UNIT – 1

Artificial Intelligence:-

History of AI :-

- The gestation of AI: Warren McCulloch and Walter Pitts(1943): a model of artificial neurons able to perform computations.
- 1950: Alan Turing's "Computing Machinery and Intelligence". First Complete Vision of AI.
- The birth of AI (1956): Dartmouth Workshop bringing together top minds on automata theory, neural nets and the study of intelligence.

What is artificial intelligence?
Definition:
- Systems that think like humans?
- Systems that think rationally?
- Systems that act like humans?
- Systems that act rationally?

Definitions of AI -

- Intelligence: "ability to learn, understand and think" (Oxford dictionary)
- Artificial intelligence is the study of how to make computers do things which, at the moment, people do better.
- Artificial Intelligence (AI) is a branch of computer science and engineering that deals with intelligent behavior, learning, and adaptation in machines.
- "The art of creating machines those perform functions that require intelligence when performed by people".
- "A field of study that seeks to explain and emulate intelligent behavior in terms of computational processes".
- Examples: Speech recognition, Learning new skills, Decision making, Abstract thinking.

What is AI?

Thinking humanly	Thinking rationally
Acting humanly	Acting rationally

Acting humanly: The Turing Test approach

- The **Turing Test,** proposed by Alan Turing (1950), was designed to provide a satisfactory operational definition of intelligence. A computer passes the test if a human interrogator, after posing some written questions, cannot tell whether the written responses come from a person or from a computer.

- The computer would need to possess the following capabilities:
  - ✓ **natural language processing** to enable it to communicate successfully in English;
  - ✓ **knowledge representation** to store what it knows or hears;
  - ✓ **automated reasoning** to use the stored information to answer questions and to draw new conclusions;
  - ✓ **machine learning** to adapt to new circumstances and to detect and extrapolate patterns.

Thinking humanly: The cognitive modeling approach

- If a given program thinks like a human, there must have some way of determining how humans think. It is need to get *inside* the actual workings of human minds.
- There are three ways to do this: through introspection—trying to catch our own thoughts as they go by; through psychological experiments—observing a person in action; and through brain imaging—observing the brain in action. Once we have a sufficiently precise theory of the mind, it becomes possible to express the theory as a computer program. If the program's input—output behavior matches corresponding human behavior, that is evidence that some of the program's mechanisms could also be operating in humans.
- **Cognitive science** brings together computer models from AI and experimental techniques from psychology to construct precise and testable theories of the human mind.

Thinking rationally: The "laws of thought" approach

- Aristotle was one of the first to attempt to codify “right thinking”, i.e., irrefutable reasoning processes. For example, "Socrates is a man; all men are mortal; therefore, Socrates is mortal."
- Formal logic provides a precise notation and rules for representing and reasoning with all kinds of things in the world.
- There are two main obstacles to this approach. First, it is not easy to take informal knowledge and state it in the formal terms required by logical notation. Second, there is a big difference between solving a problem "in principle" and solving it in practice.

Acting rationally: The rational agent approach

- A rational agent is one that acts to achieve the best outcome or, when there is uncertainty, the best expected outcome.
- Does not necessarily involve thinking.

What is Artificial Intelligence?

- According to the father of Artificial Intelligence John McCarthy, it is *“The science and engineering of making intelligent machines, especially intelligent computer programs”*.
- Artificial Intelligence is a way of **making a computer, a computer-controlled robot, or a software think intelligently**, in the similar manner the intelligent humans think.
- AI is accomplished by studying how human brain thinks, and how humans learn, decide, and work while trying to solve a problem, and then using the outcomes of this study as a basis of developing intelligent software and systems.

Philosophy of AI

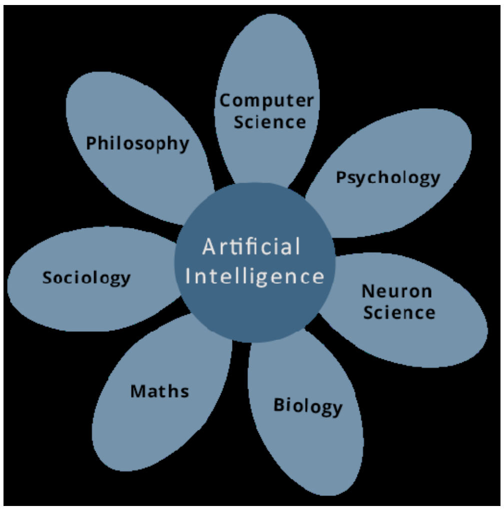
- While exploiting the power of the computer systems, the curiosity of human, lead him to wonder, “Can a machine think and behave like humans do?”
- Thus, the development of AI started with the intention of creating similar intelligence in machines that we find and regard high in humans.

Goals of AI

- **To Create Expert Systems:** The systems which exhibit intelligent behavior, learn, demonstrate, explain, and advice its users.
- **To Implement Human Intelligence in Machines:** Creating systems that understand, think, learn, and behave like humans.

What Contributes to AI?

- Artificial intelligence is a science and technology based on disciplines such as Computer Science, Biology, Psychology, Linguistics, Mathematics, and Engineering. A major thrust of AI is in the development of computer functions associated with human intelligence, such as reasoning, learning, and problem solving.
- Out of the following areas, one or multiple areas can contribute to build an intelligent system.



What is AI Technique?

In the real world, the knowledge has some unwelcomed properties:

- Its volume is huge, next to unimaginable.
- It is not well-organized or well-formatted.
- It keeps changing constantly.

AI Technique is a manner to organize and use the knowledge efficiently in such a way that:

- It should be perceivable by the people who provide it.
- It should be easily modifiable to correct errors.
- It can be used in variety of situations

Applications of AI

AI has been dominant in various fields such as:
Gaming

- AI plays crucial role in strategic games such as chess, poker, tic-tac-toe, etc., where machine can think of large number of possible positions based on heuristic knowledge.

Natural Language Processing

- It is possible to interact with the computer that understands natural language spoken by humans.

Expert Systems

- There are some applications which integrate machine, software, and special information to impart reasoning and advising. They provide explanation and advice to the users.

Vision Systems

- These systems understand, interpret, and comprehend visual input on the computer. For example,
  Doctors use clinical expert system to diagnose the patient, Police use computer software that can recognize the face of criminal with the stored portrait made by forensic artist.

Speech Recognition

- Some intelligent systems are capable of hearing and comprehending the language in terms of sentences and their meanings while a human talks to it. It can handle different accents, slang words, noise in the background, change in human's noise due to cold, etc.

Handwriting Recognition

- The handwriting recognition software reads the text written on paper by a pen or on screen by a stylus. It can recognize the shapes of the letters and convert it into editable text.

<u>Intelligent Robots</u>

- Robots are able to perform the tasks given by a human. They have sensors to detect physical data from the real world such as light, heat, temperature, movement, sound, bump, and pressure.

A Brief History of AI

- The gestation of AI (1943 − 1956):
  - − 1943: McCulloch & Pitts: Boolean circuit model of brain.
  - − 1950: Turing's "Computing Machinery and Intelligence".
  - − 1956: McCarthy's name "Artificial Intelligence" adopted.
- Early enthusiasm, great expectations (1952 − 1969):
  - − Early successful AI programs: Samuel's checkers, Newell & Simon's Logic Theorist, Gelernter's Geometry Theorem Prover.
  - − Robinson's complete algorithm for logical reasoning
- A dose of reality (1966 − 1974):
  - − AI discovered computational complexity.
  - − Neural network research almost disappeared after
- Minsky & Papert's book in 1969.
- Knowledge-based systems (1969 − 1979):
  - − 1969: DENDRAL by Buchanan et al.
  - − 1976: MYCIN by Shortliffle.
  - − 1979: PROSPECTOR by Duda et al.
- AI becomes an industry (1980 − 1988):
  - − Expert systems industry booms.
  - − 1981: Japan's 10-year Fifth Generation project.
- The return of NNs and novel AI (1986 − present):
  - − Mid 80's: Back-propagation learning algorithm reinvented.
  - − Expert systems industry busts.
  - − 1988: Resurgence of probability.
  - − 1988: Novel AI (ALife, GAs, Soft Computing, …).
  - − 1995: Agents everywhere.

− 2003: Human-level AI back on the agenda.

TASK DOMAINS OF AI

Mundane Tasks:

- Perception
    - Vision
    - Speech
- Natural Languages
    - Understanding
    - Generation
    - Translation
- Common sense reasoning
- Robot Control

Formal Tasks:

- Games : chess, checkers etc
- Mathematics: Geometry, logic,Proving properties of programs

Expert Tasks:

- Engineering (Design, Fault finding, Manufacturing planning)
- Scientific Analysis
- Medical Diagnosis
- Financial Analysis

AGENT AND ENVIRONMENTS:-

“Problem-solving agents decide what to do by finding sequences of actions that lead to desirable states.”

- *An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators*. **Example- 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 keystrokes, file contents, and network packets as sensory inputs and acts on the environment by displaying on the screen, writing files, and sending network packets.

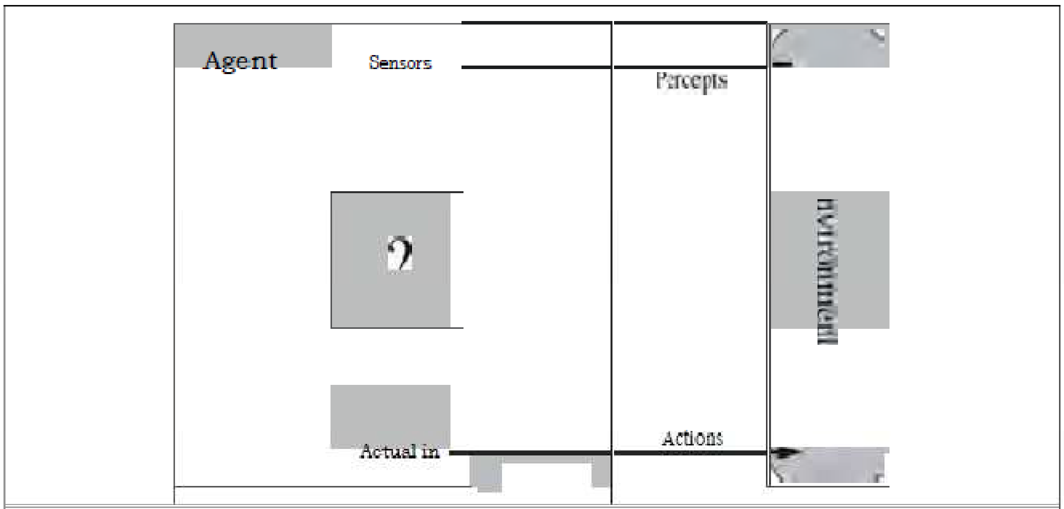


Figure 2.1 Agents interact with environments through sensors and actuators.

- The term **percept is used** to refer to the agent's perceptual inputs at any given instant. 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 the entire percept sequence observed to date, but not on anything it hasn't perceived
.
- **Agent function** are those that maps any given percept sequence to an action. Internally, the agent function for an artificial agent will be implemented by an agent program**.** It is important to keep these two ideas distinct. The agent function is an abstract mathematical description; the agent program is a concrete implementation, running within some physical system.
- To illustrate these ideas, we use a very simple example—the vacuum-cleaner world shown in Figure 2.2. This particular world has just two locations: squares A and *B*. The vacuum agent perceives which square it is in and whether there is dirt in the square. It can choose to move left, move right, 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.

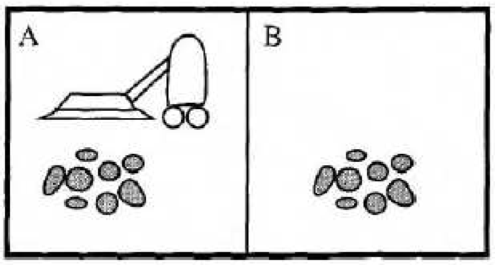
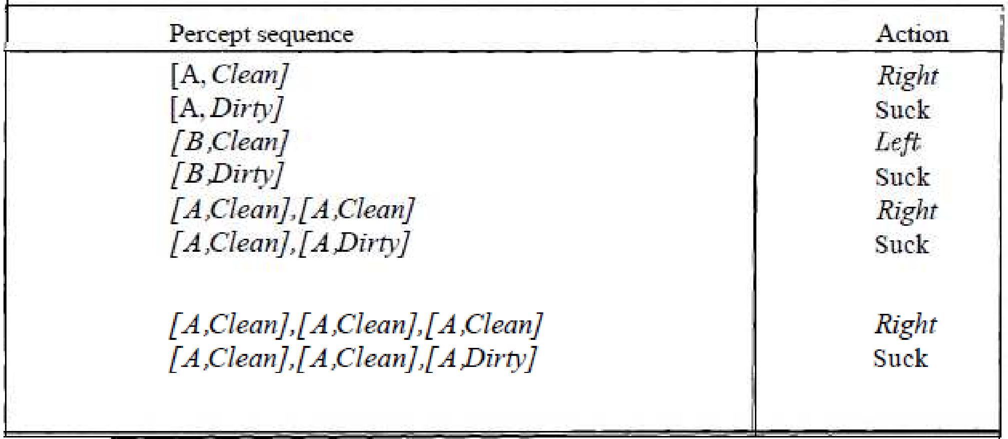


Figure 2.2 A vacuum-cleaner world with just two locations.



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.

THE CONCEPT OF RATIONALITY

- A **rational agent** is one that does the right thing—conceptually speaking, every entry in the table for the agent function is filled out correctly. Doing the right thing is better than doing the wrong thing.
- As a first approximation, we will say that the right action is the one that will cause the agent to be most successful. Therefore, we will need some way to measure success. What is rational at any given time depends on four things:

  ✓ The performance measure that defines the criterion of success.
  ✓ The agent's prior knowledge of the environment.
  ✓ The actions that the agent can perform.
  ✓ The agent's percept sequence to date.

This leads to a definition **of a rational agent:**
For each possible percept sequence, a rational agent should select an action that is ex-pected to maximize its performance measure, given the evidence provided by the percept sequence and whatever built-in knowledge the agent has.

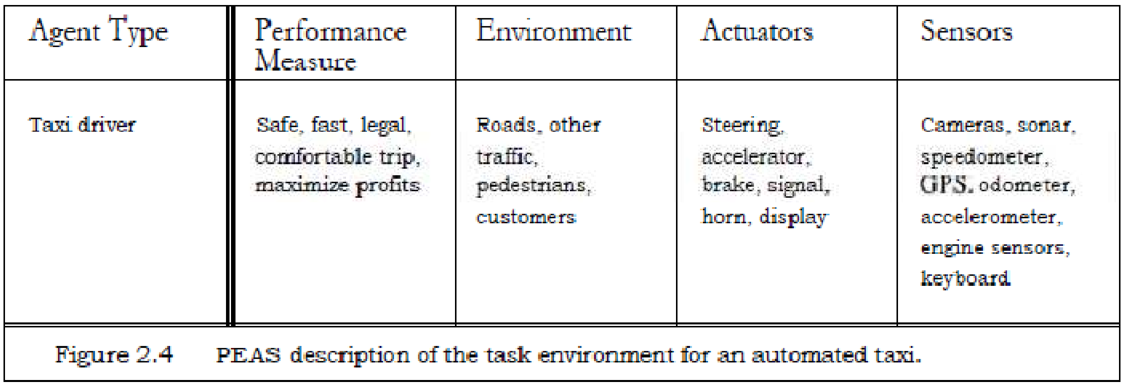
Performance measures

- **A performance measure** embodies the criterion for success of an agent's behaviour. When an agent is plunked down in an environment, it generates a sequence of actions according to the percepts it receives. This sequence of actions causes the environment to go through a sequence of states. If the sequence is desirable, then the agent has performed well.
- **An omniscient** agent knows the *actual* outcome of its actions and can act accordingly; but omniscience is impossible in reality. Rationality maximizes *expected* performance, while perfection maximizes *actual* performance
- **Example:** I am walking along the Champs ElysCes one day and I see an old friend across the street. There is no traffic nearby and I'm not otherwise engaged, so, being rational, I start to cross the street. Meanwhile, at 33,000 feet, a cargo door falls off a passing before I make it to the other side of the street I am flattened. Was I irrational to cross the street? It is unlikely that my obituary would read "Idiot attempts to cross street." **This example shows that rationality is not the same as perfection. Rationality maximizes *expected* performance, while perfection maximizes *actual* performance.**

- *A rational agent should be autonomous—it should learn what it can to compensate for partial or incorrect prior knowledge.* For example, a vacuum-cleaning agent that learns to foresee where and when additional dirt will appear will do better than one that does not. As a practical matter, one seldom requires complete autonomy from the start: when the agent has had little or no experience, it would have to act randomly unless the designer gave some assistance. So, just as evolution provides animals with enough built in reflexes to survive long enough to learn for themselves, it would be reasonable to provide an artificial intelligent agent with some initial knowledge as well as an ability to learn.

Specifying the task environment

- We had to specify the performance measure, The environment, the agent's actuators and sensors. We group all these under the heading of the **task environment. W**e call this the PEAS (Performance, Environment, Actuators, and Sensors) description.



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.

<u>Properties of task environments</u> :-

- **Fully observable** vs. **partially observable**: *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.
- *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 divers are thinking. If the agent has no sensors at all then the environment is unobservable**.**

Deterministic vs. stochastic.

- *If the next state of the environment is completely determined by the current state and the action executed by the agent, then we say the environment is deterministic; otherwise, it is stochastic*.

Episodic vs. **sequential:**

- *In an episodic task environment, the agent's experience is divided into atomic episodes. In each episode the agent receives a percept and then performs a single action. Crucially, 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 environments, on the other hand, the current decision could affect all future decisions.* **Chess and taxi driving are sequential:** in both cases, short-term actions can have long-term consequences

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.
- **Taxi driving is clearly dynamic**: the other cars and the taxi itself keep moving while the driving algorithm dithers about what to do next. **Chess, when played with a clock**, is **semi dynamic. Crossword puzzles are static**.

Discrete vs. **continuous:**

- The discrete/continuous distinction applies to the *state* of the environment, to the way *time* is handled, and to the *percept's* and *actions* of the agent**.**

- For example, 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.)

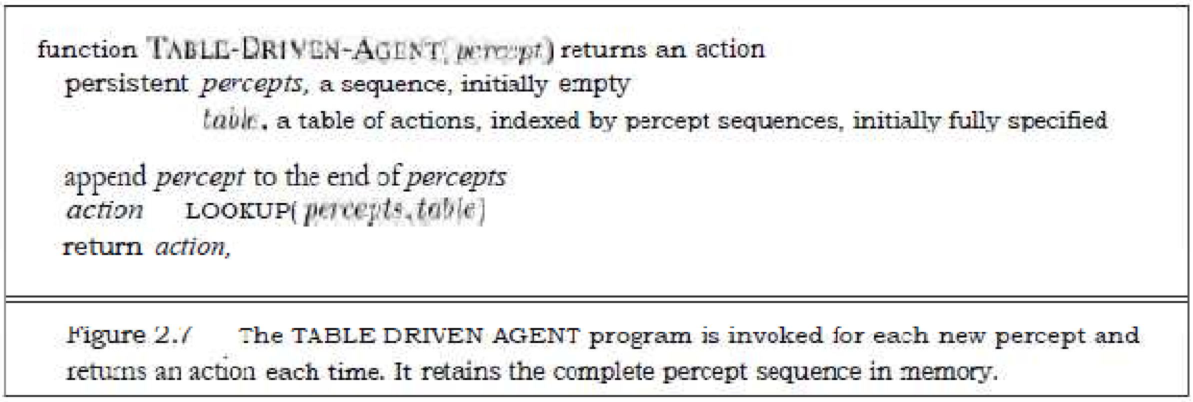
THE STRUCTURE OF AGENTS

- *The job of Al 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 take the current percept as input from the sensors and return an action to the actuators. Notice 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 theenvironment; if the agent's actions depend on the entire percept sequence, the agent will have to remember the percepts.



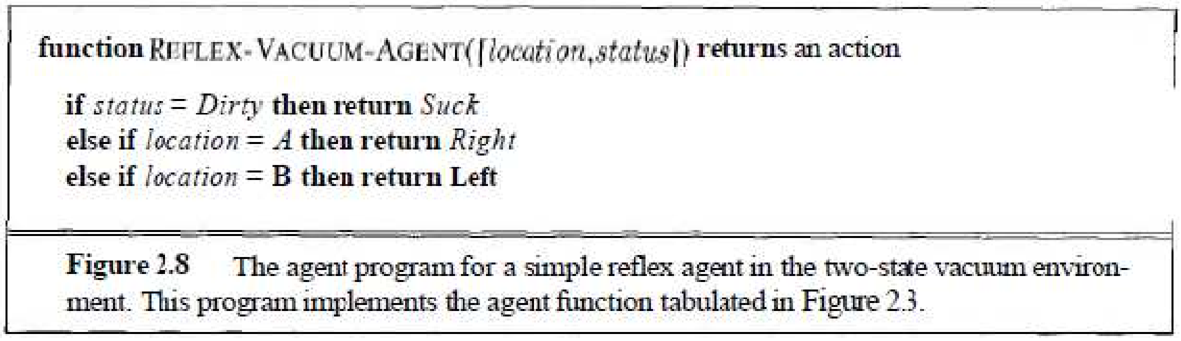
```
function TABLE-DRIVEN-AGENT(percept) returns an action
  persistent percepts, a sequence, initially empty
             table, a table of actions, indexed by percept sequences, initially fully specified

  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.

Consider the automated taxi: the visual input from a single camera comes in at the rate of roughly 27 megabytes per second (30 frames per second, 640 x 480 pixels with 24 bits of colour information). This gives a lookup table with over 10250,000,000,000 entries for an hour's driving. Even the lookup table for chess-a tiny, well-behaved fragment of the real world would have at least 10150 entries. The daunting size of these tables (the number of atoms in the observable universe is less than1080) means that ➢ No physical agent in this universe will have the space to store the table,

- ✓ The designer would not have time to create the table,
- ✓ No agent could ever learn all the right table entries from its experience, and
- ✓ 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.



```
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
```

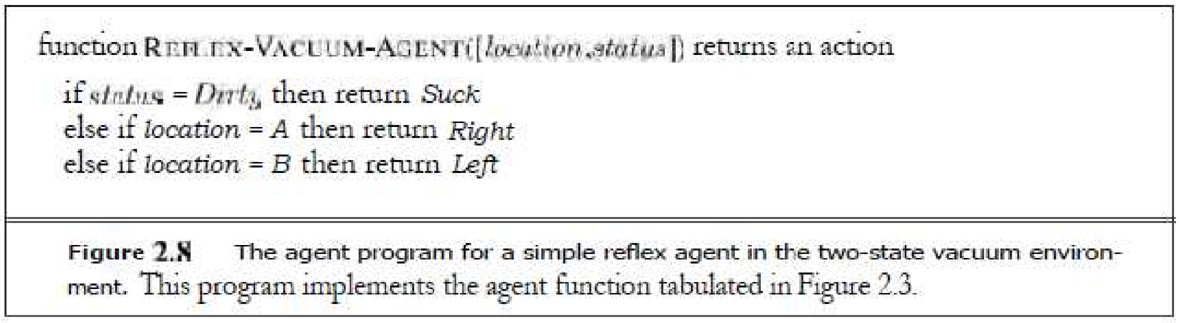
Figure 2.8 The agent program for a simple reflex agent in the two-state vacuum environment. This program implements the agent function tabulated in Figure 2.3.

There are four basic kinds of agent programs that embody the principles underlying almost all intelligent systems:

i. Simple reflex agents;
ii. Model-based reflex agents;
iii. Goal-based agents; and
iv. Utility-based agents.

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. 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.



```
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.

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

```
function SIMPLE-REFLEX-AGENT(percept) returns action
   static: rules, a set of condition-action rules

   state ← INTERPRET-INPUT(percept)
   rule ← RULE-MATCH(state, rules)
   action ← RULE-ACTION[rule]
   return action
```

Figure 2.8 A simple reflex agent. It works by finding a rule whose condition matches the current situation (as defined by the percept) and then doing the action associated with that rule.

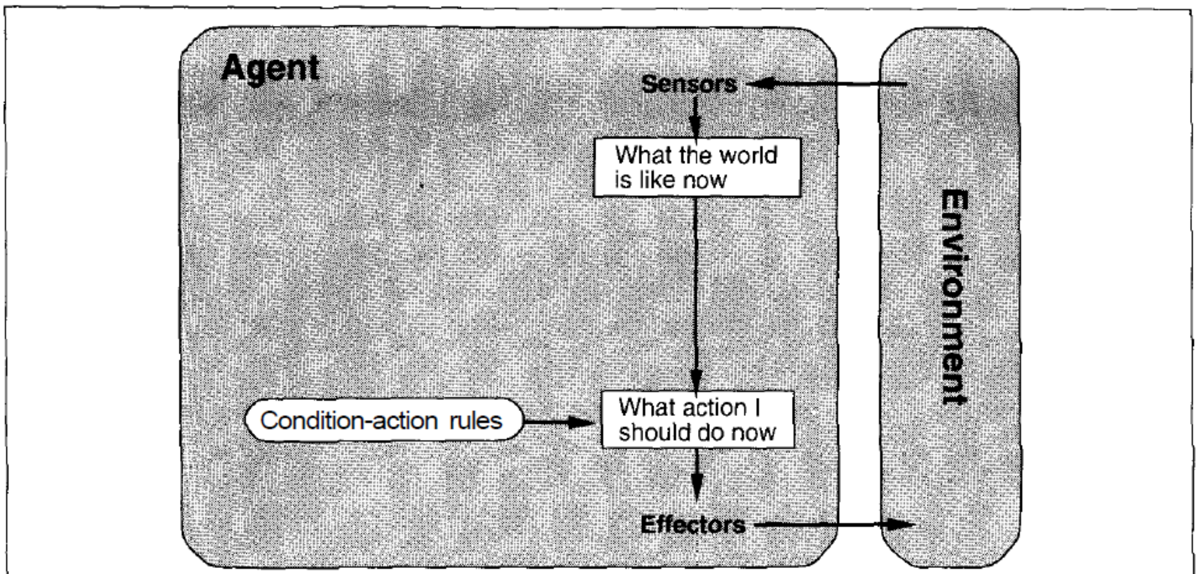


Figure 2.7 Schematic diagram of a simple reflex agent.

- We use rectangles to denote the current internal state of the agent's decision process, and ovals to represent the background information used in the process.

- **The INTERPRET-INPUT** function generates an abstracted description of the current slate from the percept, and the **RULE-MATCH** function returns the first rule in the set of rules that matches the given state description.
- The agent will work *only if the correct decision can be made on the basis of only the current percept—that is if the environment is fully observable.* Even a little bit of unobservability can cause serious trouble Suppose that a simple reflex vacuum agent is deprived of its location sensor and has only a dirt sensor. Such an agent has just two possible percepts: *[Dirty]* and *[Clean]*. It can *Suck* in response to *[Dirt];* what should it do in response to *[Clean]?* Moving *Left* fails (forever) if it happens to start in square *A*, and moving *Right* fails (forever) if it happens to start in square*B.* Infinite loops are often unavoidable for simple reflex agents **operating in partially observable environments.**

2. Model-based reflex agents

- The most effective way to handle partial observability is for the agent to *keep track of the part of the world it can't see now.* That is, 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. Updating this internal state information as time goes by requires two kinds of knowledge to be encoded in the agent program.
- **First,** we need some information about how the world evolves independently of the agent for example, that an overtaking car generally will be closer behind than it was a moment ago.
- **Second**, we need some information about how the agent's own actions affect the world for example, that when the agent turns the steering wheel clockwise, the car turns to the right, or that after driving for five minutes northbound on the freeway, one is usually about five miles north of where one was five minutes ago.
- *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 is called a **model-based agent.***

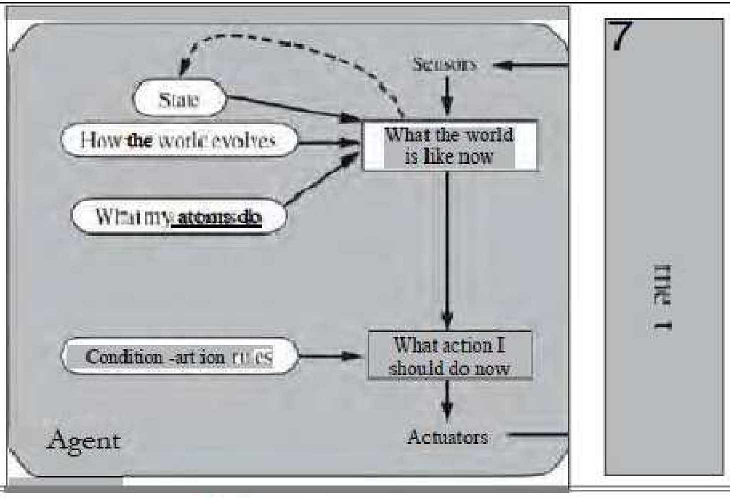


Figure 2.11 A model-based reflex agent.

```
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,
  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.

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 Goal as a current state description, the 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.
- Although the 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.

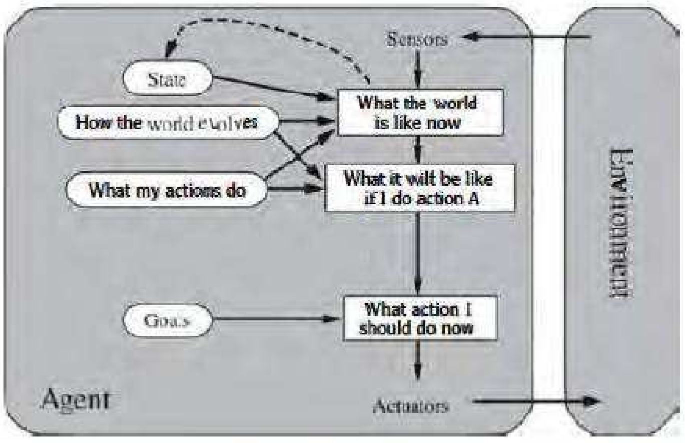

Sensors
State
How the world evolves
What my actions do
What the world is like now
What it will be like if I do action A
Goals
What action I should do now
Agent
Actuators
Environment

Figure 2.13 A model-based, goal-based agent. It keeps track of the world stale 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 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 term utility instead.*
- Furthermore, in two kinds of 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**.**

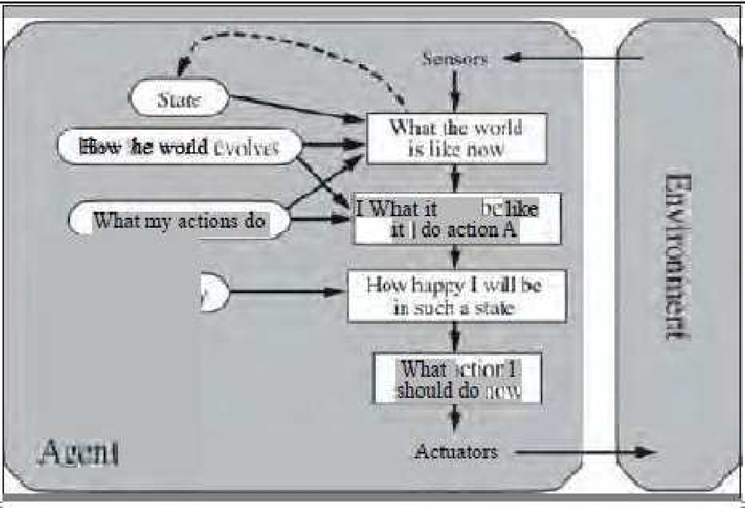

Sensors
State
How the world evolves
What the world is like now
What my actions do
What it will be like if I do action A
How happy I will be in such a state
What action I should do now
Agent
Actuators
Environment

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.

Learning agents

- 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.
- 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. The point is that if the performance element had its way, it would keep doing the actions that are best; given what it knows

  Example:-
- The taxi goes out on the road and drives, using this performance element. The critic observes the world and passes information along to the learning element. For example, after the taxi makes a quick left turn across three lanes of traffic, the critic observes the shocking language used by other drivers. From this experience, the learning element is able to formulate a rule saying this was a bad action, and the performance element is modified by

installation of the new rule. The problem generator might identify certain areas of behavior in need of improvement and suggest experiments, such as trying out the brakes on different roadsurfaces under different conditions.

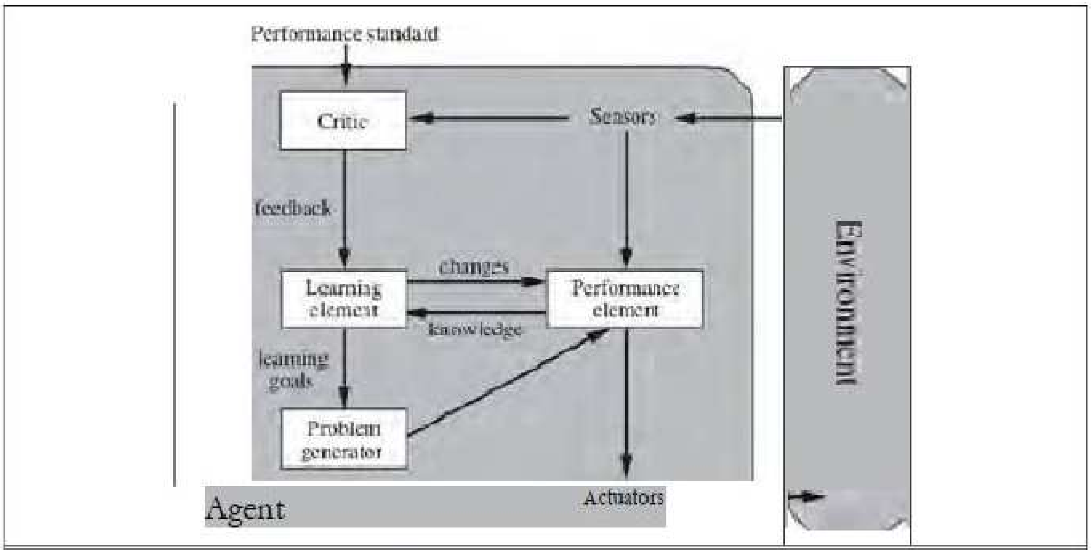


Figure 2.15 A general learning agent.

Computer vision

- *Computer vision is a field that includes methods for acquiring, processing, analyzing, and understanding images and, in general, high-dimensional data from the real world in order to produce numerical or symbolic information, e.g., in the forms of decisions.*
- A theme in the development of this field has been to duplicate the abilities of human vision by electronically perceiving and understanding an image. Computer vision has also been described as the enterprise of automating and integrating a wide range of processes and representations for vision perception. As a technological discipline, computer vision seeks to apply its theories and models to the construction of computer vision systems. Examples of applications of computer vision include systems for:

  - ✓ **Controlling processes**, *e.g.*, an industrial robot;
  - ✓ **Navigation,** *e.g.*, by an autonomous vehicle or mobile robot;
  - ✓ **Detecting events**, *e.g.*, for visual surveillance or people counting;
  - ✓ **Organizing information**, *e.g.*, for indexing databases of images and image sequences;
  - ✓ **Modelling objects or environments**, *e.g.*, medical image analysis or topographical modeling;
  - ✓ **Interaction**, *e.g.*, as the input to a device for computer-human interaction, and
  - ✓ **Automatic inspection,** *e.g.*, in manufacturing applications.

Computer vision system methods

- The organization of a computer vision system is highly application dependent. The specific implementation of a computer vision system also depends on if its functionality is pre specified or if some part of it can be learned or modified during operation. Many functions are unique to the application. Typical functions which are found in many computer vision systems are :

1. Image acquisition – A digital image is produced by one or several image sensors, which, besides various types of light-sensitive cameras, include range sensors, tomography devices, radar, ultra-sonic cameras, etc. Depending on the type of sensor, the resulting image data is an ordinary 2D image, a 3D volume, or an image sequence.

2. Pre-processing – Before a computer vision method can be applied to image data in order to extract some specific piece of information, it is usually necessary to process the data in order to assure that it satisfies certain assumptions implied by the method. Examples are

- ✓ Re-sampling in order to assure that the image coordinate system is correct.
- ✓ Noise reduction in order to assure that sensor noise does not introduce false information.
- ✓ Contrast enhancement to assure that relevant information can be detected.

3. Feature extraction – Image features at various levels of complexity are extracted from the image data.

4. Detection/segmentation – At some point in the processing a decision is made about which image points or regions of the image are relevant for further processing.

- ✓ Selection of a specific set of interest points
- ✓ Segmentation of one or multiple image regions which contain a specific object of interest.

5. High-level processing – At this step the input is typically a small set of data, for example a set of points or an image region which is assumed to contain a specific object. The remaining processing deals with, for example:

- ✓ Verification that the data satisfy model-based and application specific assumptions.
- ✓ Estimation of application specific parameters, such as object pose or object size.
- ✓ Image recognition – classifying a detected object into different categories.
- ✓ Image registration – comparing and combining two different views of the same object.

6. Decision making Making the final decision required for the application, for example:

- ✓ Pass/fail on automatic inspection applications
- ✓ Match / no-match in recognition applications
- ✓ Flag for further human review in medical, military, security and recognition applications

Natural language processing

- *Natural language processing (NLP) is a field of computer science, artificial intelligence, and linguistics concerned with the interactions between computers and human (natural) languages.* Many challenges in NLP involve natural language understanding, that is, enabling computers to derive meaning from human or natural language input, and others involve natural language generation.

NLP using machine learning

- Modern NLP algorithms are based on machine learning, especially statistical machine learning. The machine-learning paradigm calls instead for using general learning algorithms often, although not always, grounded in statistical inference to automatically learn such rules algorithms, such as decision trees, produced systems of hard if-then rules similar to the systems of hand-written rules that were then common. through the analysis of large *corpora* of typical real-world examples. Many different classes of machine learning algorithms have been applied to NLP tasks. Some of the earliest used

Major tasks in NLP

A list of some of the most commonly researched tasks in NLP

1. Automatic summarization:- Produce a readable summary of a chunk of text. Often used to provide summaries of text of a known type, such as articles in the financial section of a newspaper.

2. Machine translation:- automatically translate text from one human language to another.

3. Morphological segmentation:-Separate words into individual morphemes and identify the class of the morphemes.

4. Named entity recognition (NER):- Given a stream of text, determine which items in the text map to proper names, such as people or places, and what the type of each such name is

5. Natural language generation Convert information from computer databases into readable human language.

6. Natural language understanding: - Convert chunks of text into more formal representations such as first-order logic structures that are easier for computer programs to manipulate. Natural language understanding involves the identification of the intended semantic from the multiple possible semantics which can be derived from a natural language expression which usually takes the form of organized notations of natural languages concepts.

7. Optical character recognition (OCR):- Given an image representing printed text determine the corresponding text.

8. Part-of-speech tagging: - Given a sentence, determine the part of speech for each word. Many words, especially common ones, can serve as multiple parts of speech.

9. Parsing:-Determine the parse tree (grammatical analysis) of a given sentence. The grammar for natural languages is ambiguous and typical sentences have multiple possible analyses.

10. Question answering:- Given a human-language question, determine its answer.

11. Relationship extraction:-Given a chunk of text, identify the relationships among named entities.

12. Speech recognition:-Given a sound clip of a person or people speaking, determine the textual representation of the speech. This is the opposite of text to speech and is one of the extremely difficult problems colloquially termed "AI-complete"

13. Speech segmentation Given a sound clip of a person or people speaking, separate it into words.

14. Information retrieval (IR):- This is concerned with storing, searching and retrieving information. It is a separate field within computer science (closer to databases), but IR relies on some NLP methods (for example, stemming).

15. Information extraction (IE):-This is concerned in general with the extraction of semantic information from text. This covers tasks such as named entity recognition, Coreference resolution, relationship extraction, etc.