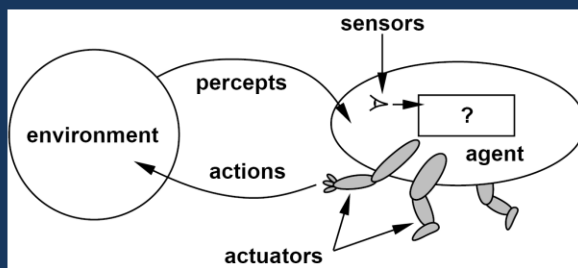
BLG435E
Artificial Intelligence

Lecture 2: Intelligent Agents

Outline

- Agents and Environments
- Rationality
- PEAS
- Environment Types
- Agent Types

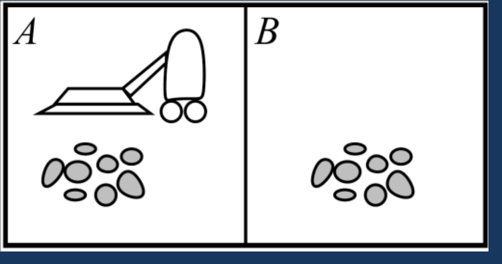
Agents



Agents

- Agents perceive their own actions
  - Effects?
- Percept: the agen’t perceptual input
- Percept sequence: the complete history
- Action choices depend on the percept sequence
- Agent function, abstract mathematical description (agent’s behavior)
- Agent program implements the function

Vacuum-Cleaner World



Percepts: location and contents, e.g., $[A, Dirty]$

Actions: $Left$, $Right$, $Suck$, $NoOp$

A Vacuum-Cleaner Agent Function

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

What is the right way to fill out the table?

What makes an agent good, bad or stupid?

A Vacuum-Cleaner Agent Function

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

```
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
```


Rationality

- A rational agent does the right thing
- What is rational at any given time depends on:
  - The performance measure that defines the criterion of success
  - The agent's prior knowledge of the environment
  - The actions that the agent can perform
  - The agent's percept sequence to date

Vacuum Cleaner Agent - PM

- The amount of dirt cleaned up in a single eight-hour shift.
- Rewarding agent for having a clean floor.
- Factoring amount of electricity consumed and the amount of noise generated

- Design PM according to what you want

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.

- With a rational agent, what you ask is what you get

Rationality vs. Perfection

- Omniscience is impossible in reality
- Agents don't estimate the actual outcome of actions
- Rationality maximizes expected outcome, while perfection maximizes actual performance

Further Dimensions in Rationality

- Information gathering
  - Exploration
  - Helps maximize the expected outcome
- Learning
- Autonomy
- With or without initial knowledge

Agents <> Environments

- Task environment forms the problem
  - Rational agents are the solutions
- The task environment affects the appropriate design of the agent

The Nature of Environments

- PEAS for task environments:
  - **P**erformance measure
  - **E**nvironment
  - **A**ctuators
  - **S**ensors
- PEAS for automated taxi driver

Properties of Task Environments

- Fully observable vs. Partially observable
- Deterministic vs. Stochastic
  - Strategic
- Episodic vs. Sequential
- Static vs. Dynamic
  - semidynamic

Properties of Task Environments

- Discrete vs. Continous
- Single agent vs. Multiagent
  - competitive
  - cooperative

Examples of Task Environments

Task Environment	Observable	Deterministic	Episodic	Static	Discrete	Agents
Crossword puzzle						
Chess with a clock						
Poker						
Backgammon						
Taxi driving						

Examples of Task Environments

Task Environment	Observable	Deterministic	Episodic	Static	Discrete	Agents
Crossword puzzle	Fully	Deterministic	Sequential	Static	Discrete	Single
Chess with a clock						
Poker						
Backgammon						
Taxi driving						

Examples of Task Environments

Task Environment	Observable	Deterministic	Episodic	Static	Discrete	Agents
Crossword puzzle	Fully	Deterministic	Sequential	Static	Discrete	Single
Chess with a clock	Fully	Strategic	Sequential	Semi	Discrete	Multi
Poker						
Backgammon						
Taxi driving						

Examples of Task Environments

Task Environment	Observable	Deterministic	Episodic	Static	Discrete	Agents
Crossword puzzle	Fully	Deterministic	Sequential	Static	Discrete	Single
Chess with a clock	Fully	Strategic	Sequential	Semi	Discrete	Multi
Poker	Partially	Stochastic	Sequential	Static	Discrete	Multi
Backgammon						
Taxi driving						

Examples of Task Environments

Task Environment	Observable	Deterministic	Episodic	Static	Discrete	Agents
Crossword puzzle	Fully	Deterministic	Sequential	Static	Discrete	Single
Chess with a clock	Fully	Strategic	Sequential	Semi	Discrete	Multi
Poker	Partially	Stochastic	Sequential	Static	Discrete	Multi
Backgammon	Fully	Stochastic	Sequential	Static	Discrete	Multi
Taxi driving						

Examples of Task Environments

Task Environment	Observable	Deterministic	Episodic	Static	Discrete	Agents
Crossword puzzle	Fully	Deterministic	Sequential	Static	Discrete	Single
Chess with a clock	Fully	Strategic	Sequential	Semi	Discrete	Multi
Poker	Partially	Stochastic	Sequential	Static	Discrete	Multi
Backgammon	Fully	Stochastic	Sequential	Static	Discrete	Multi
Taxi driving	Partially	Stochastic	Sequential	Dynamic	Continuous	Multi

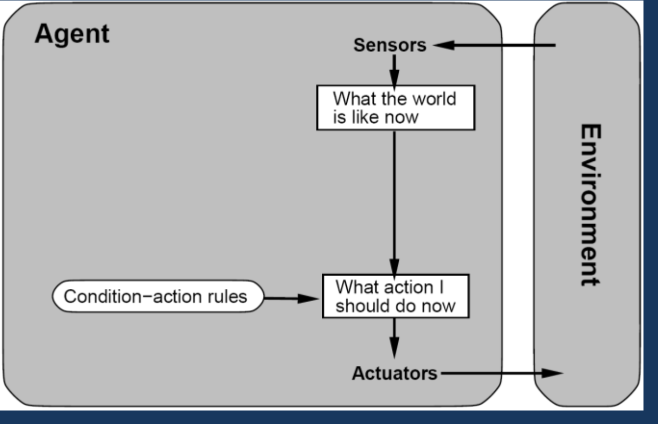
The Structure of Agents

- The job of AI is to design the agent program
- Agent architecture
- Agent = Architecture + Program

Agent Types

- Simple reflex agents
- Model-based reflex agents
- Goal-based reflex agents
- Utility-based agents
- All these agents can be converted into learning agents

Simple Reflex Agents

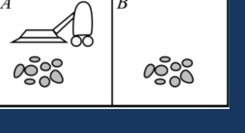


Reflex Vacuum Agent Program

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*

```
(setq joe (make-agent :name 'joe :body (make-agent-body)
                      :program (make-reflex-vacuum-agent-program))

(defun make-reflex-vacuum-agent-program ()
  #'(lambda (percept)
      (let ((location (first percept)) (status (second percept)))
        (cond ((eq status 'dirty) 'Suck)
              ((eq location 'A) 'Right)
              ((eq location 'B) 'Left)))))
```

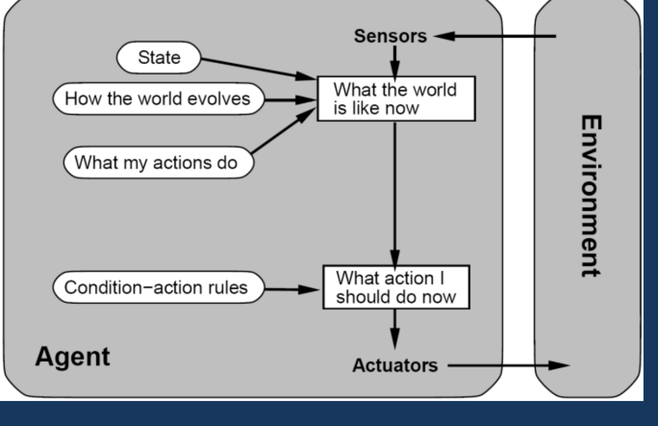


Simple-Reflex Agent Program

```
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

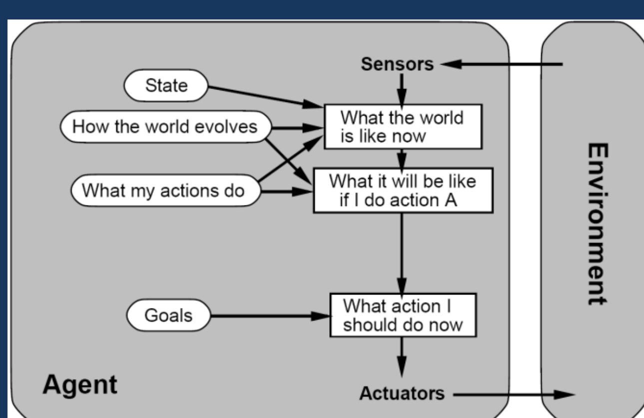


Model-based Agent Program

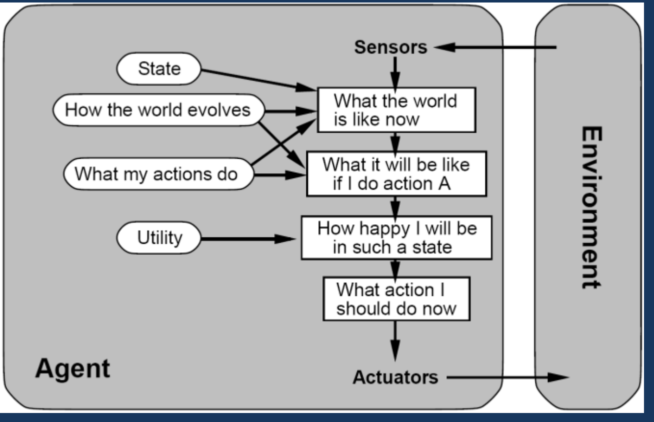
function REFLEX-AGENT-WITH-STATE(*percept*) **returns** an action
  static: *state*, a description of the current world state
    rules, a set of condition–action rules
    action, the most recent action, initially none

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

Goal-Based Agents



Utility-based Agents



Learning Agents