Artificial Intelligence

Unit – I

Chap - III

THE STRUCTURE OF AGENTS

- The job of AI is to design an **agent program that implements the agent function** the mapping from percepts to actions.
- We assume this program will run on some sort of computing device with physical sensors and actuators—we call this the **architecture:**

agent = architecture + program .

Agent programs

- The agent programs that we design in this book all have the same skeleton: they take the current percept as input from the sensors and return an action to the actuators.
- The difference between the agent program, which takes the current percept as input, and the agent function, which takes the entire percept history.
- The agent program takes just the current percept as input because nothing more is available from the environment; if the agent's actions need to depend on the entire percept sequence, the agent will have to remember the precepts.

```
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 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.

- TABLE-DRIVEN-AGENT *does do what we want: it implements the* desired agent function.
- The key challenge for AI is to find out how to write programs that, to the extent possible, produce rational behaviour from a smallish program rather than from a vast table

KINDS OF AGENT PROGRAMS

- four basic kinds of agent programs that embody the principles underlying almost all intelligent systems:
- Simple reflex agents;
- Model-based reflex agents;
- Goal-based agents; and
- Utility-based agents.
- Each kind of agent program combines particular components in particular ways to generate actions.

SIMPLE REFLEX AGENTS

- 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.*

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

CONDITION–ACTION RULE

- Simple reflex behaviours occur even in more complex environments.
- Imagine yourself as the driver of the automated taxi. If the car in front brakes and its brake lights come on, then you should notice this and initiate braking. In other words, some processing is done on the visual input to establish the condition we call "The car in front is braking."
- Then, this triggers some established connection in the agent program to the action "initiate braking." We call such a connection a **condition–action rule**

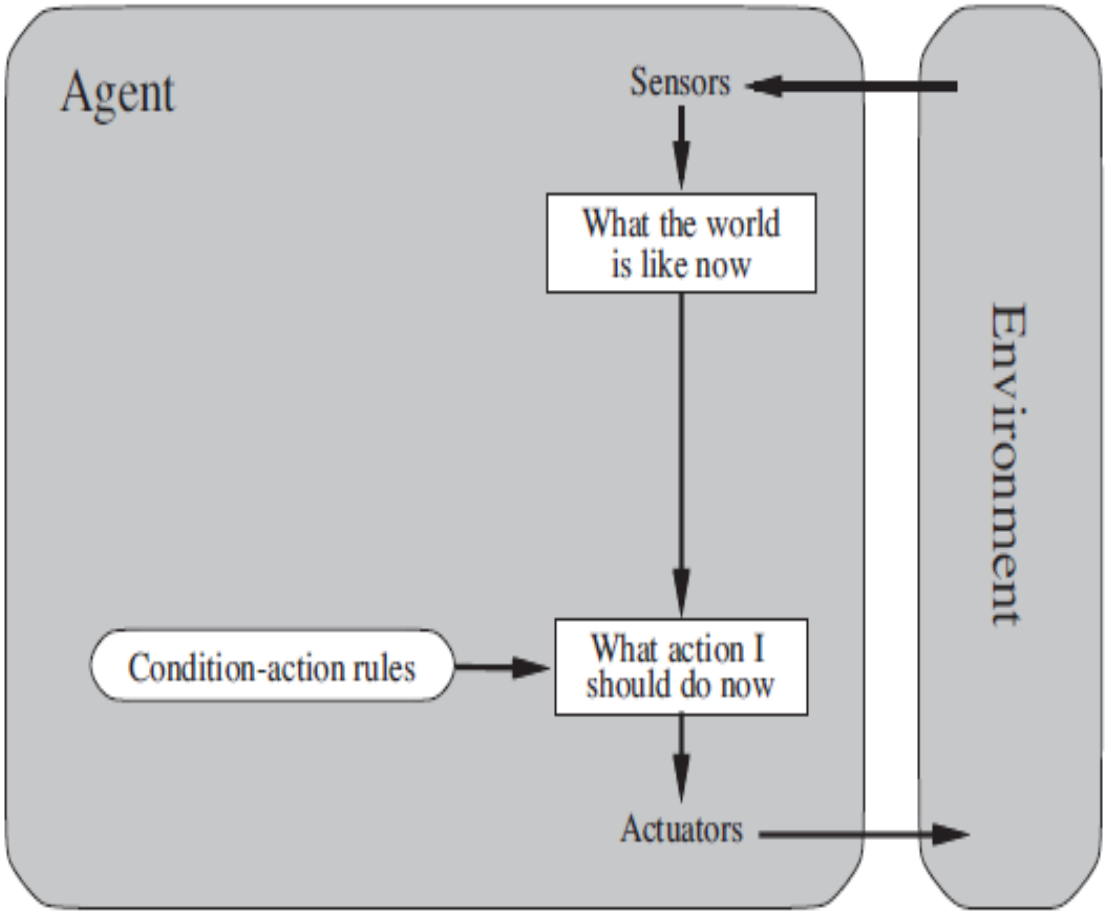

Figure Schematic diagram of a simple reflex agent.

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

Model-based reflex agents

- 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.
- 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 is called a model-based agent.

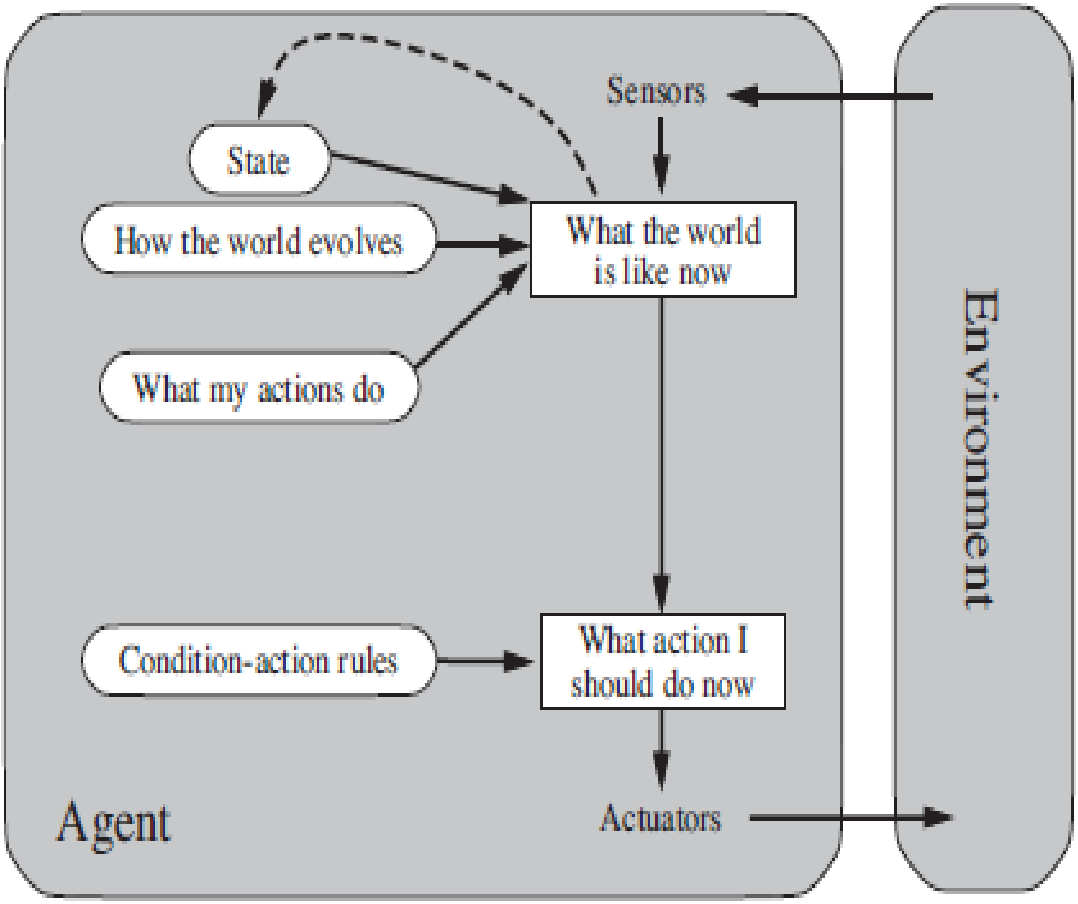

Figure A model-based reflex agent.

```
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 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.

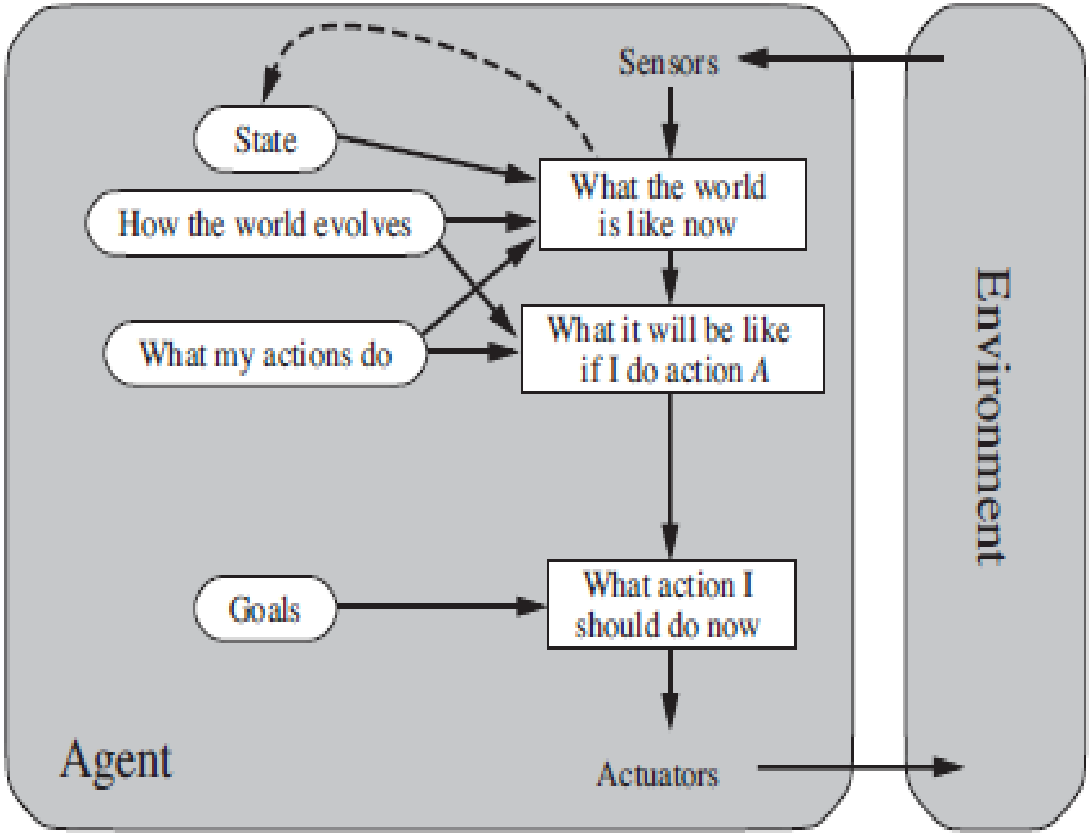

Figure A model-based, goal-based agent. It keeps track of the world state as well as a set of goals it is trying to achieve, and chooses an action that will (eventually) lead to the achievement of its goals.