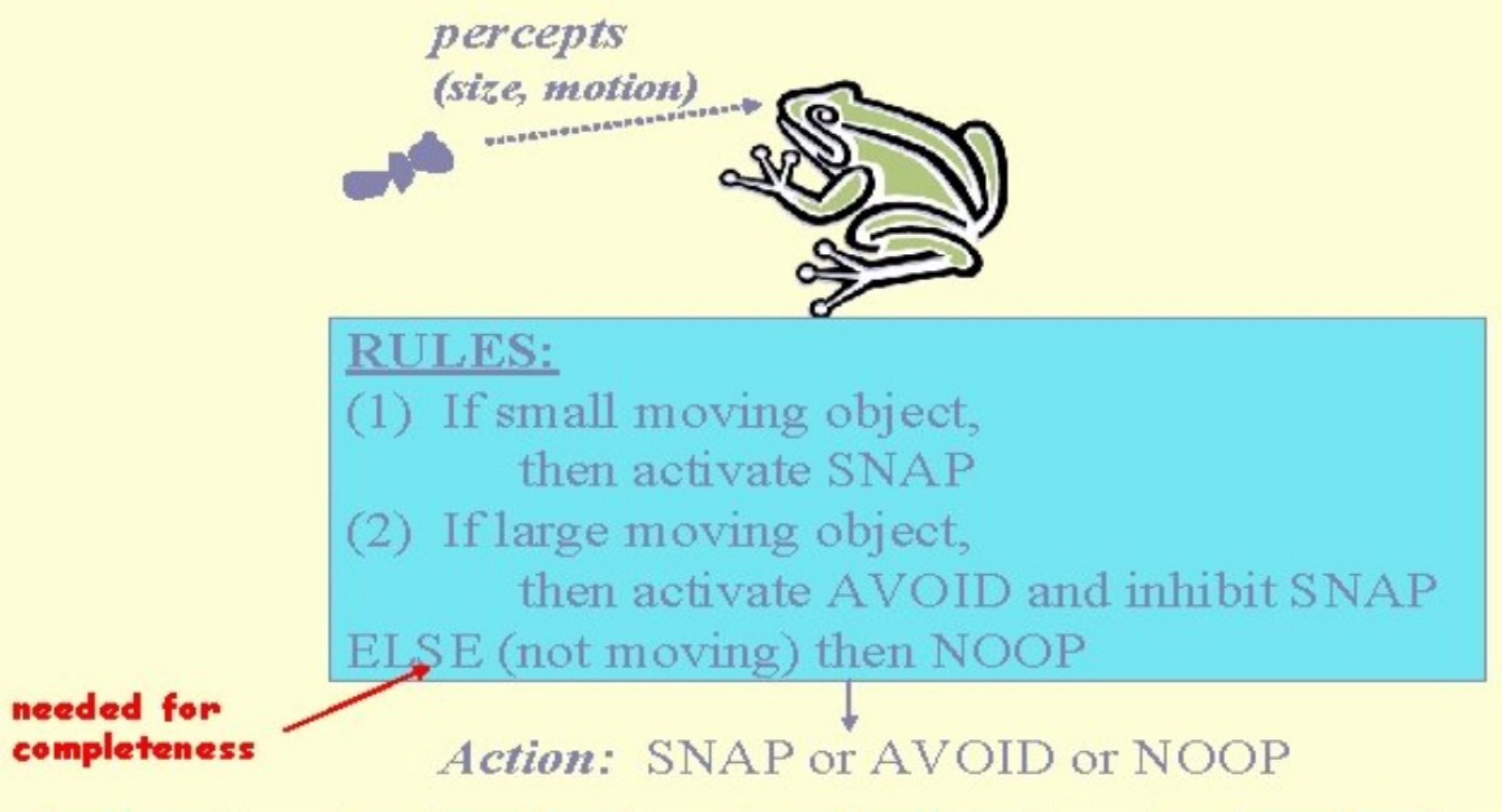
  rule ← RULE-MATCH(state, rules)

  action ← RULE-ACTION[rule]

return action

Simple reflex agent

- Example from the nature:



Simple reflex agent

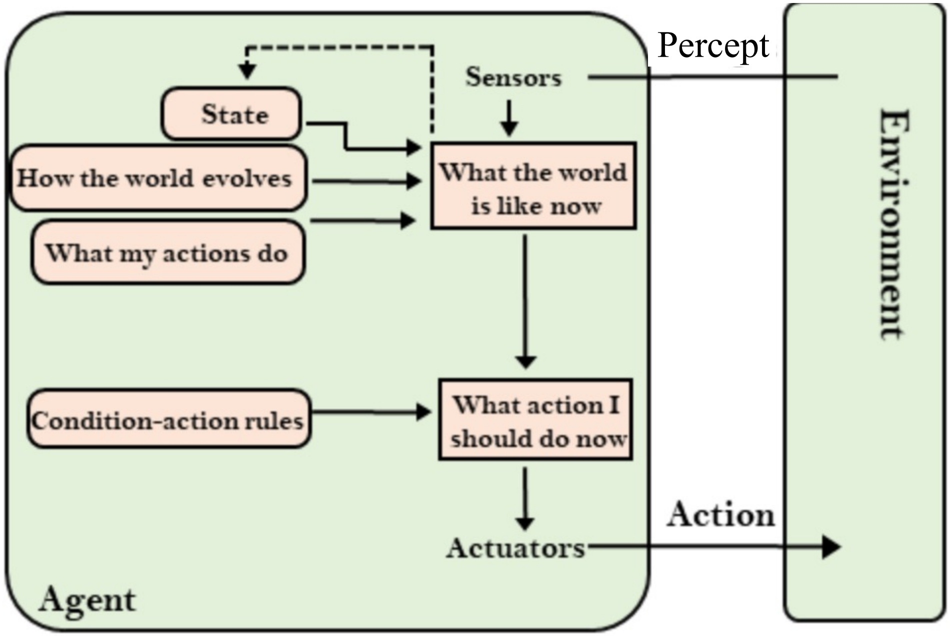
- Simple reflex agents can only be 100% successful in fully observable environments
- Example: Assume the simple reflex agent does not have location sensor, and it has only a dirt sensor
  - Then, it has only 2 possible percepts : [dirty] and [clean]
    - For [dirty], it has to Suck
    - But, what about [clean]?
      - Moving left fails if it is already in A
      - Moving right fails if it is already in B
  - So, infinite loops may occur in partially observable environments!

Simple reflex agent

- Disadvantages of the simple reflex agent design approach:
  - Not adapted to non observable or partially observable environments
  - They have very limited intelligence
  - They do not take into account the history of percepts/states/actions
  - They do not take into account the non-perceptual parts of the current state
    - Information that cannot be guessed from the percepts
  - The set of condition-action rules is often too big to generate and to store
  - Not adaptive to changes in the environment.

Model-based reflex agents

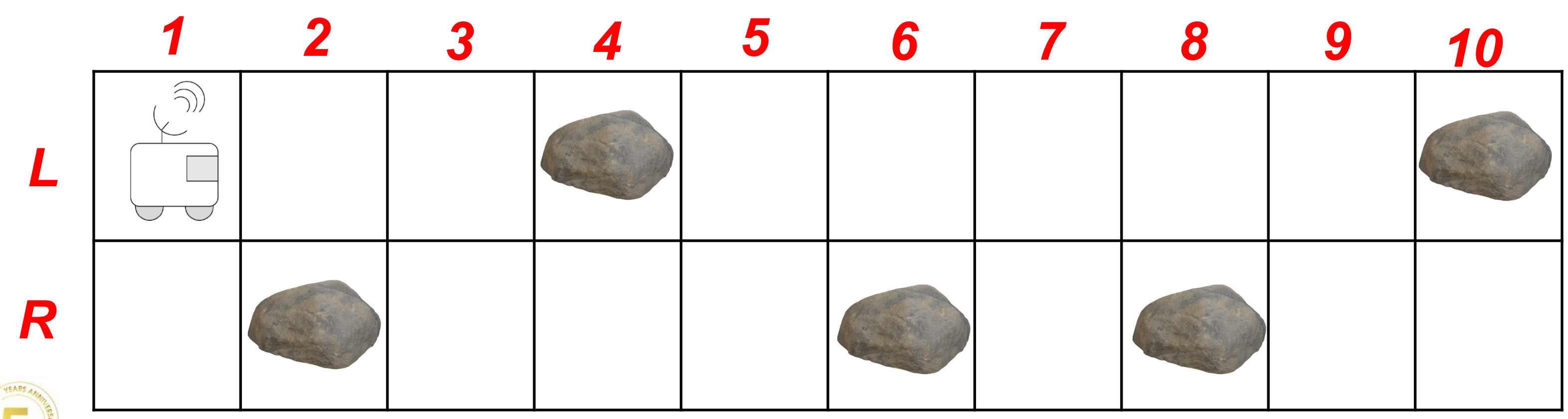
- The Model-based agent keeps track of an **internal state**
  - Internal state depends on the percept history
  - Reflects some aspects that cannot be observed right now through the sensors



- Requires a model composed of two types of knowledge
  - How the world evolves, independently of the agent's actions
    - Example: if a car just started overtaking the taxi agent right now, then in a few seconds it should be closer to the agent
  - How the agent's actions affect the world
    - When the taxi agent turns the steering clockwise, it will turn right

Model-based reflex agents

- Example: toy example
  - There are obstacles on the way; the agent must avoid them
  - The agent perceives its current location (*e.g. location* = “L1”)
    - The agent keeps a history of its successive locations
  - The agent only perceives obstacles that are in the cell in front of it
    - If there is an obstacle in front: obstacle=1 ; otherwise obstacle=0
  - The agent can only perform 4 actions:
    - Move forward (*Forward)*
    - Move to the left, if it is in a right cell (*Left)*
    - Move to the right, if it is in a left cell (*Right)*
    - Do nothing (*NoOp)*

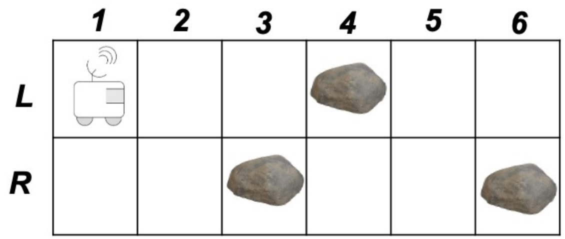


Model-based reflex agents

- Question 1: is the toy example a simple reflex agent or a model-based agent?
  - Answer: ….

Model-based reflex agents

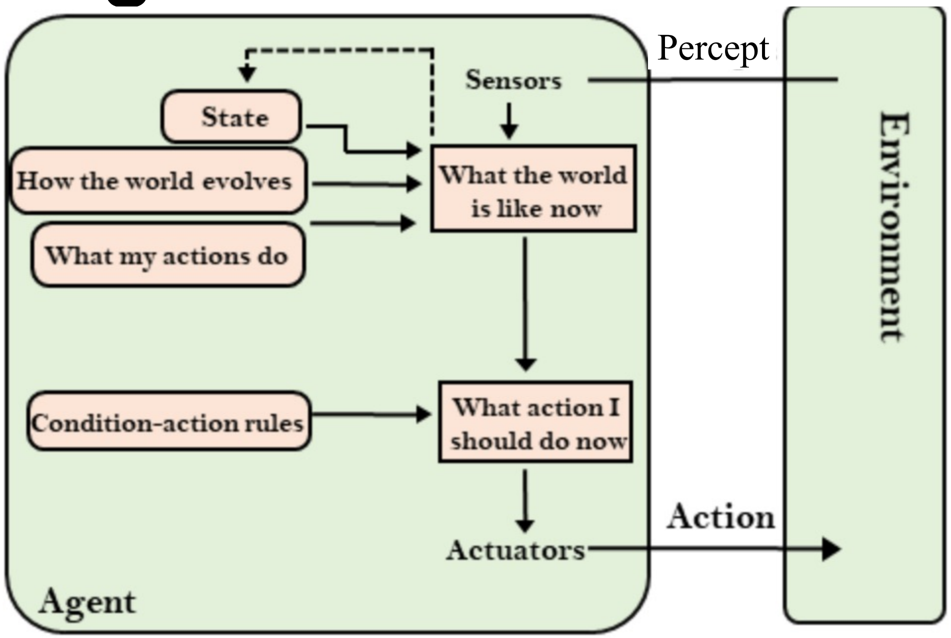
- Question 2: How to improve the agent so that it works even if there are two rocks blocking the road in diagonal?



  - Answer:
    - …

Model-based reflex agents

- More generally



function REFLEX-AGENT-WITH-STATE(percept) returns an action
static: rules, a set of condition-action rules
state, a description of the current world state
action, the most recent action, initially none

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

Model-based reflex agents

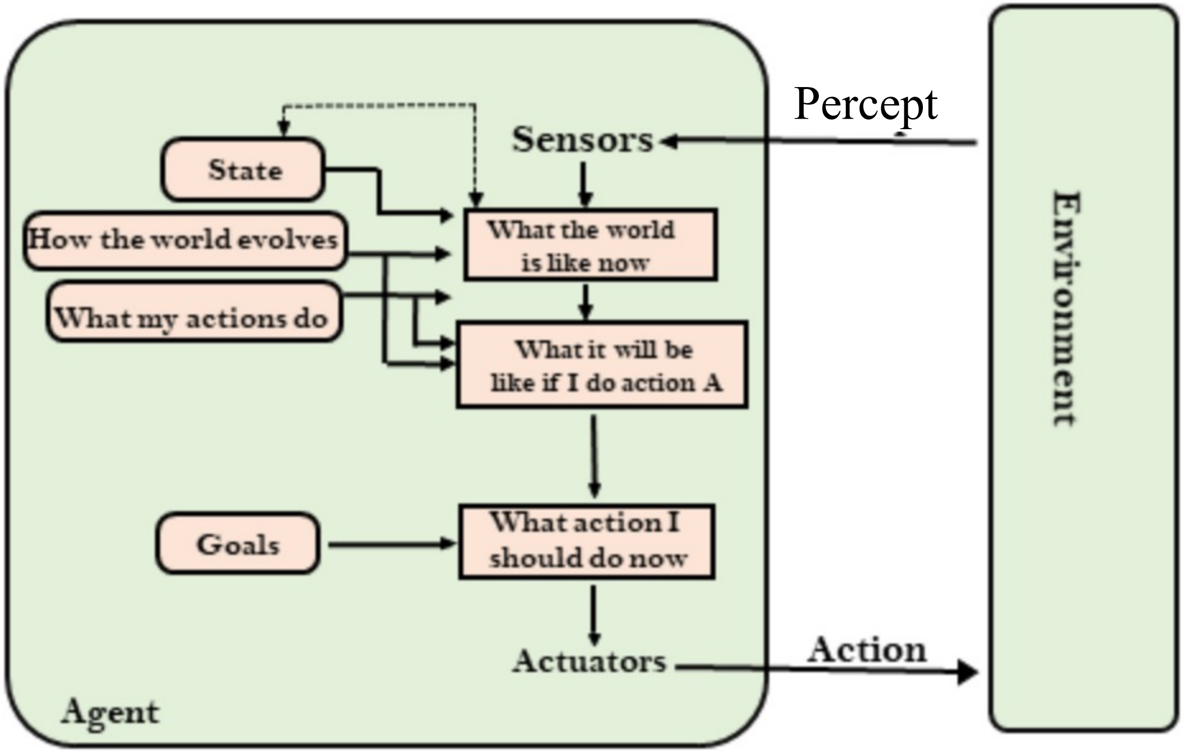
- The function UPDATE-STATE:
  - Creates a new internal state description
  - Interprets the new percept in the light of existing knowledge about the state
  - Keeps track of unseen part of the world by using information about how the world evolves

Model-based reflex agents

- Different from simple reflex agents, they can work even in a partially observable environment
  - If their sensors/models is adapted to what they observe
- But, in practice, it is seldom possible for the agent to determine the current state of a partially observable environment exactly
  - Instead, the agent will use its "best guess" in its model
  - Example: an automated taxi may not be able to see around the large truck that has stopped in front of it: it can only guess about what may be causing the truck to be stuck.
  - Thus, uncertainty about the current state is often unavoidable, but the agent still has to make a decision
- Also, knowing the current state is not always enough to decide what to do
  - Example 1: The taxi driver agent needs to know its destination, and have access to a map
  - Example 2: A shopping agent will be more effective if it has a map of the shop and a shopping list!

Goal-based agents

- Agents that take actions in the pursuit of a goal (or multiple goals)



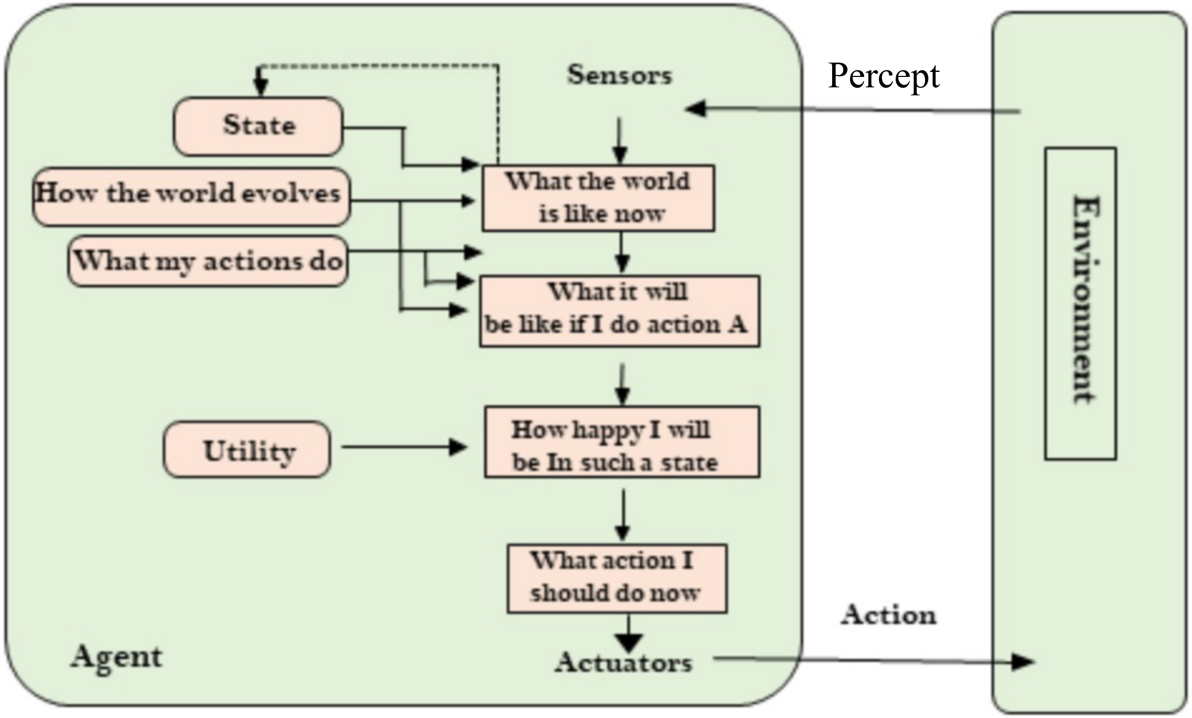
- Goals introduce the need to reason about the future or other hypothetical states.
- → Need for **search and planning** (see future chapters)
  - to find out the action sequences to achieve its goal

Goal-based agents

- Goal-based agents are often more **effective** than model-based reflex agents (for problems with a goal)
  - Reach better outcome
- But, goal-based agents are of course less **efficient**
  - Less quick in their decisions
- Also, goals are not always enough for ensuring **high-quality** behavior of the agent
  - Example: the agent's goal is to eat quickly and the canteen is the closest
  - But, what if the food at the canteen is disgusting?
  - → Notion of **utility**, initially introduced in the field of economics

Utility-based agents

- Agents that take actions making them the happiest in the long run



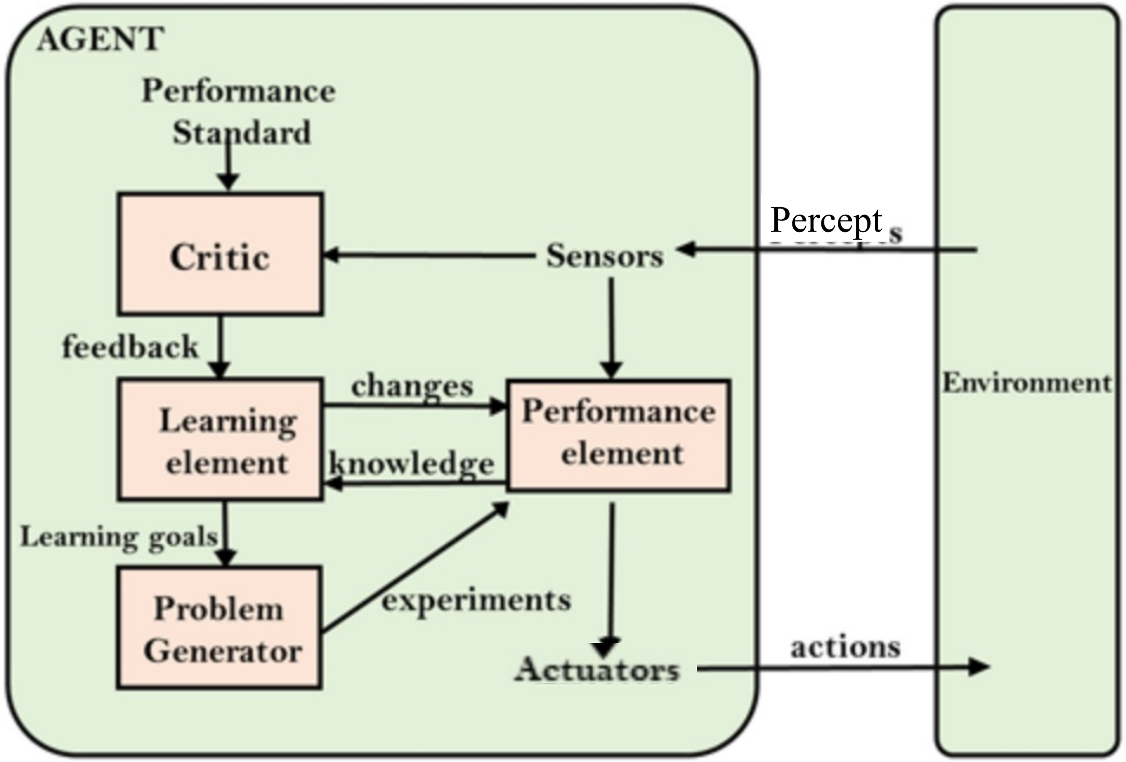
- More formally, agents that prefer actions that lead to states with higher utility
  - Utility is a function that maps a state into a real number (degree of "satisfaction")
- Utility-based agents can reason about multiple goals, conflicting goals, and uncertain situations
  - Example: eating quick but eating "good enough" food

Utility-based agents

- Utility has several advantages:
  - When there are conflicting goals, only some of the goals but not all can be achieved, still having the best "utility"
    - utility describes the appropriate trade-off
  - When there are several goals
    - None of them are achieved certainly
    - utility provides a way for the decision-making

Learning agents

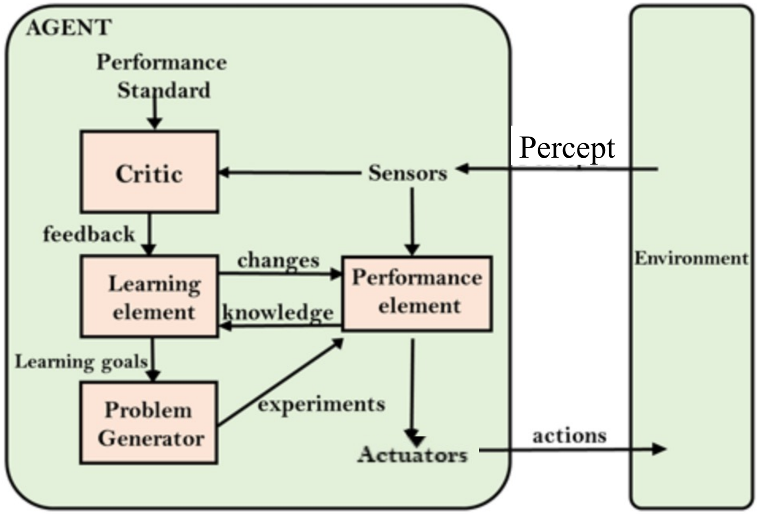
- A learning agent can learn from its past experiences
  - It starts to act with basic knowledge and then adapts automatically



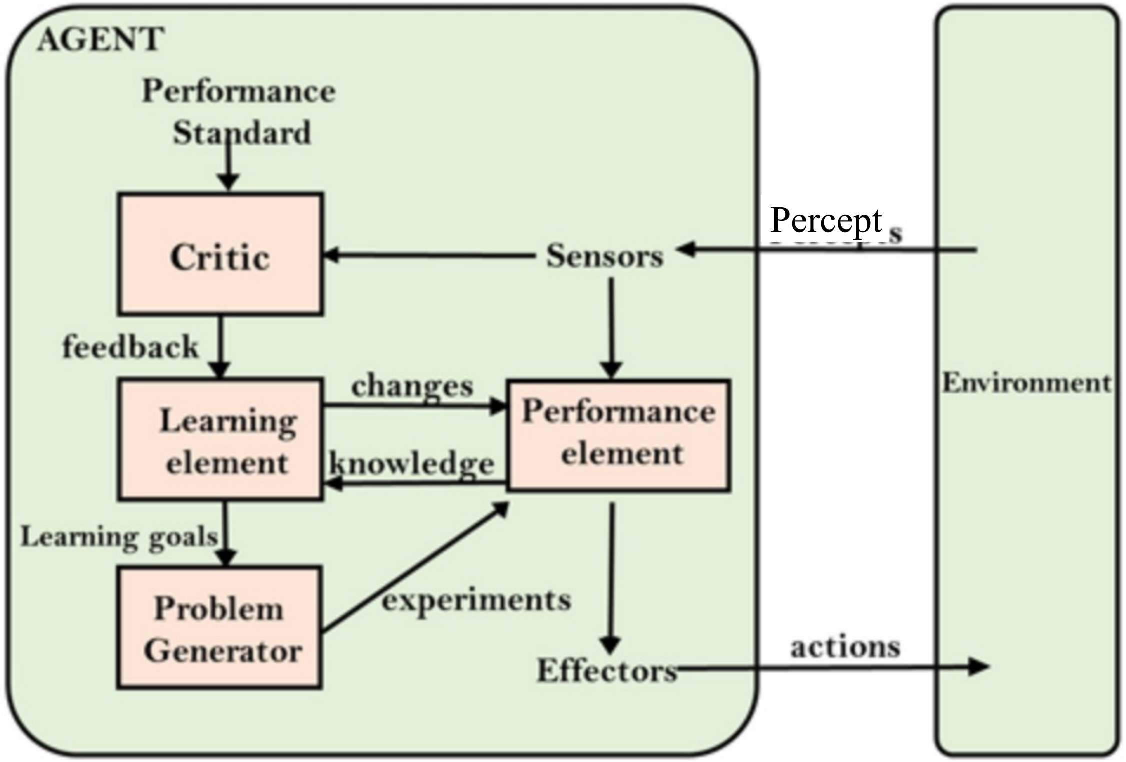
- ❑ **Learning** allows the agent to operate in initially unknown environments and to become more competent than with its initial knowledge only

Learning agents

- Four conceptual components:
  - **Performance element**
    - Used for selecting the best action
    - The performance element is what we have previously considered to be the entire agent: it takes in percepts and decides on actions
  - **Learning element**
    - Making improvements to the performance element, using feedback from the critic
  - **Critic**
    - Tells the Learning element how well the agent is doing with respect to a **fixed** performance standard
  - **Problem generator**
    - Suggest actions that will lead to new and informative experiences



Exercise on learning agents



- For a taxi driver agent, please identify what could be:
  - The performance standard
  - The performance element
  - The critic
  - The learning element
  - The problem generator

Exercise on learning agents

- For a taxi driver agent, please identify what **could** be:
    - The performance standard
        - …
        - …
    - The performance element
        - …
        - …
    - The critic
        - …
        - …
    - The learning element
        - …
        - …
    - The problem generator
        - …
        - …

Chapter 2: Intelligent agents

Knowledge Bases for agents

Knowledge bases

- Logical agents rely on Knowledge Bases

- Knowledge Base (KB) is a set of sentences in a formal language (**knowledge representation** language), telling an agent what it needs to know
    - Initially containing some background knowledge
    - can be enriched with new sentences

- An agent can **TELL** the KB what it perceives, then **ASK** the KB which action it should take, then **TELLs** the KB which action it took

Knowledge-based agents

- The agent must be able to:
  - Incorporate new percepts
  - Update internal representations of the world
  - Deduce hidden properties of the world
  - Deduce appropriate actions

- Agents can be viewed at:
  - the knowledge level: what they know, what their goals are
  - the implementation level: data structures in KB and algorithms that manipulate them

Knowledge-based agents

```
function KB-AGENT(percept) returns an action
   static: KB, a knowledge base
               t, a counter, initially 0, indicating time
   TELL(KB, MAKE-PERCEPT-SENTENCE(percept,t))
   action ← ASK(KB, MAKE-ACTION- QUERY(t))
   TELL(KB, MAKE-ACTION-SENTENCE(action,t) )
   t ←  t+1
return action
```

Chapter 2: Intelligent agents

A few words about multi-agent planning

Multi-agent planning

- Environment: **cooperative** or **competitive**
- Issue: the environment is not **static → synchronization**
- Require a model of the other agent's plans
- Cooperation: joint goals and plans, e.g., team planning in doubles tennis
  - Joint goal: returning the ball that has been hit to them and ensuring that at least one of them is covering the net
  - Joint plan: multi-body planning
  - Coordination mechanisms: decompose and distribute tasks

Multi-agent planning

- Environment: **cooperative** or **competitive**
- Competition: e.g., chess-playing
  - An agent in a competitive environment must
    - recognize that there are other agents
    - compute some of the other agent's possible plans
    - compute how the other agent's plans interact with its own plans
    - decide on the best action in view of these interactions.

How to decide which entity is an agent, and which entity is simply an object that follows the laws of physics?
  - Not always straightforward, as said on slide 40…

Chapter 2: Intelligent agents

Summary

Summary

- An agent perceives and acts in an environment
- Its agent function specifies the action taken by the agent in response to any percept sequence
- The agent program implements the agent function.
- Performance measure evaluates the behavior of the agent in its environment and is maximized by rational agents
- Task environment specification includes the performance measure, the external environment, the actuators, and the sensors. It is crucial for designing an agent.
- There are several kinds of task environments
  - Fully or partially observable, single-agent / multiagent, deterministic / stochastic, episodic / sequential, static / dynamic, discrete / continuous, known / unknown.
- There are several kinds of agents
  - Simple reflex agents, model-based reflex agents, goal-based agents, utility-based agents, knowledge-based agents
- All agents can improve their performance through learning

Homework

- Read Chapter 2 of the reference book "Artificial Intelligence – A modern approach, **THIRD EDITION**"
- Make the exercises 2.1. to 2.13. of this book, and verify yourself that your answers are correct using the solution leaflet

Chapter 2: INTELLIGENT AGENTS

Questions

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

Thank you for your attention!

soict.hust.edu.vn/ fb.com/groups/soict