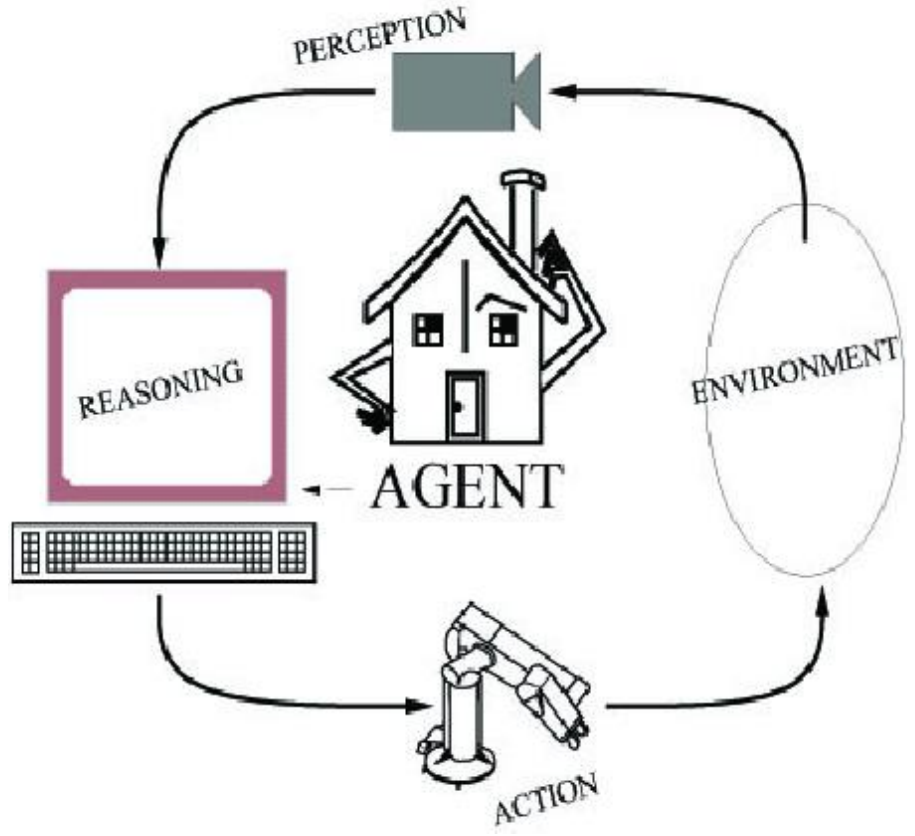
Artificial Intelligence: Intelligent Agents

F24 AI & MMG

Outlines

- Agents
- Percept & Percept Sequence
- Agent Function vs Program
- Environment Types
- Types of AI Agents

Agent



An agent perceives its environment through sensors and acts on the environment through actuators.

Human: sensors are eyes, ears, actuators (effectors) are hands, legs, mouth.

Robot: sensors are cameras, sonar, lasers, effectors are grippers, manipulators, motors

Percept and Percept Sequence

Percept:

- Percept is the raw data or input that the agent collects
- Percept refers to the raw sensory input an agent receives at any given instant.
- **Example:** For a self-driving car, a **percept could be the data from a camera** or sensor detecting the **distance to other cars or an object** in front.
- A percept for a robot could be **the visual information from a camera about its surroundings** or touch sensor data when it makes contact with an object.

Percept and Percept Sequence

Percept Sequence:

- History of everything agent has perceived so far
- Choice of action depends on everything agent has perceived so far, and doesn’t depend on anything which it hasn’t perceived
- Behavior is described by **agent function**:
  - *Map given percept sequence to an action*
- We can make a **table for agent function** that describes an agent
- Put a limit on percept sequences to avoid infinity

Agent Function vs Agent Program

Agent function is a mathematical function which does the mapping

In other words it is a mathematical rule mapping percepts → actions.

while **Agent program** is a practical implementation running within a physical system.

In other words, It is the actual software that implements the function in a machine.

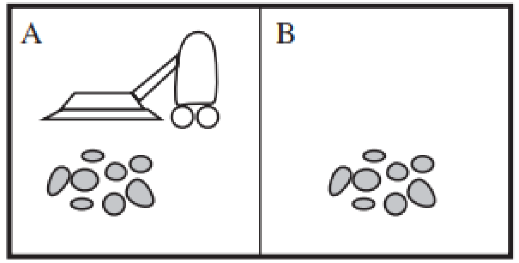
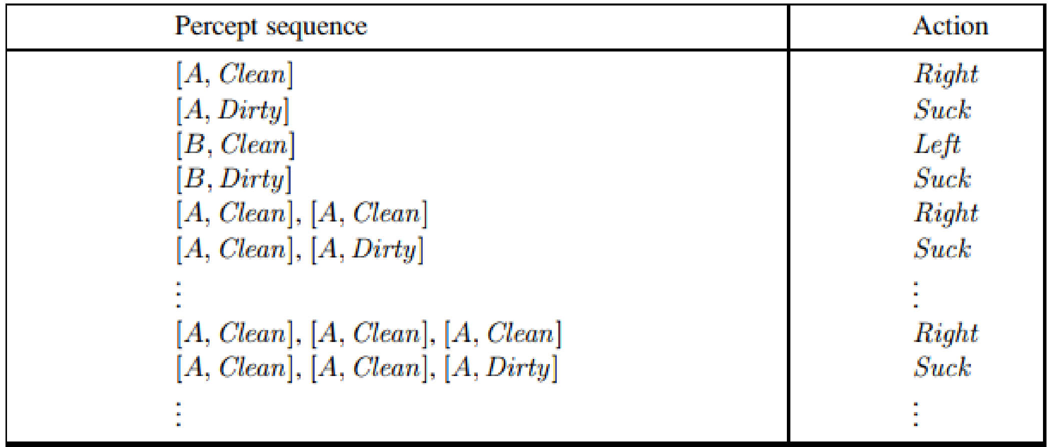
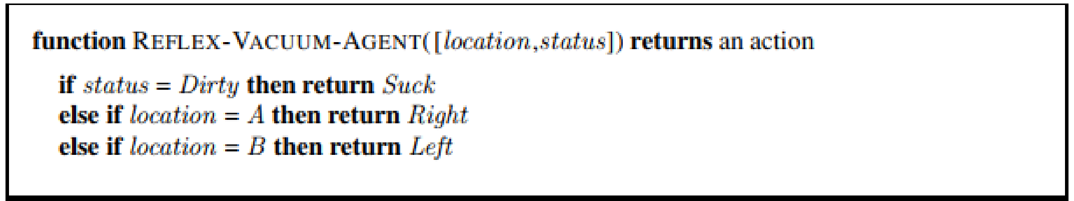


Figure 2.2 A vacuum-cleaner world with just two locations.



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 2.8 The agent program for a simple reflex agent in the two-state vacuum environment. This program implements the agent function tabulated in Figure 2.3.

Rational Agent (Good Behavior): the Concept of Rationality

A **rational agent** = does the **right thing** to maximize success.

Rationality depends on:

Performance measure (e.g., rescue robot = number of victims saved).

Prior knowledge (robot knows hazards).

Actions available (robot can turn, pick, move).

Percept sequence (past experience).

What is rational at any given time?

It depends upon four things:

- The **performance measure** that defines the criterion of success:
  - *Search-rescuce robot: Number of Victims resuced, maximize the number*
- The **agent's prior knowledge** of the environment:
  - *Robot knows certain areas are hazardous and avoids the risk*
- The **actions** that agent can perform:
  - *Rationality: Game Playing agent selects moves that lead to wining*
- The agent's **percept sequence** to date
  - *Robot detects obstacle in path: Use Information, adjust route and avoid collision*

This leads to definition of a 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 Example from the Book

1. **Performance Measure:** How do we judge if the agent is doing a good job?

   Here: The vacuum cleaner gets **1 point for every clean square** at each time step.

2. **Geography of Environment:** The world has **two squares (A & B):** The agent knows the **layout** (two squares connected), but:

   It does **not know where it starts** (could be A or B).

   It does **not know the dirt distribution** (which squares are dirty).

3. **Actions Available:** The agent has **three actions only**:

   Left → moves to the left square.

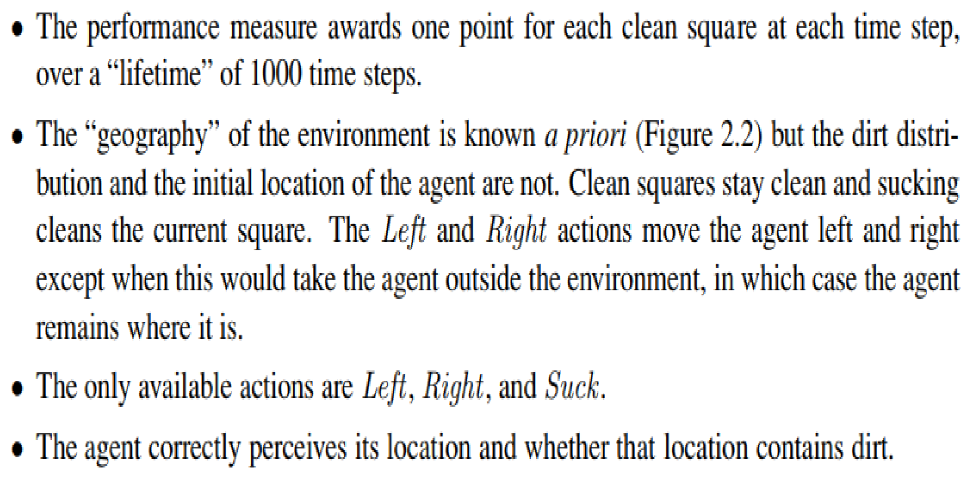
   Right → moves to the right square.

   Suck → cleans dirt in the current square.

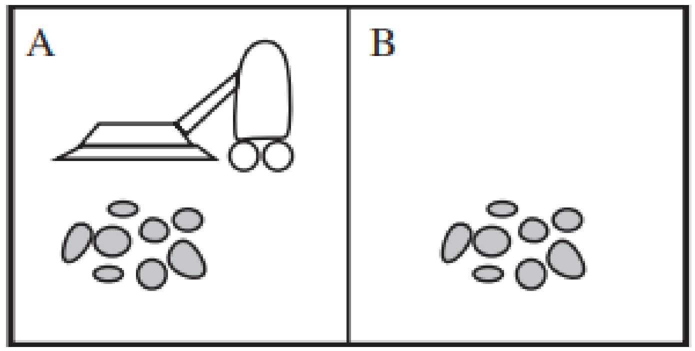
4. **Perception (Sensors):** The vacuum perceives:

   Its **current location** (A or B).

   Whether the location is **Dirty or Clean**.



- The performance measure awards one point for each clean square at each time step, over a "lifetime" of 1000 time steps.
- The "geography" of the environment is known *a priori* (Figure 2.2) but the dirt distribution and the initial location of the agent are not. Clean squares stay clean and sucking cleans the current square. The *Left* and *Right* actions move the agent left and right except when this would take the agent outside the environment, in which case the agent remains where it is.
- The only available actions are *Left*, *Right*, and *Suck*.
- The agent correctly perceives its location and whether that location contains dirt.

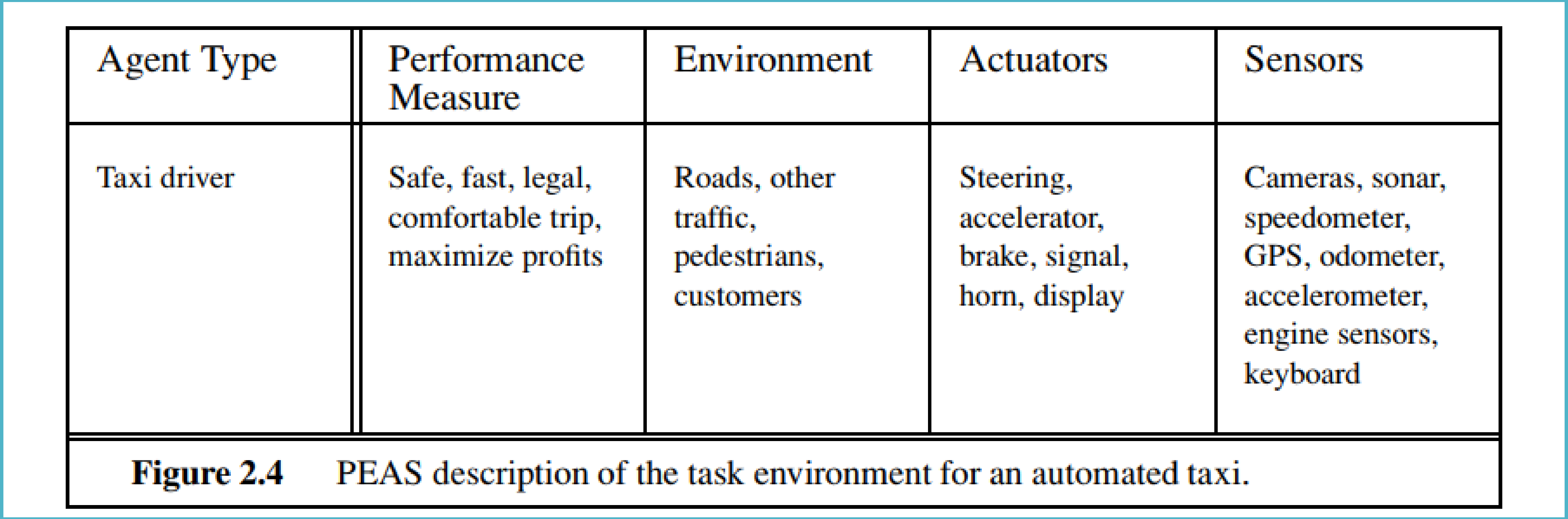


Task Environment / PEAS of an Agent

Use PEAS to describe task

- Performance measure
- Environment
- Actuators
- Sensors

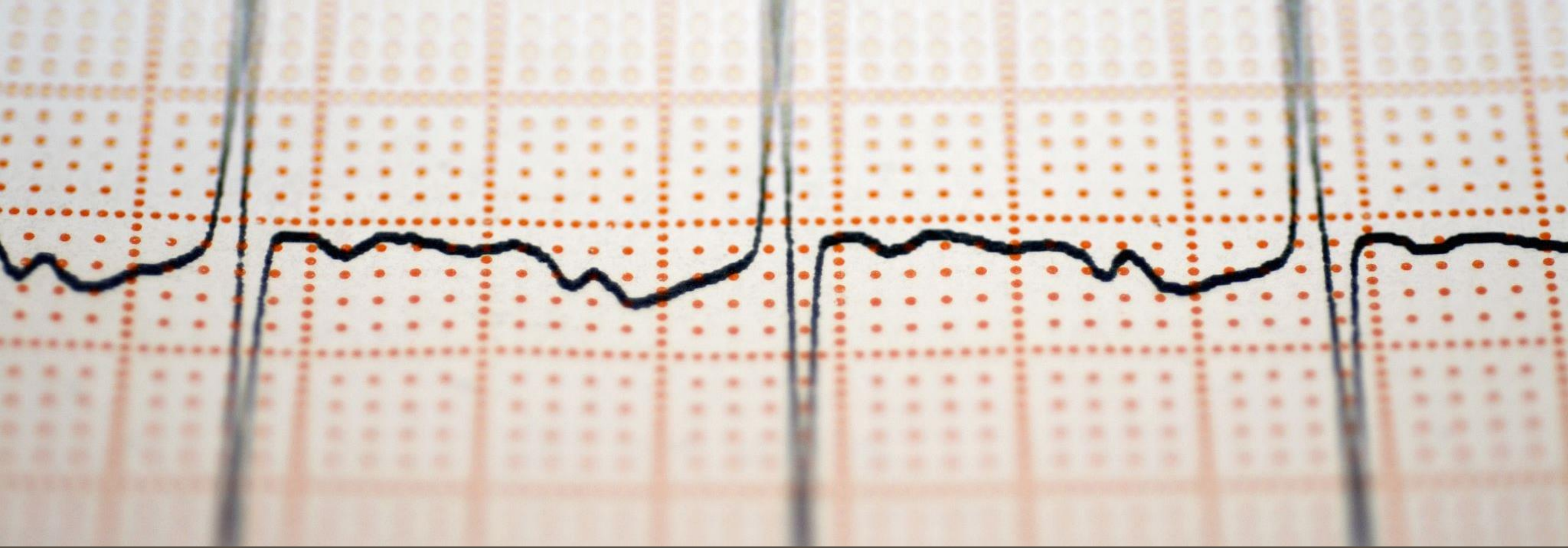
When Designing agent first step should be to describe the task as fully as possible



Agent Type	Performance Measure	Environment	Actuators	Sensors
Taxi driver	Safe, fast, legal, comfortable trip, maximize profits	Roads, other traffic, pedestrians, customers	Steering, accelerator, brake, signal, horn, display	Cameras, sonar, speedometer, GPS, odometer, accelerometer, engine sensors, keyboard

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

Automated Taxi (PEAS) / Task Description



Activity

Medical Diagnosis System **PEAS**

Environment and its types

Environment is the part of universe that surrounds intelligent systems

Accessible vs Inaccessible → Can agent see everything? (Chess vs Poker).

Deterministic vs Non-deterministic → Predictable or uncertain? (Tic-tac-toe vs picking up objects).

Static vs Dynamic → Does environment change when agent is thinking? (Crossword vs real-time traffic).

Discrete vs Continuous → Limited options vs infinite (Traffic lights vs car driving).

Fully vs Partially observable → Complete info vs incomplete (Chess vs Poker).

Episodic vs Non-episodic → Independent short episodes vs long-term linked tasks (Chatbot vs Stock trading agent).

Accessible vs Inaccessible

Accessible:

Agent can access complete and accurate information about state's environment

Exp: Room with Temperature as a state

Agent can accurately access and measure the temperature of a room

Inaccessible:

Whole information may not be in agent's reach

Exp: Any Event on earth

Agent can not Determine any event happening on the earth

Deterministic

vs

Non Deterministic

Deterministic:

- ❑ Agent's current state and selected action can completely determine the next state of environment
- ❑ Doesn't worry about uncertainty
- ❑ Predictable
- ❑ Exp: Agent playing a tic-tac-toe game, next move depends on current action taken

Non-Deterministic (Stochastic):

- ❑ Nature of environment can't be decided by agent alone
- ❑ It is random in nature (Uncertainty)
- ❑ Not Predictable
- ❑ Exp: Agent using a robot arm to pick up objects, now while picking objects the objects(ball) may slip or may be picked up successfully

Static or Dynamic

Static:

Environment doesn’t change its state with the passage of time

Exp:Rules of chess game never change

Dynamic:

Environment changes its state with the passage of time

Exp:Traffic on a road, Condition of a road

Discrete vs Continuous

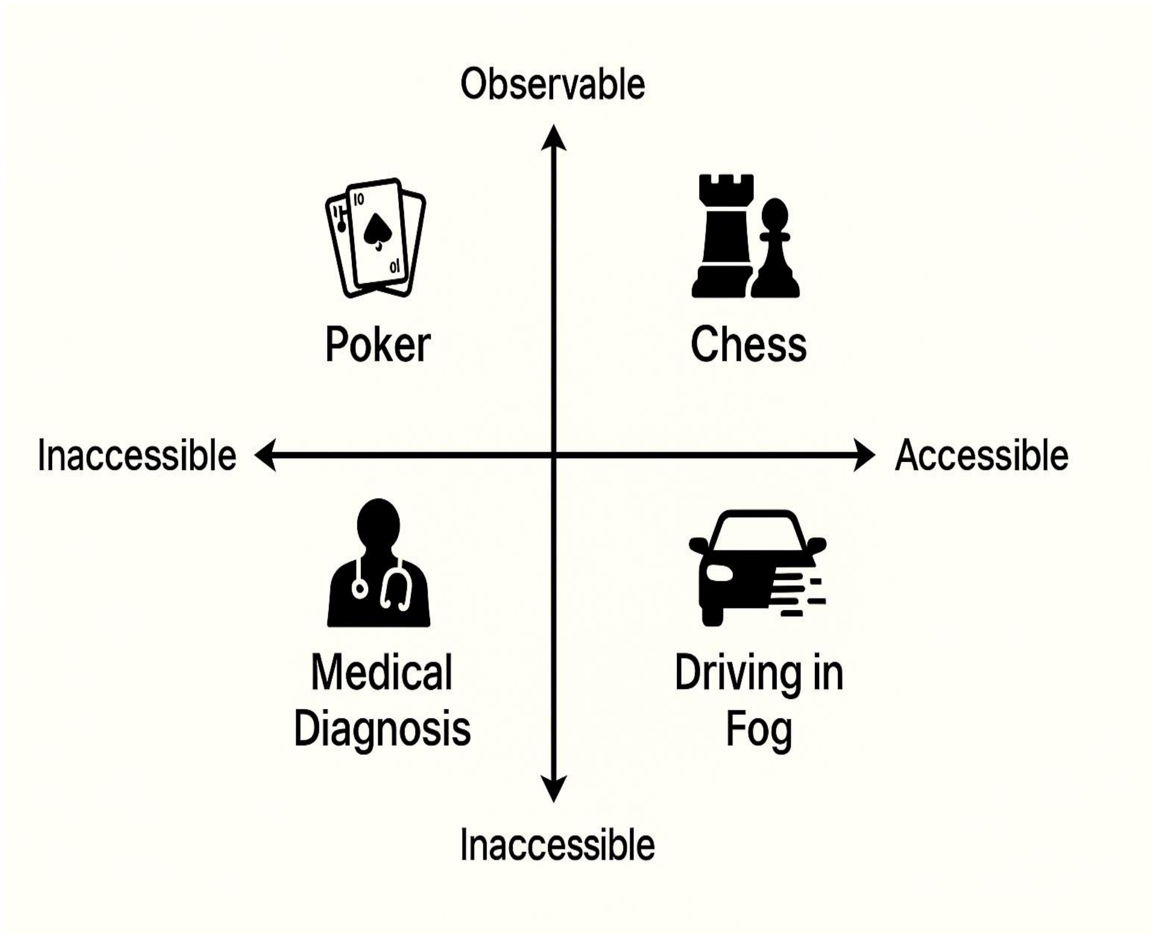
Discrete:

- Finite number of percepts and actions
- Exp: Traffic Light (Red, Green or Yellow)

Continuous:

- Actions and outcomes are continuous
- The environment can be in finite number of states
- Exp: Speed and Position of a car

Fully Observable and Partially Observable



Fully Observable:

- Agent has complete knowledge of the current state
- **Chess game:** Agent can see the entire board and all pieces

Partially Observable:

- Agent has incomplete knowledge of the current state
- Exp: The agent can see only it’s own card not the others

Episodic

vs

Non Episodic Environment

- **Episodic:**
- Agent’s experience is divided into separate independent episodes
- Each Episode has start and end
- Agent’s goal is to achieve a specific outcome in that episode
- Agent’s actions in one episode doesn’t affect the other
- Exp: A customer service chat bot

Episodic

vs

Non Episodic Environment

Non-Episodic:

- Agent's experience is continuous and interconnected
- Actions and decisions have long-term consequences that affect future outcomes
- Agent must consider potential long term effects of its actions
- Exp: **Financial Trading Agent:** The agents actions (buying, selling) have long term effects

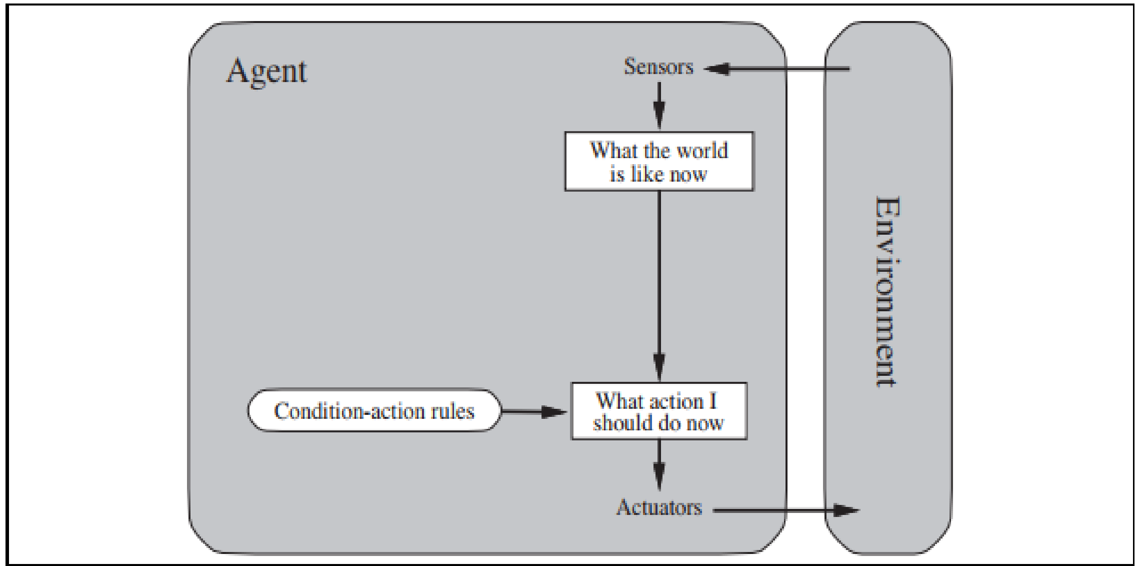
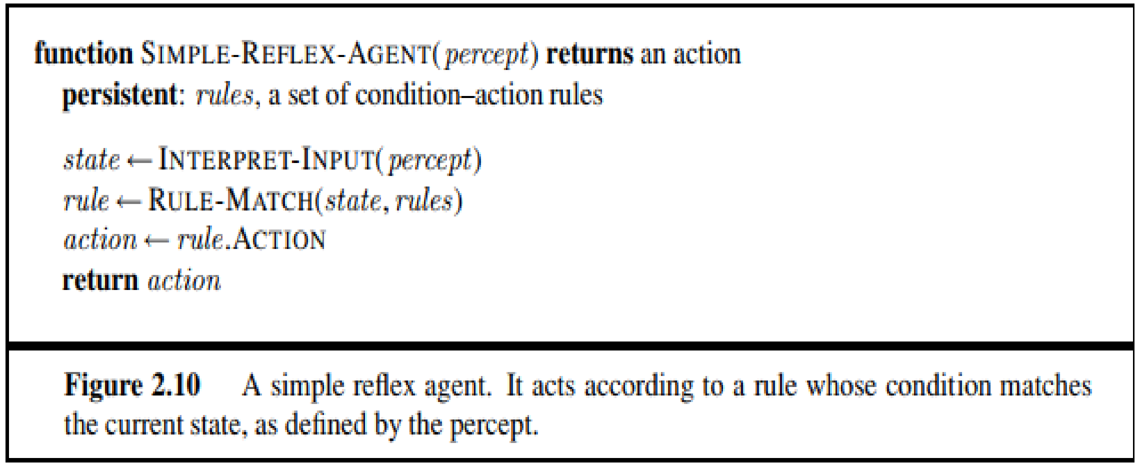


Figure 2.9 Schematic diagram of a simple reflex agent.



```
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
```

Figure 2.10 A simple reflex agent. It acts according to a rule whose condition matches the current state, as defined by the percept.

Types of AI Agents

Simple Reflex Agents:

- ❑ Work only on current situation/perception and ignores the history of previous state
- ❑ Condition-Action Rule is applied

  Exp: Automatic Door Opener: Opens when someone approaches

Limitations

Very Limited Intelligence

Operating in only fully observable environment, otherwise infinite loops

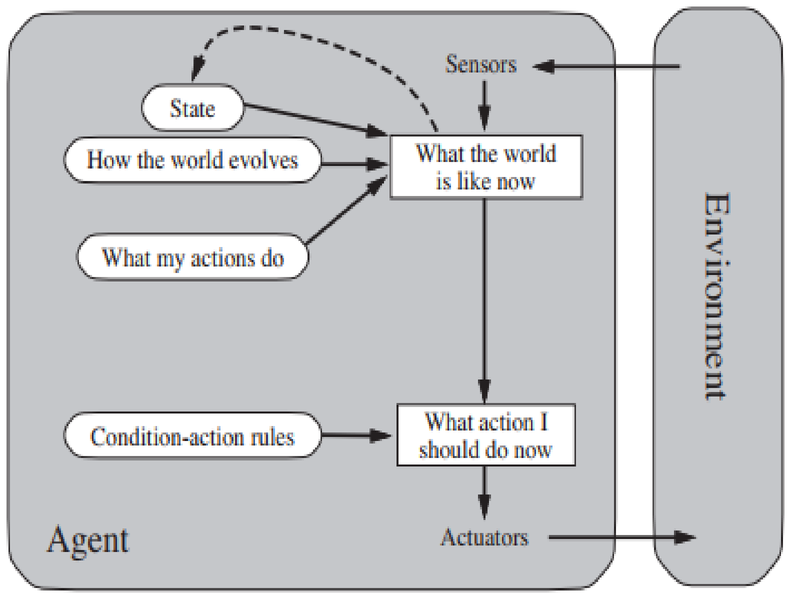
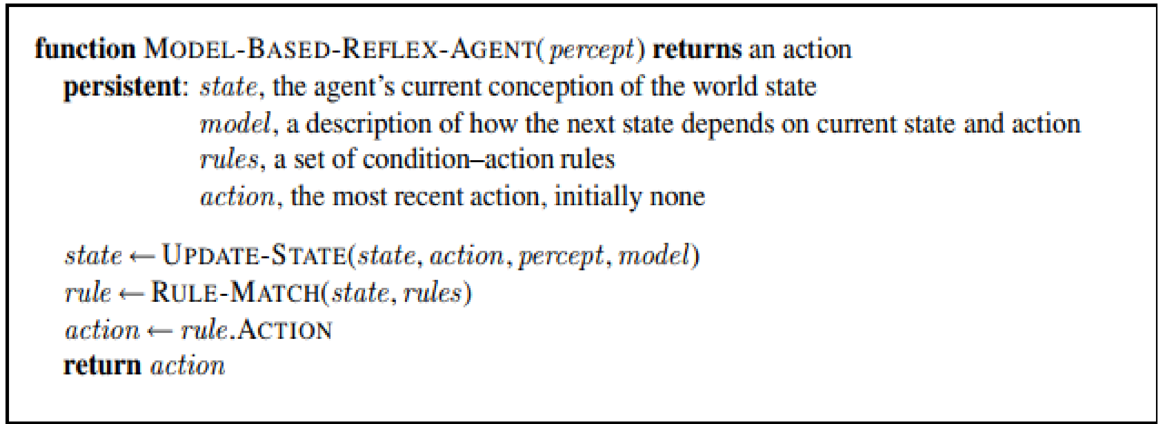


Figure 2.11 A model-based reflex agent.



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

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

Figure 2.12 A model-based reflex agent. It keeps track of the current state of the world, using an internal model. It then chooses an action in the same way as the reflex agent.

Types of AI Agents

Model based Reflex Agents:

- ❑ Work by finding the rules whose conditions matches the current situations
- ❑ It can work in partially observable environment
- ❑ Maintains internal state which relates to percept history
- ❑ Autonomous Vacuum Cleaner: Navigates around furniture based on internal map

Scenario: Vacuum Cleaner Robot

Initial State:

- The robot is in position (0,0).
- The floor at (0,0) is dirty.
- The robot's internal model reflects that (0,0) is dirty.

Process:

1. **Perception**:
    1. The robot perceives that (0,0) is dirty.
2. **Internal State Update**:
    1. The robot updates its internal state based on this perception. It now knows that the current location is dirty and needs to be cleaned.
3. **Decision Making**:
    1. The robot refers to its rule base:
        1. *Rule: If the current location is dirty, clean it.*
    2. It decides to clean (0,0).

Scenario: Vacuum Cleaner Robot

4. Action:
    1. The robot performs the cleaning action at (0,0).
    2. As a result, the floor at (0,0) is now clean.
5. World Evolution:
    1. The world has evolved: (0,0) has changed from dirty to clean.
    2. The robot’s model of the world now needs to be updated to reflect this change.
6. Feedback:
    1. After cleaning, the robot may move to the next location, for example, (0,1).
    2. The robot perceives the state of (0,1) (e.g., whether it is dirty or clean).
7. Repeat:
    1. The process repeats. The robot perceives the state at (0,1), updates its internal state, makes decisions based on its rules, performs actions, and the world continues to evolve.

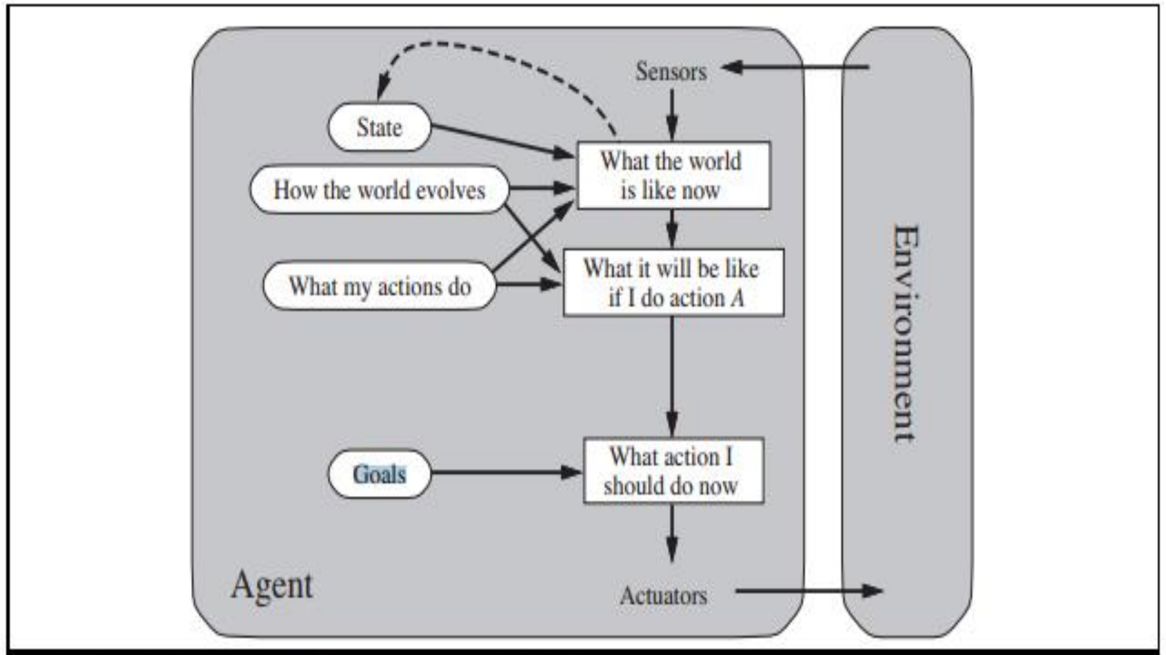

Sensors
State
What the world is like now
How the world evolves
What it will be like if I do action *A*
What my actions do
Environment
Goals
What action I should do now
Agent
Actuators

Figure 2.13 A model-based, goal-based agent. It keeps track of the world state as well as a set of goals it is trying to achieve, and chooses an action that will (eventually) lead to the achievement of its goals.

Types of AI Agents

Goal Based Agents:

- ❑ Focus on reaching the goal state
- ❑ Agent takes decision on how far it is from the goal state
- ❑ Every action is taken to minimize distance to goal state
- ❑ Exp: Chess Program where goal is to checkmate the opponent

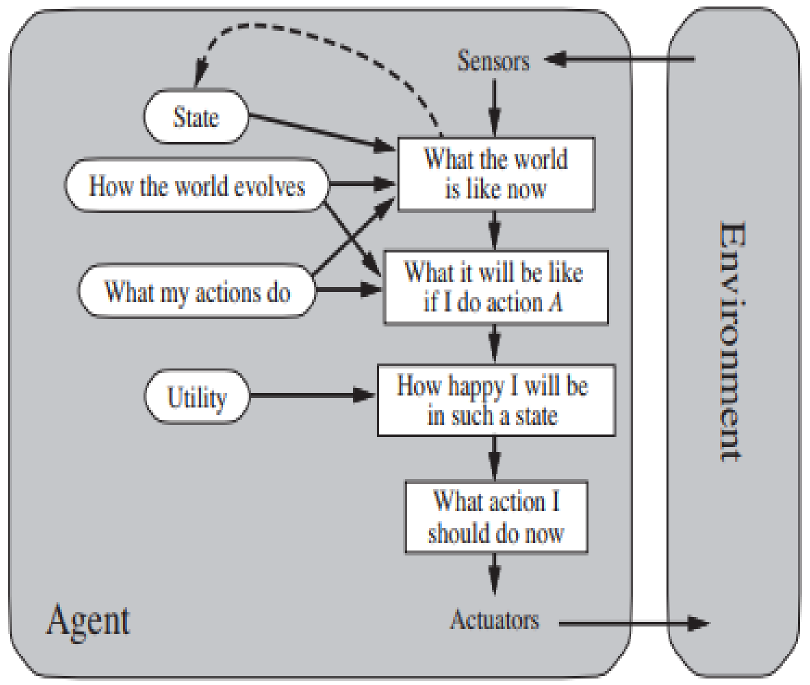

Sensors
State
How the world evolves
What my actions do
Utility
What the world is like now
What it will be like if I do action A
How happy I will be in such a state
What action I should do now
Actuators
Agent
Environment

Figure 2.14 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.

Types of AI Agents

Utility Based Agents:

- ❑ More concerned about the utility (preference) for each state
- ❑ Utility function measures happiness
- ❑ Act based not only on goals but best way to achieve the goal
- ❑ Useful when there are multiple possible alternatives, and agent has to choose the best possible.

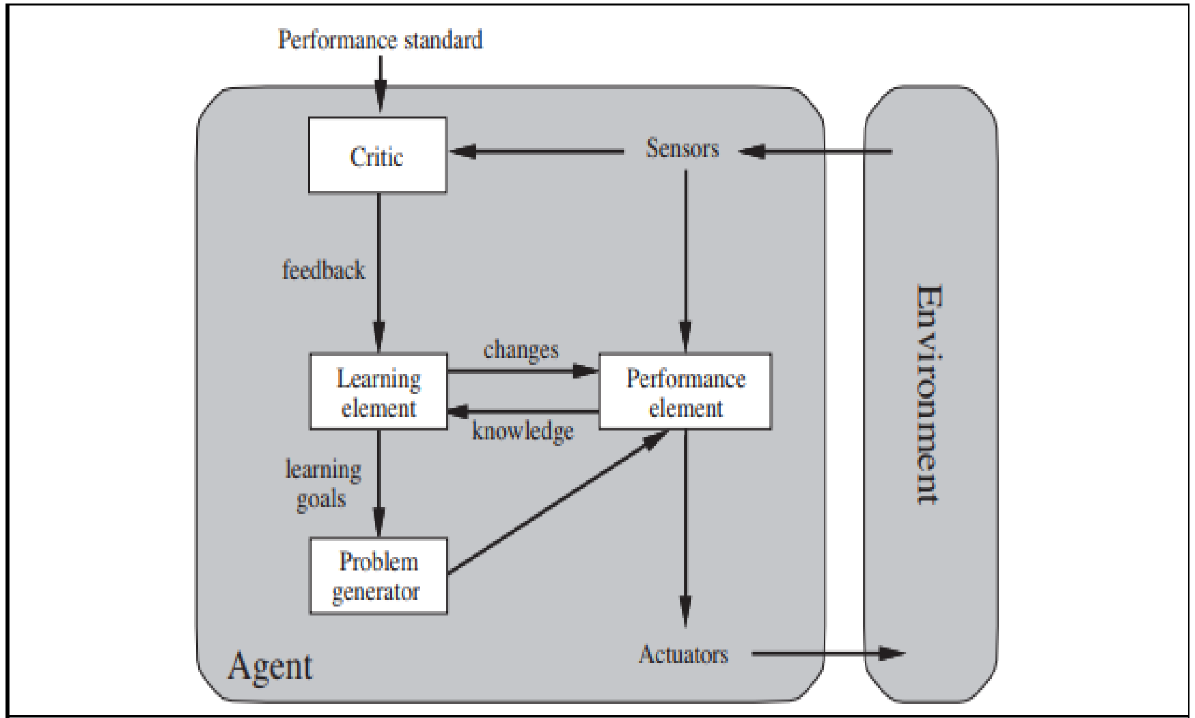


Figure 2.15 A general learning agent.

Types of AI Agents

Learning Agents:

- ❑ Can learn from its past experiences
- ❑ Starts to act with basic knowledge and then able to act by adapting learning.
- ❑ Exp: Game Playing agent improves learning by reinforcement learning