Artificial Intelligence, RCA-403

Syllabus

UNIT-I INTRODUCTION: - Introduction to Artificial Intelligence, Foundations and History of Artificial Intelligence, Applications of Artificial Intelligence, Intelligent Agents, Structure of Intelligent Agents. Computer vision, Natural Language Possessing.

UNIT-II INTRODUCTION TO SEARCH: - Searching for solutions, uniformed search strategies, informed search strategies, Local search algorithms and optimistic problems, Adversarial Search, Search for
Games, Alpha - Beta pruning.

UNIT-III KNOWLEDGE REPRESENTATION & REASONING: - Propositional logic, Theory of first order logic, Inference in First order logic, Forward & Backward chaining, Resolution, Probabilistic reasoning, Utility theory, Hidden Markov Models (HMM), Bayesian Networks.

UNIT-IV MACHINE LEARNING: - Supervised and unsupervised learning, Decision trees, Statistical learning models, learning with complete data - Naive Bayes models, Learning with hidden data – EM algorithm, Reinforcement learning,

UNIT-V PATTERN RECOGNITION: - Introduction, Design principles of pattern recognition system, Statistical Pattern recognition, Parameter estimation methods - Principle Component Analysis (PCA)
and Linear Discriminant Analysis (LDA), Classification Techniques – Nearest Neighbour (NN) Rule,
Bayes Classifier, Support Vector Machine (SVM), K – means clustering.

References: -
1. Stuart Russell, Peter Norvig, “Artificial Intelligence – A Modern Approach”, Pearson Education
2. Elaine Rich and Kevin Knight, “Artificial Intelligence”, McGraw-Hill
3. E Charniak and D McDermott, “Introduction to Artificial Intelligence”, Pearson Education
4. Dan W. Patterson, “Artificial Intelligence and Expert Systems”, Prentice Hall of India

UNIT – I

<u>Introduction to Artificial Intelligence</u>

AI is the study of computer science to make machine intelligent.
Artificial intelligence (AI) can be divided into two major categories – **weak** and **strong**.

Strong AI is about building a machine that is capable of emulating the human mind to such an extent that it matches or exceeds it. Strong AI requires the intelligent agent to think outside and beyond the programming guidelines given to it. These machines should be self-aware and their overall intellectual ability needs to be indistinguishable from that of a human being. All the examples of AI shown in movies like Jarvis (the AI system running all the home-systems and the exoskeleton) in the movie Iron Man, the movie AI by Spielburg, the movie Terminator are examples of Strong AI. Needless to say the advent of such machines is quite far-fetched. One area which makes use of this technology is the chat robots which talk to humans.

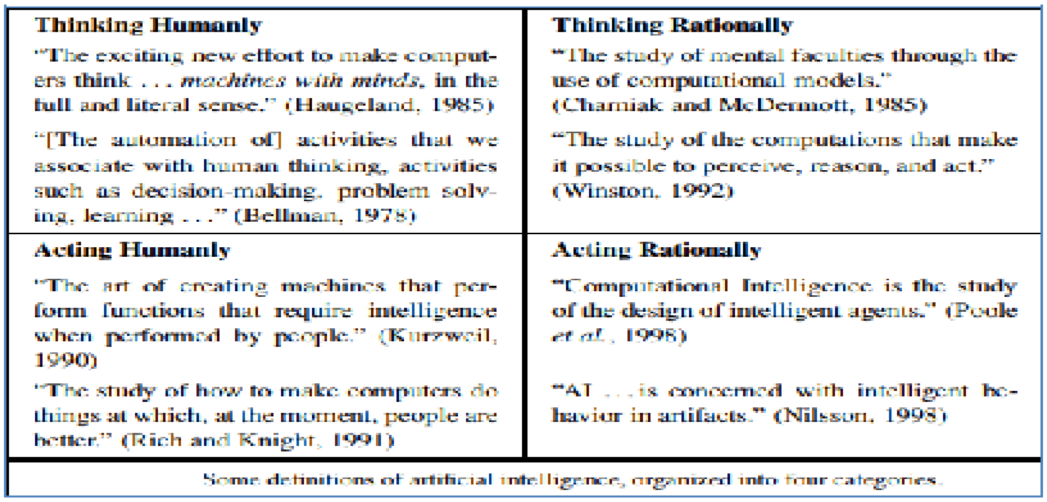
Siri and Alexa could be considered AI, but generally, they are weak AI programs. Even advanced chess programs are considered weak AI. This categorization seems to be rooted in the difference between supervised and unsupervised programming. Voice-activated assistance and chess programs often have a programmed response. If you ask Alexa to turn on the TV, the programming understands key words like On and TV. The algorithm will respond by turning on the TV, but it is only responding to its programming.

	Strong AI	Weak AI
Definition	The machine can actually think and perform tasks on its own just like a human being.	The devices cannot follow these tasks on their own but are made to look intelligent.
Functionality	Algorithm is stored by a computer program.	Tasks are entered manually to be performed.
Examples	There are no proper examples for Strong AI.	An automatic car or remote control devices.
Progress	Initial Stage	Advanced Stage

<u>Definition of AI</u>

AI is actually mapping of intelligence where intelligence is boundary less. Boundaries of AI are:

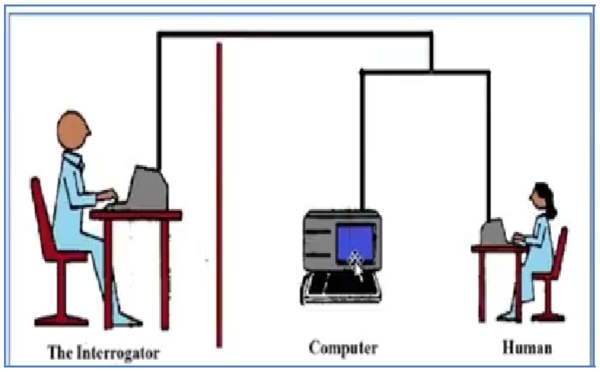
1. Acting Humanly
2. Thinking Humanly
3. Thinking Rationally
4. Acting Rationally

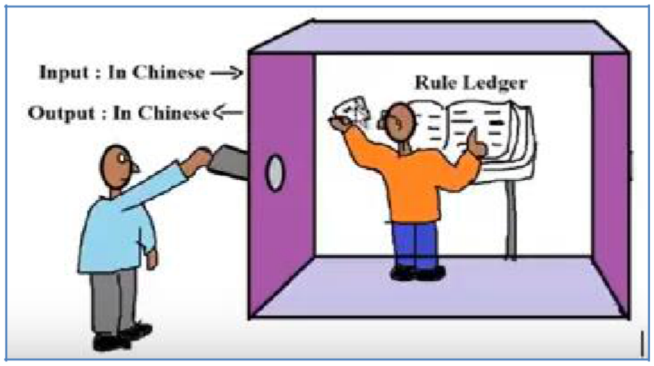


Thinking Humanly	**Thinking Rationally**
"The exciting new effort to make computers think . . . *machines with minds*, in the full and literal sense." (Haugeland, 1985) "[The automation of] activities that we associate with human thinking, activities such as decision-making, problem solving, learning . . ." (Bellman, 1978)	"The study of mental faculties through the use of computational models." (Charniak and McDermott, 1985) "The study of the computations that make it possible to perceive, reason, and act." (Winston, 1992)
Acting Humanly	**Acting Rationally**
"The art of creating machines that perform functions that require intelligence when performed by people." (Kurzweil, 1990) "The study of how to make computers do things at which, at the moment, people are better." (Rich and Knight, 1991)	"Computational Intelligence is the study of the design of intelligent agents." (Poole *et al.*, 1998) "AI . . . is concerned with intelligent behavior in artifacts." (Nilsson, 1998)

Some definitions of artificial intelligence, organized into four categories.

<u>Acting humanly: The Turing Test approach</u>





TURING TEST by ALAN TURING ***CHINESE ROOM ARGUMENT (TEST) by John Searle***

Mr. Alan Turing proposed "**Turing Test**" in the year 1950. He proposed that Turing Test can be used to determine "whether or not a machine is considered as intelligent?"

Imagine a game of three players having two humans and one computer, an interrogator (as human) is isolated from other two players. The interrogator job is to try and figure out which one is human and which one is computer by asking questions from both of them. To make the things harder computer is trying to make the interrogator guess wrongly. In other words, computer would try to indistinguishable from human as much as possible. If interrogator wouldn't be able to distinguish the answers provided by both human and computer, then the computer passes the test and machine (computer) is considered as intelligent as human.

Capabilities to pass Alan Turing test:

- 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.

But the test, which has given to the system, was having some human involvement (as operator or interrogator). Whereas, the Total Turing Test states that "System can pass the total Turing test if there is no operator i.e. avoidance of complete human intelligence from the system to answer the questions of interrogator.

To pass the total Turing Test, the computer will need:

- computer vision to perceive objects, and
- robotics to manipulate objects and move about.

In the year 1980, Mr. John Searle proposed the "**Chinese Room Argument (test)**". He argued that Turing Test could not be used to determine "whether or not a machine is considered as intelligent?". He argued that any machine could easily pass Turing Test simply by manipulating symbols of which they had no understanding. Without understanding, they could not be described as "thinking" in the same sense people do.

Compare the John with machine in Turing Test; the machine may have huge collection of database containing questions and answers. When a interrogator ask the question, the machine is simply locating the question in the database and returning the corresponding answer to the interrogator. The whole scenario would seems like that human is returning the answer unlike machine.

Hence the machine in configuration has no understanding of those questions and answers, without "understanding" (or "intentionality"), we cannot describe what the machine is doing as "thinking" and, since it does not think, it does not have a "mind" in anything like the normal sense of the word. Therefore, we can't consider machine as intelligent.

Thinking humanly: The cognitive modeling approach

If we are going to say that a given program thinks like a human, we must have some way of determining how humans think. We 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. For example, Allen Newell and Herbert Simon, who developed GPS, the "General Problem Solver" (Newell and Simon, 1961), were not content merely to have their program solve problems correctly. They were more concerned with comparing the trace of its reasoning steps to traces of human subjects solving the same problems. The interdisciplinary field of **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

The Greek philosopher Aristotle was one of the first to attempt to codify “right thinking,” that is, irrefutable reasoning processes. His syllogisms provided patterns for argument structures that always yielded correct conclusions when given correct premises—for example, “Socrates is a man; all men are mortal; therefore, Socrates is mortal.” These laws of thought were supposed to govern the operation of the mind; their study initiated the field called l**ogic**. Remember, logic is the prerequisite to study AI. There are two main obstacles to this approach.

1. It is not easy to take informal knowledge and state it in the formal terms required by logical notation, particularly when the knowledge is less than 100% certain.
2. There is a big difference between solving a problem “in principle” and solving it in practice.

Even problems with just a few hundred facts can exhaust the computational resources of any computer unless it has some guidance as to which reasoning steps to try first. Although both of these obstacles apply to any attempt to build computational reasoning systems, they appeared first in the logicist tradition.

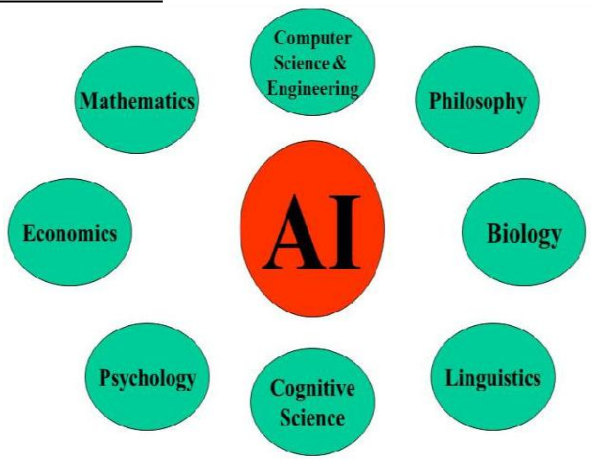
Acting rationally: The rational agent approach

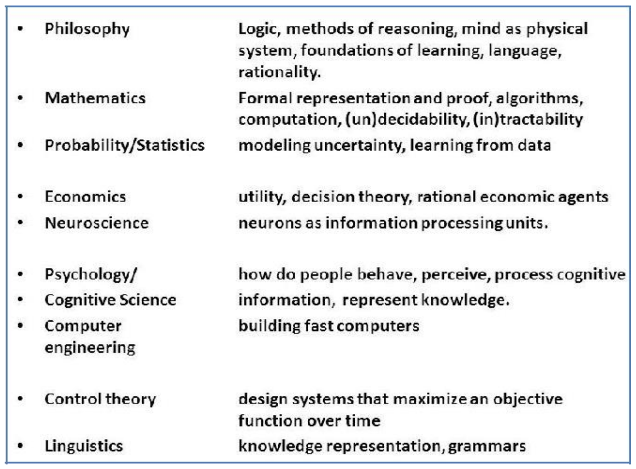
An agent is just something that acts (agent comes from the Latin agere, to do). Expectations from agents of AI: all computer programs do something, but computer agents are expected to do more: operate autonomously, perceive their environment, persist over a prolonged time period, adapt to change, and create and pursue goals. A **rational agent** is one that acts so as to achieve the best outcome or, when there is uncertainty, the best expected outcome. The rational-agent approach has two advantages over the other approaches.

- It is more general than the “laws of thought” approach because correct inference is just one of several possible mechanisms for achieving rationality.
- It is more amenable to scientific development than are approaches based on human behavior or human thought.

One important point to keep in mind that achieving perfect rationality — always doing the right thing — is not feasible in complicated environments. The computational demands are just too high. In our course curriculum of AI, we will adopt the working hypothesis that perfect rationality is a good starting point for analysis. It simplifies the problem and provides the appropriate setting for most of the foundational material in the field.

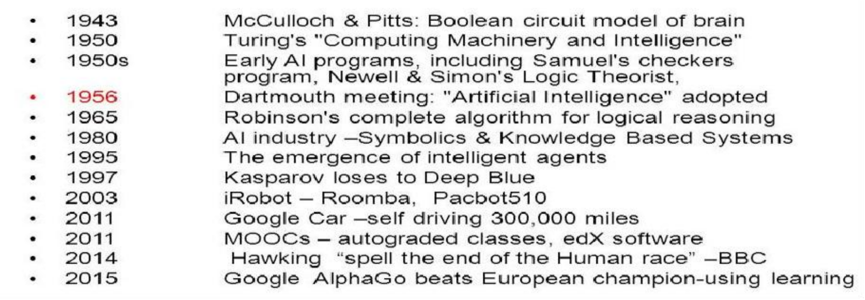
Foundation of AI



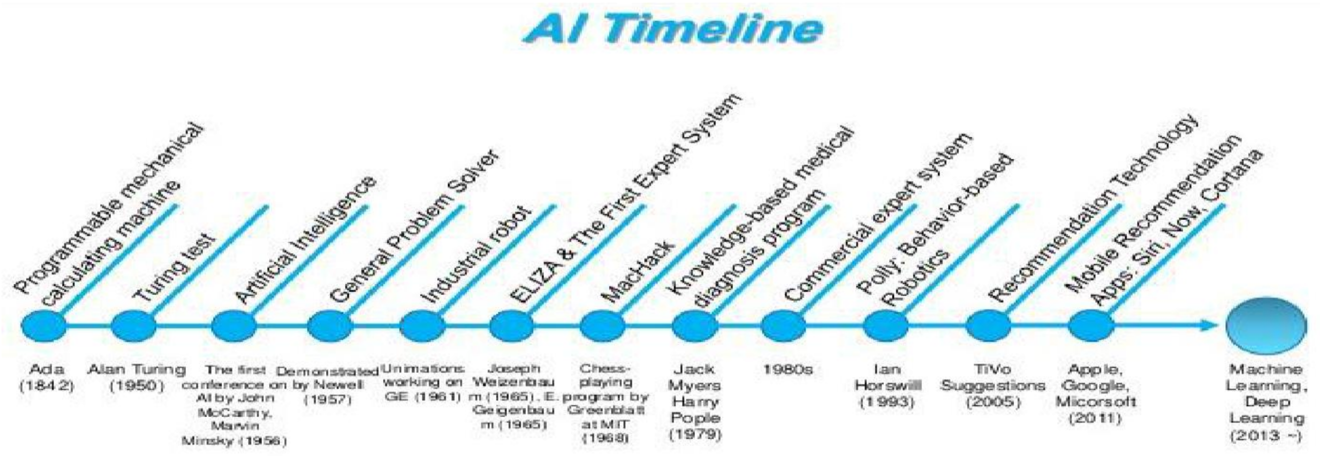


• Philosophy	Logic, methods of reasoning, mind as physical system, foundations of learning, language, rationality.
• Mathematics	Formal representation and proof, algorithms, computation, (un)decidability, (in)tractability
• Probability/Statistics	modeling uncertainty, learning from data
• Economics	utility, decision theory, rational economic agents
• Neuroscience	neurons as information processing units.
• Psychology/	how do people behave, perceive, process cognitive
• Cognitive Science	information, represent knowledge.
• Computer engineering	building fast computers
• Control theory	design systems that maximize an objective function over time
• Linguistics	knowledge representation, grammars

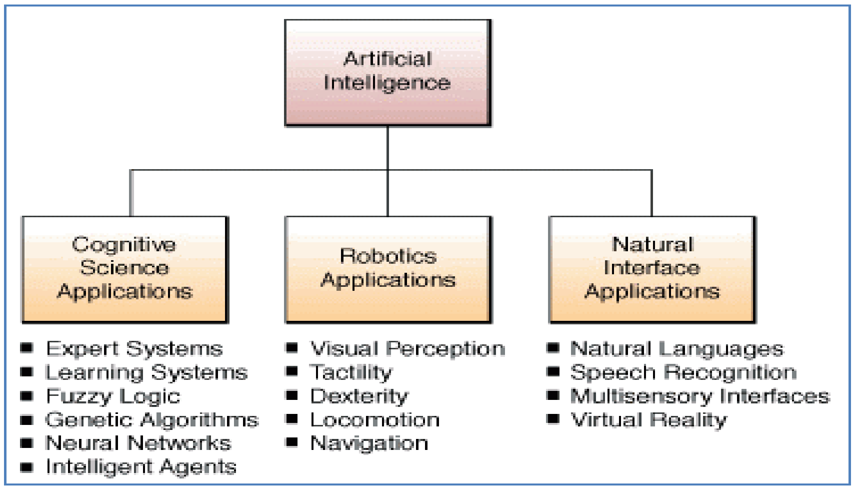
History of AI



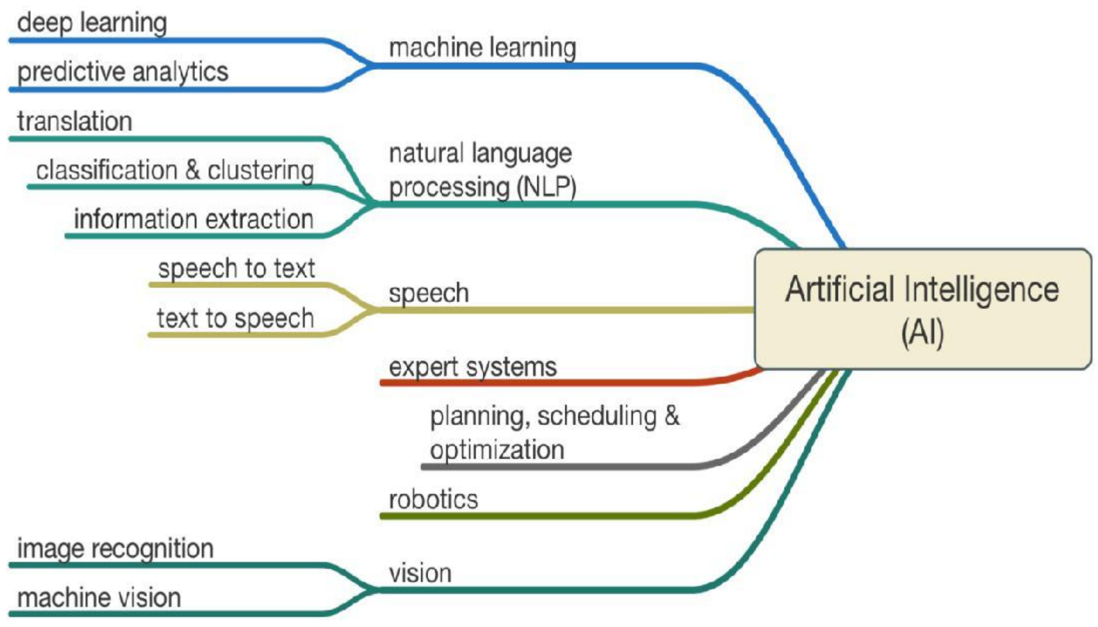
1943	McCulloch & Pitts: Boolean circuit model of brain
1950	Turing's "Computing Machinery and Intelligence"
1950s	Early AI programs, including Samuel's checkers program, Newell & Simon's Logic Theorist,
1956	Dartmouth meeting: "Artificial Intelligence" adopted
1965	Robinson's complete algorithm for logical reasoning
1980	AI industry –Symbolics & Knowledge Based Systems
1995	The emergence of intelligent agents
1997	Kasparov loses to Deep Blue
2003	iRobot – Roomba, Pacbot510
2011	Google Car –self driving 300,000 miles
2011	MOOCs – autograded classes, edX software
2014	Hawking "spell the end of the Human race" –BBC
2015	Google AlphaGo beats European champion-using learning


AI Timeline
Programmable mechanical calculating machine
Turing test
Artificial Intelligence
General Problem Solver
Industrial robot
ELIZA & The First Expert System
MacHack
Knowledge-based medical diagnosis program
Commercial expert system
Polly: Behavior-based Robotics
Recommendation Technology
Mobile Recommendation Apps: Siri, Now, Cortana
Ada (1842)
Alan Turing (1950)
The first conference on AI by John McCarthy, Marvin Minsky (1956)
Demonstrated by Newell (1957)
Unimations working on GE (1961)
Joseph Weizenbaum (1965), E. Geigenbaum (1965)
Chess-playing program by Greenblatt at MIT (1968)
Jack Myers Harry Pople (1979)
1980s
Ian Horswill (1993)
TiVo Suggestions (2005)
Apple, Google, Micorsoft (2011)
Machine Learning, Deep Learning (2013 ~)

Applications of AI



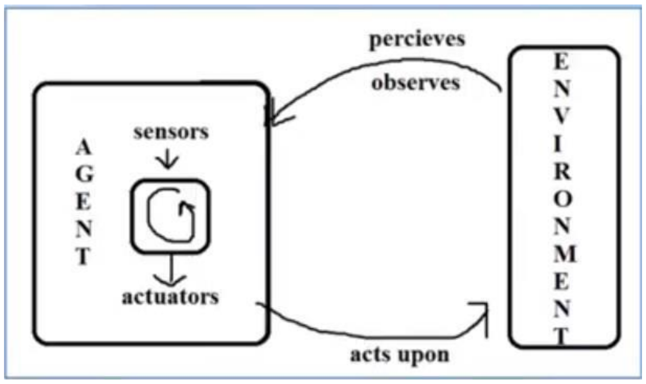
Application areas of AI



Intelligent Agents

AI is the branch of computer science, which deals with building intelligent agents; which can be able to think intelligently like humans, can be able to solve problems, and evolve by themselves.

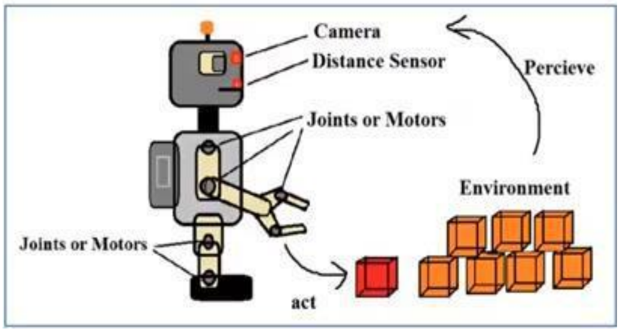
Intelligent agent is a program or software that perceives (or observes or senses) its environment through sensors, thinks intelligently and acts upon that environment through its actuators (to perform actions on the environment or produce output).



Note: Any agent carries 3 key tasks continuously in an infinite loop (or cycle), they are sense, think and act. Any agent can also call as "sense-think-act" cycle or "perception-action" cycle.

Example: Robotics Agent

- **Sensors**: Cameras (eyes), microphone (ears), touch screen (skin), balance sensors (for self-balance), distance sensors (check distance between two objects) etc.
- **Actuators (effectors)**: Motors, speakers (mouth) etc.



As we know, Robots are made of different components like cameras, microphone, touch screen etc. Here, the responsibility of this robot is to pickup this shaded (red) cube and place it on the top of stack of orange cubes. The stack of orange cubes or red cube is going to be an environment for this robot and the software installed in this robot considered as robotic agent. This robotic agent would perceive the environment through its sensors like camera, distance sensors etc. and performs the action on the environment through its actuators like motors (in this case, motors are use to rotate its joints). The robotic agent is going to receive the input from the environment through its sensors and thinks intelligently so that in the minimum number of steps efficiently the robot can be able to pick up this red (shaded) cube and place it on the top of stack of orange cube. So, robotic agent guides its legs and arms to do such task efficiently. If compare this robotic agent with human agent or human, cameras are like eyes, microphone are ears, touch screen would be skin and so on.

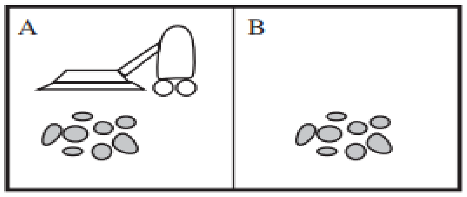
Game AI is the branch of artificial intelligence; which deals with building intelligent game agents i.e. game characters especially NPCs (Non-Player Characters) which act like opponents or co-players and stimulate human like intelligence.

For example, in the year 1997, the world chess champion Mr. Garry Kasparov was defeated by the first chess playing system known as **deep blue** (built by IBM)

<u>Agents and its environment</u>

1. Percept
2. Percept sequence
3. Agent function
4. Agent table
5. Agent program

To illustrate these ideas, we use a very simple example—the vacuum-cleaner world



A vacuum-cleaner world with just two locations.

- 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.

Percept:

The term “percept” is used to refer agent’s perceptual inputs at any given instant. So percepts for vacuum-cleaner world:

- Location: Square A/ Square B
- Status: Dirty/ Clean

Percept sequence:

An Agent’s percept sequence is the complete history of everything the agent has even perceived. Percept sequence for vacuum-cleaner world:

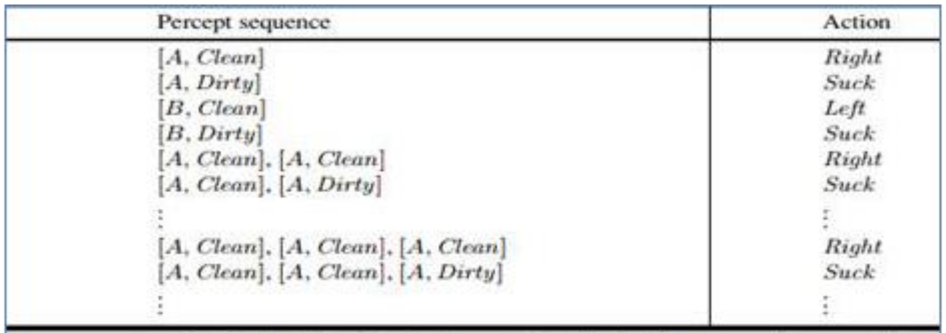
- {A, Dirty}
- {A, Clean}
- {B, Dirty}
- {B, Clean} ….

An agent’s behaviour is described by the **agent function** that maps any given percept sequence to an action.

Percept	Action
{A, Dirty}	Suck
{A, Clean}	Right
{B, Dirty}	Suck
{B, Clean}	Left

Agent table (percept table): We can imagine tabulating the agent function that describes any given agent. For most agents, this could be very large table – infinite, in fact, unless we place a bound on the length of percept sequences we want to consider: Given an agent to experiment with, we can, in principle, construct the table by trying out all possible percept sequences and recording which actions the agent does in response.

Partial percept table:



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*
⋮	⋮

Partial tabulation of a simple agent function for the vacuum-cleaner world

Agent program

The **percept table** is an external characterization of an agent. Internally the **agent function** for an artificial agent will be implemented by an **agent program**.

The obvious question, then, is this: What is the right way to fill out the table? In other words, what makes an agent good or bad, intelligent or stupid?

We should emphasize that the notion of an agent is meant to be a tool for analyzing systems, not an absolute characterization that divides the world into agents and non-agents.

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. Obviously, doing the right thing is better than doing the wrong thing, but what does it mean to do the right thing?

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 expected to maximize its performance measure, given the evidence provided by the percept sequence and whatever built-in knowledge the agent has.

Consider the simple vacuum-cleaner agent that cleans a square if it is dirty and moves to the other square if not; this is the agent function tabulated the figure. Is this a rational agent? That depends! First, we need to say what the performance measure is, what is known about the environment, and what sensors and actuators the agent has.

Let us assume the following:

- 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 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.

We claim that under these circumstances the agent is indeed rational; its expected performance is at least as high as any other agent's.

One can see easily that the same agent would be irrational under different circumstances. For example, once all the dirt is cleaned up, the agent will oscillate needlessly back and forth; if the performance measure includes a penalty of one point for each movement left or right, the agent will fare poorly. A better agent for this case would do nothing once it is sure that all the squares are clean. If clean squares can become dirty again, the agent should occasionally check and re-clean them if needed. If the geography of the environment is unknown, the agent will need to explore it rather than stick to squares A and B.

Omniscience, learning, and autonomy

We need to be careful to distinguish between rationality and omniscience. An omniscient agent knows the actual outcome of its actions and can act accordingly; but omniscience is impossible in reality. Consider the following example: I am walking along the Champs 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 airliner and 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. Retreating from a requirement of perfection is not just a question of being fair to agents. The point is that if we expect an agent to do what turns out to be the best action after the fact, it will be impossible to design an agent to fulfill this specification—unless we improve the performance of crystal balls or time machines.

Our definition requires a rational agent not only to gather information but also to **learn** as much as possible from what it perceives. The agent's initial configuration could reflect some prior knowledge of the environment, but as the agent gains experience this may be modified and augmented. There are extreme cases in which the environment is completely known a priori. In such cases, the agent need not perceive or learn; it simply acts correctly.

To the extent that an agent relies on the prior knowledge of its designer rather than AUTONOMY on its own percepts, we say that the agent lacks autonomy. 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.

After sufficient experience of its environment, the behavior of a rational agent can become effectively independent of its prior knowledge. Hence, the incorporation of learning allows one to design a single rational agent that will succeed in a vast variety of environments.

<u>Nature of environments</u>

Task environments are essentially the "problems" to which rational agents are the "solutions". Task environments come in a variety of flavors. The flavor of the task environment directly affects the appropriate design for the agent program.

<u>Properties of TASK environment:</u>

1. Fully observable vs. partially observable
2. Single agent vs. multi-agent
3. Deterministic vs. stochastic
4. Episodic vs. sequential
5. Static vs. dynamic
6. Discrete vs. continuous
7. 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. Example: Automated taxi driver.

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. Example: Automated taxi driver - night driving. If the agent has no sensors at all then the environment is unobservable.

<u>Single agent vs. multi-agent</u>

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. There are, however, some subtle issues. First, we have described how an entity may be viewed as an agent, but we have not explained which entities must be viewed as agents. Does an agent A (the taxi driver for example) have to treat an object B (another vehicle) as an agent, or can it be treated merely as an object behaving according to the laws of physics, analogous to waves at the beach or leaves blowing in the wind? The key distinction is whether B's behavior is best described as maximizing a performance measure whose value depends on agent A's behavior. For example, in chess, the opponent entity B is trying to maximize its performance measure, which, by the rules of chess, minimizes agent A's performance measure. Thus, chess is a competitive multi-agent environment. In the taxi-driving environment, on the other hand, avoiding collisions maximizes the performance measure of all agents, so it is a partially **cooperative multi-agent** environment. It is also partially competitive because, for example, only one car can occupy a parking space. The agent-design problems in multi-agent environments are often quite different from those in single-agent environments; for example, communication often emerges as a rational behavior in multi-agent environments; in some competitive environments, randomized behavior is rational because it avoids the pitfalls of predictability.

<u>Deterministic vs. stochastic</u>

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. Taxi driving is clearly stochastic in this sense, because one can never predict the behavior of traffic exactly; moreover, one's tires blow out and one's engine seizes up without warning. The vacuum world as we described it is deterministic, but variations can include stochastic elements such as randomly appearing dirt and an unreliable suction mechanism. We say an environment is uncertain if it is not fully observable or not deterministic.

<u>Episodic vs. sequential</u>

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. Example: part-picking robot. 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. Episodic environments are much simpler than sequential environments because the agent does not need to think ahead.

<u>Static vs. dynamic</u>

If the environment can change while an agent is deliberating, then we say the environment is dynamic for that agent; otherwise, it is static. 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.

<u>Discrete vs. continuous</u>

The discrete/continuous distinction applies to the state of the environment, to the way time is handled, and to the percepts 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.).

<u>Known vs. unknown</u>

In a known environment, the outcomes for all actions are given. Obviously, if the environment is unknown, the agent will have to learn how it works in order to make good decisions. Note that the distinction between known and unknown environments is not the same as the one between fully and partially observable environments. It is quite possible for a known environment to be partially observable—for example, in solitaire card games, I know the rules but am still unable to see the cards that have not yet been turned over. Conversely, an unknown environment can be fully observable—in a new video game, the screen may show the entire game state but I still don't know what the buttons do until I try them.

Examples of task environments and their characteristics are as:

Task Environment	Observable	Agents	Deterministic	Episodic	Static	Discrete
Crossword puzzle	Fully	Single	Deterministic	Sequential	Static	Discrete
Chess with a clock	Fully	Multi	Deterministic	Sequential	Semi	Discrete
Poker	Partially	Multi	Stochastic	Sequential	Static	Discrete
Backgammon	Fully	Multi	Stochastic	Sequential	Static	Discrete
Taxi driving	Partially	Multi	Stochastic	Sequential	Dynamic	Continuous
Medical diagnosis	Partially	Single	Stochastic	Sequential	Dynamic	Continuous
Image analysis	Fully	Single	Deterministic	Episodic	Semi	Continuous
Part-picking robot	Partially	Single	Stochastic	Episodic	Dynamic	Continuous
Refinery controller	Partially	Single	Stochastic	Sequential	Dynamic	Continuous
Interactive English tutor	Partially	Multi	Stochastic	Sequential	Dynamic	Discrete

Examples of task environments and their characteristics.

Specifying the task environment

For the acronymically minded, we call this the PEAS (Performance, Environment, Actuators, Sensors) description. In designing an agent, the first step must always be to specify the task environment as fully as possible.

- Performance – which qualities it should have?
- Environment – where it should act?
- Actuators – how will it perform actions?
- Sensors – how will it perceive environment?

Let us consider a more complex problem: an automated taxi driver. The PEAS description for the taxi's task environment is as:

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

PEAS description of the task environment for an automated taxi.

Examples:

Agent Type	Performance Measure	Environment	Actuators	Sensors
Medical diagnosis system	Healthy patient, reduced costs	Patient, hospital, staff	Display of questions, tests, diagnoses, treatments, referrals	Keyboard entry of symptoms, findings, patient's answers
Satellite image analysis system	Correct image categorization	Downlink from orbiting satellite	Display of scene categorization	Color pixel arrays
Part-picking robot	Percentage of parts in correct bins	Conveyor belt with parts; bins	Jointed arm and hand	Camera, joint angle sensors
Refinery controller	Purity, yield, safety	Refinery, operators	Valves, pumps, heaters, displays	Temperature, pressure, chemical sensors
Interactive English tutor	Student's score on test	Set of students, testing agency	Display of exercises, suggestions, corrections	Keyboard entry

Examples of agent types and their PEAS descriptions.

Structure of Agent

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

Agent = architecture + program

Agent program take the current percept as input from the sensors and return an action to the actuators.

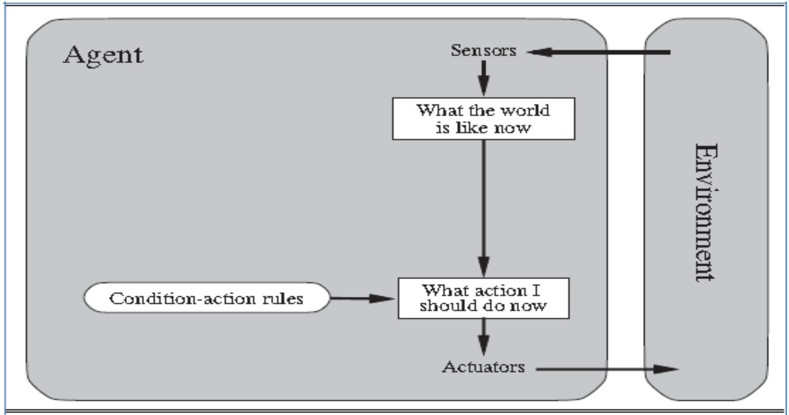
Intelligent Agent Types & its functionality

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.
- Learning 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, vacuum cleaner agent.



Schematic diagram of a simple reflex agent.

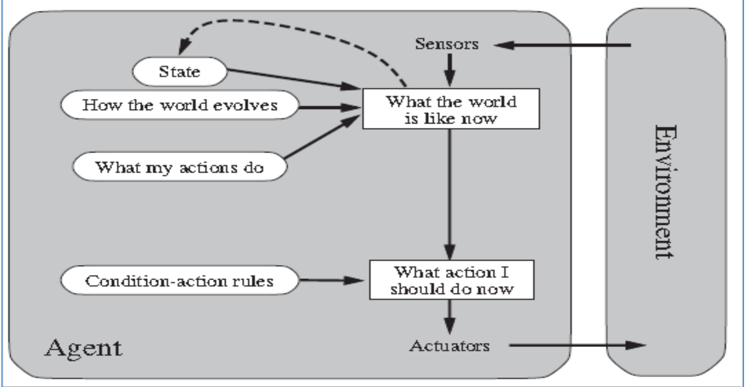
Simple reflex behaviors occur even in more complex environments. Such a connection is a **condition–action rule.** Suppose an **automated taxi driver** agent is driving on road. The car in front of the taxi applying the brake and its break lights are turned on and speed becomes slow. In this situation agent takes the percepts from camera and other sensors and able to identify that current state of front car is applying the brake. So it will take the decision to initiate braking system to avoid collision.

$$\left.\begin{array}{l}\text{Q. What is the world like now?}\\ \text{A. It's applying brakes.}\\ \text{Q. What action should I take?}\\ \text{A. Initiate braking.}\end{array}\right\}\quad \text{Condition-action}$$

<u>Model-based reflex agents:</u>

Model-based reflex agents are used in situations of partial visibility. This agent don't know what will happen if it will take some action against the current percept. 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 no. 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

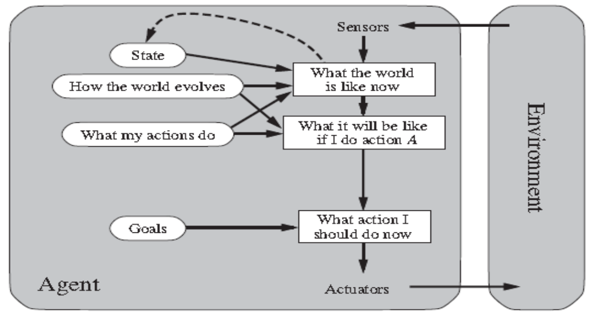


A model-based reflex agent.

So in nut shell we can say that model based reflex agent maintains the internal state to take decision what action should take upon the environment. These agents are extension of condition-action rule based reflexive agent.

<u>Goal-based agent:</u>

If you are having specified goals, it makes the problem simpler. Knowing something about the current state of the environment is not always enough to decide what to do. In this situation goal will help you to take the decision.
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 GOAL agent needs some sort of goal information that describes situations that are desirable.

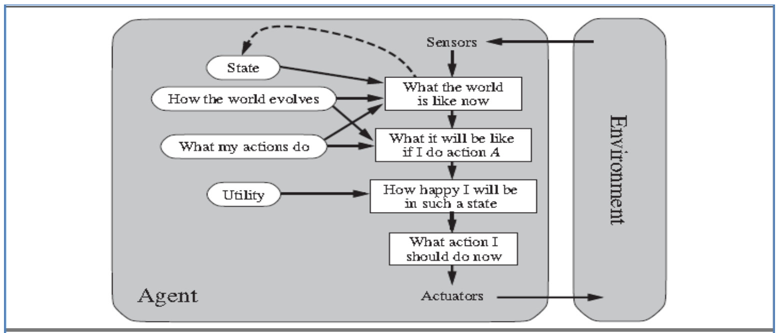

Sensors
State
How the world evolves
What the world is like now
What my actions do
What it will be like if I do action A
Goals
What action I should do now
Agent
Actuators
Environment

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 generate high-quality behaviour in most environments. 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. 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. Because "happy" does not sound very scientific, economists and computer scientists use the term **utility** instead.

An 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.



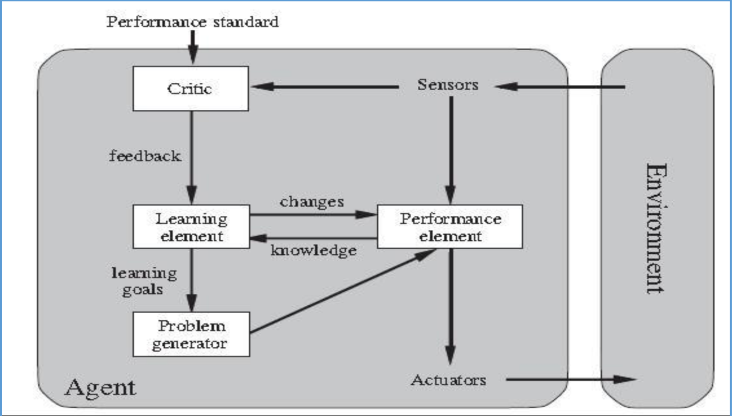
Sensors
State
What the world is like now
How the world evolves
What it will be like if I do action A
What my actions do
Utility
How happy I will be in such a state
What action I should do now
Agent
Actuators
Environment

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.

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.

Learning agent:



A general learning agent.

A learning agent can be divided into four conceptual components:

- Learning element: responsible for making improvements
- Performance element: responsible for selecting external actions
- problem generator: responsible for suggesting actions that will lead to new and informative experiences
- 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 design of the learning element depends very much on the design of the performance element.

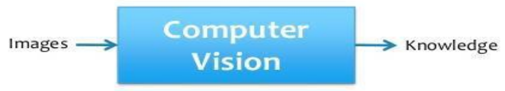
Performance measure vs. utility function

- A performance measure (typically imposed by the designer) is used to evaluate the behaviour of the agent in environment. It tells "does agent do what it's supposed to do in the environment".
- A utility function is used by an agent itself to evaluate how desirable states are. Some paths to the goal are better (more efficient) than others –which path is the best
- Does agent do what it's supposed to do vs. does agent do it in optimal way
- The utility function may not be the same as the performance measure
- An agent may have no explicit utility function at all, whereas there is always a performance measure

Computer Vision

- Computer vision is a field of computer science that works on enabling computers to see, identify and process images in the same way that human vision does, and then provide appropriate output. Computer vision's goal is not only to see, but also process and provide useful results based on the observation
- Computer vision is concerned with the theory and technology for building artificial systems that obtain information from images or multi-dimensional data.

Computer vision is a field that includes methods for acquiring, processing, analysing, 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.



Applications are: face recognition, object recognition, location recognition and tracking, Forensics, Virtual Reality, Robotics, Navigation & Security and so on.

Computer Vision Hierarchy: Computer vision is divided into three basic categories that are as following:

- Low-level vision: includes process image for feature extraction (like edge, corners, optical flow etc)
- Intermediate-level vision: includes object recognition, motion analysis and 3D reconstruction using features obtained from the low-level vision.
- High-level vision: includes interpretation of the evolving information provided by middle-level vision as well as directing what middle and low-level vision tasks should be performed. Interpretation may include conceptual description of a scene like activity, intention and behavior.

Natural Language Processing (NLP)

- Natural Language Processing (NLP) refers to AI method of communicating with an intelligent system using a natural language such as English.
- Processing of Natural Language is required when you want an intelligent system like robot to perform as per your instructions, when you want to hear decision from a dialogue based clinical expert system, etc.

Definition of NLP: Natural language processing (NLP) is a branch of artificial intelligence that helps computers understand, interpret and manipulate human language. NLP draws from many disciplines, including computer science and computational linguistics, in its pursuit to fill the gap between human communication and computer understanding.

The field of NLP involves making computers to perform useful tasks with the natural languages humans use. The input and output of an NLP system can be –

- Speech
- Written Text

Components of NLP: There are two components of NLP as given –

1. Natural Language Understanding (NLU)

Natural languages can be ambiguous (i.e. different meaning of same sentence), 3 different types of ambiguity:

- ✓ Lexical ambiguity: related to noun/ verb/ adjectives and so on.
- ✓ Syntactical ambiguity: related to grammar of sentence.
- ✓ Referential ambiguity: related to the meaning which may not well refer from the sentence.

Understanding involves the following tasks –

- Mapping the given input in natural language into useful representations.
- Analyzing different aspects of the language.

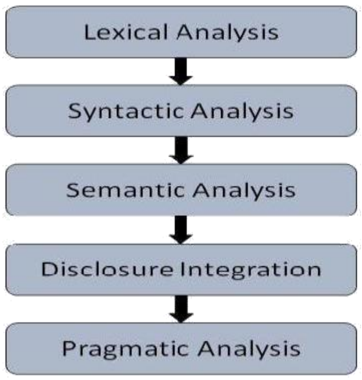
2. Natural Language Generation (NLG)

It is the process of producing meaningful phrases and sentences in the form of natural language from some internal representation. It involves –

- **Text planning** − It includes retrieving the relevant content from knowledge base.
- **Sentence planning** − It includes choosing required words, forming meaningful phrases, setting tone of the sentence.
- **Text Realization** − It is mapping sentence plan into sentence structure.

Note: The NLU is harder than NLG.

FLOWCHART for NLP:



**********************************END OF UNIT-1 ***************************************