ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

Artificial Intelligence

Lecture 2 - Agent

School of Information and Communication Technology - HUST

Outline

1. Agents and environments
2. PEAS (Performance measure, Environment, Actuators, Sensors)
3. Environment types
4. Agent types

Agents and environments

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

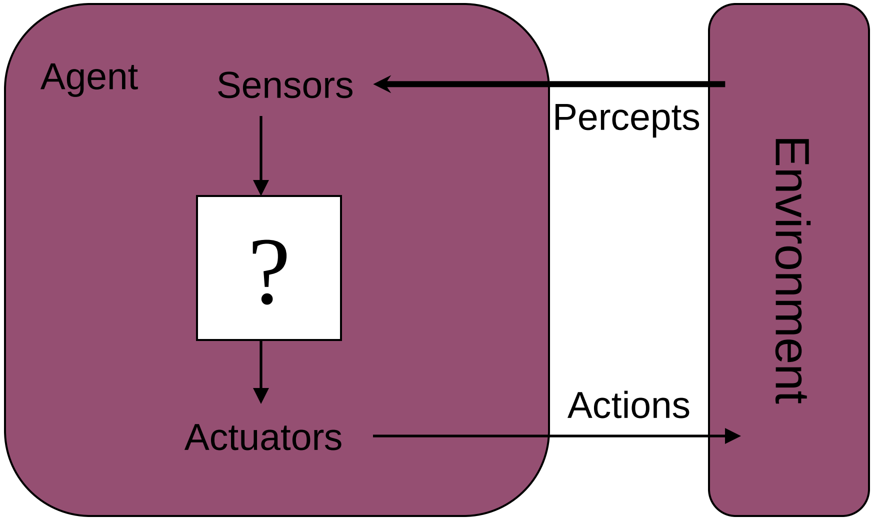
- Example 1: human agent
  - Sensors: eyes, ears, …
  - Actuators: hands, legs, mouth, …

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

Agents and environments (con't)

- The agent function maps from percept histories to actions:

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



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

  agent = architecture + program

Agent function based on conditional table

Function TABLE-DRIVEN-AGENT(percept) **returns** an action

static: *percepts*, a sequence, initially empty

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

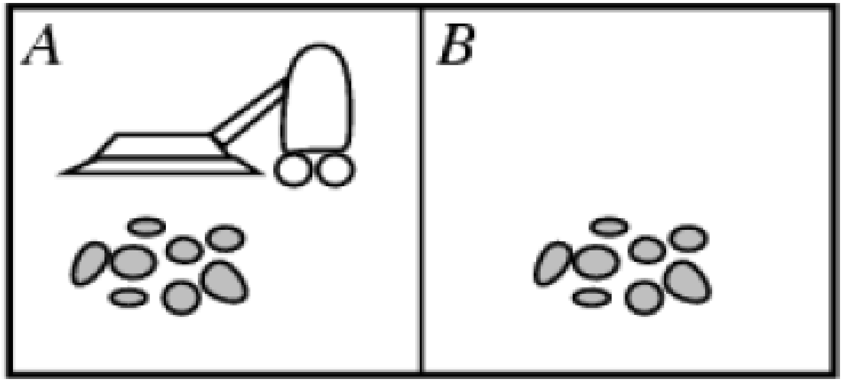
Append *percept* to the end of *percepts*

action ← LOOKUP(*percepts, table*)

Return *action*

Drawback: huge table!

Vacuum-cleaner world



- Percepts: location (A or B), state (clean or dirty)
- Actions: *Left*, *Right*, *Suck*, *NoOp*

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

Vacuum-cleaner world

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

If state = Dirty **then return** Suck

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

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

End Function

- Does the agent act reasonably?

Rational agent

- A **rational agent** is one that does the right thing - the one that will cause the agent to be most successful

- Performance measure embodies the criterion for success of an agent's behavior.
  - E.g., performance measure of a vacuum-cleaner agent:
    - amount of dirt cleaned up
    - amount of time taken
    - amount of electricity consumed
    - amount of noise generated
    - …

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.

- An agent is autonomous if its behavior is determined by its own experience (with ability to learn and adapt)

PEAS

- 4 factors should be considered when design 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 - Medical diagnosis system

- Performance measure: Healthy patient, minimize costs, lawsuits, …
- Environment: Patient, hospital, staff
- Actuators: Screen display (questions, tests, diagnoses, treatments, referrals)
- Sensors: Keyboard (entry of symptoms, findings, patient's answers)

PEAS - Spam Filtering Agent

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

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.
- Deterministic (vs. stochastic): The next state of the environment is completely determined by the current state and the action executed by the agent.
- Episodic (vs. sequential): The agent's experience is divided into atomic "episodes" (each episode consists of the agent perceiving and then performing a single action.

Environment types

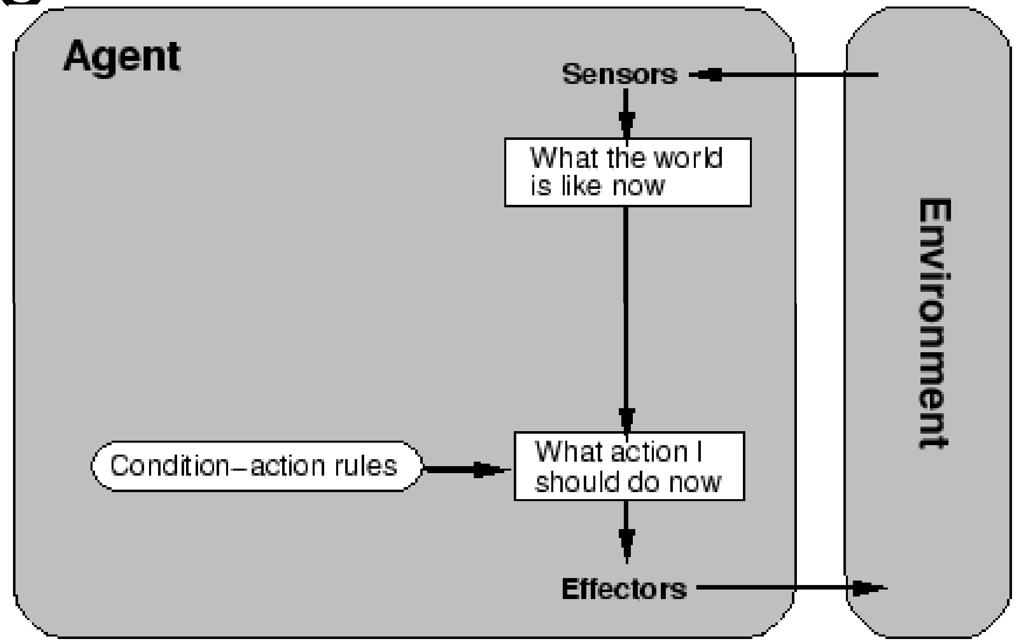
- Static (vs. dynamic): The environment is unchanged while an agent is deliberating.
- Discrete (vs. continuous): A limited number of distinct, clearly defined percepts and actions.
- Single agent (vs. multiagent): An agent operating by itself in an environment.

Agent types

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

Simple reflex agent

- These agents select actions on the basis of the *current* percept, ignoring the rest of the percept history



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

static: rules, a set of condition-action rules

state ← INTERPRET-INPUT(percept)

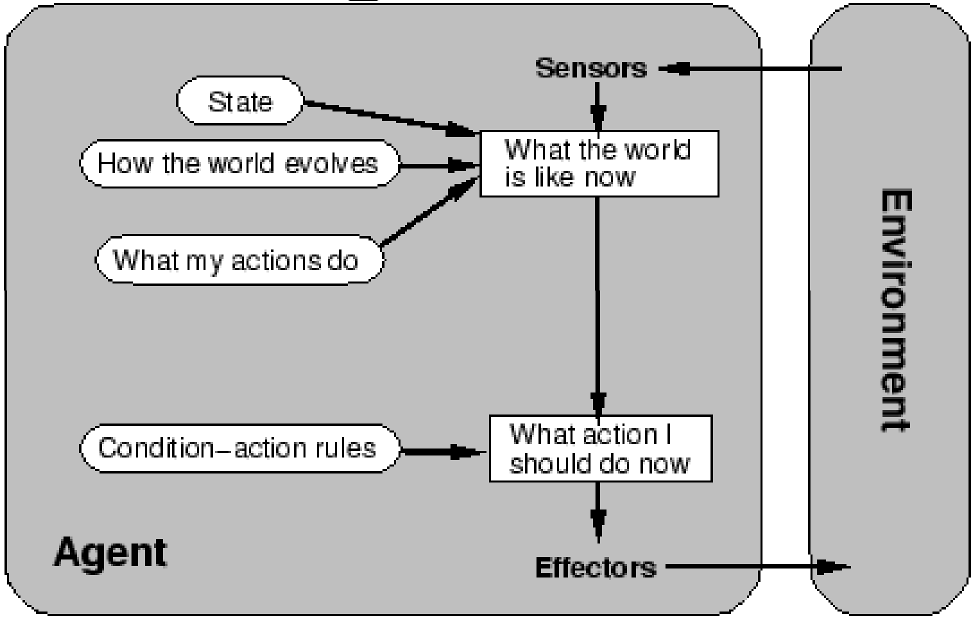
rule ← RULE-MATCH(state, rules)

action ← RULE-ACTION[rule]

return action

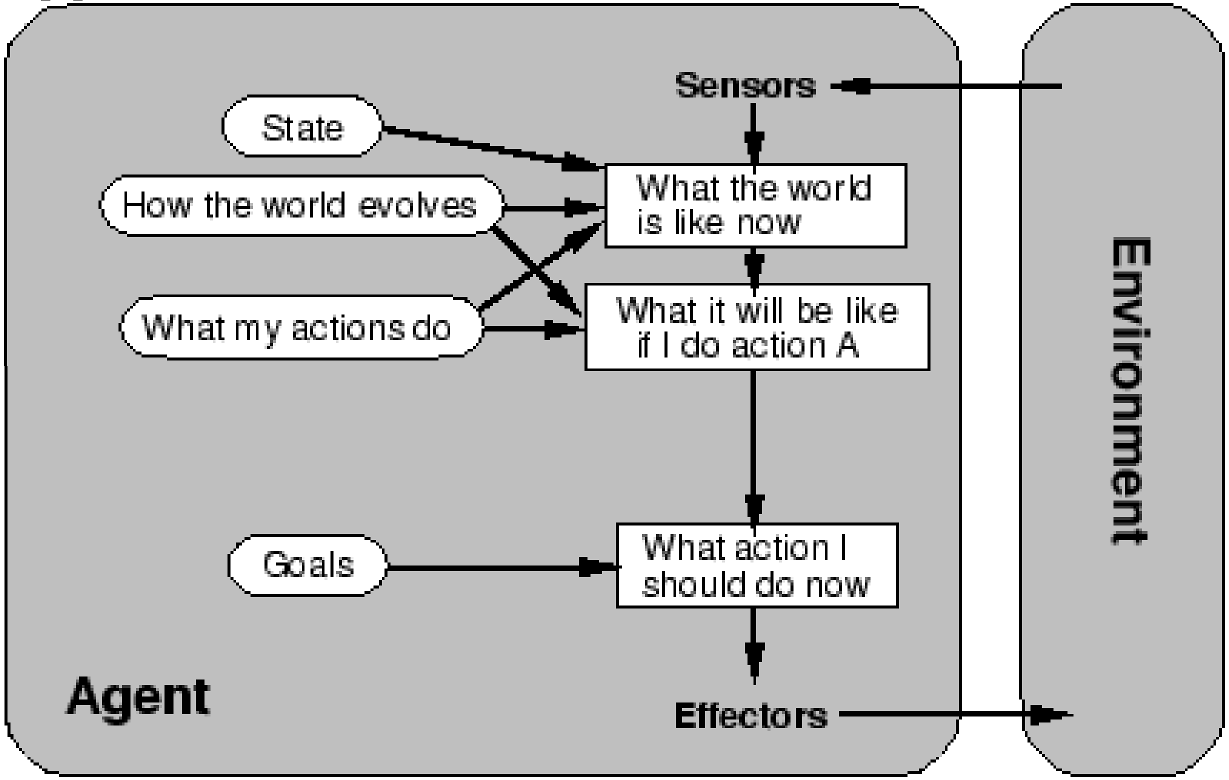
Model-based reflex agents

- These agents maintain **internal states** that depends on the percept history and thereby reflects at least some of the unobserved aspects of the current state.



function REFLEX-AGENT-WITH-STATE(percept) returns an action
static: state, a description of the current world state
rules, a set of condition-action rules
action, the most recent action, initially none
state ← UPDATE-STATE(state, action, percept)
rule ← RULE-MATCH(state, rules)
action ← RULE-ACTION[rule]
return action

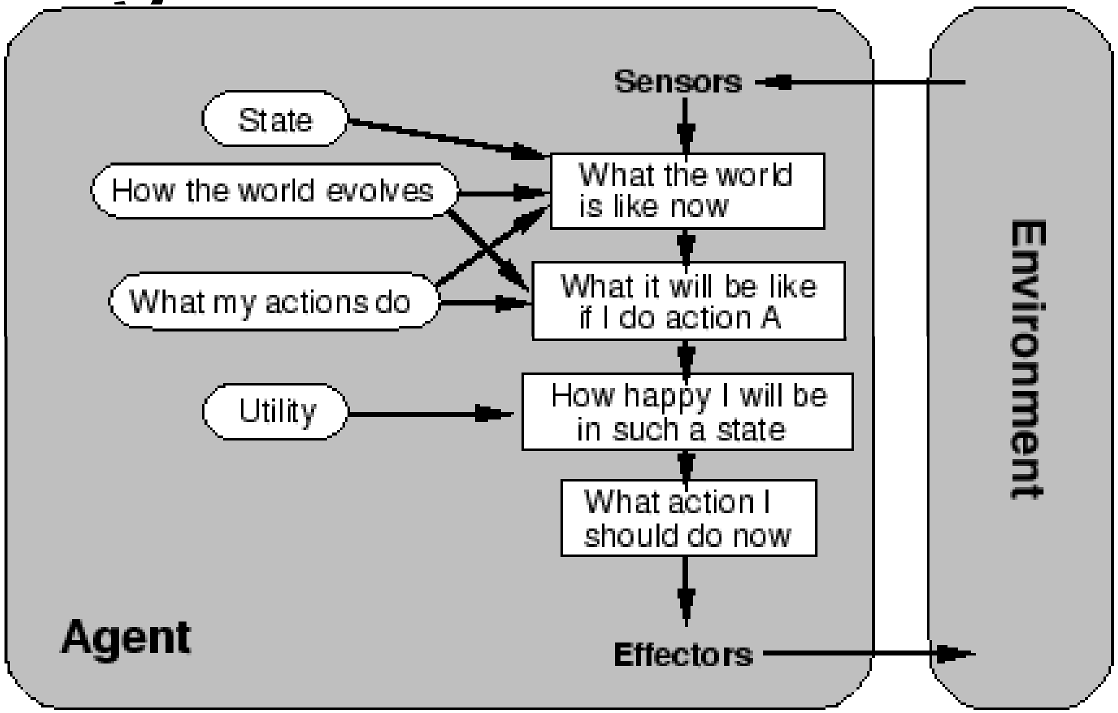
Goal-based agents



- Agents that take actions in the pursuit of a goal or goals.

- Goals introduce the need to reason about the future or other hypothetical states. It may be the case that none of the actions an agent can currently perform will lead to a goal state.

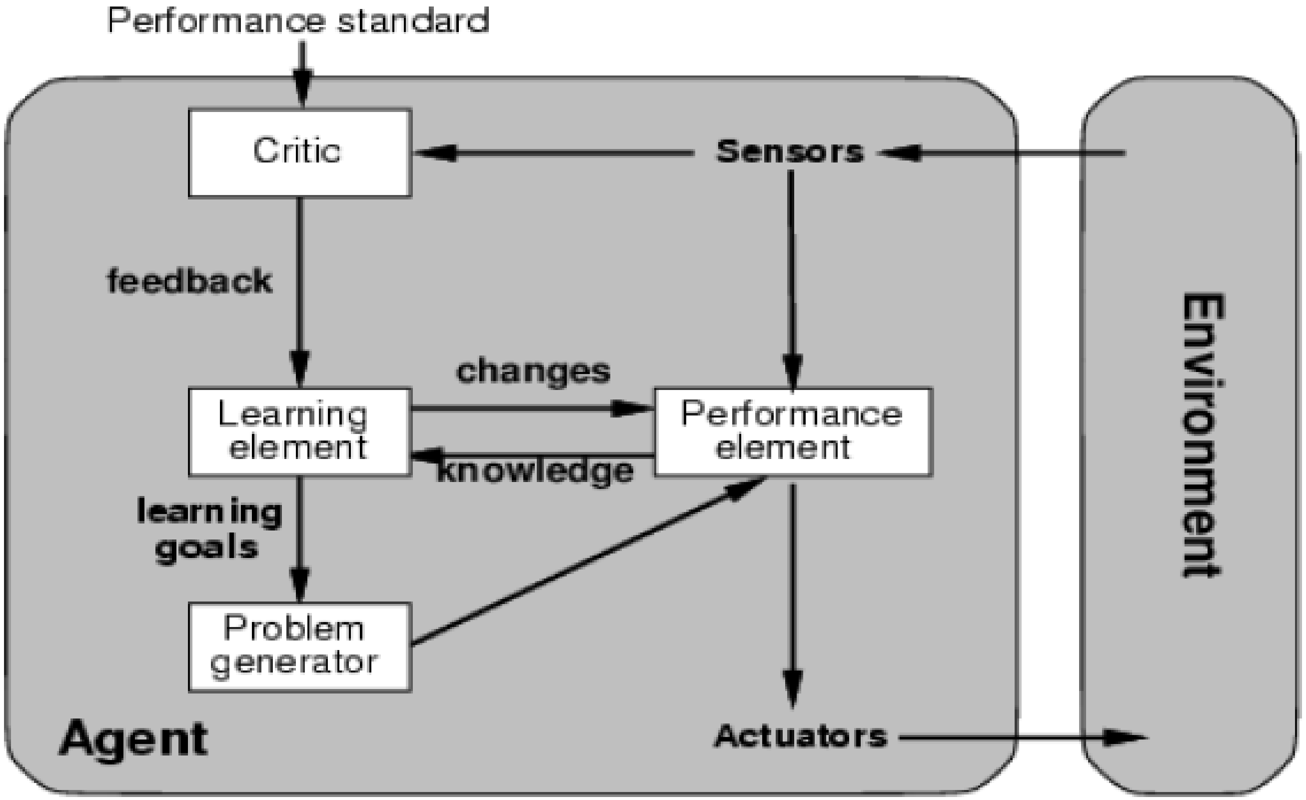
Utility-based agents



- Agents that take actions that make them the most happy in the long run.

- More formally agents that prefer actions that lead to states with higher utility.
- Utility-based agents can reason about multiple goals, conflicting goals, and uncertain situations.

Learning agents



- Learning allows the agent to operate in initially unknown environments and to become more competent than its initial knowledge alone might allow.
- The most important question: “What kind of performance element will my agent need to do this once it has learned how?”

Knowledge bases

- Knowledge base is a set of sentences in a formal language, telling an agent what it needs to know
- Agent can ASK itself what to do, the answer should follow from the KB
- Agents can be viewed at:
  - the knowledge level: what they know, what its goals are
  - the implementation level: data structures in KB and algorithms that manipulate them
- The agent must be able to:
  - Incorporate new percepts
  - Update internal representations of the world
  - Deduce hidden properties of the world
  - Deduce appropriate actions

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(^))
  TELL(KB, MAKE-ACTION-SENTENCE(action,t))
  t ← t+1
return action

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: multibody planning
  - Coordination mechanisms: decompose and distribute tasks

- 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.