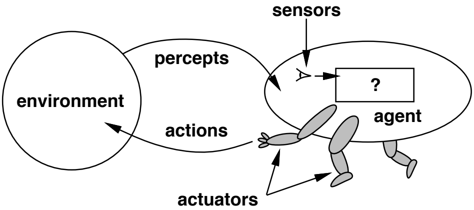
Intelligent Agents

Jihoon Yang

Machine Learning Research Laboratory
Department of Computer Science & Engineering
Sogang University

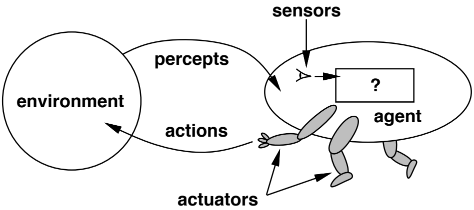
Agents and environments



- An agent: perceives and acts
- Percept: perceptual inputs at any given instant
- Percept sequence: complete history of percepts
- An agent's behavior is described by the *agent function* that maps percept sequence to actions

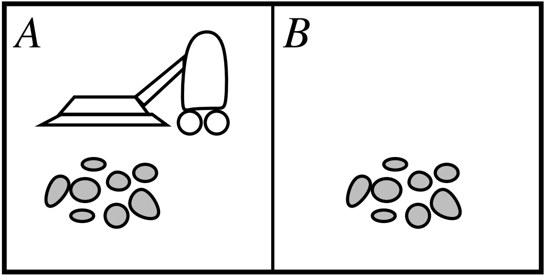
$$f : \mathcal{P}^* \to \mathcal{A}$$

Agents and environments



- The agent function will internally be implemented by the *agent program*
- The agent program runs within some physical system to produce f
- Job of AI is to design agent programs

The vacuum-cleaner world



- Environment: square A and B
- Percepts: [location and content] (e.g. [A, $Dirty$])
- Actions: left, right, suck, and no-op

The vacuum-cleaner world

Percept sequence	Action
$[A, Clean]$	$Right$
$[A, Dirty]$	$Suck$
$[B, Clean]$	$Left$
$[B, Dirty]$	$Suck$
$[A, Clean]$, $[A, Clean]$	$Right$
$[A, Clean]$, $[A, Dirty]$	$Suck$
$\vdots$	$\vdots$

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*

Is this agent a rational agent?

The concept of rationality

- A **Rational agent** is one that does the right thing
  - Every entry in the table is filled out correctly
- What is the right thing?
  - Approximation: cause the agent to be most *successful*
  - *Measure of success?*
- **Performance measure**: a criterion for success of an agent's behavior
  - E.g. the amount of dirt cleaned within a certain time
  - E.g. how clean the floor is
  - ...

Rationality

- What is rational at a given time depends on four things:
  - Performance measure 성능평가
  - Prior environment knowledge 사전 지식
  - Actions that the agent can perform
  - Percept sequence to date

- **Definition**: A rational agent chooses an action that is expected to maximize its performance measure given the percept sequence to date and built-in environment knowledge the agent has

Rationality

- Rationality $\neq$ omniscience, $\neq$ perfection
  - An omniscient agent knows the actual outcome of its actions
  - Rationality maximizes *expected* performance
  - Perfection maximizes *actual* performance
- Rationality requires: 탐색 활용
  - Information gathering/exploration $\longleftrightarrow$ exploit
    - To maximize future rewards
  - Learn from percepts
    - Extending prior knowledge
  - Being automomous 자율성
    - Compensate for partial prior knowledge, adapt

이미 알고있는 지식 보완

Task Environment

- To design a rational agent we must specify its *task environment*
- PEAS description of the task environment:
  - Performance measure
  - Environment
  - Actuators
  - Sensors

PEAS

Consider the task of designing an automated taxi:

- Performance measure: safety, destination, profits, comfort, $\cdots$
- Environment: US streets/freeways, traffic, pedestrians, weather, $\cdots$
- Actuators: steering, acclerator, brake, horn, speaker/display, $\cdots$
- Sensors: video, accelerometers, gauges, engine sensors, GPS, $\cdots$

PEAS

Consider the task of designing an Internet shopping agent:

- Performance measure: price, quality, appropriateness, efficiency
- Environment: current and future WWW sites, vendors, shippers
- Actuators: display to user, follow URL, fill in form
- Sensors: HTML pages (text, graphics, scripts)

Environment types

- Categorize task environments according to properties
- These properties may determine appropriate families of techniques for agent implementation

	Chess	Backgammon	Taxi driving
Observable??			
Deterministic??			
Static??			
Discrete??			
Single-agent??			

Environment types

- Fully vs. partially observable: An environment is full observable when the sensors can detect all aspects that are *relevant* to the choice of action

	Chess	Backgammon	Taxi driving
Observable??	FULL	FULL	PARTIAL
Deterministic??			
Static??			
Discrete??			
Single-agent??			

episodic?
sequential?

Environment types

- Deterministic vs. stochastic: If the next environment state is completely determined by the current state and the executed action then the environment is deterministic

	Chess	Backgammon	Taxi driving
Observable??	FULL	FULL	PARTIAL
Deterministic??	YES	NO	NO
Static??			
Discrete??			
Single-agent??			

Environment types

- Static vs. dynamic: If the environment can change while the agent is choosing an action, the environment is dynamic. *Semi-dynamic* if the agent's performance score changes with the passage of time even when the environment remains the same.

	Chess	Backgammon	Taxi driving
Observable??	FULL	FULL	PARTIAL
Deterministic??	YES	NO	NO
Static??	YES/Semi	YES/Semi	NO
Discrete??			
Single-agent??			

Environment types

- Discrete vs. continuous: This distinction can be applied to the *state* of the environment, to the way *time* is handled, and to the *percepts/actions* of the agent

	Chess	Backgammon	Taxi driving
Observable??	FULL	FULL	PARTIAL
Deterministic??	YES	NO	NO
Static??	YES/Semi	YES	NO
Discrete??	YES	YES	NO
Single-agent??			

Environment types

- Single vs. multi-agent: Does the environment contain other agents who are also maximizing some performance measure that depends on the current agent's actions?

	Chess	Backgammon	Taxi driving
Observable??	FULL	FULL	PARTIAL
Deterministic??	YES	NO	NO
Static??	YES/Semi	YES	NO
Discrete??	YES	YES	NO
Single-agent??	NO	NO	NO

Environment types

- The simplest environment is
  - Fully observable, deterministic, static, discrete, and single-agent
- Most real situations are
  - Partially observable, stochastic, dynamic, continuous, and multi-agent

Agent types

- The job of AI is to design agent programs
  - Agent = architecture + program
- Agent program implements agent function mapping percepts to actions
- All agent programs have the same skeleton:
  - Input = current percepts
  - Output = action
  - Program = manipulates input to produce output

Table-lookup Agent

hard coding table에서 적절한 행동 찾아서 반환

Function TABLE-DRIVEN_AGENT(*percept*) **returns** an action

static: *percepts*, a sequence initially empty
table, a table of actions, indexed by percept sequence
- explicit representation of agent function

append *percept* to the end of *percepts*
action $\leftarrow$ LOOKUP(*percepts*, *table*)
return *action*

Does this approach appear to implement any possible agent function?

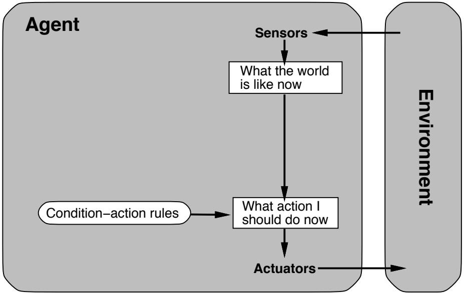
Agent types

- Four basic kinds of agent programs:
  - Simple reflex agents
  - Model-based reflex agents
  - Goal-based agents
  - Utility-based agents

효율성

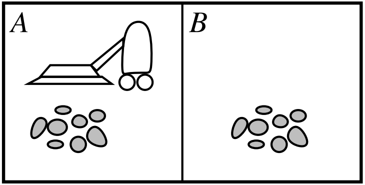
- All these can be turned into learning agents

Simple reflex agents



- Select actions on the basis of *only* the current percept
    - E.g. the vacuum-agent
- Implemented through *condition-action* rules
    - if dirty then suck

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*

Reduction from 4^T to 4 entries (T is the lifetime of the agent, in other words the total number of percepts it will receive)

Simple reflex agents

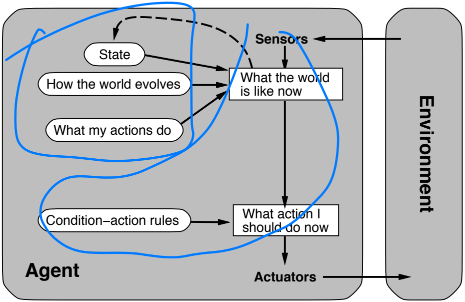
function SIMPLE-REFLEX-AGENT(*percept*) **returns** an action

static: *rules*, a set of condition-action rules

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

Will work only if the correct decision can be made based on only the current percept (e.g. the environment is fully observable)

Model-based reflex agents



- To tackle *partially* observable environments
  - Maintain internal state
- Over time update state using world knowledge
  - How does the world change
  - How do actions affect world

  $\Rightarrow$ *Model of World*

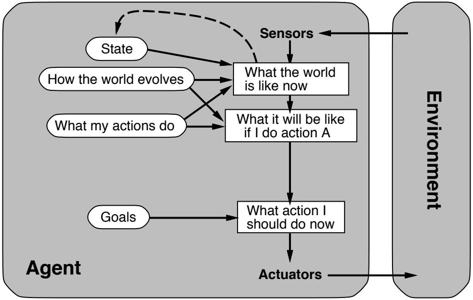
Model-based reflex agents

function MODEL-BASED-REFLEX-AGENT(*percept*) **returns** an action

static: *rules*, a set of condition-action rules
state, a description of the current world state
model, a model of the world
action, the most recent action.

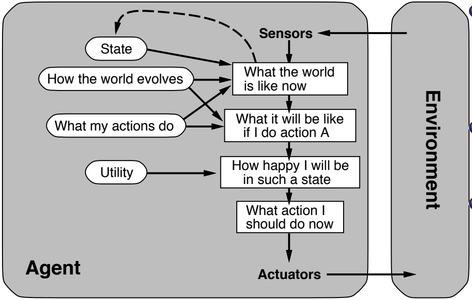
state ← UPDATE-STATE(*state*, *action*, *percept*,*model*)
rule ← RULE-MATCH(*state*, *rules*)
action ← rule.ACTION
return *action*

Goal-based agents



- The agent seeks to achieve certain goals
- Things become difficult when long sequences of actions are required to find the goal
  - Search
  - Planning
- Fundamental difference: future is taken into account

Utility-based agents

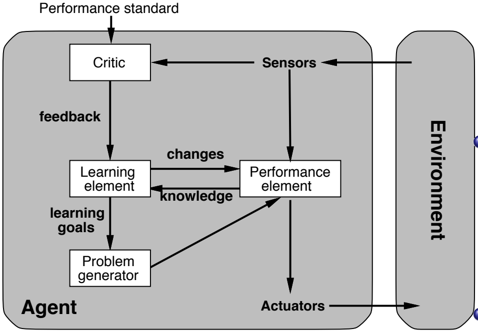


- Certain goals can be reached in different ways; Conflicting goals
- *Utility function* maps a (sequence of) state(s) onto a real number (utility)
- Rational agents try to maximize expected utility
- Improves on goals:
  - Selecting between conflicting goals
  - Select appropriately between several goals based on likelihood of success and importance of the goals

Learning agents

- All previous agent-programs describe methods for selecting *actions*
- All use knowledge
  - Where does these knowledge come from?
  - Learning mechanisms can be used
  - Teach them instead of instructing them
- Advantage is the robustness of the program
  - Environment changes over time – adapt to changes
  - Learning is essential for unknown

Learning agents



- *Learning element*: introduces improvements in performance element
  - Critic provides feedback on agent's performance based on fixed performance standard
- *Performance element*: selects actions based on percepts
  - Corresponds to the previous agent programs
- *Problem generator*: suggests actions that will lead to new and informative experiences
  - Exploration vs. exploitation

Summary

- **Agents** interact with environments through actuators and sensors
- The **agent function** describes what the agent does
- The **performance measure** evaluates the behavior of the agent
- A perfectly **rational agent** maximizes expected performance
- **Agent programs** implement agent functions
- **PEAS** descriptions define task environments
- Environments are categorized along several dimensions:
  observable? deterministic? static? discrete? single-agent?
- Several basic agent architectures exist:
  reflex, model-based reflex, goal-based, utility-based
- All agents can improve their performance through **learning**