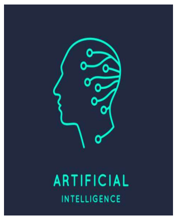
# Artificial Intelligence Lecture 5



Btech CSE 7th

GNDU REGIONAL CAMPUS JALANDHAR

Dr Vinit Grewal , GNDU RC Jalandhar

The job of AI is to design an **agent program** that implements the agent function— the mapping from

percepts to actions $\longrightarrow$ Structure of agents

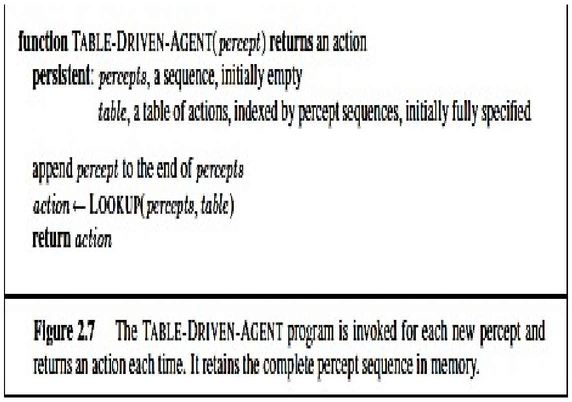
## The Structure of Agents

- **Agent architecture:** computing device with physical sensors and actuators
- **Agent = architecture + program**
- ***the architecture makes the percepts from the sensors available to the program runs the program, and feeds the program's action choices to the actuators as they are generated.***
-

## Agent program vs Agent function

- The difference is that the **agent program takes the current percept as input**, and the **agent function takes the entire percept history.**
- The agent program takes just the current percept as input because nothing more is available from the environment;

**But if the agent's actions need to depend on the entire percept sequence, the agent will have to remember the percepts.**



```
function TABLE-DRIVEN-AGENT(percept) returns an action
  persistent: percepts, a sequence, initially empty
              table, a table of actions, indexed by percept sequences, initially fully specified

  append percept to the end of percepts
  action ← LOOKUP(percepts, table)
  return action
```

**Figure 2.7** The TABLE-DRIVEN-AGENT program is invoked for each new percept and returns an action each time. It retains the complete percept sequence in memory.

The daunting size of these tables means that
(a) no physical agent in this universe will have the space to store the table
(b) the designer would not have time to create the table
(c) no agent could ever learn all the right table entries from its experience
(d) even if the environment is simple enough to yield a feasible table size, the designer still has no guidance about how to fill in the table entries

- **The key challenge for AI is to find out how to write programs that, to the extent possible, produce rational behavior from a small program rather than from a vast table.**

- **Four basic types in order of increasing generality:**

➢ **Simple reflex agents**

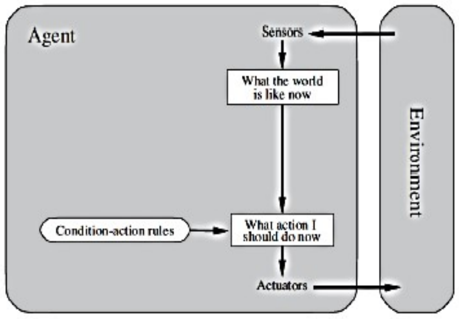
➢ **Model-based reflex agents**

➢ **Goal-based agents**

➢ **Utility-based agents**

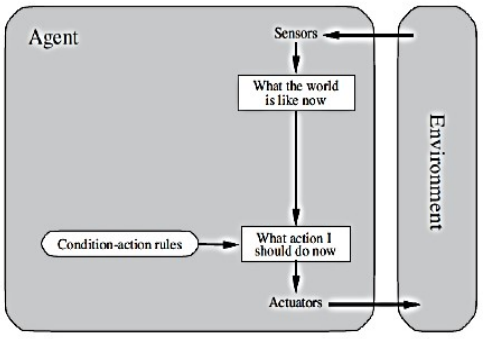
## Simple Reflex Agents

```
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
```



- **The simplest kind of agent is the simple SIMPLE REFLEX reflex agent.**
- **These agents select actions on the basis of the *current* percept, ignoring the rest of the percept history.**
- **For example, the vacuum agent is a simple reflex agent, because its decision is based only on the current location and on whether that location contains dirt.**
- **The most obvious reduction comes from ignoring the percept history, which cuts down the number of possibilities from 4^T to just 4. and the fact that when the current square is dirty, the action does not depend on the location.**
- **Simple reflex behaviors occur even in more complex environments.**
- **such a connection a condition–action rule**

```
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
```

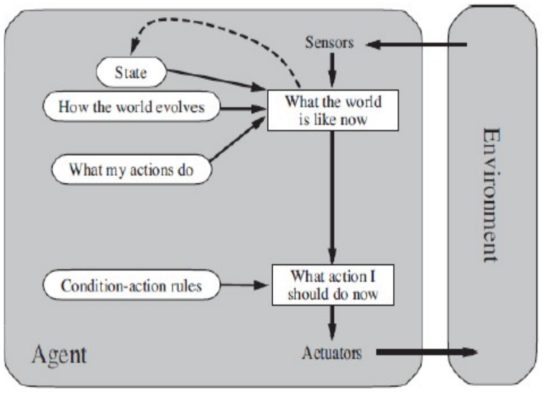


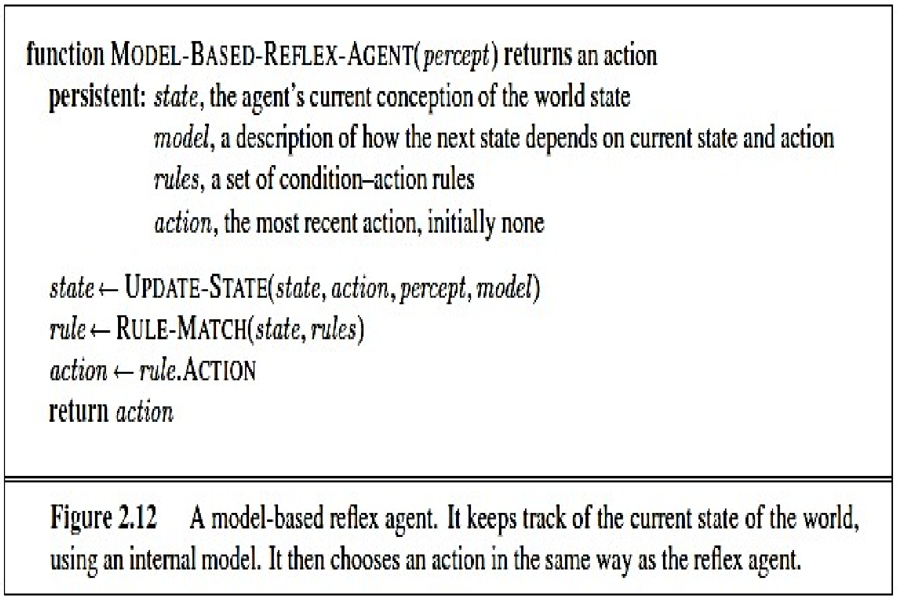
- rectangles denote the current internal state of the agent's decision process
- ovals represent the background information used in the process.
- The agent program
  - INTERPRET-INPUT function generates an abstracted description of the current state from the percept
  - the RULE-MATCH function returns the first rule in the set of rules that matches the given state description

**The agent in Figure will work** ***only if the correct decision can be made on the basis of only the current percept—that is, only if the environment is fully observable***

## Model-based reflex agents

- 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.





```
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.

- The agent should maintain some sort of **internal state** that depends on the percept history and reflects at least some of the unobserved aspects of the current state.
- *For the braking problem, the internal state is not too extensive— just the previous frame from the camera, allowing the agent to detect when two red lights at the edge of the vehicle go on or off simultaneously. For other driving tasks such as changing lanes, the agent needs to keep track of where the other cars are if it can't see them all at once.*
- Updating this internal state information as time goes by requires two kinds of knowledge to be encoded in the agent program.

➢ **some information about how the world evolves independently of the agent.**

➢ **some information about how the agent's own actions affect the**

- This knowledge about "how the world works"—whether implemented in simple Boolean circuits or in complete scientific theories—is called a **model** of the world.
- An agent that uses such a MODEL-BASED model is called a **model-based agent**

- structure of the model-based reflex agent with internal state, that ***the current percept is combined with the old internal state*** to ***generate the updated description of the current state, based on the agent's model of how the world works.***
- The agent program is responsible for creating the **new internal state description**

## Goal-based agents

- Knowing 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.*

- GOAL agent needs some sort of **goal** information that describes situations that are *desirable—for example, being at the passenger's 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.
- **Search** and **planning** are the subfields of AI devoted to finding action sequences that achieve the agent's goals.

✓ ***"What will happen if I do such-and-such?"***
✓ ***"Will that make me happy?***

## Reflex vs Goal

- **The information is not explicitly represented, because the built-in rules map directly from percepts to actions.**
- *The reflex agent brakes when it sees brake lights. A goal-based agent, in principle, could reason that if the car in front has its brake lights on, it will slow down :the only action that will achieve the goal of not hitting other cars is to brake.*
- **Goal-based agent appears less efficient, it is more flexible because the knowledge that supports its decisions is represented explicitly. and can be modified.**
- *If it starts to rain, the agent can update its knowledge of how effectively its brakes will operate; this will automatically cause all of the relevant behaviors to be altered to suit the new conditions.*
- **For the reflex agent, we have to rewrite many condition–action rules. Behavior can easily be changed to go to a different destination, simply by specifying that destination as the goal.**
- ***The reflex agent's rules for when to turn and when to go straight will work only for a single destination; they must all be replaced to go somewhere new.***

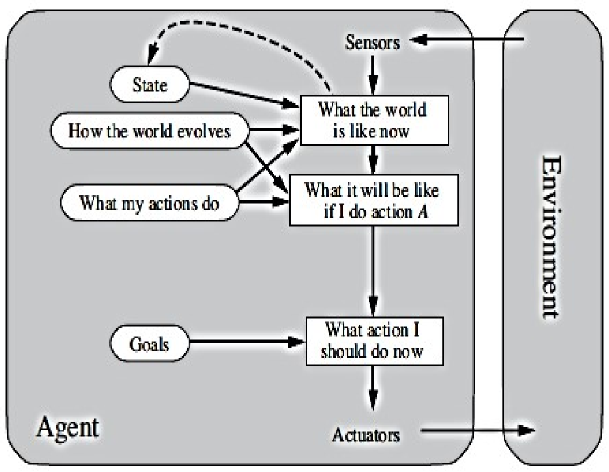

Sensors
State
How the world evolves
What my actions do
What the world is like now
What it will be like if I do action A
Goals
What action I should do now
Agent
Actuators
Environment

**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.

## Utility based Agent

- Goals alone are not enough to achieve high-quality behaviour in most environments
- Some action sequences are better (cheaper, quicker, more reliable, …) than others.
- A utility function maps a state (or a sequence of states) onto a real number, which describes the associated degree of happiness of the agent

- agent's **utility function** is essentially an internalization of the performance measure. If the internal utility function and the external performance measure are in agreement, then an agent that chooses actions to maximize its utility will be rational according to the external performance measure.

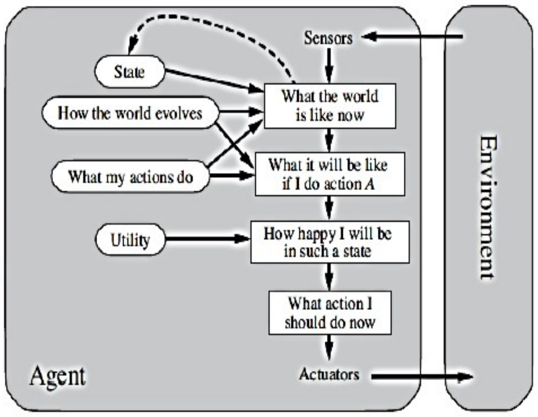
- In two cases, goals are inadequate but a utility-based agent can still make rational decisions.

✓ **First, when there are conflicting goals, only some of which can be achieved (for example, speed and safety), the utility function specifies the appropriate tradeoff.**

✓ **Second, when there are several goals that the agent can aim for, none of which can be achieved with certainty, utility provides a way in which the likelihood of success can be weighed against the importance of the goals**.

- *rational utility-based agent chooses the action that maximizes the **expected utility** of the action outcomes*
- Any rational agent must behave *as if* it possesses a utility function whose expected value it tries to maximize.
- An agent that possesses an *explicit* utility function can make rational decisions with a general-purpose algorithm that does not depend on the specific utility function being maximized.

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


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

## Learning Agents

- Programming intelligent machines by hand is extremely time consuming
- Better method: program a learning algorithm and then teach machines what they should learn
  - Example: Supervised learning of artificial neural networks (ANNs) or support vector machines (SVMs)
- Or better still, let these machines learn by themselves (under some constraints)
  - Reinforcement learning of artificial neural networks
- Learning is nowadays **the** big issue in AI!

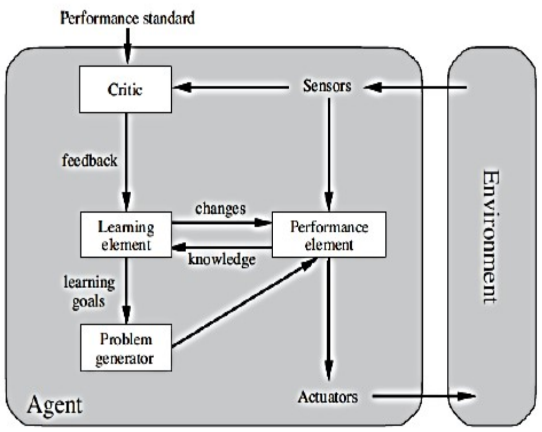
- A learning agent can be divided into four conceptual components,
- The most important distinction is between
- **the learning element,** which is responsible for making improvements, and
- **the performance element,** which is responsible for selecting external actions. T
- The performance element is what we have previously considered to be the entire agent: 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**

- The **critic** tells the learning element how well the agent is doing with respect to a fixed performance standard.
- The critic is necessary because the percepts themselves provide no indication of the agent's success.
- *For example, a chess program could receive a percept indicating that it has checkmated its opponent, but it needs a performance standard to know that this is a good thing*; the percept itself does not say so.
- It is important that the performance standard be fixed

- The last component of the learning agent is the **problem generator**. It is responsible for suggesting actions that will lead to new and informative experiences.
- If the performance element had its way, it would keep doing the actions that are best, given what it knows. But if the agent is willing to explore a little and do some perhaps suboptimal actions in the short run, it might discover much better actions for the long run

- **The learning element can make changes to any of the "knowledge" components shown in the agent diagrams .**
- **The simplest cases involve learning directly from the percept sequence.**
- **Observation of pairs of successive states of the environment can allow the agent to learn "How the world evolves," and observation of the results of its actions can allow the agent to learn "What my actions do."**

- A general learning agent, consisting of 4 components:
  - Learning element
  - Performance element
  - Critic
  - Problem generator



## How the components of agent programs work

- the representations along an axis of increasing complexity and expressive power—**atomic**, **factored**, and **structured**.
- ***To illustrate these ideas, it helps to consider a particular agent component, such as the one that deals with "What my actions do."***

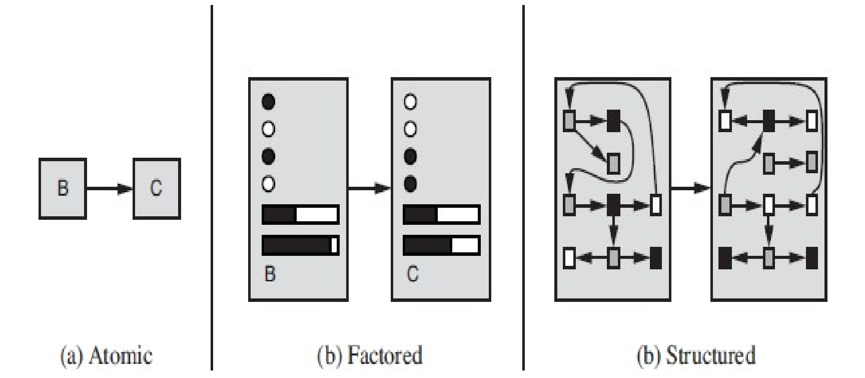


**Figure 2.16** Three ways to represent states and the transitions between them. (a) Atomic representation: a state (such as B or C) is a black box with no internal structure; (b) Factored representation: a state consists of a vector of attribute values; values can be Boolean, real-valued, or one of a fixed set of symbols. (c) Structured representation: a state includes objects, each of which may have attributes of its own as well as relationships to other objects.

- In an **atomic representation** ATOMIC each state of the world is indivisible—it has no internal structure.
- A **factored representation** splits up each state into a fixed set of **variables** or **attributes**, each of which can have a **value**.
- While two different atomic states have nothing in common—they are just different black boxes—two different factored states can share some attributes (such as being at some particular GPS location) and not others; this makes it much easier to work out how to turn one state into another.

- Structured representations represent attributes to be defined and underlie **relational databases** and **first-order logic  first-order probability models, knowledge-based learning** and much of **natural language understanding**