Intelligent Agents

Contents

- Introduction of agents, Structure of Intelligent agent, Properties of Intelligent Agents, Configuration of Agents,
- PEAS description of Agents,
- Types of Agents: Simple Reflexive, Model Based, Goal Based, Utility Based, Learning Agent,
- Environment Types: Deterministic, Stochastic, Static, Dynamic, Observable, Semi-observable, Single Agent, Multi Agent

Agents and environments

- An agent is anything that can be viewed as **perceiving** its environment through **sensors** and **acting** upon that environment through **actuators**. An Agent runs in the cycle of perceiving, thinking, and acting.
  - A **human agent** has eyes, ears, and other organs for sensors and hands, legs, vocal tract, and so on for actuators.
  - A **robotic agent** might have cameras and infrared range finders for sensors and various motors for actuators.
  - A **software agent** receives file contents, network packets, and human input (keyboard/mouse/touchscreen/voice) as sensory inputs and acts on the environment by writing files, sending network packets, and displaying information or generating sounds.

- The term **percept** to refer to the content an agent's sensors are perceiving.
- An agent's **percept sequence** is the complete history of everything the agent has ever perceived.
- In general, an agent's choice of action at any given instant can depend on its built-in knowledge and on the entire percept sequence observed to date, but not on anything it hasn't perceived.
- Mathematically speaking, we say that an agent's behavior is described by the **agent function** that maps any given percept sequence to an action.

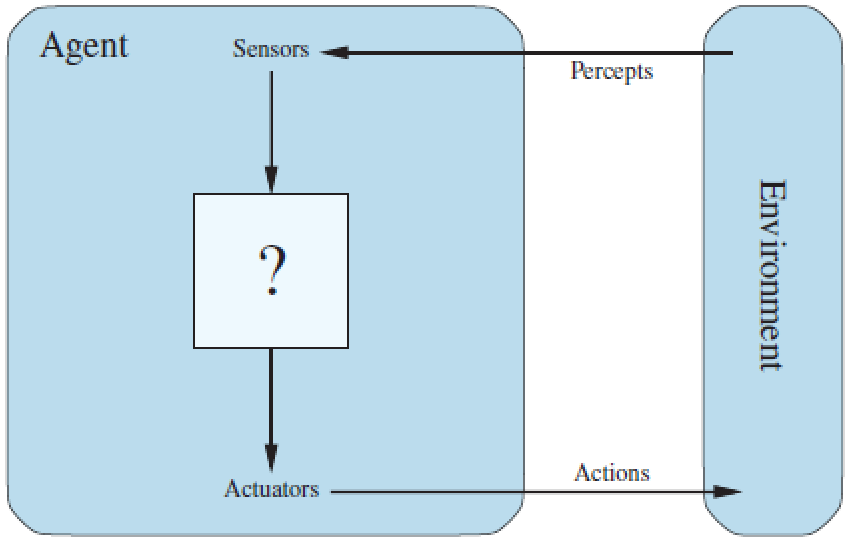


Figure 2.1: Agents interact with environments through sensors and actuators.

Rational agent:

- An ideal rational agent is the one, which is capable of doing expected actions to maximize its performance measure, on the basis of −
  - Its percept sequence
  - Its built-in knowledge base
- Rationality of an agent depends on the following −
  - The performance measures, which determine the degree of success.
  - Agent’s Percept Sequence till now.
  - The agent’s prior knowledge about the environment.
  - The actions that the agent can carry out.
- A rational agent always performs right action, where the right action means the action that causes the agent to be most successful in the given percept sequence. The problem the agent solves is characterized by Performance Measure, Environment, Actuators, and Sensors (PEAS).

- Agent’s structure can be viewed as −
    - Agent = Architecture + Agent Program
        - Architecture = the machinery that an agent executes on.
        - Agent Program = an implementation of an agent function.

Example: performance measure of a vacuum-cleaner agent could be amount of dirt cleaned up, amount of time taken, amount of electricity consumed, amount of noise generated, etc.

- To illustrate these ideas, we use a simple example—the vacuum-cleaner world, which consists of a robotic vacuum-cleaning agent in a world consisting of squares that can be either dirty or clean.
- Figure 2.2 shows a configuration with just two squares, A and B. The vacuum agent perceives which square it is in and whether there is dirt in the square. The agent starts in square A. The available actions are to move to the right, move to the left, suck up the dirt, or do nothing.
- One very simple agent function is the following: if the current square is dirty, then suck; otherwise, move to the other square.
- A partial tabulation of this agent function is shown in Figure 2.3 and an agent program that implements it appears in Figure 2.4

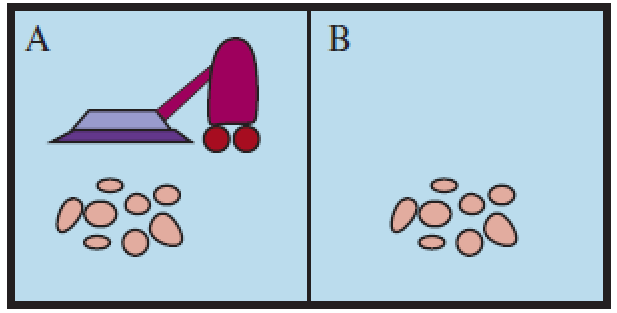


Figure 2.2 : vacuum-cleaner world with just two locations.

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*
⋮	⋮
$[A, Clean], [A, Clean], [A, Clean]$	*Right*
$[A, Clean], [A, Clean], [A, Dirty]$	*Suck*
⋮	⋮

Figure 2.3 Partial tabulation of a simple agent function for the vacuum-cleaner world shown in Figure 2.2.

```
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
```

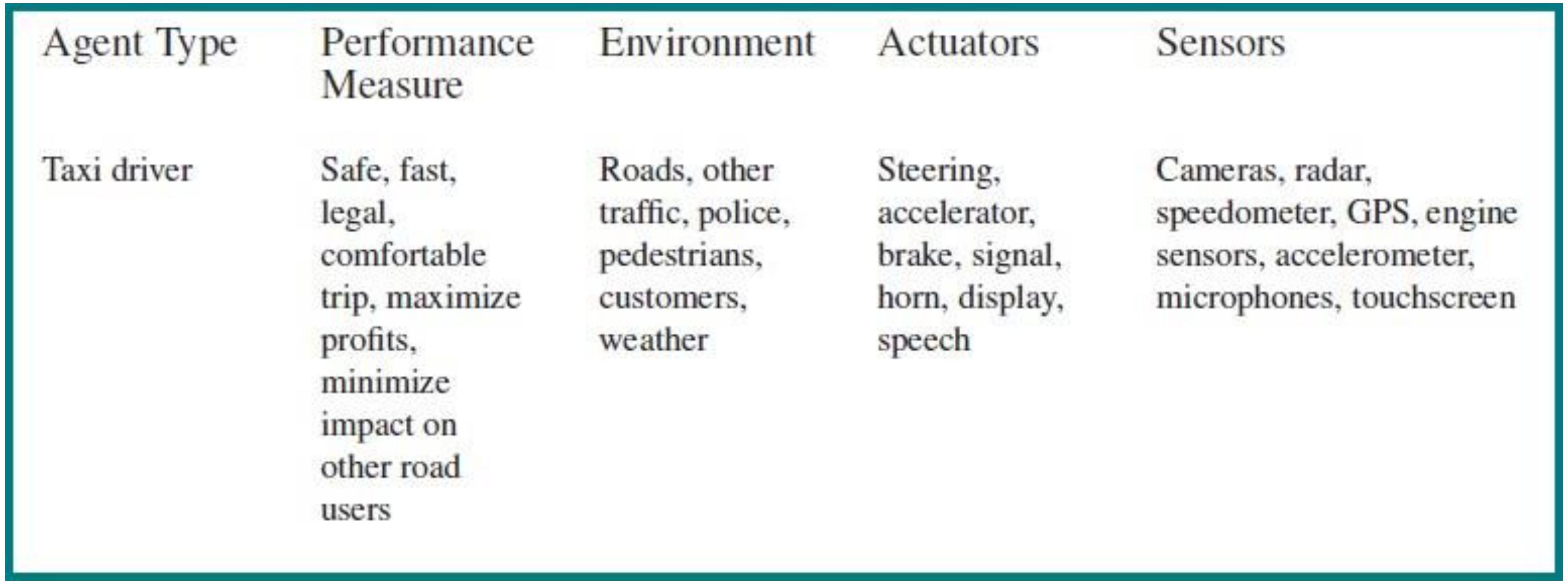
Figure 2.4: The agent program for a simple reflex agent in the two-location vacuum environment.

Properties of Intelligent Agents:

- **Autonomy:** They operate independently, making decisions and taking actions without direct human intervention.
- **Perception:** They perceive their environment through sensors, which could include cameras, microphones, or other input devices.
- **Reasoning:** They use logic, algorithms, and knowledge representation to process information and make decisions.
- **Learning:** Intelligent agents can improve their performance over time through learning algorithms, such as reinforcement learning or deep learning.
- **Adaptability:** They can adapt to changes in their environment or goals, adjusting their behavior accordingly.
- **Goal-oriented:** Agents are typically designed to achieve specific goals or tasks, and they work towards optimizing their actions to achieve these objectives.
- **Communication:** Many agents can communicate with other agents or humans, either to share information or to collaborate on tasks.

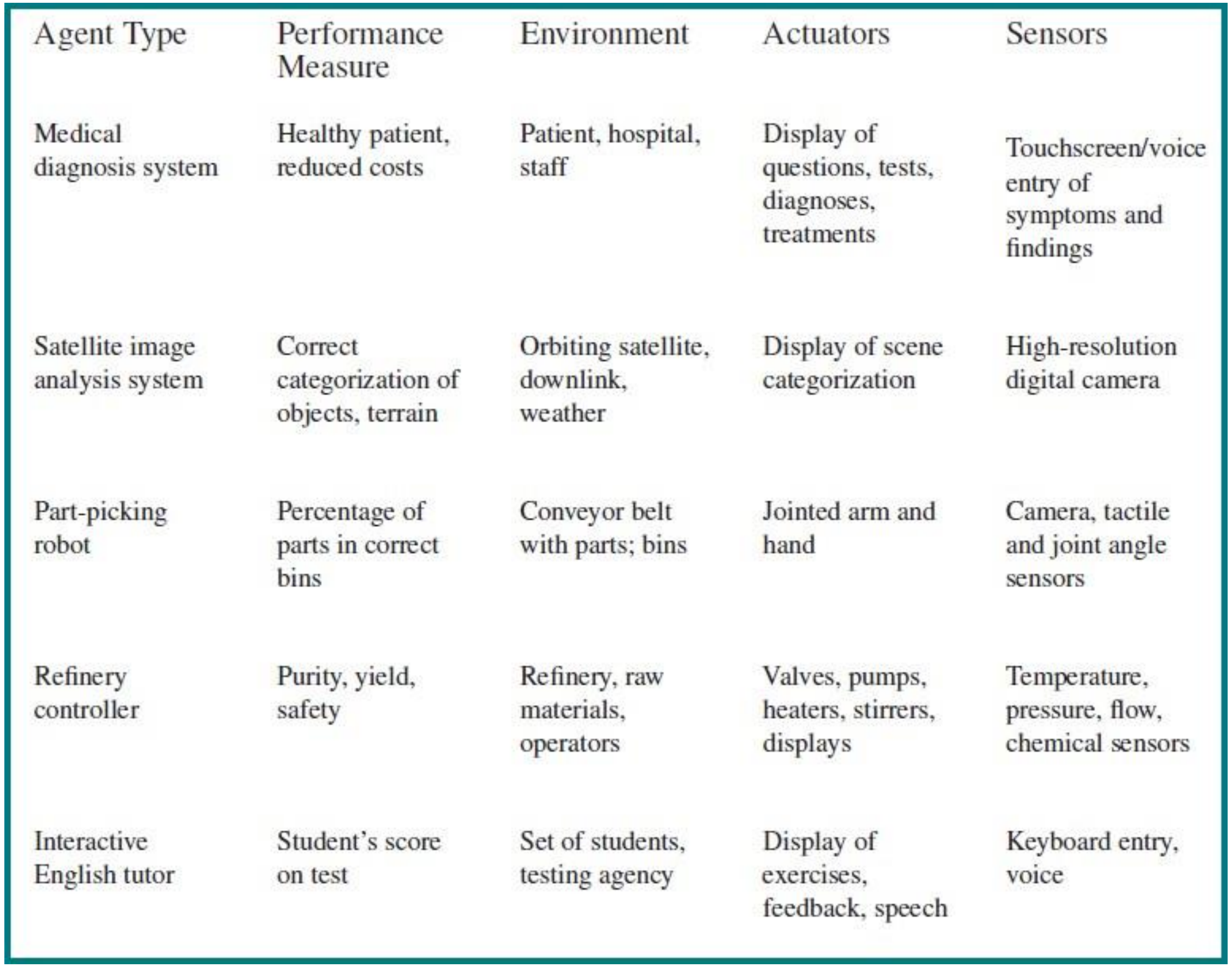
Specifying the task environment (PEAS)

- PEAS:
  - Performance measure,
  - Environment,
  - Actuators,
  - Sensors
- In designing an agent, the first step must always be to specify the task environment as fully as possible.
- The vacuum world was a simple example; let us consider a more complex problem: an automated taxi driver.



Agent Type	Performance Measure	Environment	Actuators	Sensors
Taxi driver	Safe, fast, legal, comfortable trip, maximize profits, minimize impact on other road users	Roads, other traffic, police, pedestrians, customers, weather	Steering, accelerator, brake, signal, horn, display, speech	Cameras, radar, speedometer, GPS, engine sensors, accelerometer, microphones, touchscreen

Figure 2.4: PEAS description of the task environment for an automated taxi driver.



Agent Type	Performance Measure	Environment	Actuators	Sensors
Medical diagnosis system	Healthy patient, reduced costs	Patient, hospital, staff	Display of questions, tests, diagnoses, treatments	Touchscreen/voice entry of symptoms and findings
Satellite image analysis system	Correct categorization of objects, terrain	Orbiting satellite, downlink, weather	Display of scene categorization	High-resolution digital camera
Part-picking robot	Percentage of parts in correct bins	Conveyor belt with parts; bins	Jointed arm and hand	Camera, tactile and joint angle sensors
Refinery controller	Purity, yield, safety	Refinery, raw materials, operators	Valves, pumps, heaters, stirrers, displays	Temperature, pressure, flow, chemical sensors
Interactive English tutor	Student's score on test	Set of students, testing agency	Display of exercises, feedback, speech	Keyboard entry, voice

Figure 2.5: Examples of agent types and their PEAS descriptions.

Types of Agents

- Simple reflex agents;
- Model-based reflex agents;
- Goal-based agents;
- Utility-based agents; and
- Learning Agent

1. Simple reflex agents:

- The simplest kind of agent is the simple reflex agent. These agents select actions on the basis of the current percept, ignoring the rest of the percept history.
- In Figure, rectangles to denote the current internal state of the agent's decision process, and ovals to represent the background information used in the process.

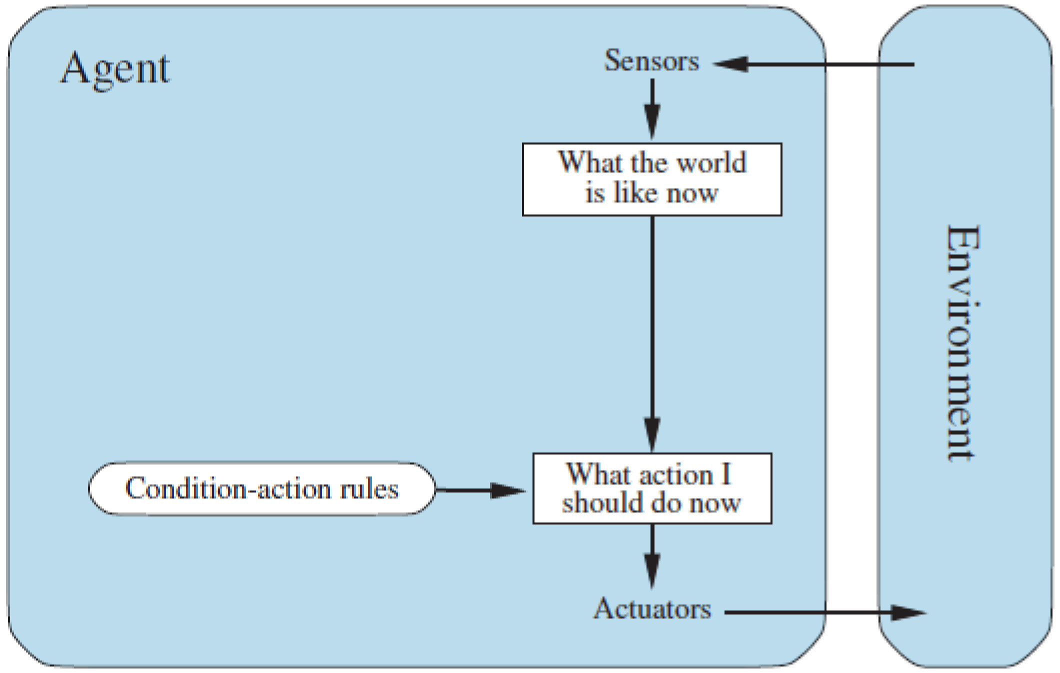


Figure 2.6: Schematic diagram of a simple reflex agent.

Examples,

<u>Simple reflex vacuum cleaner agent</u>

- An agent program for this agent as,

```
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
```

<u>Automated taxi</u>

Condition–action rule, written as

if car-in-front-is-braking **then** initiate-braking.

- An agent program for this agent as,

```
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
```

2. Model-based reflex agents:

- The agent should keep track of the part of the world it can't see now. i.e. the agent should maintain some sort of **internal state** that depends on the percept history and thereby reflects at least some of the unobserved aspects of the current state.

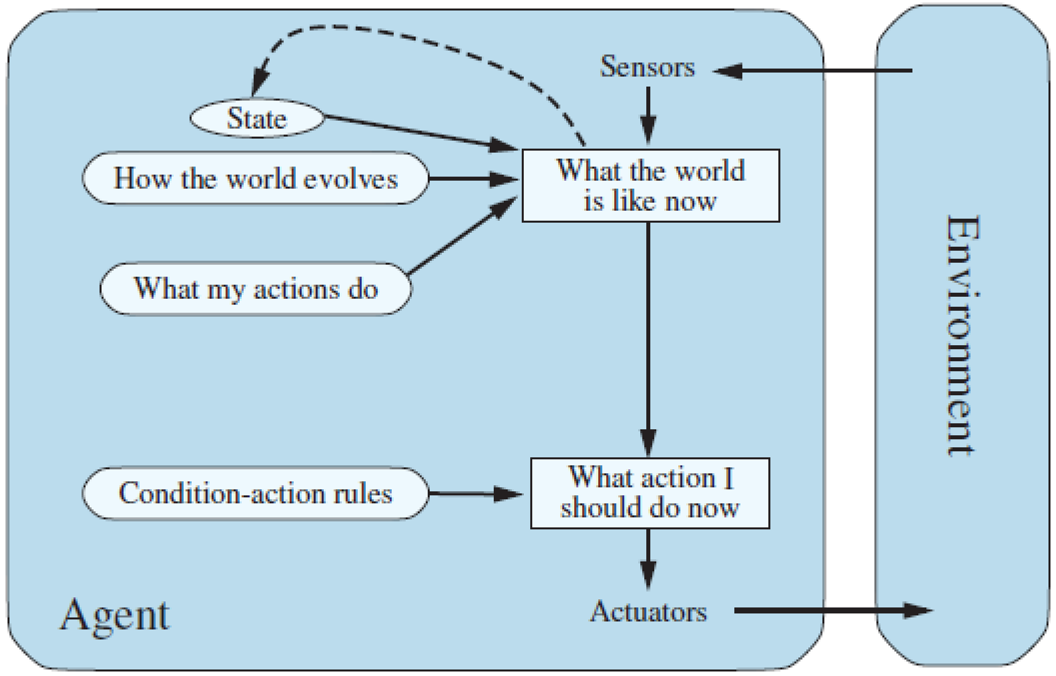


Figure 2.7: A model-based reflex agent.

- Updating this **internal state** information as time goes by requires two kinds of knowledge to be encoded in the agent program in some form.
    - **First, we need some information about how the world changes over time,**
    - For example, when the agent turns the steering wheel clockwise, the car turns to the right, and when it’s raining the car’s cameras can get wet. This knowledge about “how the world works”—whether implemented in simple Boolean circuits or in complete scientific theories—is called a **transition model** of the world.
    - **Second, we need some information about how the state of the world is reflected in the agent’s percepts**.
    - For example, when the car in front initiates braking, one or more illuminated red regions appear in the forward-facing camera image, and, when the camera gets wet, droplet-shaped objects appear in the image partially obscuring the road. This kind of knowledge is called a **sensor model**.

For example, **an automated taxi**

```
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
```

3. Goal-based agents:

- Knowing something about the current state of the environment is not always enough to decide what to do.
- **For example,** at a road junction, the taxi can turn left, turn right, or go straight on. The correct decision depends on where the taxi is trying to get to.
- In other words, as well as a current state description, the agent needs some sort of goal information that describes situations that are desirable—for example, being at a particular destination.
- The agent program can combine this with the model (the same information as was used in the model-based reflex agent) to choose actions that achieve the goal.

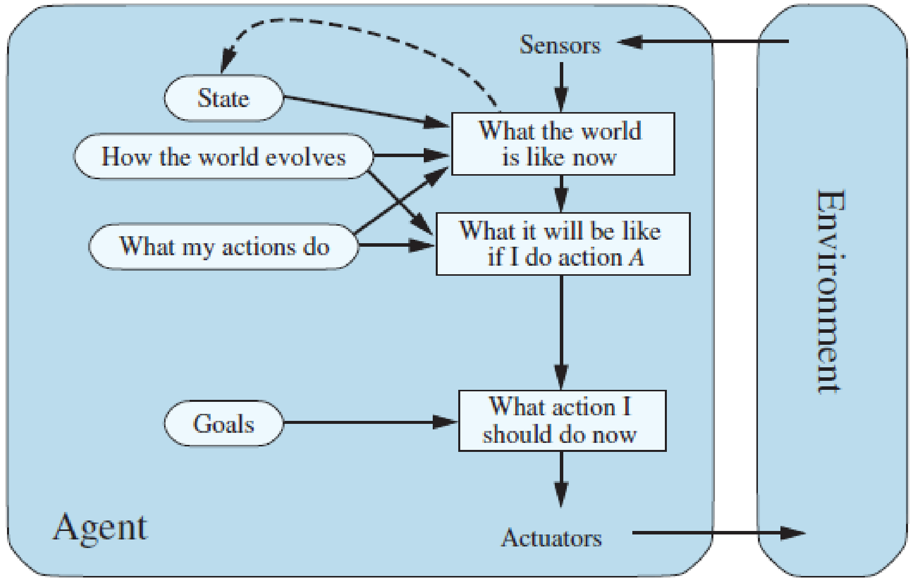

Sensors
State
What the world is like now
How the world evolves
What it will be like if I do action *A*
What my actions do
Goals
What action I should do now
Agent
Actuators
Environment

Figure 2.8: A model-based, goal-based agent.

Goal-based agents vs reflex-based agents

- Although goal-based agents appears less efficient, it is more flexible because the knowledge that supports its decision is represented explicitly and can be modified.
- On the other hand, for the reflex-agent, we would have to rewrite many condition-action rules
- The goal based agent's behavior can easily be changed
- The reflex agent's rules must be changed for a new situation

4. Utility-based agents:

- Goals alone are not enough to generate high-quality behavior in most environments. For example, many action sequences will get the taxi to its destination (thereby achieving the goal), but some are quicker, safer, more reliable, or cheaper than others.
- Goals just provide a crude binary distinction between “happy” and “unhappy” states.
- A more general performance measure should allow a comparison of different world states according to exactly how happy they would make the agent. Because “happy” does not sound very scientific, economists and computer scientists use the Utility term utility instead.
- An agent’s **utility function** is essentially an internalization of the performance measure.

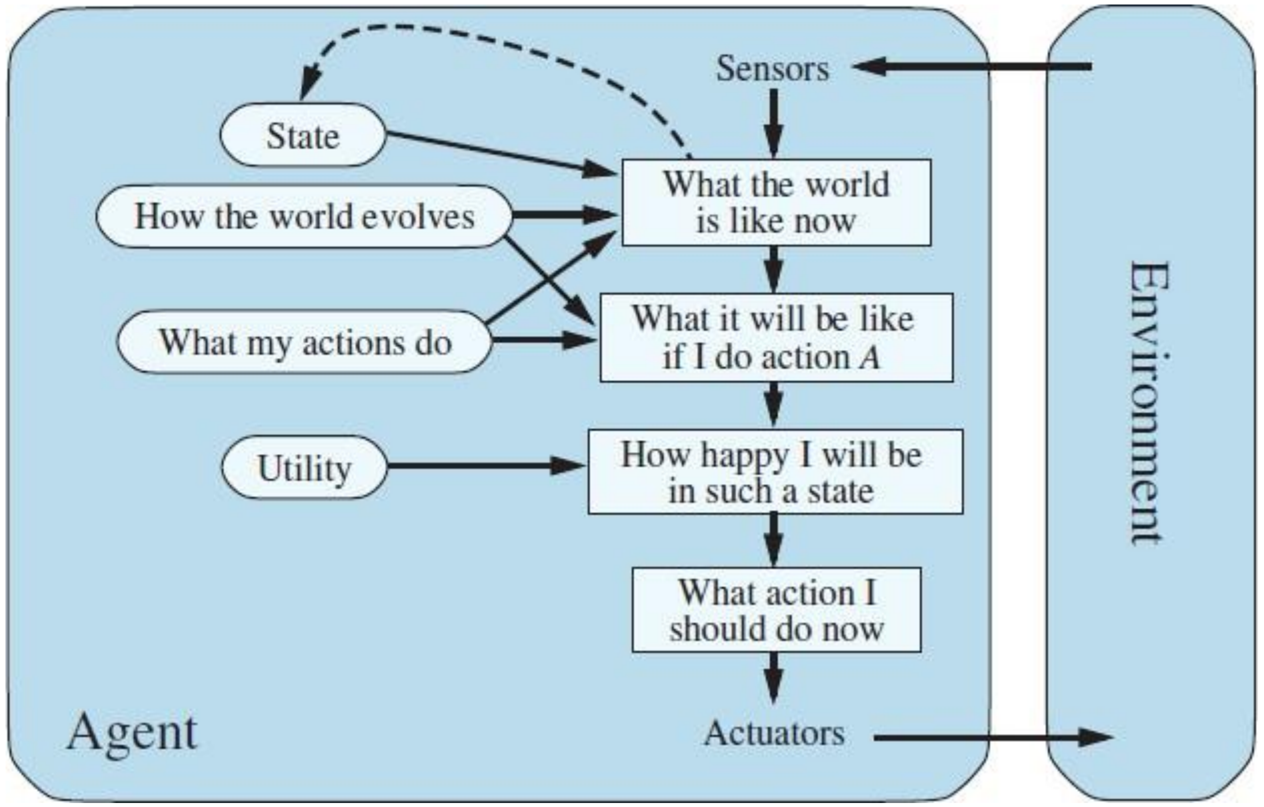

Sensors
State
What the world is like now
How the world evolves
What it will be like if I do action *A*
What my actions do
Utility
How happy I will be in such a state
What action I should do now
Agent
Actuators
Environment

Figure 2.9: A model-based, utility-based agent.

- It uses a model of the world, along with a utility function that measures its preferences among states of the world. Then it chooses the action that leads to the best expected utility, where expected utility is computed by averaging over all possible outcome states, weighted by the probability of the outcome.

5. Learning agents:

- In his famous early paper, Turing (1950) considers the idea of actually programming his intelligent machines by hand.
- Instead of actually programming intelligent machines by hand, which is too much work, build learning machines and then teach them.
- Learning allows the agent to operate in initially unknown environments and to become more competent than its initial knowledge alone might allow.

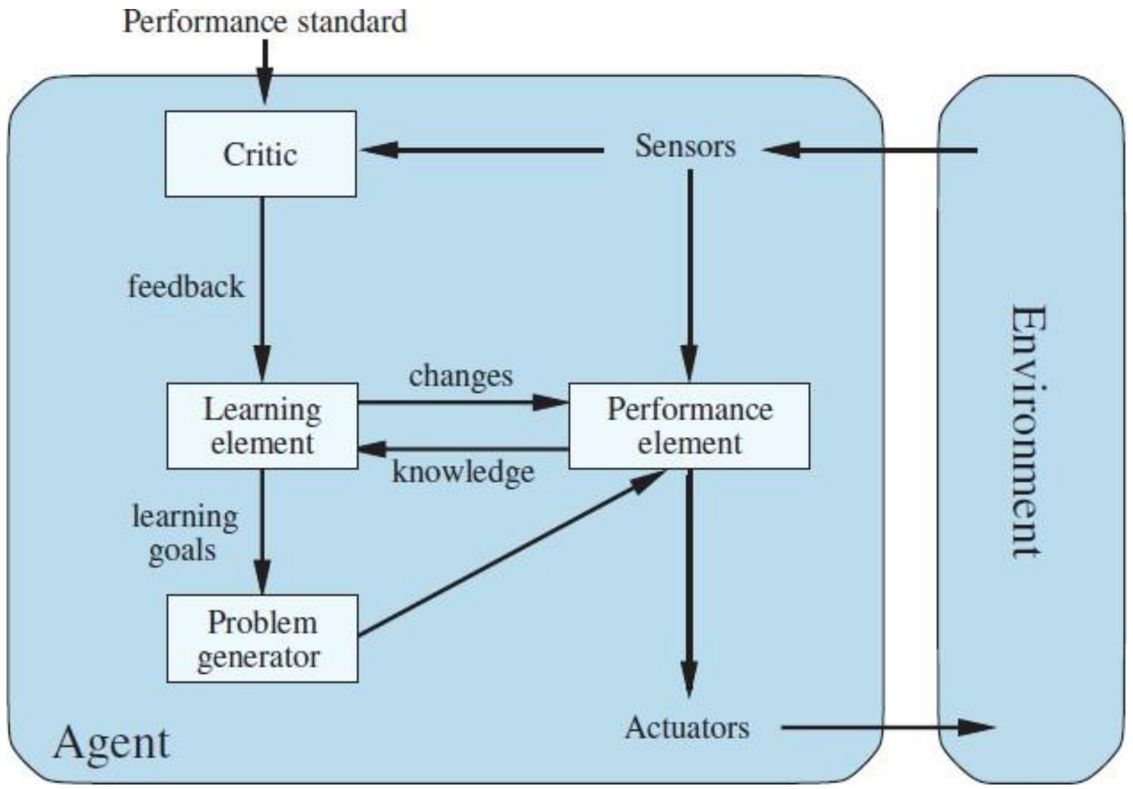


Figure 2.10: A general learning agent.

- A learning agent can be divided into four conceptual components:
    - "**learning element**", which is responsible for making improvements.
    - "**performance element**“ (entire agent), which is responsible for selecting external actions. i.e., it takes in percepts and decides on actions.
    - The learning element uses feedback from the **"critic**" on how the agent is doing and determines how the performance element should be modified to do better in the future.
    - For example when you were in school you would do a test and it would be marked the test is the critic. The teacher would mark the test and see what could be improved and instructs you how to do better next time, the teacher is the learning element and you are the performance element.
    - "**problem generator**“, It is responsible for suggesting actions that will lead to new and informative experiences.

Environment Types

- An environment in artificial intelligence is the surrounding of the agent. The agent takes input from the environment through sensors and delivers the output to the environment through actuators. There are several types of environments:
    - Fully Observable vs Partially Observable
    - Deterministic vs Stochastic
    - Competitive vs Collaborative
    - Single-agent vs Multi-agent
    - Static vs Dynamic
    - Discrete vs Continuous
    - Episodic vs Sequential
    - Known vs Unknown

<u>Fully observable vs. partially observable:</u>

- If an agent's sensors give it access to the complete state of the environment at each point in time, then we say that the task environment is fully observable. A task environment is effectively fully observable if the sensors detect all aspects that are relevant to the choice of action; relevance, in turn, depends on the performance measure.
- Fully observable environments are convenient because the agent need not maintain any internal state to keep track of the world.
- **For example, Chess –** the board is fully observable, and so are the opponent's moves.
- An environment might be partially observable because of noisy and inaccurate sensors or because parts of the state are simply missing from the sensor data.
- **For example, a vacuum agent** with only a local dirt sensor cannot tell whether there is dirt in other squares, and an **automated taxi** cannot see what other drivers are thinking. If the agent has no sensors at all then the environment is unobservable.

Single-agent vs Multi-agent:

- An environment consisting of only one agent is said to be a single-agent environment.
- An environment involving more than one agent is a multi-agent environment.
- **For examples**, an agent solving a crossword puzzle by itself is clearly in a single-agent environment, whereas an agent playing chess is in a two agent environment. The game of football is multi-agent as it involves 11 players in each team.

Competitive vs Collaborative:

- An agent is said to be in a competitive environment when it competes against another agent to optimize the output.
- An agent is said to be in a collaborative environment when multiple agents cooperate to produce the desired output.
- **For Examples,**
    - The game of chess is competitive as the agents compete with each other to win the game which is the output.

- When multiple self-driving cars are found on the roads, they cooperate with each other to avoid collisions and reach their destination which is the output desired.

Deterministic vs Stochastic:

- If the next state of the environment is completely determined by the current state and the action executed by the agent(s), then we say the environment is deterministic; otherwise, it is nondeterministic.
- The stochastic environment is random in nature which is not unique and cannot be completely determined by the agent.
- **Examples,**
  - **Chess –** there would be only a few possible moves for a coin at the current state and these moves can be determined.
  - **Self-Driving Cars-** the actions of a self-driving car are not unique, it varies time to time.

Static vs dynamic:

- If the environment can change while an agent is deliberating, then we say the environment is dynamic for that agent; otherwise, it is static.
- Static environments are easy to deal with because the agent need not keep looking at the world while it is deciding on an action, nor need it worry about the passage of time.
- Dynamic environments, on the other hand, are continuously asking the agent what it wants to do; if it hasn't decided yet, that counts as deciding to do nothing.
- The environment is semi-dynamic if the environment itself does not change with the passage of time but the agent's performance score does.
- **Examples,** taxi driving is dynamic, chess when played with a clock is semi-dynamic, crossword puzzles are static.

Discrete vs Continuous:

- If an environment consists of a finite number of actions that can be deliberated in the environment to obtain the output, it is said to be a discrete environment.
- The environment in which the actions are performed cannot be numbered i.e. is not discrete, is said to be continuous.
- **For examples,**
  - The chess environment has a finite number of distinct states (excluding the clock). Chess also has a discrete set of percepts and actions.
  - Taxi driving is a continuous-state and continuous-time problem: the speed and location of the taxi and of the other vehicles sweep through a range of continuous values and do so smoothly over time. Taxi-driving actions are also continuous (steering angles, etc.).

Episodic vs sequential:

- The agent’s experience is divided into atomic episodes. In each episode the agent receives a percept and then performs a single action, the next episode does not depend on the actions taken in previous episodes.
- Many classification tasks are episodic.
- **For example**, an agent that has to spot defective parts on an assembly line bases each decision on the current part, regardless of previous decisions; moreover, the current decision doesn’t affect whether the next part is defective.
- In sequential environment, the current decision could affect all future decisions.
- **For example**, Chess and taxi driving are sequential: in both cases, short-term actions can have long-term consequences.
- Episodic environments are much simpler than sequential environments because the agent does not need to think ahead.

Known vs unknown:

- In a known environment, the outcomes (or outcome probabilities if the environment is nondeterministic) for all actions are given.
- If the environment is unknown, the agent will have to learn how it works in order to make good decisions.

Here are some real-life examples of learning agent programs:

- **Self-Driving Cars:** Learning agents are used extensively in self-driving cars to navigate roads, avoid obstacles, and make decisions in real-time. These agents learn from data collected during driving sessions to improve their driving behavior and safety.
- **Recommendation Systems:** Learning agents power recommendation systems in platforms like Netflix, Amazon, and Spotify. They learn from user interactions and preferences to recommend personalized content, products, or music based on past behavior.
- **Chatbots:** Chatbots use learning agents to understand and respond to user queries. They learn from conversations with users to improve their language understanding, context handling, and overall conversation quality.

- **Healthcare:** Learning agents are used in healthcare for various tasks such as medical diagnosis, personalized treatment planning, and drug discovery. They learn from patient data, medical research, and historical outcomes to improve decision-making and patient outcomes.
- **Finance and Trading:** Learning agents are employed in financial applications for tasks like stock market prediction, fraud detection, and algorithmic trading. They learn from market data, trading patterns, and historical trends to make informed financial decisions.
- **Gaming:** In video games, learning agents are used for creating intelligent non-player characters (NPCs) that can adapt to player strategies, learn from past gameplay, and provide challenging and engaging gameplay experiences.
- **Industrial Automation:** Learning agents are utilized in industrial settings for optimizing processes, predictive maintenance, and autonomous control of robots and machinery. They learn from sensor data, production metrics, and historical performance to improve efficiency and reduce downtime.

Assignments

Uni1 & Unit2

Q1. What is AI? How can you define AI from the perspective of thought process?

Q2. What is Turing test? How it can be used to measure intelligence of machine?

Q3. How agent can be configured used PEAS framework? Illustrate with example.

Q4. Discuss the types of environment where an agent can work on.

Q5. What do you mean by Rational Agent? Explain different types of agents with advantages and disadvantages.