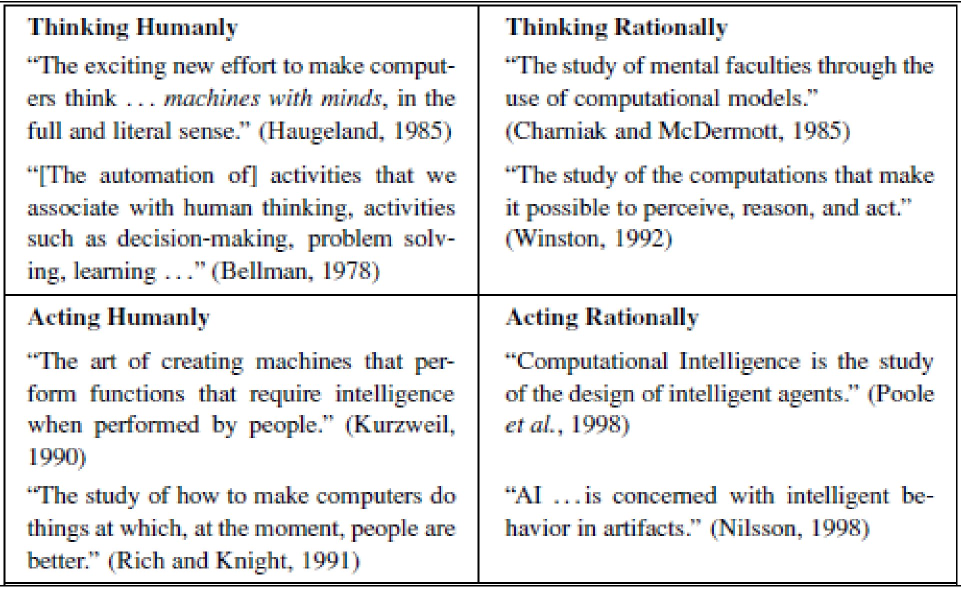
	INTRODUCTION TO AI, INTELLIGENT AGENTS
	CONTENTS
1	**What is AI?**
2	**The Foundations of Artificial Intelligence**
3	**Agents and Environments**
4	**Good Behaviour:The concept of Rationality**
5	**The Nature of Environments**
6	**The Structure of Agents**
7	**Empiricist Approaches in Artificial Intelligence**
8	**Applications of Artificial Intellience**
9	**Limitations of Artificial Intelligence**

1 **What is AI?**

Artificial intelligence is defined as methods and techniques used for implementing human intelligence in computer.



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)

Definitions of Artificial Intelligence organized into 4 categories.

The definitions on top are concerned with *thought processes* and *reasoning*.
The ones on the bottom address *behavior*.
The definitions on the left measure success in terms of fidelity to *human* performance.
The ones on the right measure against an *ideal* performance measure, called **rationality**.
A system is rational if it does the "right thing," given what it knows.

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:

1) through introspection—trying to catch our own thoughts as they go by;
2) through psychological experiments—observing a person in action;
3) 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.

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

The **Turing Test**, proposed by Alan Turing TURING TEST (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:

1) **natural language processing** to enable it to communicate successfully in English;
2) **knowledge representation** to store what it knows or hears;
3) **automated reasoning** to use the stored information to answer questions and to draw new conclusions;
4) **machine learning** to adapt to new circumstances and to detect and extrapolate patterns.

The Total Turing Test includes a video signal so that the interrogator can test the subject's perceptual abilities, as well as the opportunity for the interrogator to pass physical objects "through the hatch."
To pass the total Turing Test, the computer will need

1) **computer vision** to perceive objects,
2) **robotics** to manipulate objects and move about.

<u>Thinking rationally: The "laws of thought" approach</u>

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 **logic**.
Logicians in the 19th century developed a precise notation for statements about all kinds of objects in the world and the relations among them.
By 1965, programs existed that could, in principle, solve *any* solvable problem described in

logical notation. The so-called **logicist** tradition within artificial intelligence hopes to build on such programs to create intelligent systems.
There are two main obstacles to this approach.

1) First, 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) Second, there is a big difference between solving a problem "in principle" and solving it in practice.

<u>Acting rationally: The rational agent approach</u>
An **agent** is just something that acts.
Computer agents are expected to:

- 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.
In the "laws of thought" approach to AI, the emphasis was on correct inferences. Making correct inferences is sometimes *part* of being a rational agent, because one way to act rationally is to reason logically to the conclusion that a given action will achieve one's goals and then to act on that conclusion. On the other hand, correct inference is not *all* of rationality; in some situations, there is no provably correct thing to do, but something must still be done. There are also ways of acting rationally that cannot be said to involve inference.
For example, recoiling from a hot stove is a reflex action that is usually more successful than a slower action taken after careful deliberation.
All the skills needed for the Turing Test also allow an agent to act rationally.

1) Knowledge representation and reasoning enable agents to reach good decisions. We need to be able to generate comprehensible sentences in natural language to get by in a complex society.
2) We need learning not only for erudition, but also because it improves our ability to generate effective behavior.

The rational-agent approach has two advantages over the other approaches.

1) First, it is more general than the "laws of thought" approach because correct inference is just one of several possible mechanisms for achieving rationality.
2) Second, it is more amenable to scientific development than are approaches based on human behavior or human thought. The standard of rationality is mathematically well defined and completely general, and can be "unpacked" to generate agent designs that provably achieve it. Human behavior, on the other hand, is well adapted for one specific environment and is defined by, well, the sum total of all the things that humans do.

2 **The Foundations of Artificial Inteligence:**

Philosophy

- Can formal rules be used to draw valid conclusions?
- How does the mind arise from a physical brain?
- Where does knowledge come from?
- How does knowledge lead to action?

Aristotle was the first person to formulate a precise set of laws governing the mind's rational part. He developed an informal system of ***syllogisms*** for proper reasoning, which allowed one to generate conclusions with the given initial premises.

[*Syllogism:* It is a Logical Argument consisting of two premises and a conclusion**]**

- At that time, the study of human intelligence began with no formal expression.
- They Initiated the idea of the mind as a machine and its internal operations.

Mathematics

- What are the formal rules to draw valid conclusions?
- What can be computed?
- How do we reason with uncertain information?

Mathematics formalizes the three main areas of AI: *computation, logic*, and *probability.* Boole introduced formal language for making logical inference and used *formal logic* methods such as Boolean Logic & Fuzzy Logic. At the same time, we consider these bases for most modern approaches that handle *uncertainty* in AI Applications. Besides logic and computation, another contribution of mathematics to AI is the theory of probability used to deal with uncertain measurements and incomplete theories.

Economics

- How should we make decisions so as to maximize payoff?
- How should we do this when others may not go along?
- How should we do this when the payoff may be far in the future?

Most people think that economics is being money, but it will say that they study how people make choices that lead to preferred outcomes. So, the mathematical treatment of the preferred outcome, *"Utility"***,** was first formalized. Here they considered a "Decision Theory" that combines the Probability Theory & Utility Theory for making decisions.

Neuroscience

- How do brains process information?

It is the study of the Nervous System, particularly the brain. More recent studies used accurate sensors to correlate brain activity to human thought. This is done by monitoring individual neurons. As we know, there are 1000 times more neurons in a typical human brain than the gates in the CPU of a typical high-end computer. Moore's Law predicts that the CPU gate count will be equal to brains neuron count around 2020.

Psychology

- How do humans and animals think and act?

The origins of scientific psychology are usually traced to the German physicist Hermann von Helmholtz's work and his student Wilhelm Wundt. He applied some scientific methods to the study of human vision. He also used his view on humans' behaviourism movement and the brain's view as an Information-Processing-Device, Which is a principle characteristic of cognitive psychology.

Computer engineering

- How can we build an efficient computer?

We need two things: intelligence and an artifact for artificial intelligence to succeed. AI also requires the software side of Computer Science, which has supplied the operating systems, Programming Languages, Tools, etc.…

Control theory and cybernetics

- How can artifacts operate under their own control?

The artifacts adjust their actions:

- *To do better for the environment over time*
- *Based on an objective function and feedback from the environment*

Machines can modify their behaviour in response to the environment **(*Sense/Action*)**. These machines are called self-controlling machines. The goal is to build systems that transition from initial state to goal state with minimum energy. Examples of self-regulating feedback control systems include the steam engine, created by James Watt and the thermostat, invented by Colnelis Drebbel, who also invented the submarine.

Linguistics

· How does language relate to thought?

Modem linguistics and AI, then, were "born" at about the same time and grew up together, intersecting in a hybrid field called computational linguistics or natural language processing. The speech demonstrates so much of human intelligence. Children can create sentences they have never heard before. So language and thought are believed to be tightly intertwined.

3 **Agents and Environments**

An **agent** is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators.

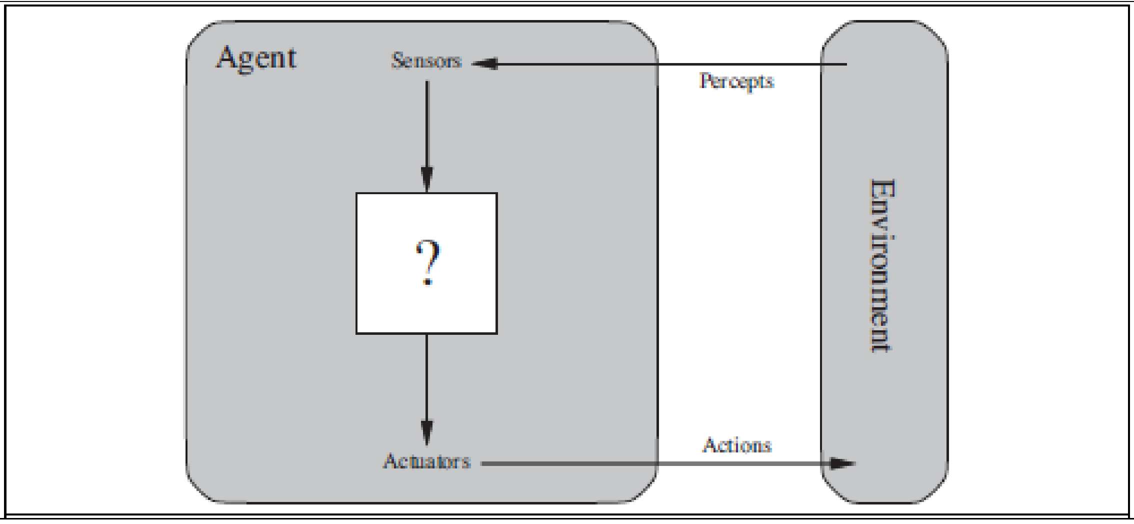
Example:

1) A human agent has eyes, ears, and other organs for sensors and hands, legs, vocal tract, and so on for actuators.
2) A robotic agent might have cameras and infrared range finders for sensors and various motors for actuators.
3) 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.

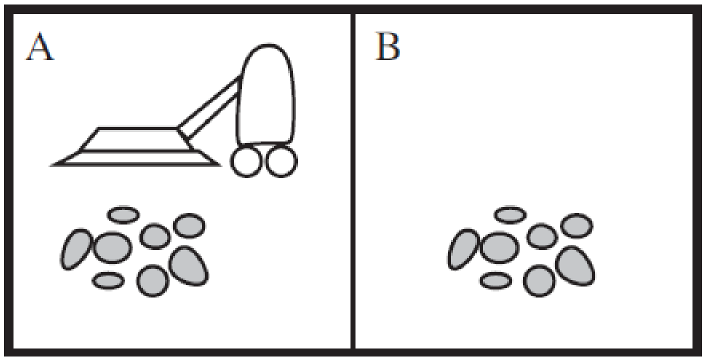
Percept refers to the agent's perceptual inputs at any given instant.

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.*



Mathematically speaking, we say that an agent's behavior is described by the **agent function** that maps any given percept sequence to an action.
Given an agent to experiment with, we can construct a table by trying out all possible percept sequences and recording which actions the agent does in response.
The table is of course an *external* characterization of the agent.
Internally, the agent function for an artificial agent will be implemented by an **agent program**.
The agent function is an abstract mathematical description;
the agent program is a concrete implementation, running within some physical system.



Vaccum cleaner world with 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*
⋮	⋮

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

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.

4 **<u>Good Behaviour:The concept of Rationality</u>**

A **rational agent** is one that does the right thing.

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. This notion of desirability is captured by a **performance measure** that evaluates any given sequence of environment states.

Consider, for example, the vacuum-cleaner agent.

- We might propose to measure performance by the amount of dirt cleaned up in a single eight-hour shift. With a rational agent, of course, what you ask for is what you get. A rational agent can maximize this performance measure by cleaning up the dirt, then dumping it all on the floor, then cleaning it up again, and so on.
- A more suitable performance measure would reward the agent for having a clean floor. For example, one point could be awarded for each clean square at each time step.

As a general rule, it is better to design performance measures according to what one actually wants in the environment, rather than according to how one thinks the agent should behave.

Rationality

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. 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.

Omniscience:
An omniscient agent knows the *actual* outcome of its actions and can act accordingly; but omniscience is impossible in reality. Our definition of rationality does not require omniscience, then, because the rational choice depends only on the percept sequence *to date*.Therefore a Rational agent is not Omniscient.

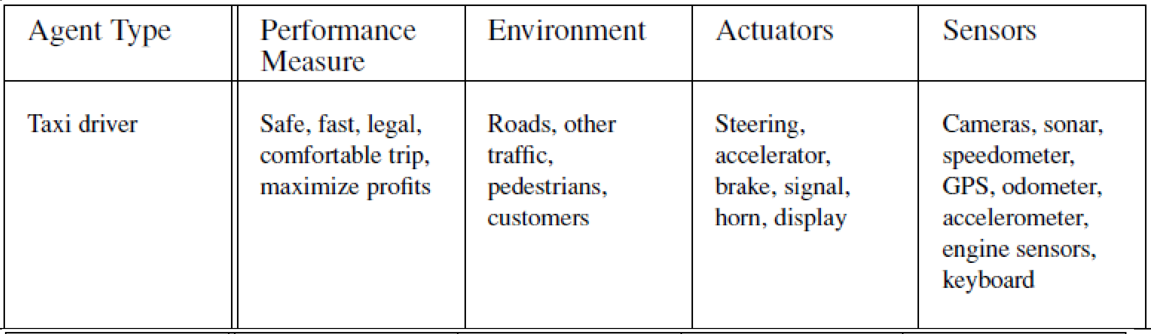
Learning:
Doing actions *in order to modify future percepts*—sometimes called **information gathering** is an important part of rationality.
A second example of information gathering is provided by the **exploration** that must be undertaken by a vacuum-cleaning agent in an initially unknown environment.
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.

Autonomy:
To the extent that an agent relies on the prior knowledge of its designer rather than 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.

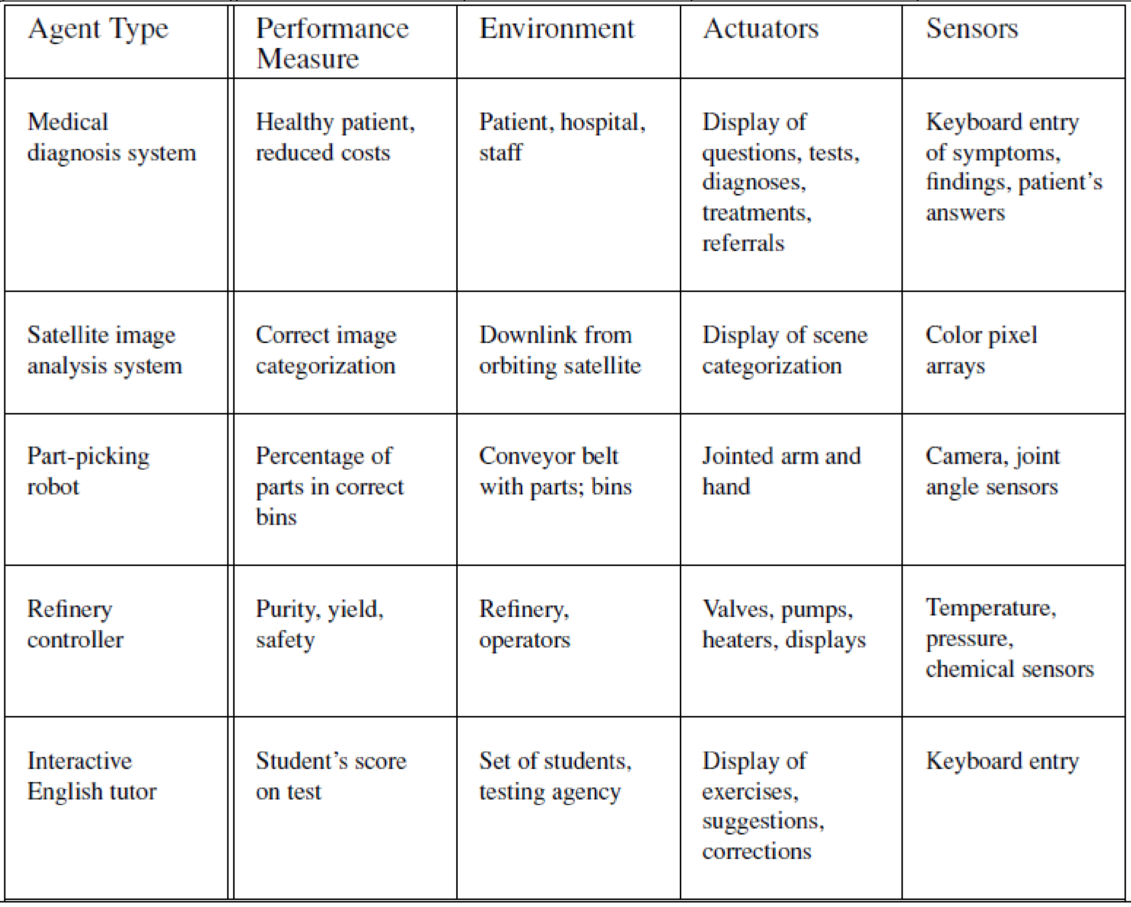
5 **<u>The Nature of Environments</u>**

Specifying the Task Environment

We group the performance measure, the environment, and the agent's actuators and sensors as the **task environment**. For the acronymically minded, we call this the **PEAS** (**P**erformance, **E**nvironment, **A**ctuators, **S**ensors) description. In designing an agent, the first step must always be to specify the task environment as fully as possible.



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



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.

Properties of task environments:

1) **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. Fully observable environments are convenient because the agent need not maintain any internal state to keep track of the world.
   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 drivers are thinking. If the agent has no sensors at all then the environment is **unobservable**.
2) **Single agent** vs. **multiagent**:
   The distinction between single-agent and multiagent may seem simple enough.
   For example, 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. The chess is a **competitive** multiagent 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** multiagent environment. It is also partially competitive because, for example, only one car can occupy a parking space.
3) **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.
   For example:
   A game can be deterministic even though each agent may be unable to predict the actions of
   the others. If the environment is partially observable, however, then it could *appear* to be stochastic. Most real situations are so complex that it is impossible to keep track of all the unobserved aspects; for practical purposes, they must be treated as stochastic.
   For Example:
   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.

4) **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.
Episodic environments are much simpler than sequential environments because the agent does not need to think ahead.

5) **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.
Dynamic environments, on the other hand, are continuously asking the agent what it wants to do; if it hasn't decided yet, that counts as deciding to do nothing. If the environment itself does not change with the passage of time but the agent's performance score does, then we say the environment is **semidynamic**.
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 semidynamic.
Crossword puzzles are static.

6) **Discrete** vs. **continuous**:
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.). Input from digital cameras is discrete, strictly speaking, but is typically treated as representing continuously varying intensities and locations.

7) **Known** vs. **unknown**:
This distinction refers not to the environment itself but to the agent's (or designer's) state of knowledge about the "laws of physics" of the environment.
In a known environment, the outcomes (or outcome probabilities if the environment is stochastic) 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.
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.

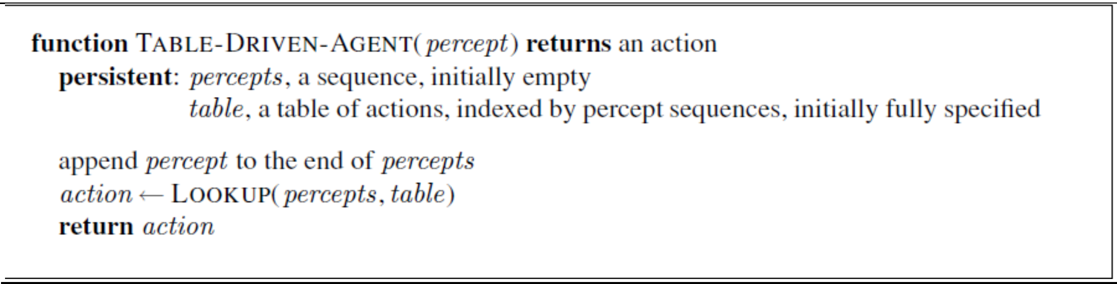
	For example, in solitaire card games, I know the rules but am still unable to see the cards that have not yet been turned over. Conversely, an *unknown* environment can be *fully* observable—in a new video game, the screen may show the entire game state but I still don't know what the buttons do until I try them.

6 **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 .
The architecture might be just an ordinary PC, or it might be a robotic car with several onboard
computers, cameras, and other sensors. The architecture makes the percepts from the sensors available to the program, runs the program, and feeds the program's action choices to the actuators as they are generated.

Agent 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 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 percepts.



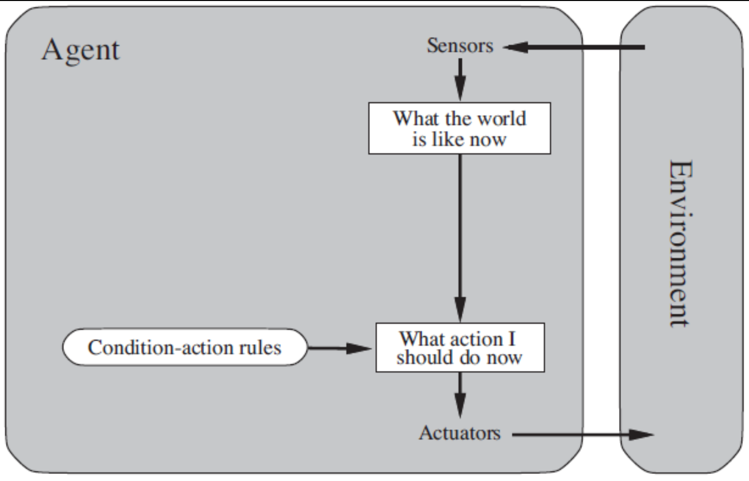
```
function TABLE-DRIVEN-AGENT(percept) returns an action
  persistent: percepts, a sequence, initially empty
              table, a table of actions, indexed by percept sequences, initially fully specified

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

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

The Table Driven Agent Program has the following disadvantages:
(a) no physical agent in this universe will have the space to store the large sized table,
(b) the designer would not have time to create the table,
(c) no agent could ever learn all the right table entries from its experience, and
(d) even if the environment is simple enough to yield a feasible table size, the designer still has no guidance about how to fill in the table entries.
Despite all this, 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 behavior from a smallish program rather than from a vast table.

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 whose agent function is tabulated is a simple reflex agent, because its decision is based only on the current location and on whether that location contains dirt.
Simple reflex behaviors occur even in more complex environments.
Example:
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*.



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
```

Simple reflex agent Program

The INTERPRET-INPUT function generates an abstracted description of the current state from the
Percept.
The RULE-MATCH function returns the first rule in the set of rules that matches the given state description.

Limitations of simple Reflex agents

Simple reflex agents have the admirable property of being simple, but they turn out to be of limited intelligence. The agent will work *only if the correct decision can be made on the basis of only the current percept—that is, only if the environment is fully observable.*

Even a little bit of unobservability can cause serious trouble.

For example, the braking rule given earlier assumes that the condition *car-in-front-is-braking* can be determined from the current percept—a single frame of video. This works if the car in front has a centrally mounted brake light. Unfortunately, older models have different configurations of taillights,brake lights, and turn-signal lights, and it is not always possible to tell from a single image whether the car is braking. A simple reflex agent driving behind such a car would either brake continuously and unnecessarily, or, worse, never brake at all.

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

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.

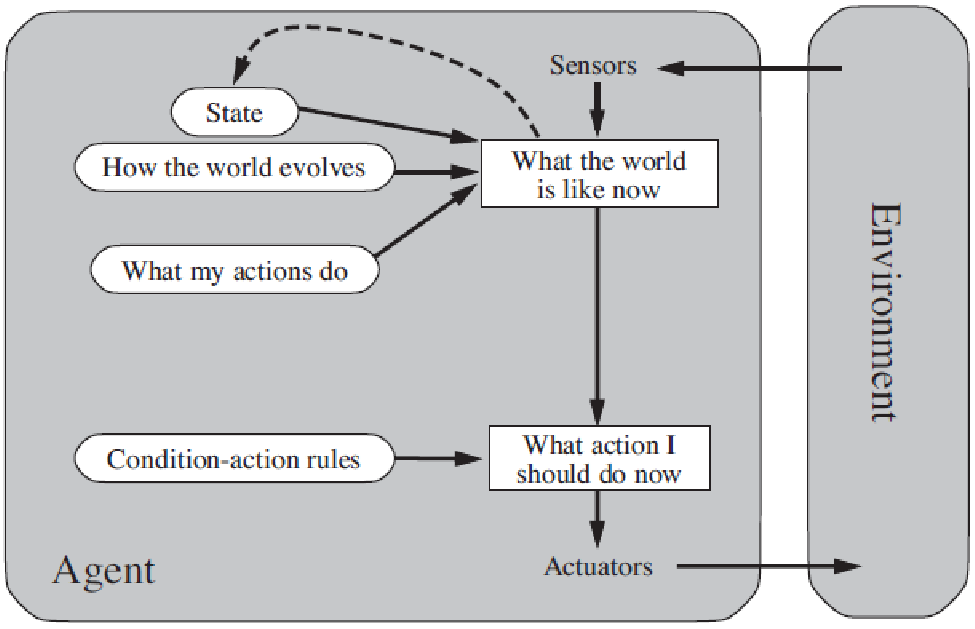
For example:

For the braking problem, the internal state is not too extensive— just the previous frame from the camera, allowing the agent to detect when two red lights at the edge of the vehicle go on or off simultaneously. For other driving tasks such as changing lanes, the agent needs to keep track of where the other cars are if it can't see them all at once. And for any driving to be possible at all, the agent needs to keep track of where its keys are.

Updating this internal state information as time goes by requires two kinds of knowledge to be encoded in the agent program:

1. 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.
2. 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**.



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
```

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

The function UPDATE-STATE,is responsible for creating the new internal state description.

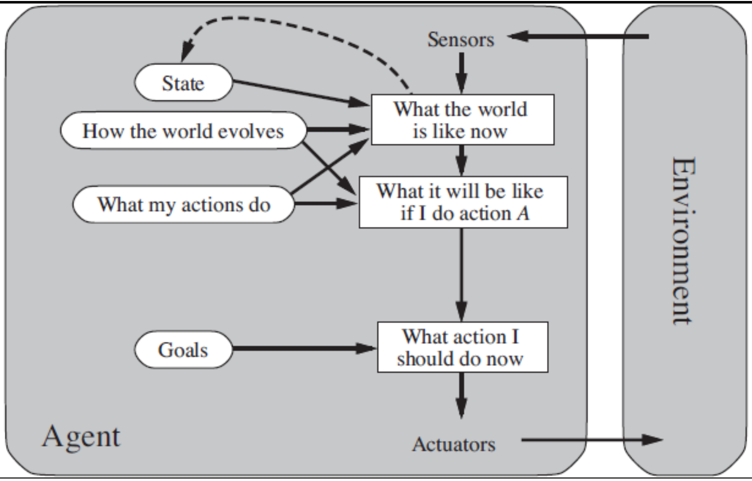
Goal-based agents

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 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.


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
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.

Sometimes goal-based action selection is straightforward—for example, when goal satisfaction results immediately from a single action.
Sometimes it will be more tricky—for example, when the agent has to consider long sequences of twists and turns in order to find a way to achieve the goal.
Search and **planning** are the subfields of AI devoted to finding action sequences that achieve the agent's goals.

A goal-based agent, in principle, could reason that if the car in front has its brake lights on, it will slow down. Given the way the world usually evolves, the only action that will achieve the goal of not hitting other cars is to brake.
Advantage of Goal based agents over Reflex agents:
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.
For the reflex agent, on the other hand, we would have to rewrite many condition–action rules. The goal-based agent's behavior can easily be changed to go to a different destination, simply by specifying that destination as the goal. The reflex agent's rules for when to turn and when to go straight will work only for a single destination; they must all be replaced to go somewhere new.

Utility-based agents

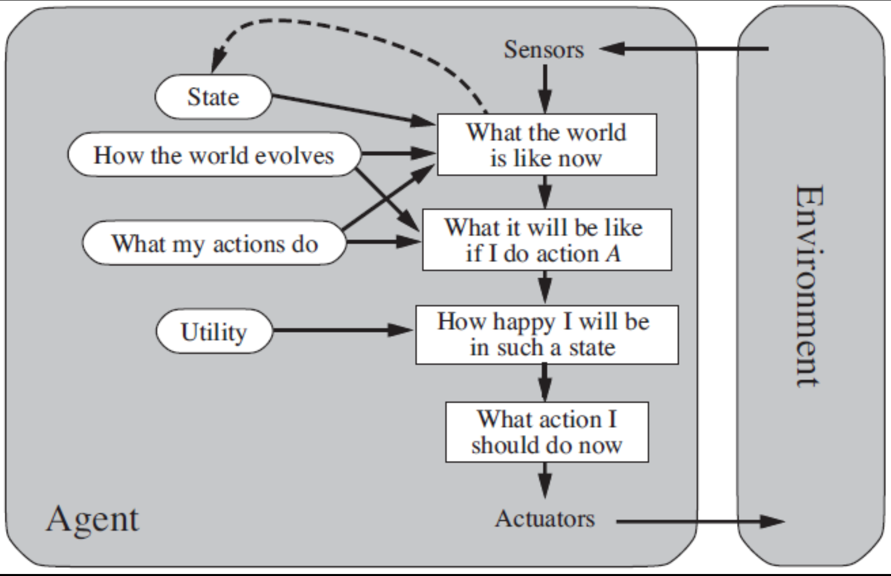
Goals alone are not enough to generate high-quality behavior in most environments.
For example, many action sequences will get the taxi to its destination (thereby achieving the goal) but some are quicker, safer, more reliable, or cheaper than others.
Goals just provide a crude binary distinction between "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.
A performance measure assigns a score to any given sequence of environment states, so it can easily distinguish between more and less desirable ways of getting to the taxi's destination.
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.

Advantage of Utility based agents over goal based agents

A utility-based agent has many advantages in terms of flexibility and learning.
Furthermore, in two kinds of cases, goals are inadequate but a utility-based agent can still make rational decisions.

1. 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.
2. 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.

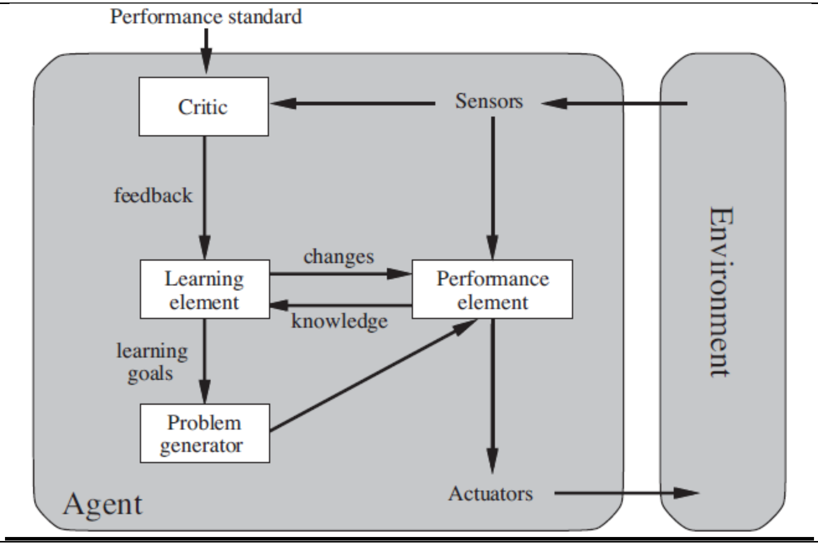
A rational utility-based agent chooses the action that maximizes the **expected utility** of the action outcomes—that is, the utility the agent expects to derive, on average, given the probabilities and utilities of each outcome.


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

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

Learning has an advantage: it allows the agent to operate in initially unknown environments and to become more competent than its initial knowledge alone might allow.
A learning agent can be divided into four conceptual components:
the **learning element**, which is responsible for making improvements,
performance element, which is responsible for selecting external actions. 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. Conceptually, one should think of it as being outside the agent altogether because the agent must not modify it to fit its own behavior.
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. But if the agent is willing to explore a little and do some perhaps suboptimal actions in the short run, it might discover much better actions for the long run.



A general learning agent

7	**Empiricist Approaches in Artificial Intelligence:**
	Empiricist approaches in artificial intelligence (AI) are based on the philosophy of empiricism, which emphasizes the importance of observation, evidence, and experience in acquiring knowledge. These approaches rely heavily on data-driven methods and learning from real-world experiences to build intelligent systems. Here are some key characteristics and examples of empiricist approaches in AI: 1. **Data-Driven Learning:** Empiricist AI methods heavily rely on data to train machine learning models. The more data these models have access to, the better they can learn and generalize from real-world examples. 2. **Inductive Reasoning:** Inductive reasoning is a fundamental aspect of empiricism, and it plays a significant role in empiricist AI. By observing specific instances or examples, AI systems attempt to infer general rules and patterns. 3. **Machine Learning**: Many machine learning techniques fall under the umbrella of empiricist approaches. Supervised learning, unsupervised learning, and reinforcement learning are common examples, where models learn from labeled data, patterns, or rewards and punishments, respectively. 4. **Big Data**: Empiricist AI benefits from large-scale datasets. These datasets contain diverse and representative samples of real-world scenarios, enabling AI models to generalize well and make accurate predictions. 5. **Pattern Recognition**: Empiricist AI excels in tasks that involve recognizing patterns and correlations within data. For example, image recognition, natural language processing, and speech recognition are areas where data-driven approaches have shown significant success. 6. **Deep Learning**: Deep learning is a subset of machine learning that employs neural networks with multiple layers to learn hierarchical representations of data. Deep learning models, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), are widely used in empiricist AI due to their ability to process large amounts of data and learn complex patterns. 7. **Reinforcement Learning**: This type of learning is often used in empiricist AI when dealing with tasks where an agent interacts with an environment and learns from the feedback received in the form of rewards and penalties. 8. **Online Learning:** Empiricist AI can adapt to changing data and environments through online learning, where the system updates its knowledge continuously as new data becomes available. 9. **Data Augmentation**: In order to further improve generalization, data augmentation techniques are used, creating variations of existing data to augment the training dataset. 10. **Bayesian Inference**: Bayesian methods are also considered part of empiricist

	approaches in AI, as they involve updating probabilities based on evidence and observed data. One of the main advantages of empiricist approaches is their ability to handle complex and real-world tasks by learning from vast amounts of data. However, they may require substantial computational resources and data to achieve optimal performance. Additionally, they might not always provide transparent explanations for their decisions, which can be a concern in critical applications. It's important to note that empiricist approaches are just one paradigm in AI, and there are other philosophies and methodologies, such as symbolic or logic-based approaches, which emphasize the use of formal rules and knowledge representation. In practice, a combination of various approaches is often employed to create robust and intelligent AI systems.
8	**<u>Applications of Artificial Intelligence:</u>** **Robotic vehicles**: A driverless robotic car named STANLEY sped through the rough terrain of the Mojave dessert at 22 mph, finishing the 132-mile course first to win the 2005 DARPA Grand Challenge. STANLEY is a Volkswagen Touareg outfitted with cameras, radar, and laser rangefinders to sense the environment and onboard software to command the steering, braking, and acceleration (Thrun, 2006). The following year CMU's BOSS won the Urban Challenge, safely driving in traffic through the streets of a closed Air Force base, obeying traffic rules and avoiding pedestrians and other vehicles. **Speech recognition**: A traveler calling United Airlines to book a flight can have the entire conversation guided by an automated speech recognition and dialog management system. **Autonomous planning and scheduling**: A hundred million miles from Earth, NASA's Remote Agent program became the first on-board autonomous planning program to control the scheduling of operations for a spacecraft (Jonsson *et al.*, 2000). REMOTE AGENT generated plans from high-level goals specified from the ground and monitored the execution of those plans—detecting, diagnosing, and recovering from problems as they occurred. Successor program MAPGEN plans the daily operations for NASA's Mars Exploration Rovers, and MEXAR2 did mission planning—both logistics and science planning—for the European Space Agency's Mars Express mission in 2008. **Game playing**: IBM's DEEP BLUE became the first computer program to defeat the world champion in a chess match when it bested Garry Kasparov by a score of 3.5 to 2.5 in an exhibition match (Goodman and Keene, 1997). Kasparov said that he felt a "new kind of intelligence" across the board from him. *Newsweek* magazine described the match as "The brain's last stand." The value of IBM's stock increased by $18 billion. Human champions studied Kasparov's loss and were able to draw a few matches in subsequent years, but the most recent human-computer matches have been won convincingly by the computer. **Spam fighting**: Each day, learning algorithms classify over a billion messages as spam, saving the recipient from having to waste time deleting what, for many users, could comprise 80% or 90% of all messages, if not classified away by algorithms. Because the spammers are continually updating their tactics, it is difficult for a static programmed approach to keep up, and learning algorithms work best. **Logistics planning**: During the Persian Gulf crisis of 1991, U.S. forces deployed a Dynamic Analysis and Replanning Tool, DART , to do automated logistics planning and

	scheduling for transportation. This involved up to 50,000 vehicles, cargo, and people at a time, and had to account for starting points, destinations, routes, and conflict resolution among all parameters. The AI planning techniques generated in hours a plan that would have taken weeks with older methods. The Defense Advanced Research Project Agency (DARPA) stated that this single application more than paid back DARPA's 30-year investment in AI. **Robotics**: The iRobot Corporation has sold over two million Roomba robotic vacuum cleaners for home use. The company also deploys the more rugged PackBot to Iraq and Afghanistan, where it is used to handle hazardous materials, clear explosives, and identify the location of snipers. **Machine Translation**: A computer program automatically translates from Arabic to English, allowing an English speaker to see the headline "Ardogan Confirms That Turkey Would Not Accept Any Pressure, Urging Them to Recognize Cyprus." The program uses a statistical model built from examples of Arabic-to-English translations and from examples of English text totaling two trillion words. None of the computer scientists on the team speak Arabic, but they do understand statistics and machine learning algorithms. These are just a few examples of artificial intelligence systems that exist today.
9	**<u>Limitations of Artificial Intelligence:</u>** AI, while incredibly powerful and promising, has its limitations. Some of the main limitations include: <u>Lack of common sense:</u> AI systems lack innate common sense and understanding of the world that humans possess. They can perform specific tasks extremely well within their programmed scope, but they struggle with generalizing knowledge and applying it in novel situations. <u>Data dependency:</u> AI models require large amounts of high-quality data to be trained effectively. Without sufficient data, their performance can be subpar, and they may even produce inaccurate or biased results. <u>Bias and fairness issues:</u> AI systems can inherit biases from the data they are trained on, reflecting the biases present in society. This can lead to discriminatory or unfair outcomes, especially in applications like hiring, lending, and law enforcement. <u>Lack of creativity:</u> While AI can generate content like text, images, and music, it is still far from being truly creative in the way humans are. AI-generated content is typically based on patterns in the data it was trained on, lacking genuine inspiration or understanding. <u>Interpretability and explainability:</u> Many AI models, especially deep learning ones, are often seen as "black boxes" because they can be challenging to interpret and explain. This lack of transparency raises concerns in critical applications, like healthcare or autonomous vehicles, where understanding the decision-making process is crucial. <u>Security and privacy concerns:</u> AI systems are susceptible to attacks, and malicious actors can exploit vulnerabilities for harmful purposes. Moreover, the collection and

use of vast amounts of personal data by AI systems raise significant privacy concerns.

Contextual limitations: AI often performs well in specific contexts and struggles to adapt to changes in the environment or tasks it was not designed for. This limitation hampers its ability to handle complex and dynamic real-world scenarios.

Lack of emotional intelligence: AI lacks emotional intelligence and understanding of human emotions. While AI can analyze emotions based on data, it does not truly comprehend emotions and may struggle to respond appropriately to emotionally charged situations.

Energy and computational resources: Many AI models are computationally intensive and require significant energy consumption, limiting their accessibility in resource-constrained environments and contributing to environmental concerns.

. Dependence on human input: AI systems still require human guidance, validation, and continuous monitoring. They are not entirely autonomous and can't replace human expertise in many critical tasks.

Despite these limitations, AI continues to advance rapidly, and ongoing research and development aim to address these challenges and unlock its full potential while mitigating risks.