WPI

CS 4341
Introduction to Artificial Intelligence

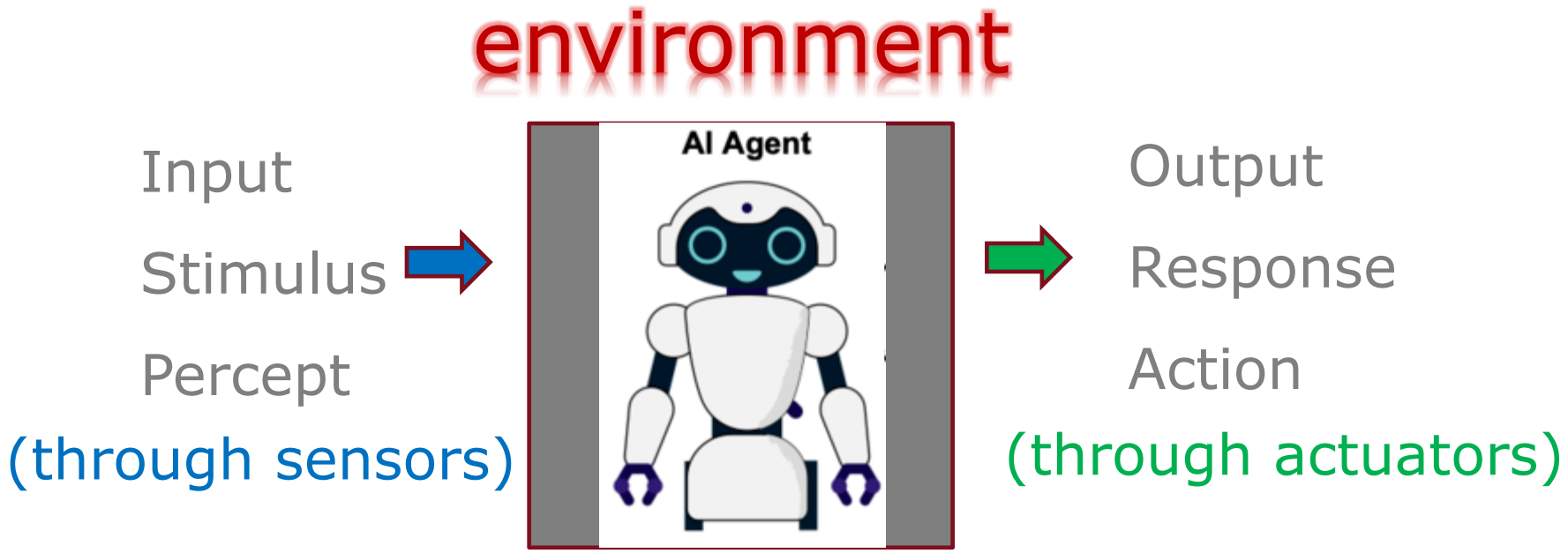
Lecture 2 and 3: Intelligent Agents

By

Ben C.K. Ngan

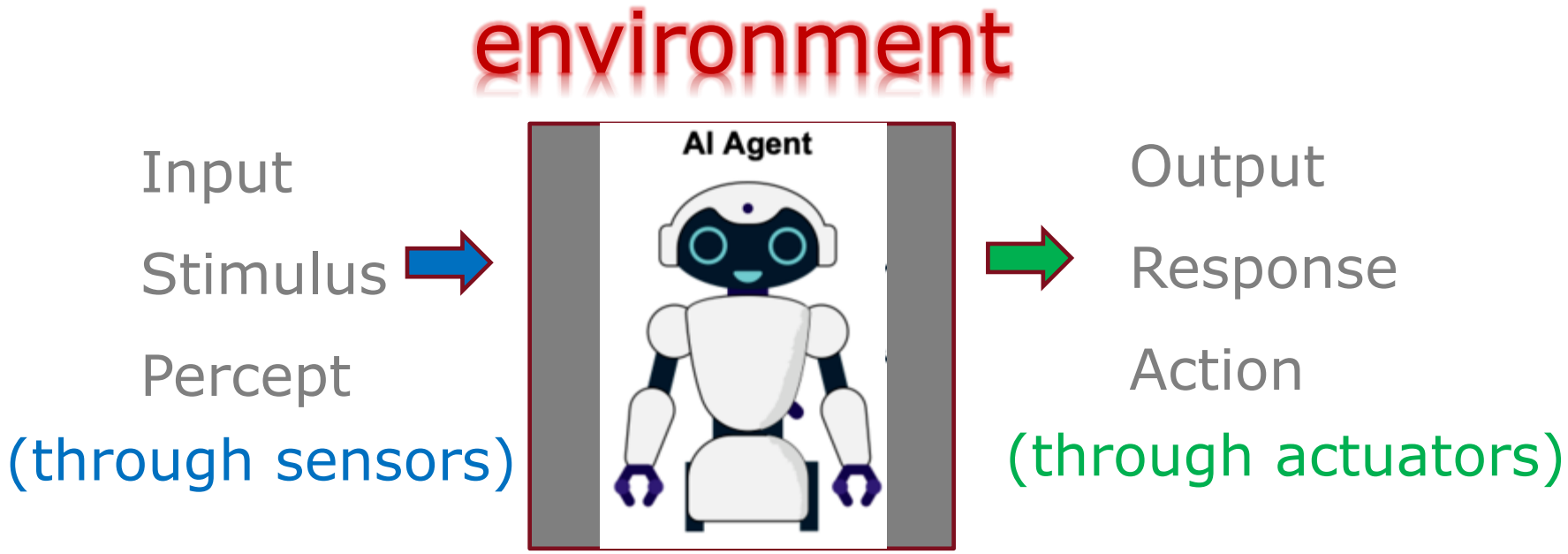
Worcester Polytechnic Institute

What is an Intelligent Agent?



- An **intelligent agent** (IA) is an entity (e.g., Robot, Chatbot, Self-driving Car, etc., to name a few) that includes a software program and a hardware architecture (sensors, mics, cameras, and actuators) to **perceive** (e.g., to see, smell, taste, hear, or touch) information from and **interact** with the environment, then **understand and think humanly/rationally/autonomously/adaptively** (i.e., performs computations), and finally decide to **take a sequence of decisions**, i.e., actions and responses, **humanly/rationally/autonomously/adaptively**, based upon the changes in its environment to maximize its **expected utility**, i.e., achieve a particular goal or optimize a particular objective function of the AI applications.
  - A loss function or a utility function is the **expected performance value** (e.g., accuracy, correctness, # of wins, etc.,) used to explicitly quantified each objective.
  - The **actual performance metric** (e.g., accuracy, correctness, # of wins, etc.,) is used to evaluate the effectiveness of IAs.

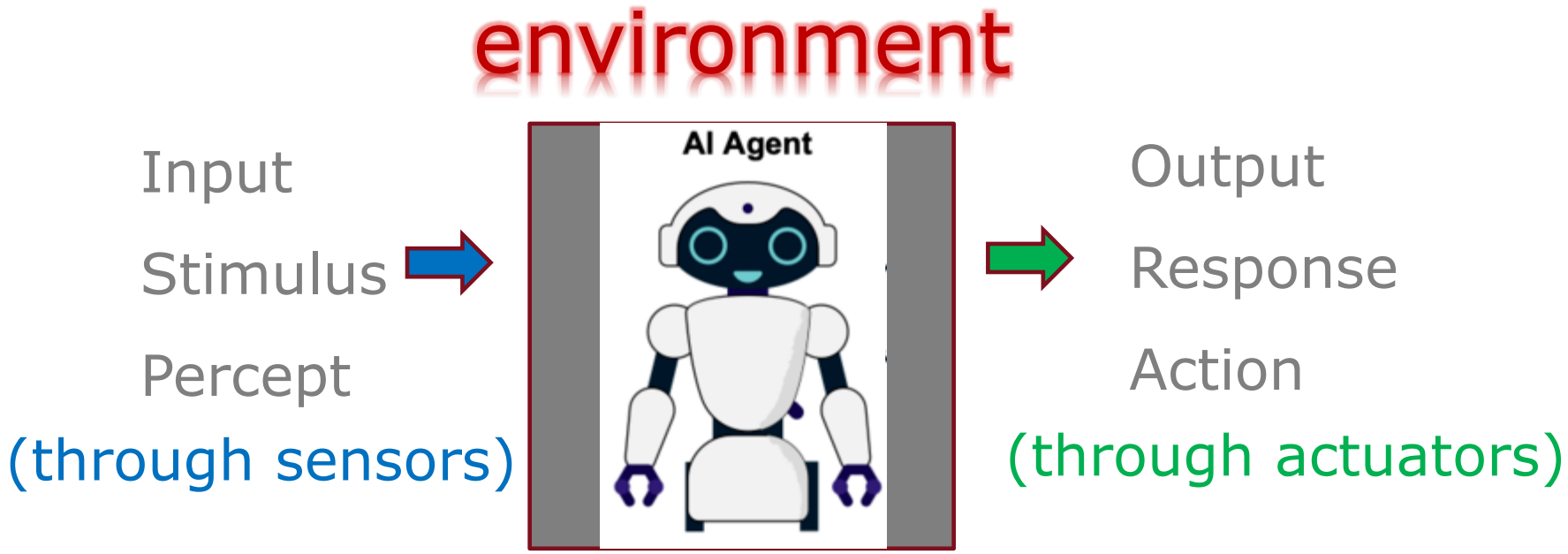
What is an Intelligent Agent?



- The Intelligent Agent **NOT ONLY** interacts with the environment **BUT ALSO** interacts with other Intelligent Agents and then reacts to the changes in its environment.

- An Intelligent Agent, like a robot, exhibits several of the properties given below:
  - Autonomous: Proactively initiates the activities
  - Mobile: Move itself from the current point to another
  - Adaptable: Adapt to a new changing environment
  - Knowledgeable: Reason about the information and knowledge
  - Collaborative: Communicate with other intelligent agents and work in a cooperative manner
  - Persistent: Retain their knowledge and **state** over an extended period of time.

What is an Intelligent Agent?



- In the context of an AI application, **a state** corresponds to **the internal/current configuration/situation of the all the variables and parameters of an intelligent agent that represents an environment to solve a problem at hand**.

  - Some examples are the current position of the robot, the piece position on a chess board in a chess-playing system, or the spatial location of the agent in a routing application.
  - Each next action of an agent leads to a transition to either the **same** state or a **different** state.
  - A transition may lead to **a reward/a penalty**, i.e., a utility, that directs the next actions of the agent.

What is an Intelligent Agent?

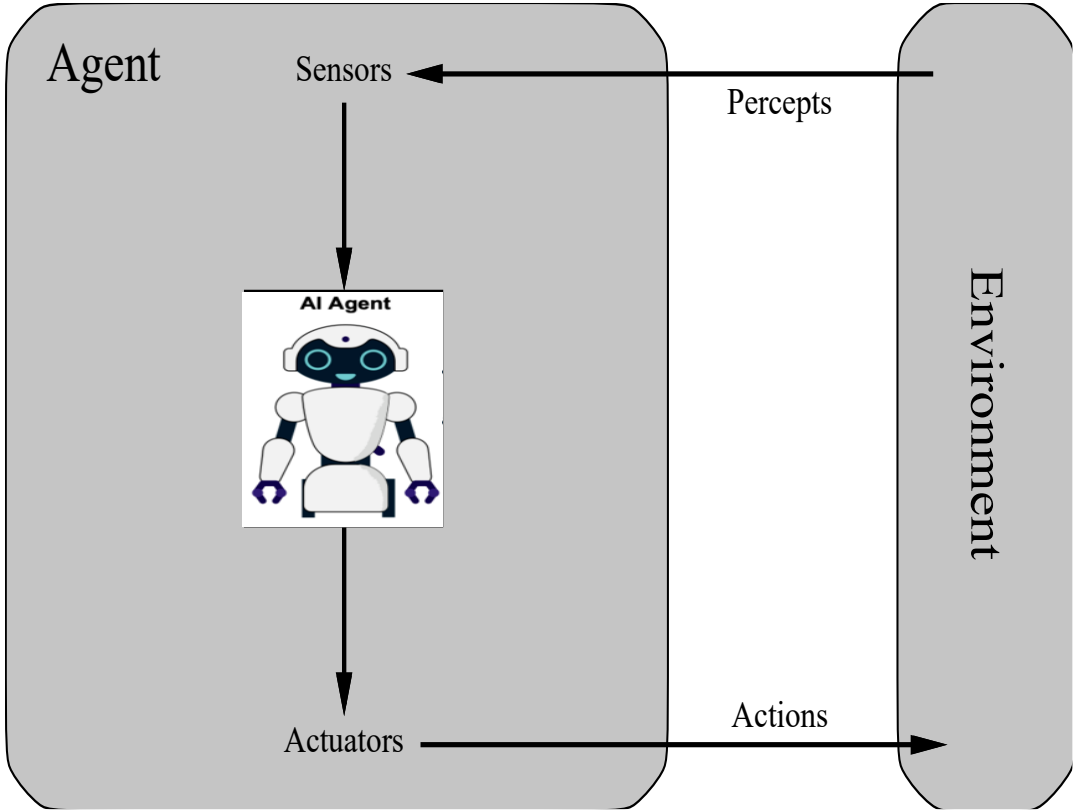
- Can perceive the state of the environment through its sensors
  - For humans, e.g., eyes, ears, noises, skins, etc.
  - For intelligent agents, e.g., motion cameras, mics, infrared range finders, etc.

- An intelligent agent's decision is described by the agent function that maps sensors (any given percept sequence) to actuators (an action), called the control policy for the agent.

- Can act upon its states through its actuators
  - For humans, e.g., hands, legs, mouths, vocal tract, etc.
  - For agents, e.g., motors, artificial limbs, speakers, etc.

- Decisions on which actuators to carry out based on the sensors are repeated in the loop

- The whole process is called Perception-Effect Cycle



What is an Intelligent Agent?

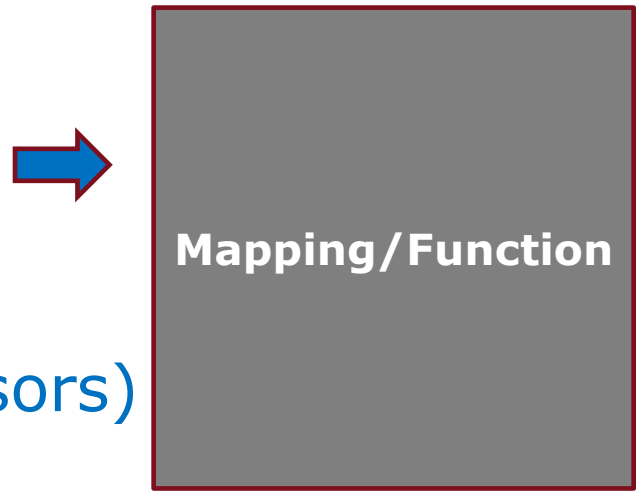
environment

Input

Stimulus

Percept

(through sensors)



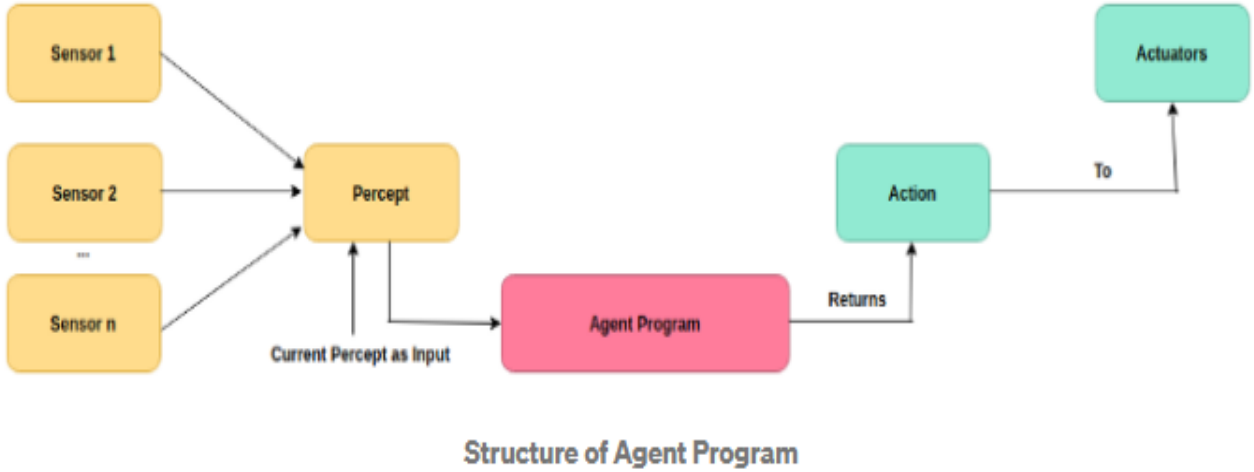
Output

Response

Action

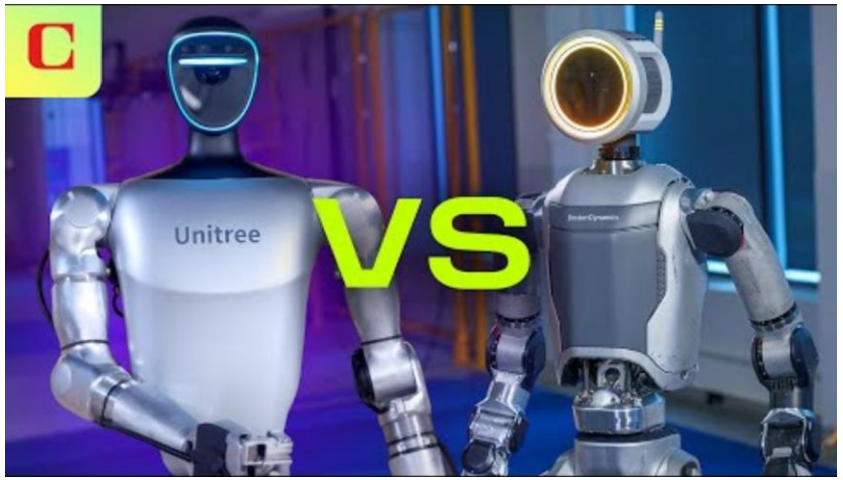
(through actuators)

- Intelligent Agent = Software Program + Hardware Architecture
- This agent mapping / function is implemented using an agent program that runs on an architecture, i.e., hardware.
- The program runs on a computing device with physical sensors and actuators, i.e., an agent architecture.



Structure of Agent Program

What is an Intelligent Agent?

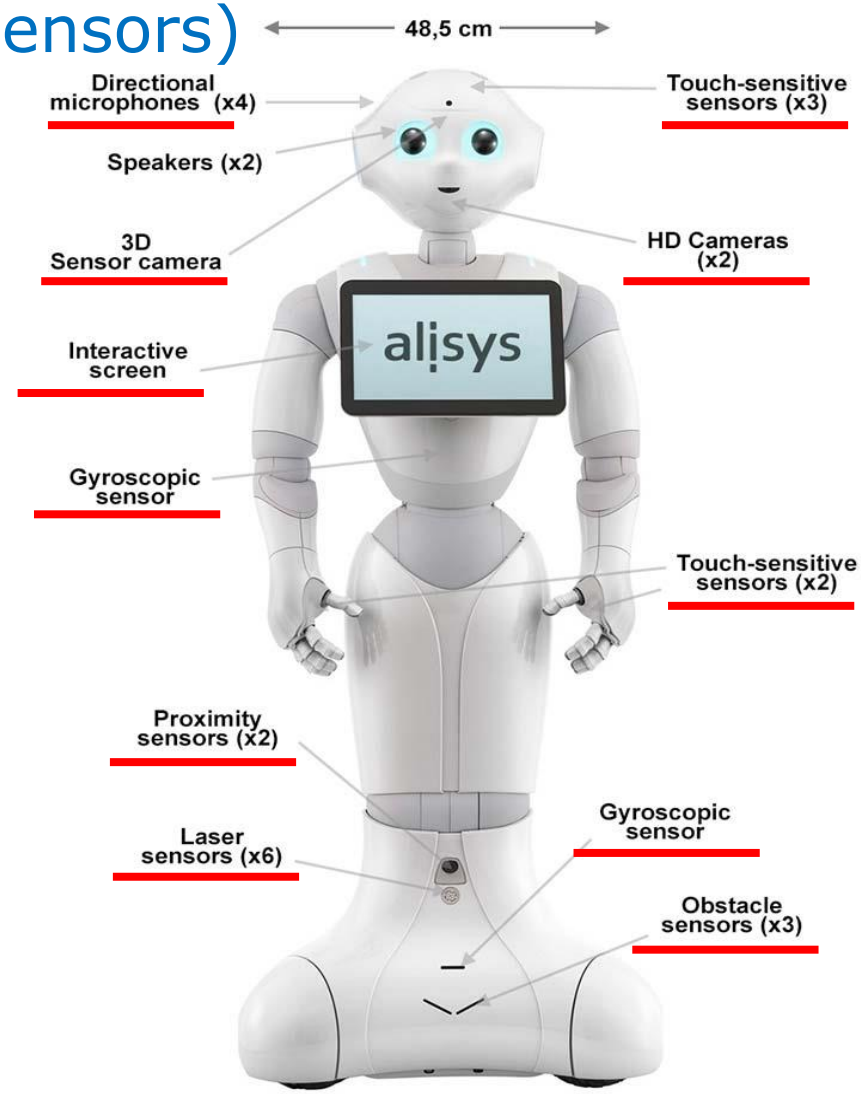


What is an Intelligent Agent?

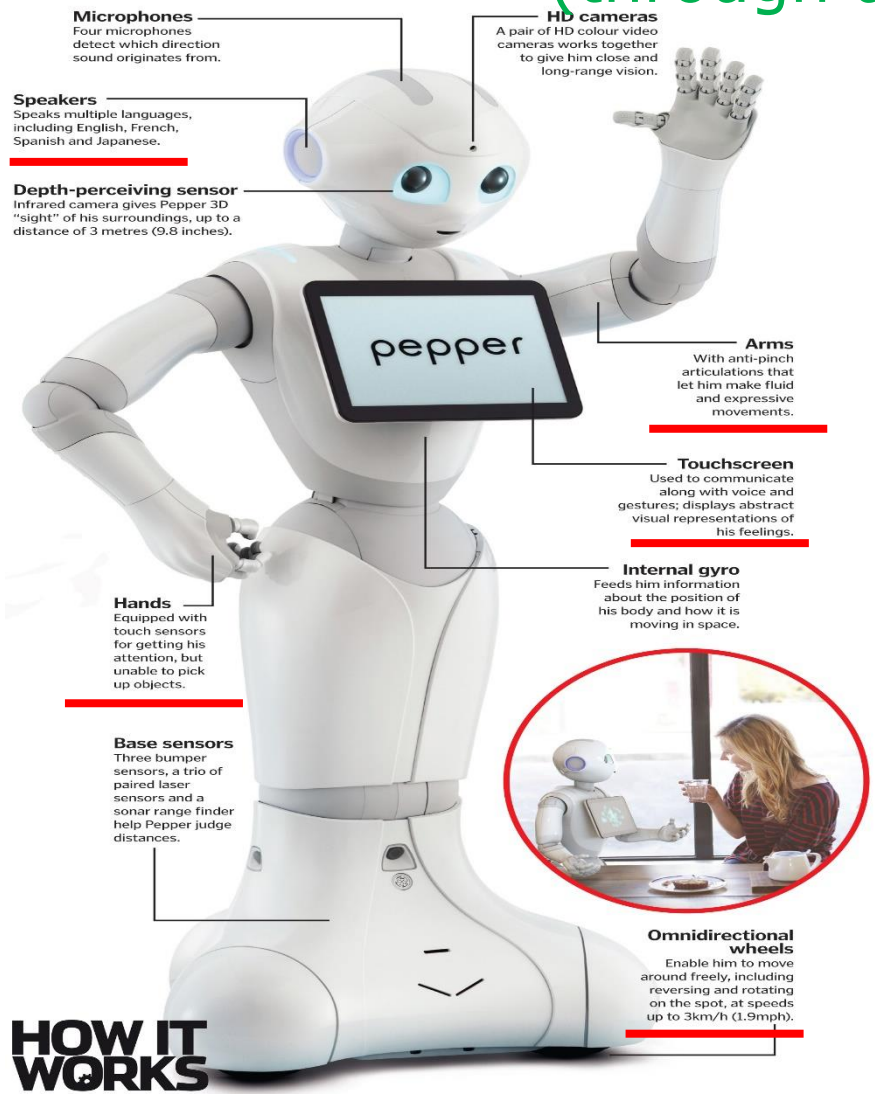
What is an Intelligent Agent?

What is an Intelligent Agent?

(through sensors)



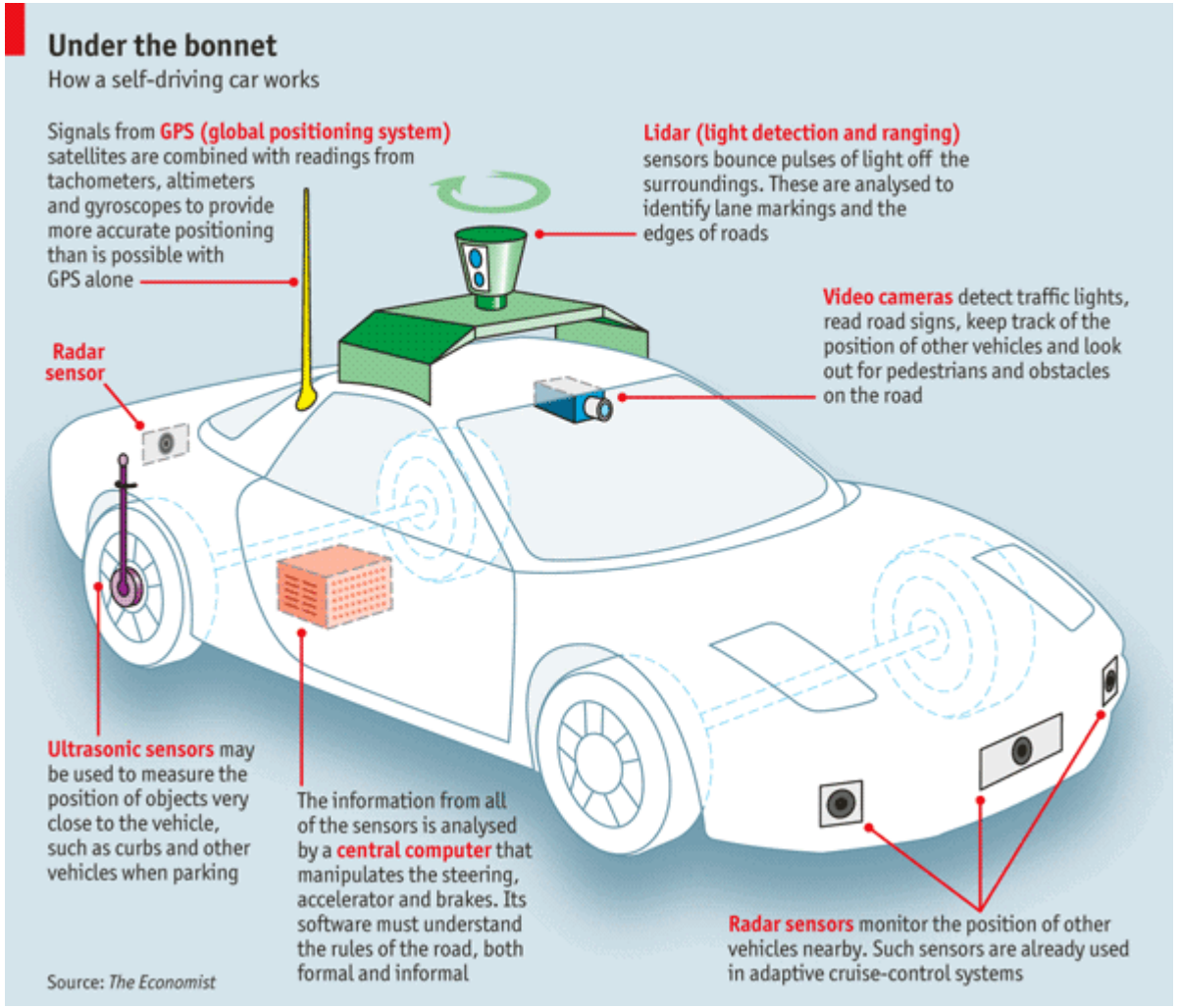
(through actuators)

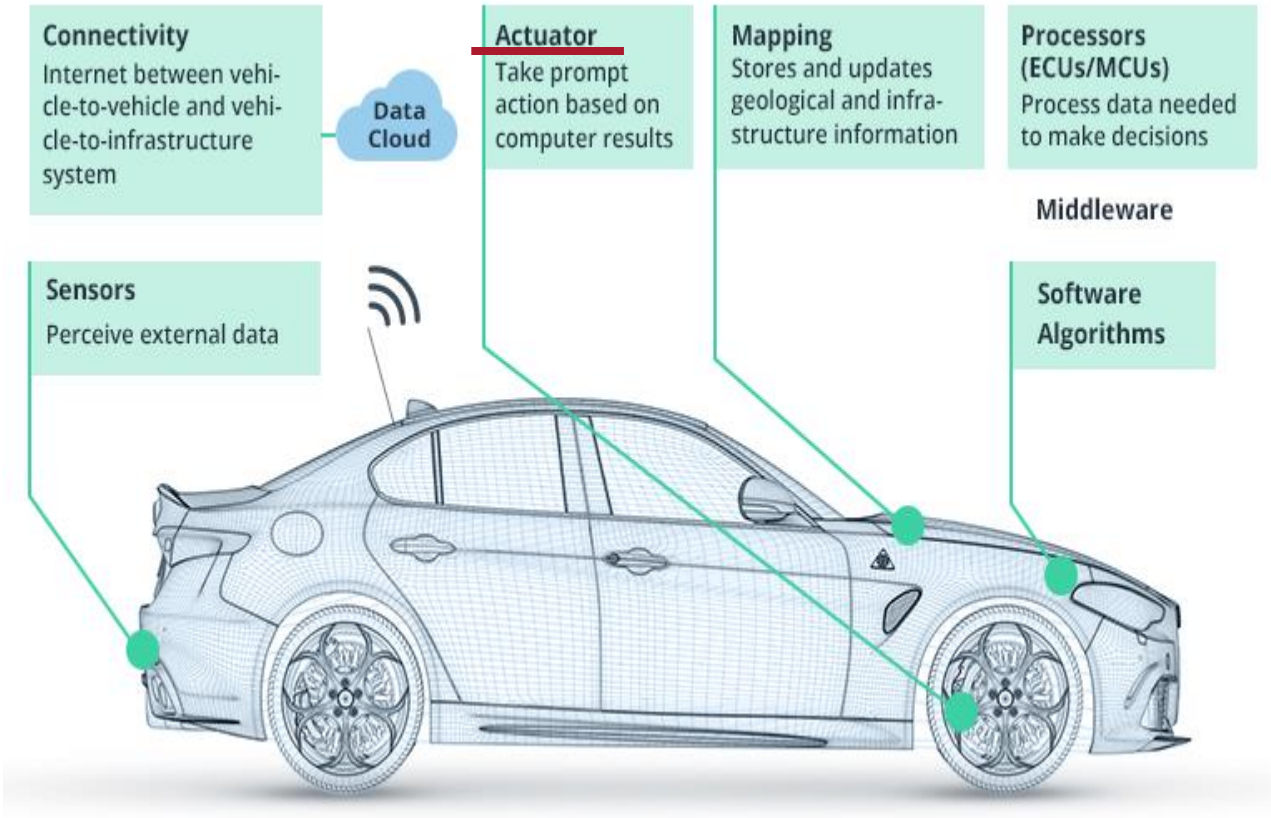


ic Institute

What is an Intelligent Agent?

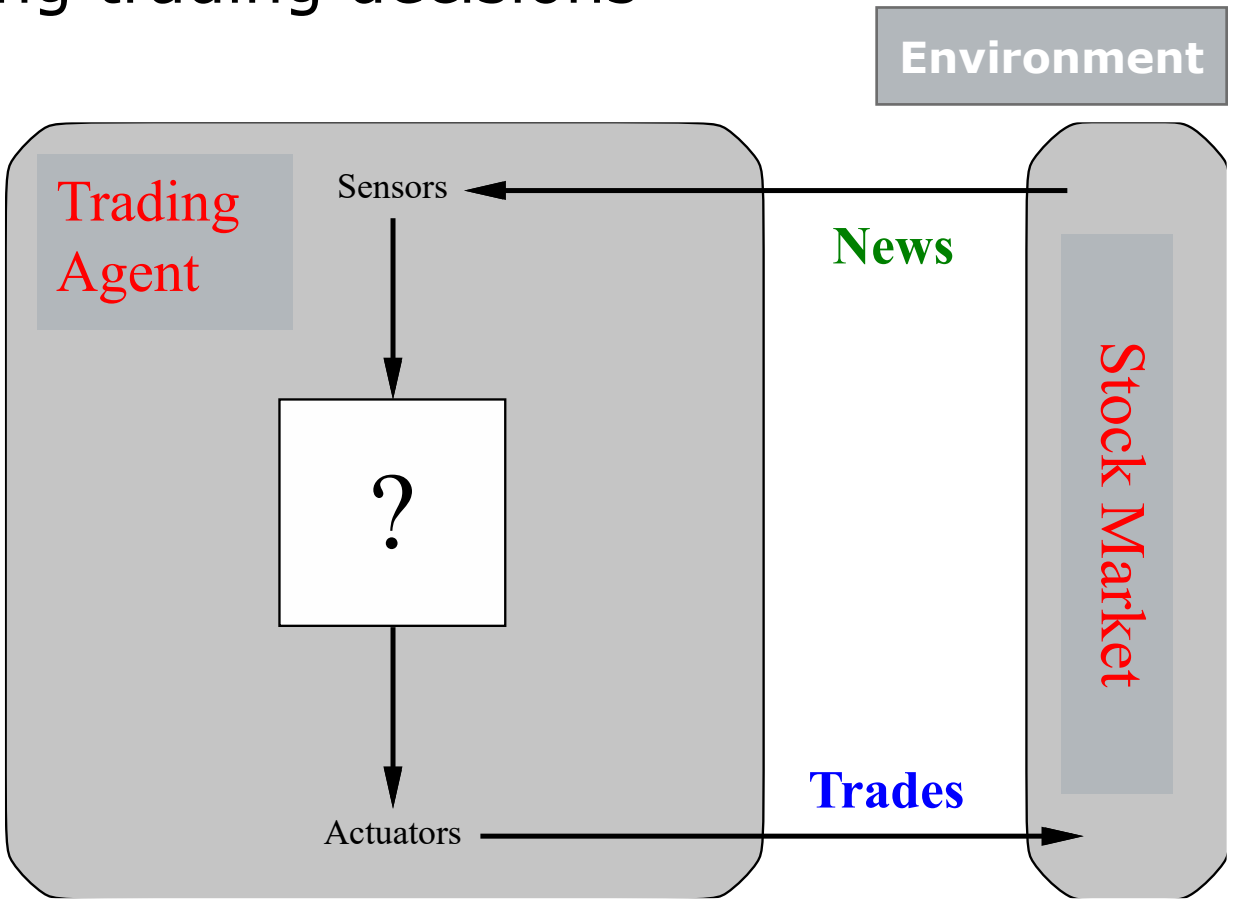
What is an Intelligent Agent?





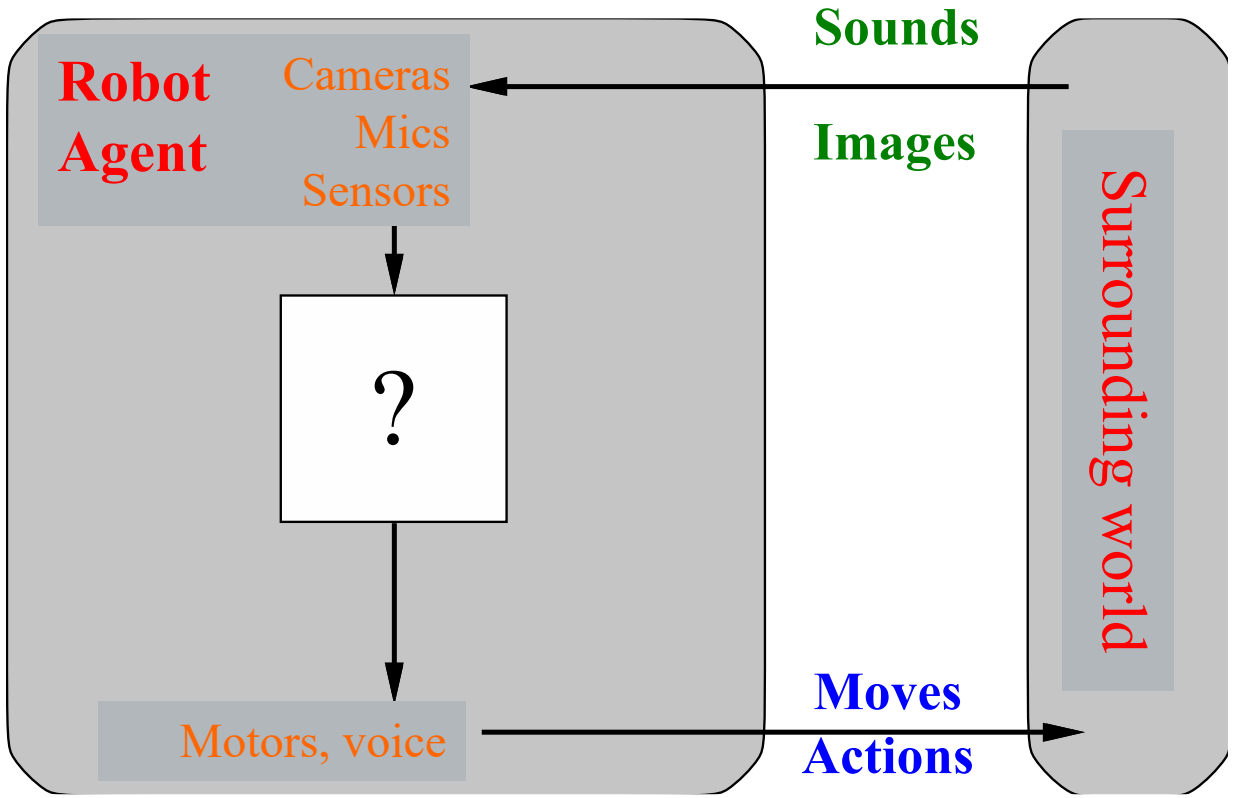
More Specific Example: AI in Finance

Trading Agent: Making trading decisions



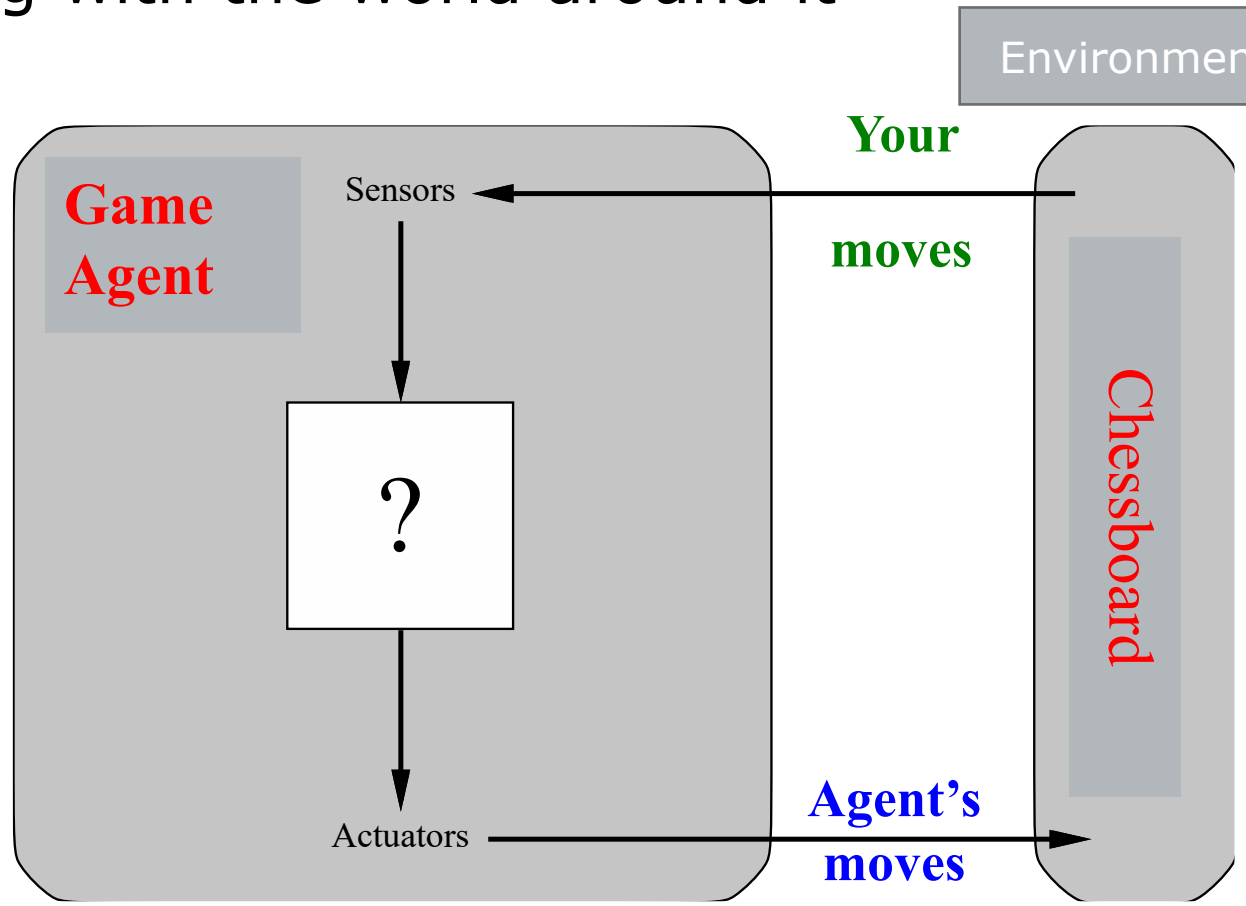
More Specific Example: AI in Robotics

Robot Agent: Interacting with the world around it

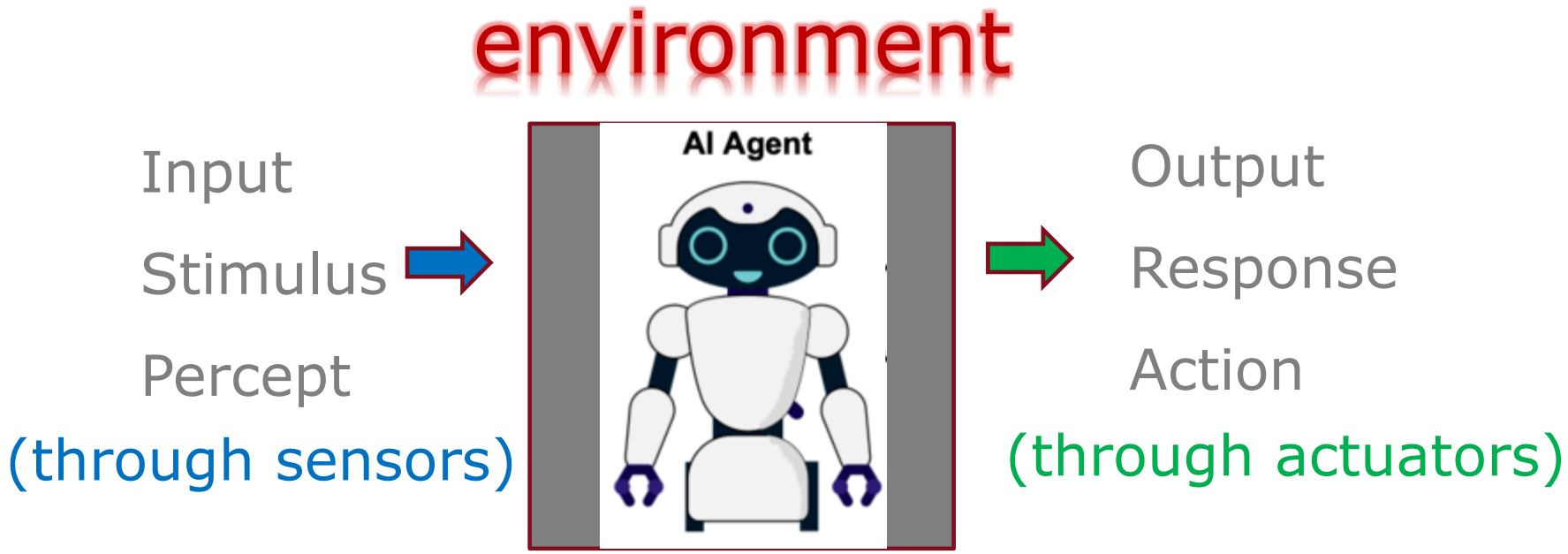


More Specific Example: AI in Games

Game Agent: Interacting with the world around it



Types of Environments



- An environment could be everything that affects what an intelligent agent perceives and that is affected by the agent's actions.
  - ❑ Fully Observable or Partially Observable
  - ❑ Deterministic or Non-deterministic (Stochastic)
  - ❑ Episodic or Sequential
  - ❑ Static or Dynamic
  - ❑ Discrete or Continuous
  - ❑ Single Agent or Multiple Agents

Fully Observable / Partially Observable

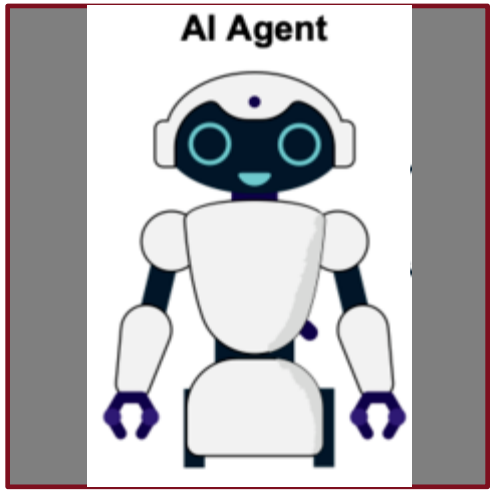
environment

Input

Stimulus

Percept

(through sensors)



Output

Response

Action

(through actuators)

- Fully Observable is that an intelligent agent's sensors give it access to **complete state** of the environment **at each point in time**, or else it is partially observable.
- **Chess Game's environment** is **fully observed**: An AI player gets to see the entire chess board.
- **Poker's environment** is **partially observed**: An AI player gets to see only his own cards and the cards on the table, but not the cards of everyone in the game.

Deterministic / Non-Deterministic (Stochastic)

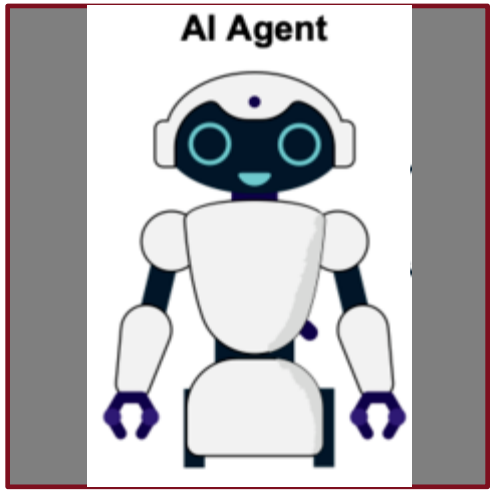
environment

Input

Stimulus

Percept

(through sensors)



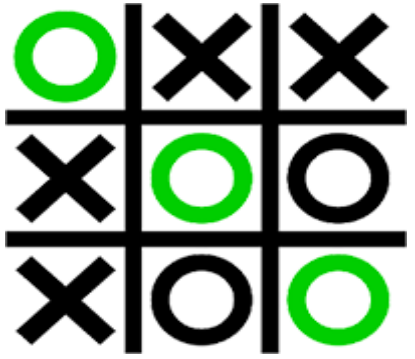
Output

Response

Action

(through actuators)

- Deterministic is that if **the next state of the environment** is completely determined by the current state and the next actions of the agent, or else it is non-deterministic (stochastic).
- **Tic Tac Toe's environment** is **Deterministic**
- **Self-driving Car's environment** is **Non-deterministic**



Episodic / Sequential

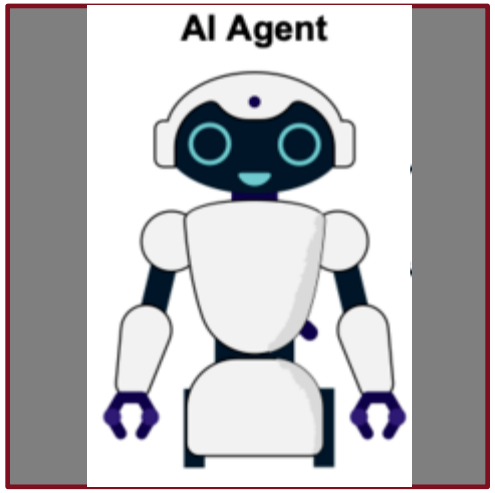
environment

Input

Stimulus

Percept

(through sensors)



Output

Response

Action

(through actuators)

- Episodic: Each single action performed by an intelligent agent depends on only the current atomic episode itself, independent of the previous episode, e.g., **Mail Sorting Machine**
- Sequential: Each next action is affected by the current action, e.g., **Chess Game**

Static / Dynamic

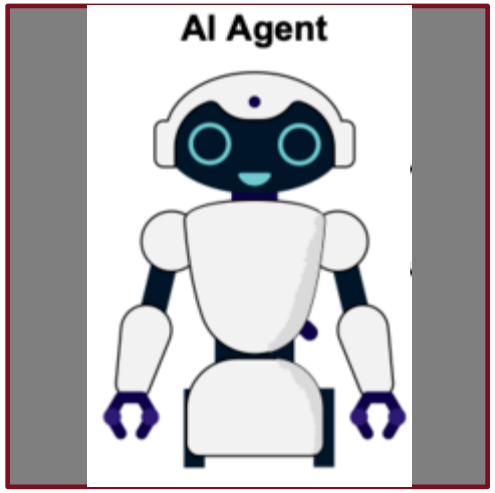
environment

Input

Stimulus

Percept

(through sensors)



Output

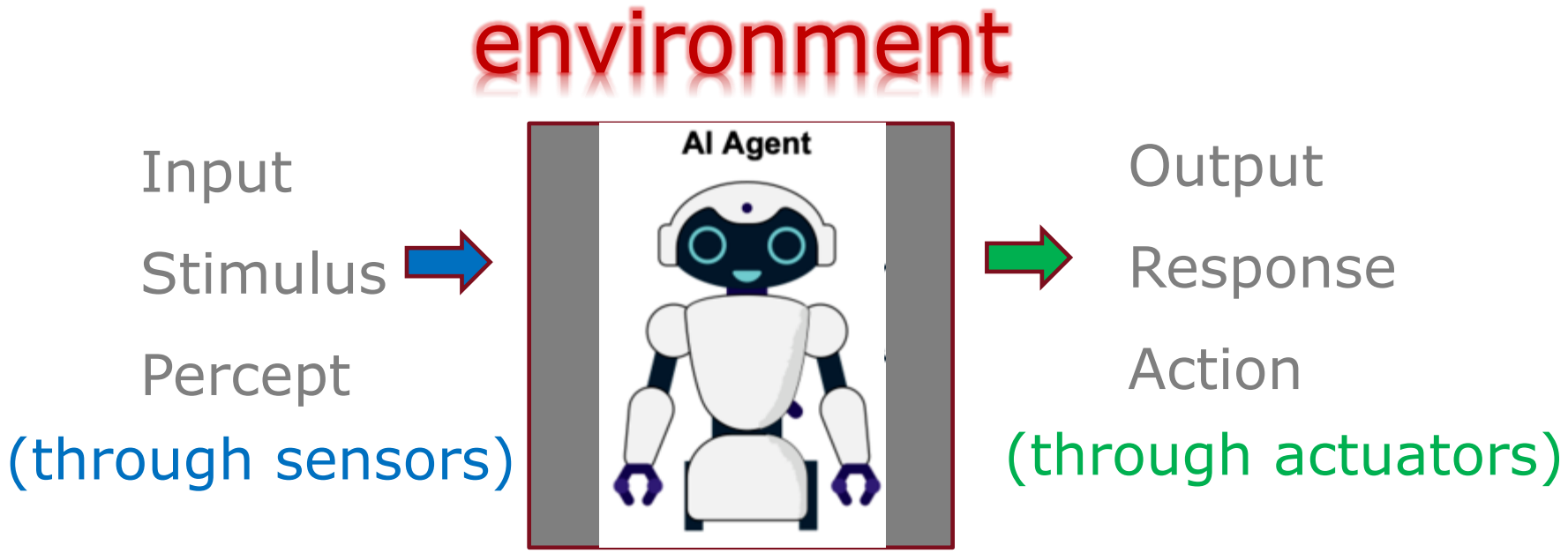
Response

Action

(through actuators)

- Static is that if the environment **does not change over time constantly** while an agent is acting, or else it is dynamic.
- Static: **Crossword Puzzles**
- Dynamic: **Self-driving Taxi**.

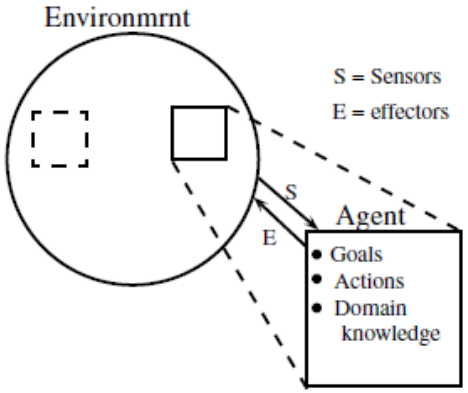
Discrete / Continuous



- Discrete: If there are **a limited number of distinct, clearly defined states** of the environment (percepts and actions), the environment is discrete, e.g., **Chess Game** – a set of numbers of discrete/distinct moves of black and white pieces

- Continuous: Signals **constantly and periodically** coming into sensors, actions continually **changing**, e.g., **Self-driving car**, accordingly.

Single Agent / Multiple Agents



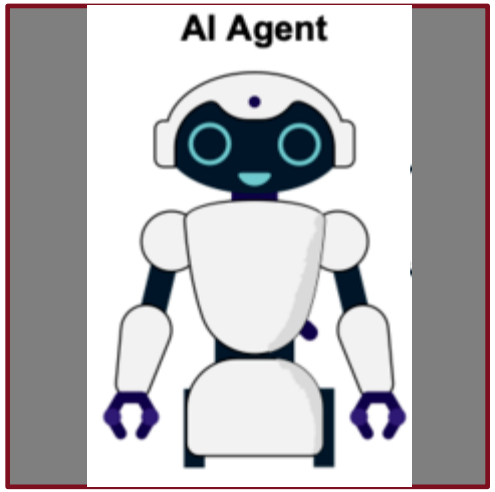
environment

Input

Stimulus

Percept

(through sensors)

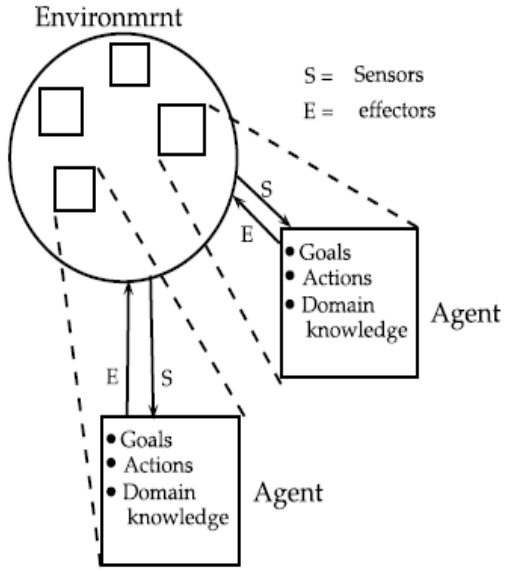


Output

Response

Action

(through actuators)



- An AI agent operating by itself in **an** environment is a **single agent**, e.g., Crossword Puzzle.
- **Multi-agents** is when other AI agents are present, e.g., other players in a soccer team or the opposing team.

More AI Agent Examples

Agent	Sensor	Actuator	Objective	State	Environment
Cleaning Robot	Camera Joint sensor	Limbs Joints	Cleanliness evaluation	Object/joint positions	Cleaned space
Chess agent	Board input interface	Move output interface	Position evaluation	Chess position	Chess board
Self-driving car	Camera Sound sensor	Car control interface	Driving safety and goals	Speed/Position in traffic	Traffic conditions
Chatbot	Keyboard	Screen	Chat evaluation	Dialog history	Chat participants

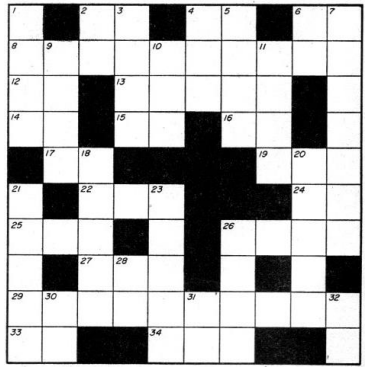
	Fully observable?	Deterministic?	Episodic?	Static?	Discrete?	Single agent?
Solitaire	No	Yes	Yes	Yes	Yes	Yes
Backgammon	Yes	No	No	Yes	Yes	No
Taxi driving	No	No	No	No	No	No
Internet shopping	No	No	No	No	Yes	No
Medical diagnosis	No	No	No	No	No	Yes

Environment	Accessible	Deterministic	Episodic	Static	Discrete
Chess with a clock	Yes	Yes	No	Semi	Yes
Chess without a clock	Yes	Yes	No	Yes	Yes
Poker	No	No	No	Yes	Yes
Backgammon	Yes	No	No	Yes	Yes
Taxi driving	No	No	No	No	No
Medical diagnosis system	No	No	No	No	No
Image-analysis system	Yes	Yes	Yes	Semi	No
Part-picking robot	No	No	Yes	No	No
Refinery controller	No	No	No	No	No
Interactive English tutor	No	No	No	No	Yes

Task Environments: What Type?

Task Environment	Observable	Agent	Determ.	Episodic	Static	Discrete
Crossword Puzzle	Fully	S	Determ.	Sequen.	Static	Discrete
Chess w. a Clock	Fully	M	Determ.	Sequen.	Semi	Discrete
Poker	Partially	M	Stoch.	Sequen.	Static	Discrete
Backgammon	Fully	M	Stoch.	Sequen.	Static	Discrete
Taxi Driving	Partially	M	Stoch.	Sequen.	Dynamic	Continuous
Medical Diagnostics	Partially	S	Stoch.	Sequen.	Dynamic	Continuous
Image Analysis	Fully	S	Determ.	Episod.	Semi	Continuous
English Tutor	Partially	M	Stoch.	Sequen.	Dynamic	Discrete

Task Environments: More Examples

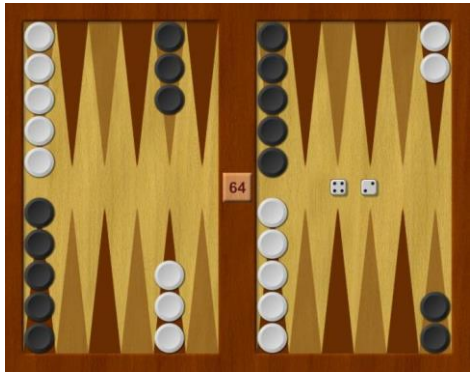


crossword

chess with timer

poker

autonomous taxi



backgammon

medical diagnostics

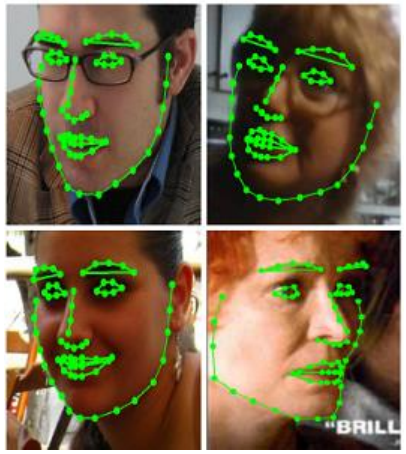


image processing

interactive English tutor

Class Exercise 1

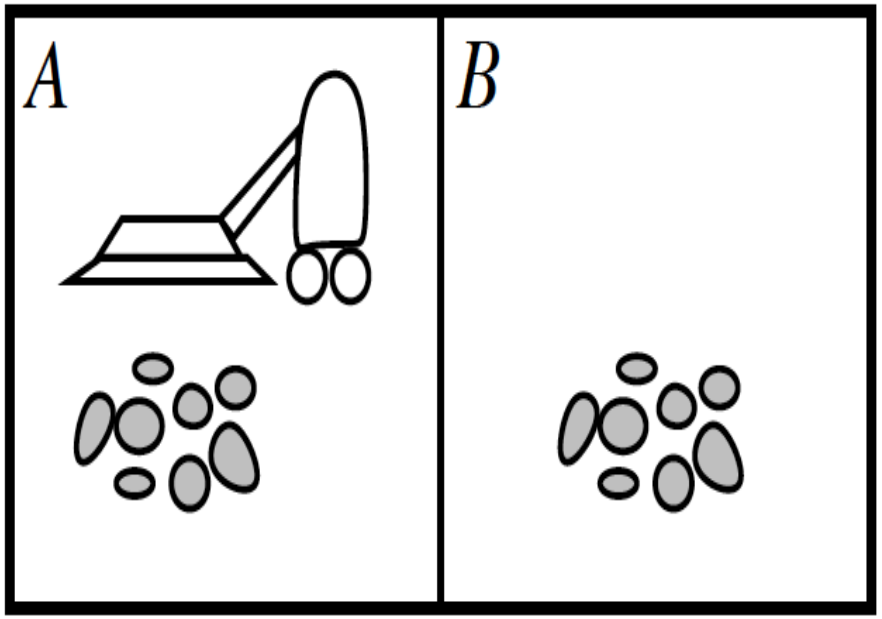
- Playing a table tennis match.

- Knitting a sweater.

- Fully Observable or Partially Observable
- Deterministic or Non-deterministic (Stochastic)
- Episodic or Sequential
- Static or Dynamic
- Discrete or Continuous
- Single Agent or Multiple Agents

The Vacuum Agent in the Reference Textbook

Vacuum Agent



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

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

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

```
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
```

Define it Implement it

An agent is **a function *F*** from percept environments to actions, i.e., F: $P^* \rightarrow A$

Vacuum Agent

https://github.com/aimacode/aima-python/blob/master/agents4e.py

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*

Define it

Implement it

```
def ReflexVacuumAgent():
    """
    [Figure 2.8]
    A reflex agent for the two-state vacuum environment.
    >>> agent = ReflexVacuumAgent()
    >>> environment = TrivialVacuumEnvironment()
    >>> environment.add_thing(agent)
    >>> environment.run()
    >>> environment.status == {(1,0):'Clean' , (0,0) : 'Clean'}
    True
    """

    def program(percept):
        location, status = percept
        if status == 'Dirty':
            return 'Suck'
        elif location == loc_A:
            return 'Right'
        elif location == loc_B:
            return 'Left'

    return Agent(program)
```

An agent is **a function *F*** from percept environments to actions, i.e., F: $P^* \rightarrow A$

Definition of a Rational Agent

Definition: For each possible percept sequence, a rational agent should select the next optimal action that is **expected** to maximize its performance measure, i.e., the expected utility value, given the evidence provided by the percept sequence and whatever built-in knowledge the agent has at any given time.

- A utility function maps a state (or a sequence of states) onto an evaluation value (usually a real number), e.g., cost, profit, score, etc.

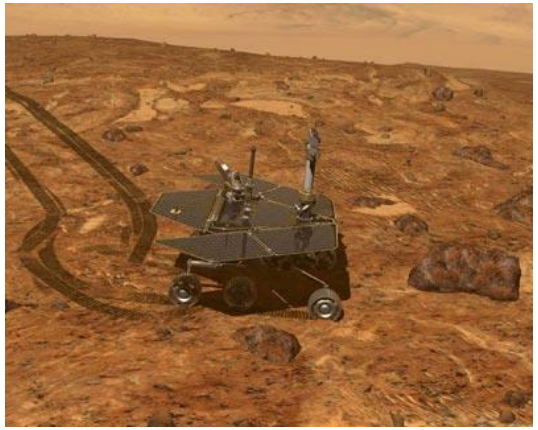
$$\text{Utility} \times \text{Percepts} \times \text{Knowledge} \rightarrow \text{Action}$$

- A selected action is optimal if it takes the agent to a state of maximum expected utility given available percepts and knowledge. The agent is rational if it always chooses optimal actions

- Rational is related to exploration, learning, and autonomy

Exploration, Learning, and Autonomy

Exploration: Information gathering in an initially unknown environment



Learning: Updating the knowledge about the environment during perception

Exploration + Learning:

- Maximizing the expected performance
- Doing actions in order to modify future percepts

Autonomy: Extend to which an agent relies on the prior knowledge of its designer – lacks autonomy.

Task Environments

What is the **first step** in designing **a rational AI agent**? That is to specify the task environment.

The task environment is defined using **PEAS** descriptors:

- **P**erformance measure
- **E**nvironment
- **A**ctuators
- **S**ensors

Example: Pepper

Performance Measure	Number of correctly answered questions
Environment	Rigi Kaltbad Station Shop
Actuators	loud speaker
Sensors	camera, microphones

Example: Internet Shopping Agent

Performance measures: 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)

Example: Internet Shopping Agent

Institute

Example: AI Healthcare Agent

- 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)

Example: AI Healthcare Agent

Example: Automated Taxi

Performance measures: safety, destination, profits, legality, comfort, *etc*.

Environment: US streets/freeways, traffic, pedestrians, customers, weather, *etc*.

Actuators: steering, accelerator, brake, horn, speakers, display, *etc*.

Sensors: cameras, accelerometers, gauges, engine sensors, keyboard, GPS, *etc*.

Waymo One: Google-launched autonomous taxi service

Example: Automated Taxi

Other PEAS Examples

Agent Type	Performance Measure	Environment	Actuators	Sensors
Medical diagnosis system	healthy patient, costs, lawsuits	patient, hospital, stuff	display 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 categorization of scene	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 exercises, suggestions, corrections	keyboard entry

Class Exercise 2

- AI-Powered Microwave Oven.

- Autonomous Supply Delivery Plane.

- Performance Measure
- Sensors
- Actuators
- Environment

Agents: Types of Agents (Programs)

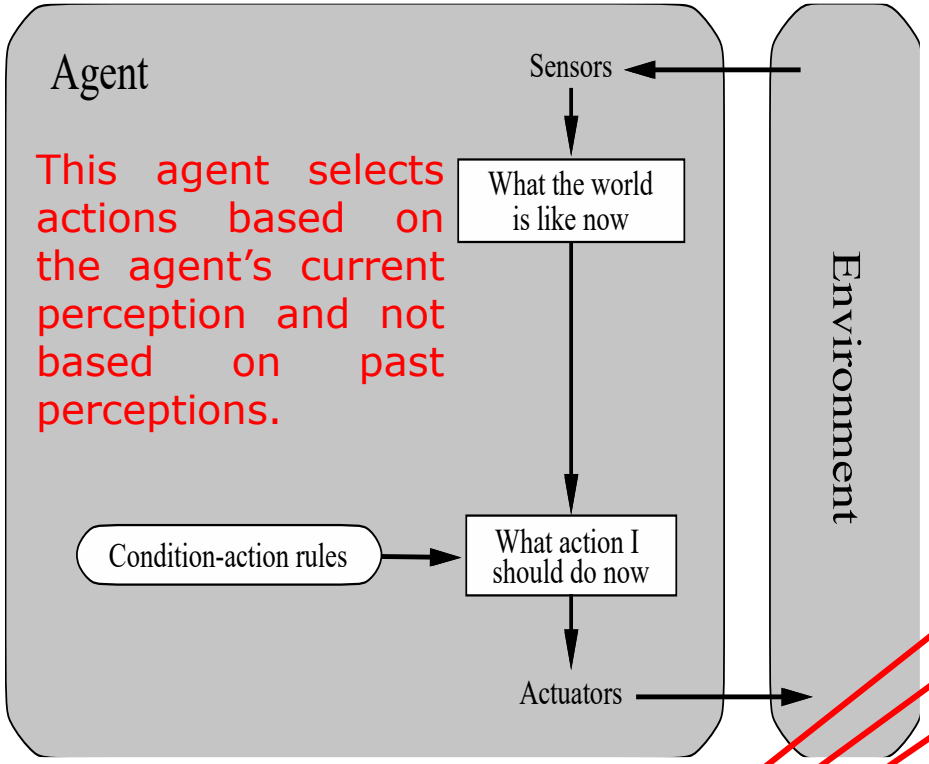
- **Simple Reflex (Reactive) Agents:** an agent whose action depends only on the current percept.
- **Model–based Reflex Agents**: an agent whose action is derived directly from an internal model of the current world state that is updated over time.
- **Goal-based Agents**: an agent that selects actions that it believes will achieve explicitly represented goals.
- **Utility-based Agents**: an agent that selects actions that it believes will maximize the expected utility of the outcome state.
- **Learning Agents:** an agent whose behavior/action improves over time based on its experience.

https://github.com/aimacode/aima-python/blob/master/agents4e.py

Simple Reflex (Reactive) Agent

INTERPRET-INPUT function generates an abstracted description of the **current state** from the percept.

- Senses the world and responds immediately based on the **simple rules whose condition matches the current state** based on percept and then acts accordingly.
- **No explicit world model, no "memory"**
- This agent will only work if the environment is fully observable.
- Agent: Mail sorting robot
- Environment: Conveyor belt of letters
- Rule: *e.g. if* city = Worcester, then put Massachusetts bag



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*

RULE-MATCH function returns the **1st rule in the set of rules** that matches the given state description.

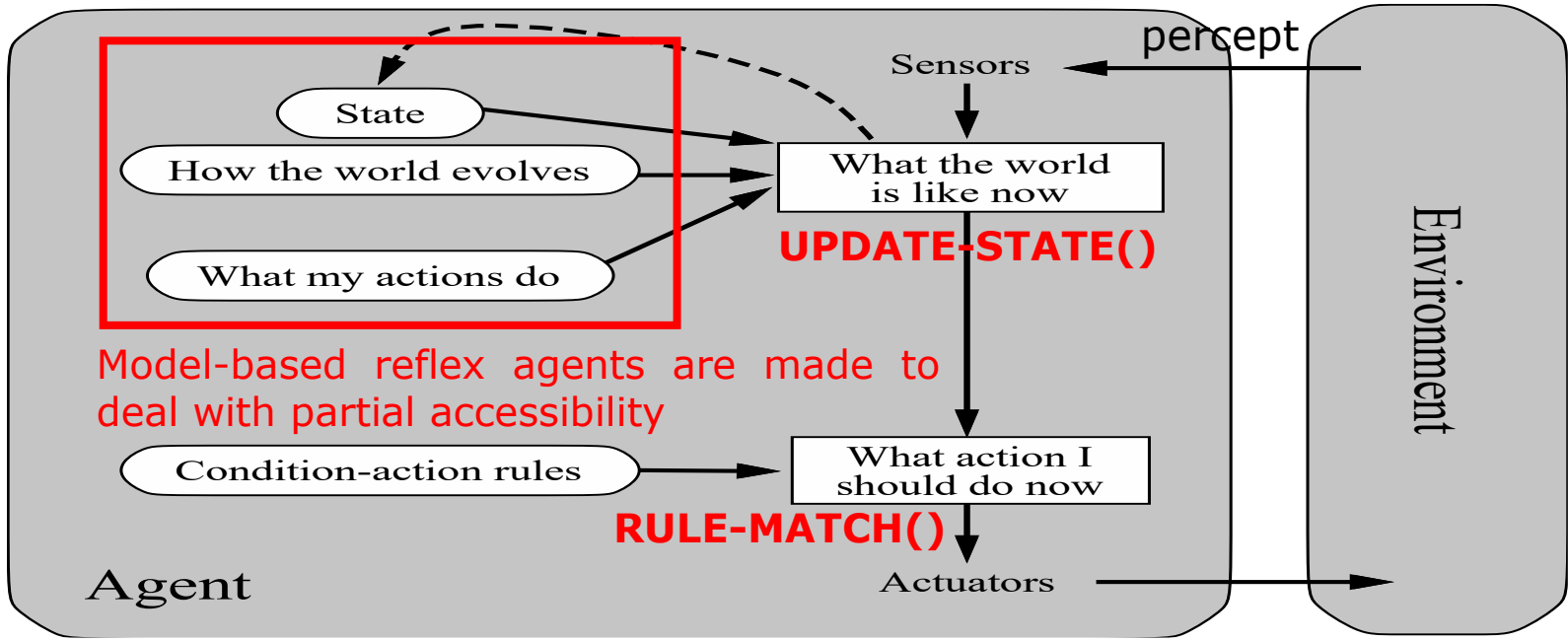
```
def SimpleReflexAgentProgram(rules, interpret_input):
    """
    [Figure 2.10]
    This agent takes action based solely on the percept.
    """

    def program(percept):
        state = interpret_input(percept)
        rule = rule_match(state, rules)
        action = rule.action
        return action

    return program
def rule_match(state, rules):
    """Find the first rule that matches state."""
    for rule in rules:
        if rule.matches(state):
            return rule
```

Model-based Reflex Agents

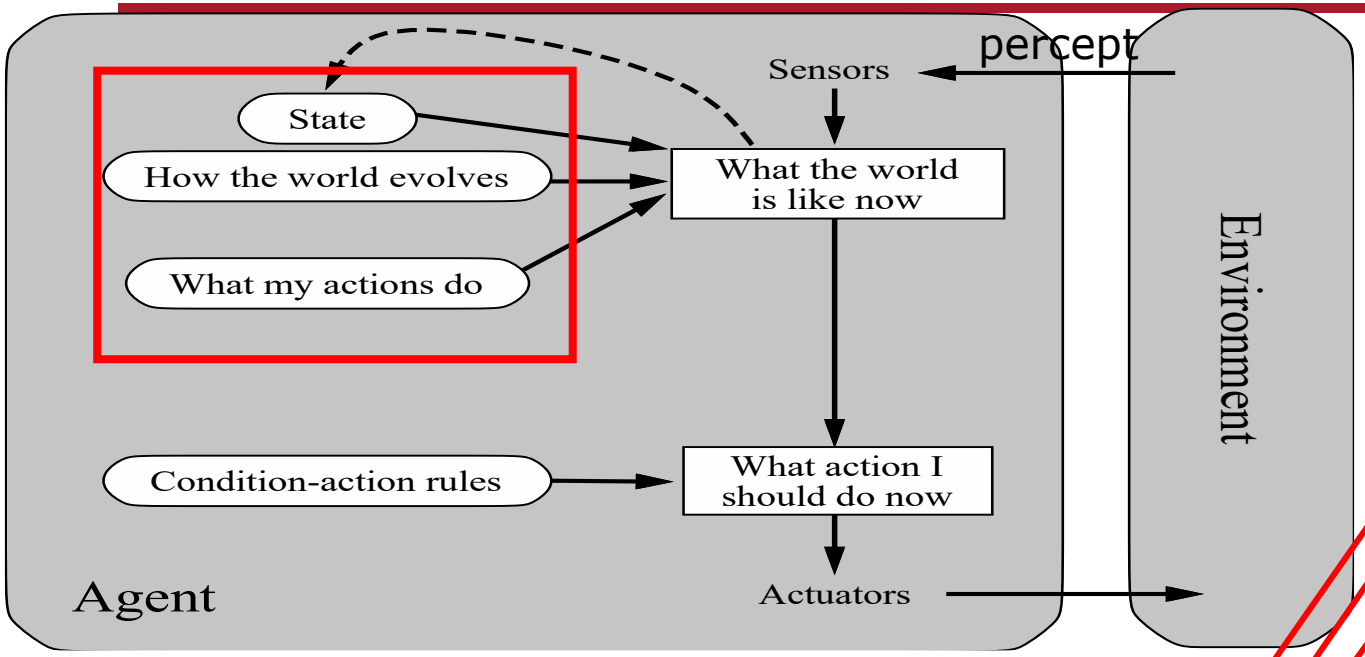
- Keep track of **the previous state** of the world that the agent cannot perceive right now, i.e., **the internal world model (agent state) that depends on what it has seen before, so it holds information on the unobserved aspects of the current state.**
- Uncertainty about the world state is unavoidable because of limited sensing capabilities and the limited world model that represents **the agent's "best guess" of the world state, the evolution of the world and the effects of its actions**
- New Current State = Previous State + Current Percept
- Action may depend on the new current state.



function MODEL-BASED-REFLEX-AGENT(*percept*) **returns** an action
  persistent: *state*, the agent's current conception of the world state
    transition_model, a description of how the next state depends on the current state and action
    sensor_model, a description of how the current world state is reflected in the agent's percepts
    rules, a set of condition–action rules
    action, the most recent action, initially none

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

Model-based Reflex Agents



function MODEL-BASED-REFLEX-AGENT(*percept*) **returns** an action
  persistent: *state*, the agent's current conception of the world state
    transition_model, a description of how the next state depends on the current state and action
    sensor_model, a description of how the current world state is reflected in the agent's percepts
    rules, a set of condition–action rules
    action, the most recent action, initially none

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

```
def ModelBasedReflexAgentProgram(rules, update_state, transition_model, sensor_model):
    """
    [Figure 2.12]
    This agent takes action based on the percept and state.
    """
    def program(percept):
        program.state = update_state(program.state, program.action, percept, transition_model,sensor_model)
        rule = rule_match(program.state, rules)
        action = rule.action
        return action
    program.state = program.action = None
    return program
```

https://github.com/aimacode/aima-python/blob/master/agents4e.py

RULE-MATCH function returns the **1st rule in the set of rules** that matches the given state description.

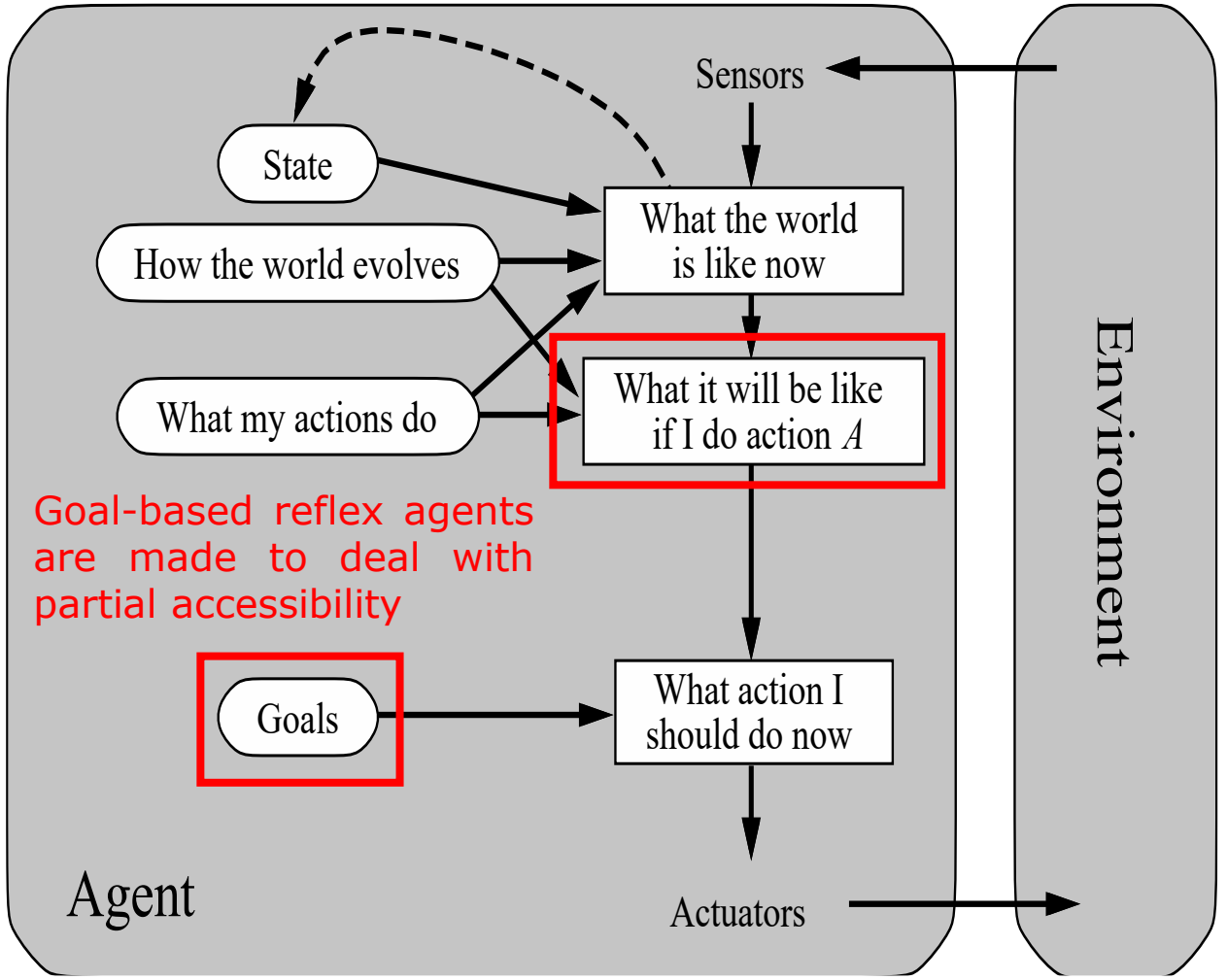
```
def rule_match(state, rules):
    """Find the first rule that matches state."""
    for rule in rules:
        if rule.matches(state):
            return rule
```

Goal-based Reflex Agents

- Builds a model of the world and uses an explicit representation of goals
- **Evaluates the effects/impacts of actions on the world** model before selecting an action
- This agent **reviews many actions** and chooses the **first** available one which can achieve its goals

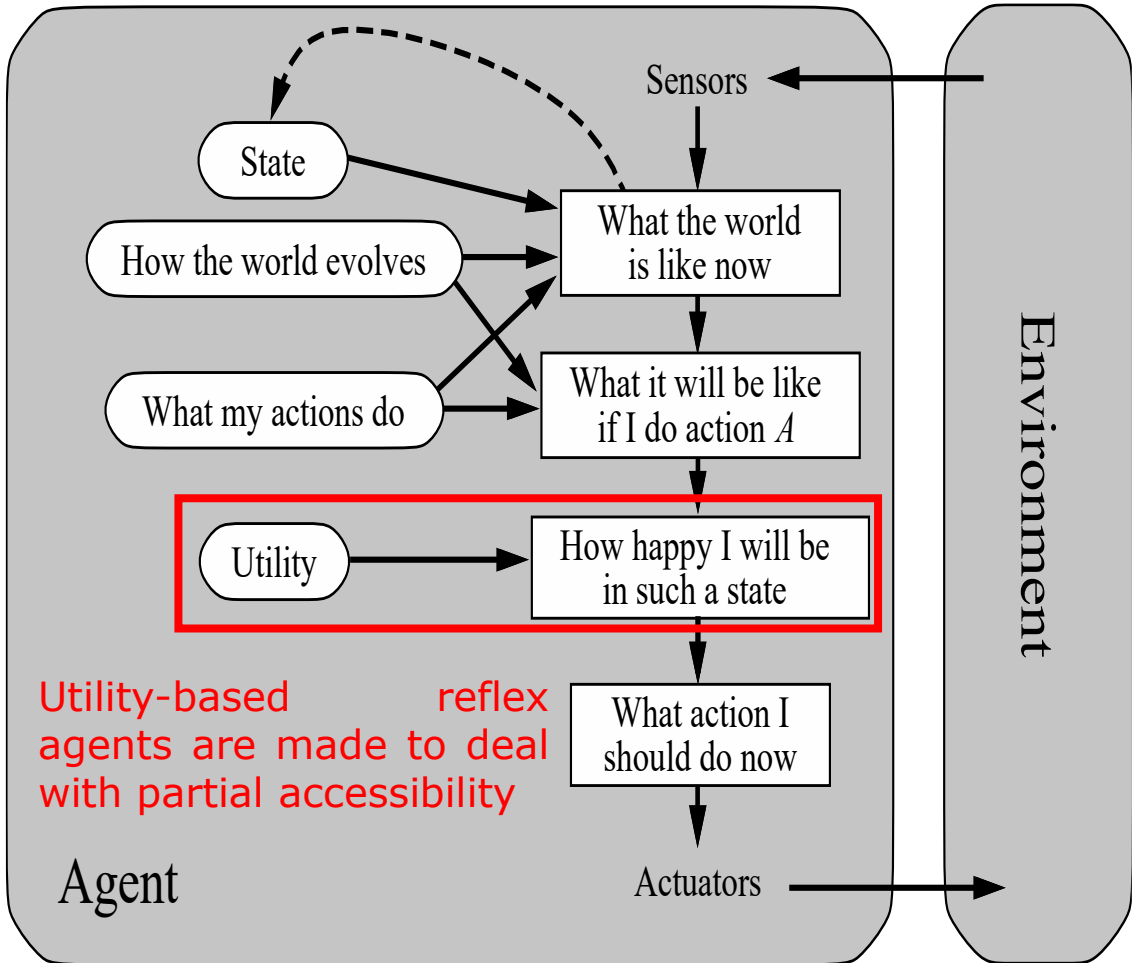
Goal based agent = Goal information + Model-based Agent

To choose action that achieve the goal



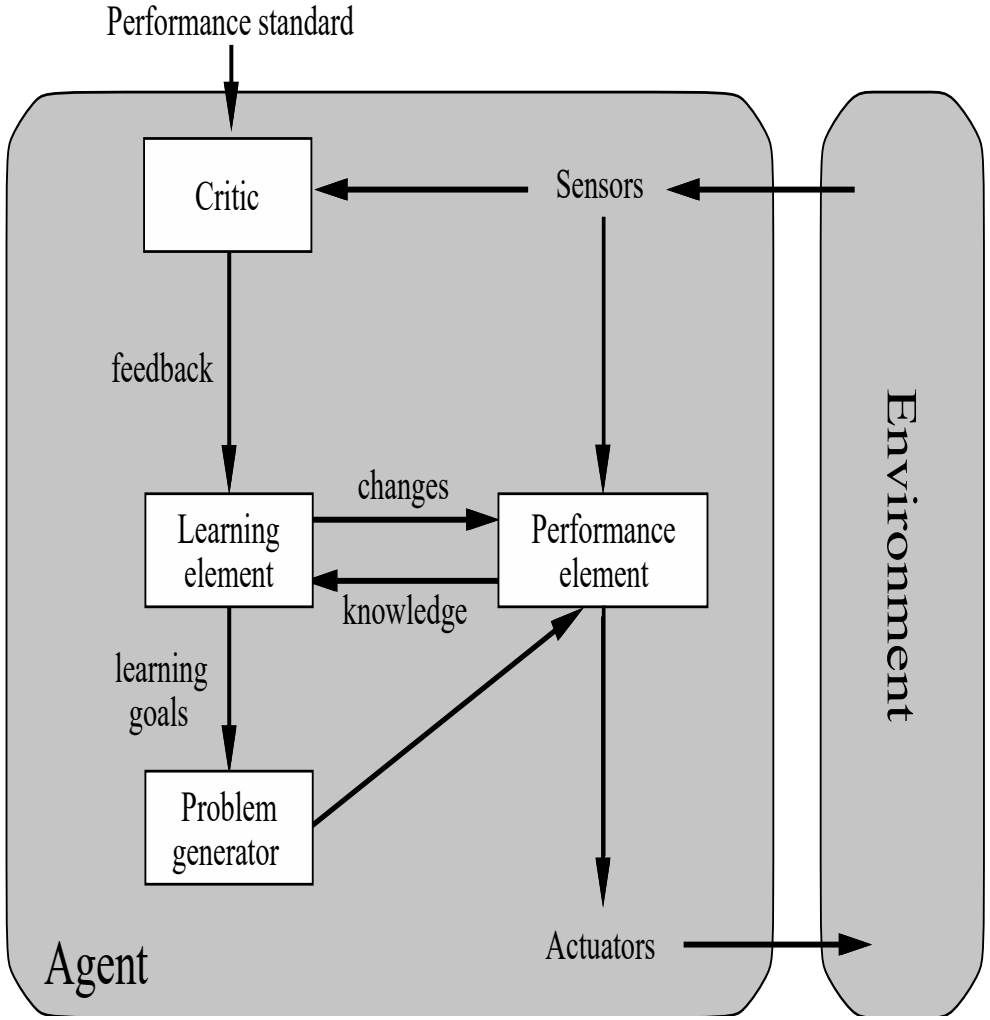
Utility-based Agents

- We may have several actions which all satisfy our goal, so we need some way of working out the most efficient one.
- A utility function maps each state after each action to a real number representing how **efficiently** each action achieves the goal.
- A goal-based navigation agent is tasked with getting from point A to point B. If the agent succeeds, the goal has been satisfied.
- A utility-based navigation agent could seek to get from point A to point B in the shortest amount of time, with the minimum expenditure of fuel, or both.

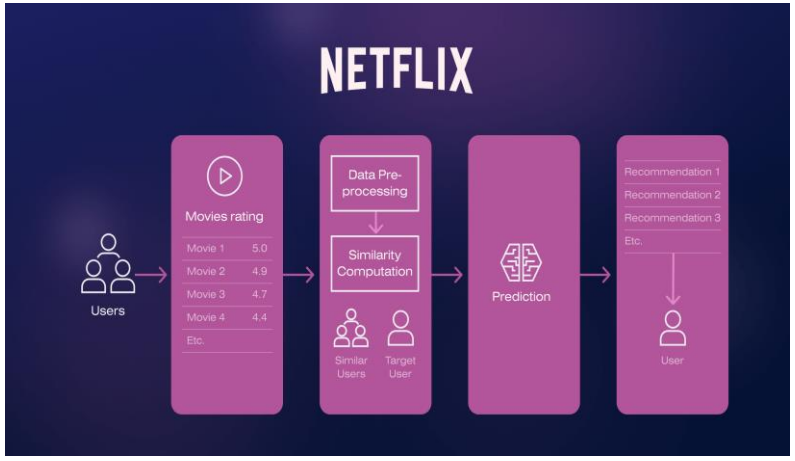


Learning Agents

- This agent can acquire new skills and reflect on its own performance to improve over time.
- Consists of four basic components: Critic, Learning Element, Performance Element, Problem Generator
- Performance element: select external actions to act on the environment
- Critic: evaluates the behavior of the agent based on its performance and gives the evaluation as feedback to the learning element
- Learning element: improves the performance element by posing **new tasks**
- Problem generator: suggests **actions** that will lead to new and informative experiences



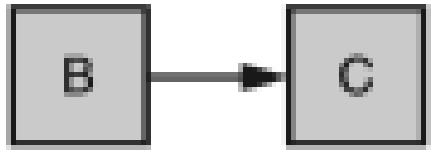
Class Exercise 3



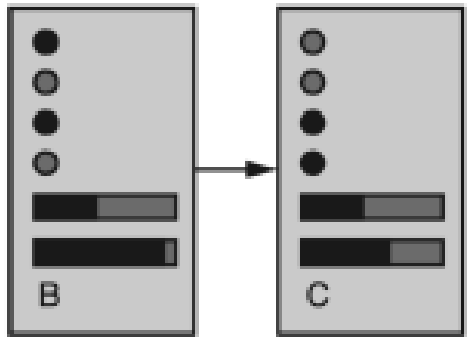
Simple Reflex (Reactive) Agents
Model–based Reflex Agents
Goal-based Agents
Utility-based Agents
Learning Agents

How Can We Represent The State of Environment That The Agent Inhibits?

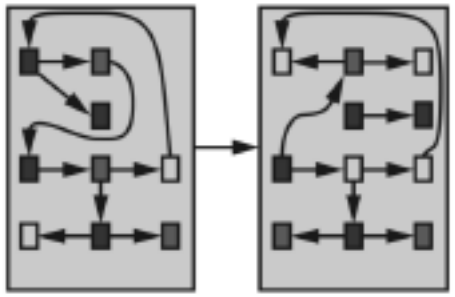
- Atomic Representation
  - each state of the world is indivisible (a black box with no internal structure)
  - used in search and game-playing algorithms, Markov Decision Processes



- Factored Representation
  - each state splits into a fixed set of **variables (attributes)**, each of which can have a value, e.g., Boolean, real-valued, or one of a fixed set of symbols.
  - used in constraint satisfaction, propositional logic, and planning



- Structured Representation
  - each state consists of a set of objects (each may have attributes) with various and varying **relationships**
  - used in first-order logic



Summary

- Agents interact with environments through actuators and sensors
- The agent function describes what the agent does in all circumstances
- The expected performance measure evaluates the environment sequence
- A perfectly rational agent maximizes expected performance
- Agent programs implement (some) agent functions
- PEAS descriptions define task environments
- Environments are categorized along several dimensions: observable; deterministic; episodic; static; discrete; single-agent
- Several basic agent architectures exist: simple reflex, model-based reflex, goal-based, utility-based, and learning-based.