ITDO6014
AI AND DS-1

Module 1: Introduction to AI

Course Objectives

- To introduce the students' with different issues involved in trying to define and simulate intelligence.
- To familiarize the students' with specific, well known Artificial Intelligence methods, algorithms and knowledge representation schemes.
- To introduce students' different techniques which will help them build simple intelligent systems based on AI/IA concepts.
- To introduce students to data science and problem solving with data science and statistics.
- To enable students to choose appropriately from a wider range of exploratory and inferential methods for analyzing data, and interpret the results contextually.
- To enable students to apply types of machine learning methods for real world problems.

Course Outcomes

- Develop a basic understanding of the building blocks of AI as presented in terms of intelligent agents.
- 2 Apply an appropriate problem-solving method and knowledge-representation scheme.
- 3 Develop an ability to analyze and formalize the problem (as a state space, graph, etc.). They will be able to evaluate and select the appropriate search method.
- 4 Apply problem solving concepts with data science and will be able to tackle them from a statistical perspective.
- Choose and apply appropriately from a wider range of exploratory and inferential methods for analyzing data and will be able to evaluate and interpret the results contextually.
- 6 Understand and apply types of machine learning methods for real world problems.

Grading

Internal Assessment	20 Marks
Theory	80 Marks
Total	100 Marks

Text Book:

1. Stuart Russell and Peter Norvig, Artificial Intelligence: A Modern Approach, 2nd Edition, Pearson Education.

2. Elaine Rich, Kevin Knight, Shivshankar B Nair, Artificial Intelligence, McGraw Hill, 3rd Edition

Grading

Online References:

Website/Reference link

1. **https://nptel.ac.in/noc/courses/noc19/SEM2/noc19-cs83/**
2. **https://nptel.ac.in/courses/106/105/106105077/**
3. **https://www.coursera.org/specializations/jhu-data-science**
4. **https://www.coursera.org/learn/machine-learning**
5. **https://www.udemy.com/course/statistics-for-data-science-and-business-analysis/**

Introduction to AI

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

Introduction to AI

- Definitions of artificial intelligence, organized into four categories:

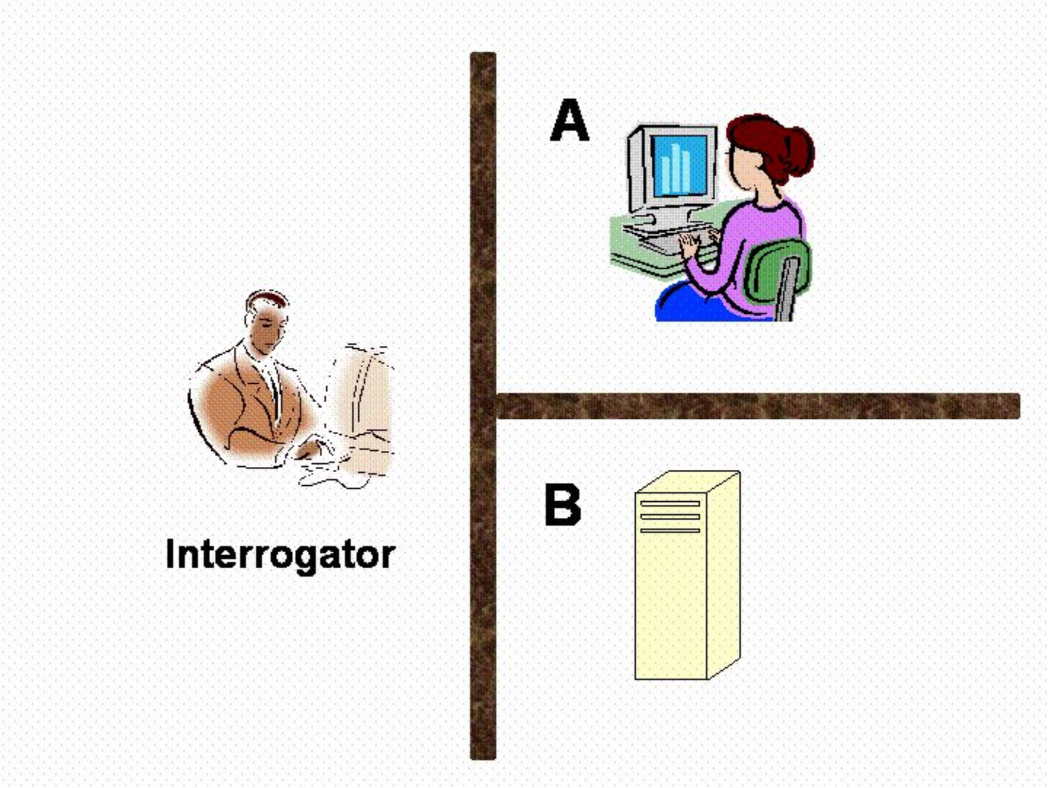
1. System that reasons (thinks) like human
2. System that reasons (thinks) rationally
3. System that acts like human
4. System that acts rationally

Introduction to AI

To Reason Human	*To Reason 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, act (Winston, 1992).
To Act Human	***To Act Rationally***
The art of creating machines that perform functions that require intelligence when performed by people. (Kurzwil, 1990). The study of how to make computers do things at which, at the moment, people do better (Rich Knight, 1991).	The field of study that seeks to explain and emulate intelligent behavior in terms of computational processes (Schalkoff, 1990). The branch of computer science concerned with automation of intelligent behavior. (Luger Stubblefield, 1993).

Introduction to AI

- The Turing Test approach



Introduction to AI

- Consider the following setting. There are two rooms, A and B. One of the rooms contains a computer. The other contains a human. The interrogator is outside and does not know which one is a computer. He can ask questions through a teletype and receives answers from both A and B. The interrogator needs to identify whether A or B are humans. To pass the Turing test, the machine has to fool the interrogator into believing that it is human.

Introduction to AI

- programming a computer to pass the test provides plenty to work on. The computer would need to possess the following capabilities:
- **Natural language processing** to enable it to communicate successfully in English
- **Knowledge representation** to store what it knows or liears;
- **Automated reasoning** to use the stored inforrnation to answer questions and to draw new conclusions;
- **Machine learning** to adapt to new circumstances and to detect and extrapolate patterns.

Introduction to AI

- **Computer vision** to perceive objects, and
- **Robotics** to manipulate objects and move about.
- These six disciplines compose most of AI

History of AI

1956 - John McCarthy coined the term 'artificial intelligence' and had the first AI conference.

1969 - Shakey was the first general-purpose mobile robot built. It is now able to do things with a purpose vs. just a list of instructions.

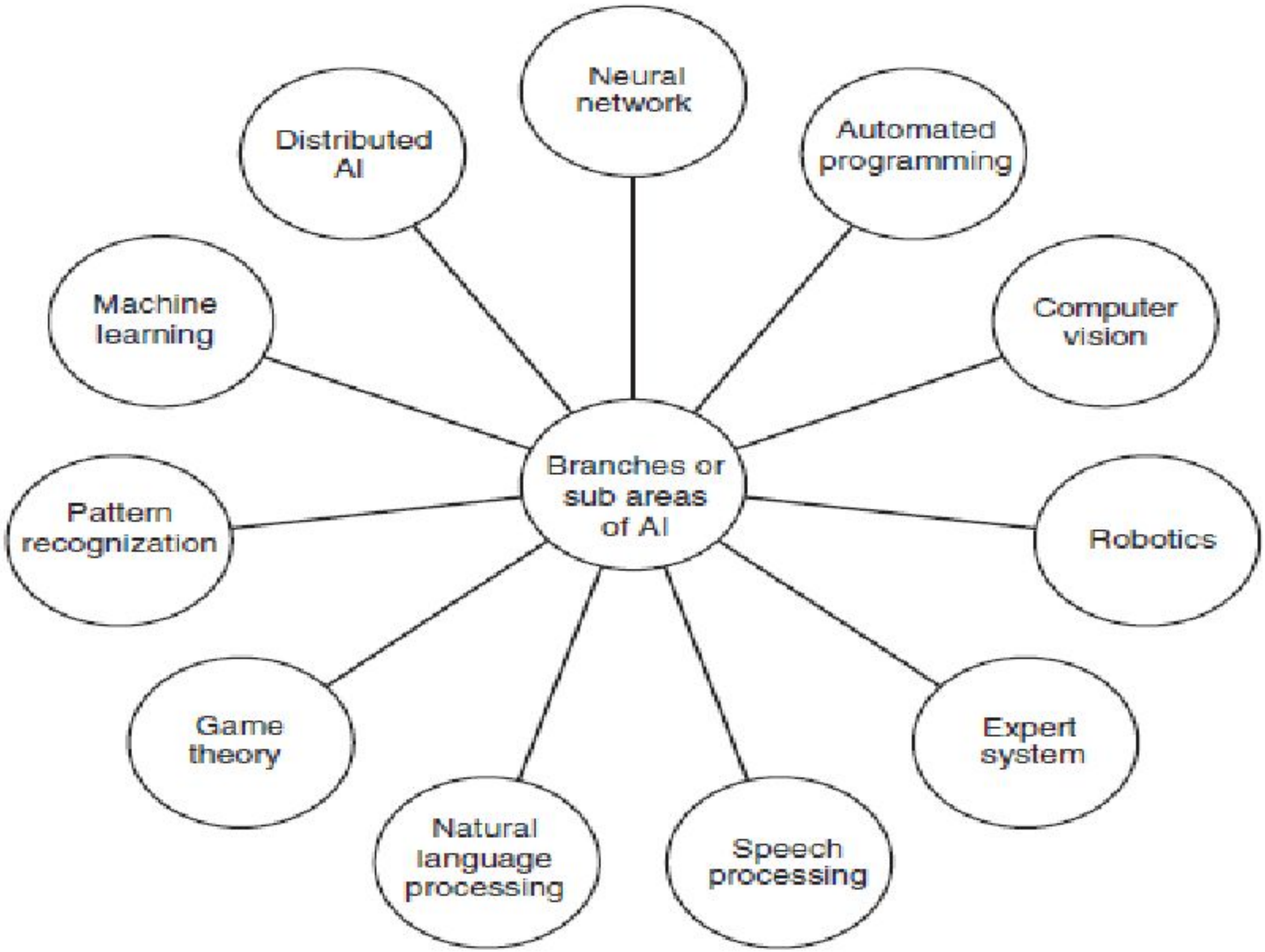
1997 - Supercomputer 'Deep Blue' was designed, and it defeated the world champion chess player in a match. It was a massive milestone by IBM to create this large computer.

2002 - The first commercially successful robotic vacuum cleaner was created.

2005 - 2019 - Today, we have speech recognition, robotic process automation (RPA), a dancing robot, smart homes, and other innovations make their debut.

2020 - Baidu releases the LinearFold AI algorithm to medical and scientific and medical teams developing a vaccine during the early stages of the SARS-CoV-2 (COVID-19) pandemic. The algorithm can predict the RNA sequence of the virus in only 27 seconds, which is 120 times faster than other methods.

Branches of AI



AI Programming

A number of programming languages exist that are used to build AI systems. General programming languages, such as C++, R, Java, Python, and LISP (List Processing) are frequently used, because these are the languages with which most computer scientists have got experience.

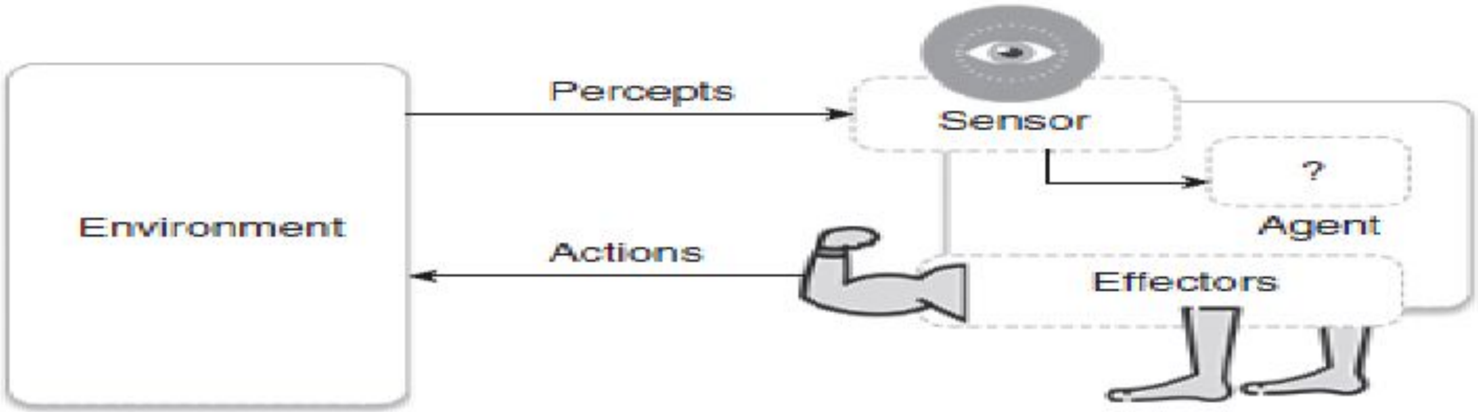
Here are some languages that are most typically used for creating the AI projects:

- PROLOG, LISP, R, Python, Java, C++

AI Agent

- An agent is anything that can be viewed as perceiving its environment through sensors and
- SENSOR acting upon that environment through actuators.
- ACTUATOR A human agent has eyes, ears, and other organs for sensors and hands, legs, mouth, and other body parts for actuators.
- A robotic agent might have cameras and infrared range finders for sensors and various motors for actuators.
- A software agent receives keystrokes, file contents, and network packets as sensory inputs and acts on the environment by displaying on the screen, writing files, and sending network packets

AI Agent



AI Agent

Agent terminology

- **Performance measure of agent:** It is the criteria determining the success of an agent.
- **Behaviour/action of agent:** It is the action performed by an agent after any specified sequence of the percepts.
- **Percept:** It is defined as an agent's perceptual inputs at a specified instance.
- **Percept sequence:** It is defined as the history of everything that an agent has perceived till date.
- **Agent function:** It is defined as a map from the precept sequence to an action.

Agent function, $a = F(p)$

where *p is the current percept, a is the action carried out, and F is the agent function.*

AI Agent

Agent program

On each invocation, the memory of the agent is updated to mirror the new percept, the best action selected and the fact that the action was taken is also stored inside the memory. The memory persists from one invocation to the next.

Function SKELETON-AGENT(Percept) **returns**
Action **Static:** Memory, the agents memory of the
world Memory <- UPDATE-MEMORY (Memory,
Percept) Action <- CHOOSE-BEST-
ACTION(Memory) Memory <- UPDATE-
MEMORY(Memory, Percept) **return** action

AI Agent

function REFLEX-VACUUM-AGENT([*location*,*status*]) **returns** an action

if *status* = *Dirty* **then return** *Suck*
else if *location* = *A* **then return** *Right*
else if *location* = **B then return Left**

Properties of task environments

- *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.*

Properties of task environments

- *Fully observable vs. partially observable.*
- **Fully Observable Environment:**
- Imagine a grid world where the robot can see the entire grid at any given time.
- The robot knows the exact positions of obstacles, goals, and any other relevant information in the entire grid.
- Each grid cell is visible to the robot, and it has complete information about the state of the environment.
- **Example:**
- The robot is in a 5x5 grid world, and it knows the exact positions of walls, goals, and other objects in every cell.

Properties of task environments

- *Fully observable vs. partially observable.*
- **Partially Observable Environment:**
- Now, let's consider a partially observable environment where the robot has limited visibility or sensor range.
- The robot can only perceive a limited portion of the grid at any given time, making it unaware of the entire state of the environment.
- The robot may need to remember its past observations or use sensors to gather information about the surroundings.
- **Example:**
- The robot is in the same 5x5 grid world, but it has a limited sensor range of 2 cells. It can only see the contents of the cells within this range.
- As the robot moves, it updates its knowledge based on the newly observed cells, but it doesn't have complete information about the entire grid.

Properties of task environments

- *Single agent vs. multiagent.*
- *The distinction between single-agent and multiagent environments 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.*
- *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.*

Properties of task environments

- *Single agent vs. multiagent.*
- **Single Agent Environment:**
- In a single-agent environment, there is only one entity or agent responsible for performing the tasks within the environment.
- **Example:**
  - A robot is assigned to clean a single room. The robot is the only entity responsible for navigating through the room, identifying dirty areas, and cleaning them. There are no other agents involved in the task.

Properties of task environments

- *Single agent vs. multiagent.*
- **Multiagent Environment:**
- In a multiagent environment, there are multiple agents, each with its own goals, and they may interact with or influence each other.
- **Example:**
  - Consider a scenario where two robots are assigned to clean different rooms in a house. Each robot is a separate agent with the goal of cleaning its designated room. The robots may need to coordinate their actions to avoid collisions or share information about their progress. In this case, the environment involves multiple agents working simultaneously.

Properties of task environments

- *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. Dynamic environments are continuously asking the agent what it wants to do; if it hasn't decided yet, that counts as deciding to do nothing. Taxi driving is clearly dynamic: the other cars and the taxi itself keep moving while the driving algorithm dithers about what to do next.*
- *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. Crossword puzzles are static.*
- *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. Chess, when played with a clock, is semidynamic.*

Properties of task environments

- *Static vs, dynamic.*
- **Static Environment:**
- In a static environment, the elements and features of the environment do not change over time.
- **Example:**
  - A robot navigating through a museum where the layout, positions of exhibits, and obstacles remain constant. The robot can plan its path without worrying about changes in the environment during its navigation. The environment remains fixed, and there are no dynamic elements.

Properties of task environments

- *Static vs, dynamic.*
- **Dynamic Environment:**
- In a dynamic environment, the elements and features of the environment can change over time.
- **Example:**
  - Now, consider the same robot navigating through a busy city street. In this dynamic environment, the positions of pedestrians, vehicles, and other obstacles can change rapidly. The robot needs to adapt its navigation in real-time, taking into account the dynamic nature of the surroundings. The environment is not fixed, and various elements can move or change positions.

Properties of task environments

- *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.*
- *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*

Properties of task environments

- **Deterministic Environment:**
- In a deterministic environment, the outcome of an action is entirely predictable, given the current state of the environment and the action taken.
- **Example:**
  - Imagine a board game where a player moves a token based on the roll of a fair six-sided die. In a deterministic environment, each roll of the die produces a predictable and fixed outcome. If the player rolls a 3, the token will move three spaces, and this result is consistent every time they roll a 3.

Properties of task environments

- **Stochastic Environment:**
- In a stochastic environment, the outcome of an action has some level of randomness or uncertainty, even if the current state and action are the same.
- **Example:**
  - Now, consider a modified version of the board game where the player moves the token based on the roll of a biased die. The biased die has a chance of producing different outcomes, and the result may vary even if the player takes the same action multiple times. For example, rolling a 3 might result in the token moving three spaces most of the time, but occasionally it could move four or two spaces due to the biased nature of the die.

Properties of task environments

- *Discrete vs. continuous.*
- *The discrete/continuous distinction can be applied to the state of the environment, to the way time is handled, and to the percepts and actions of the agent.*
- *For example, a discrete-state environment such as a chess game has a finite number of distinct states. 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.). .*

Properties of task environments

- *Discrete vs. continuous.*
- **Discrete Environment:**
- In a discrete environment, the set of possible states and actions is finite, countable, or distinct. There is a clear separation between different states and actions.
- **Example:**
  - Think of a robot navigating through a grid world. The robot can move from one grid cell to another, and each cell represents a distinct, discrete state. The robot's movement is limited to discrete actions, such as moving up, down, left, or right. The state and action spaces are well-defined and discrete.

Properties of task environments

- *Discrete vs. continuous.*
- **Continuous Environment:**
- In a continuous environment, the set of possible states and actions is uncountable and can take on a continuous range of values. There is no clear separation between different states or actions.
- **Example:**
  - Now, consider a robotic arm in a factory. The position of the robotic arm can take on a continuous range of values, and its movement is not constrained to discrete steps. The state space, representing the position of the arm, and the action space, representing the movement, are continuous. The arm can move smoothly between any two positions.

Properties of task environments

- *Episodic vs. sequential*
- *In an episodic task environment, the agent's experience is divided into atomic episodes. Each episode consists of the agent perceiving and then performing a single action. Crucially, the next episode does not depend on the actions taken in previous episodes. In episodic environments, the choice of action in each episode depends only on the episode itself. 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.*

Properties of task environments

- *Episodic vs. sequential*
- **Episodic Environment:**
- In an episodic environment, the agent's experience is divided into distinct episodes, and each episode is a self-contained task with a clear beginning and end. The agent's actions within one episode do not affect subsequent episodes.
- **Example:**
  - Think of a game of chess. Each game is an episode with a clear start (initial board setup) and end (checkmate or stalemate). The outcome of one game doesn't influence the next game. The agent (player) starts fresh with a new board at the beginning of each game.

Properties of task environments

- <u>*Episodic vs. sequential*</u>
- **Sequential Environment:**
- In a sequential environment, the agent's actions have a lasting impact on future states, and the current state depends on the agent's history of actions and observations.
- **Example:**
  - Consider a video game where a character explores a virtual world. The character's actions, such as collecting items or defeating enemies, affect its future state and capabilities. The environment is sequential because the consequences of the character's actions persist over time, and the current state depends on the sequence of actions taken.

Properties of task environments

- **Scenario: Robot Vacuum in a House**
- **Fully Observable vs. Partially Observable:**
- **Fully Observable:**
  - The robot vacuum has sensors covering the entire house, providing complete information about the location of furniture, walls, and dirty areas in the entire environment.
- **Partially Observable:**
  - The robot vacuum has a limited sensor range, so it can only perceive and gather information about a small portion of the room at any given time. As it moves, it updates its knowledge based on new observations, but it doesn't have complete information about the entire house.

Properties of task environments

- **Scenario: Robot Vacuum in a House**
- **Single Agent vs. Multiagent:**
- **Single Agent:**
  - There is only one robot vacuum in the house responsible for cleaning all the rooms.
- **Multiagent:**
  - There are multiple robot vacuums, each assigned to clean a specific room. The robots may need to coordinate their movements to efficiently cover the entire house without collisions.

Properties of task environments

- **Scenario: Robot Vacuum in a House**
- **Static vs. Dynamic:**
- **Static:**
  - The layout of the house remains constant, and there are no changes in the positions of furniture or obstacles during the cleaning process.
- **Dynamic:**
  - Family members or pets are moving around the house, introducing dynamic elements that the robot must navigate around. The positions of these dynamic elements change over time.

Properties of task environments

- **Scenario: Robot Vacuum in a House**
- **Deterministic vs. Stochastic:**
- **Deterministic:**
  - The robot's cleaning actions have predictable outcomes. If it detects a dirty area, it will always follow the same algorithm to clean it.
- **Stochastic:**
  - The robot encounters different levels of dirtiness in the house, and its cleaning efficiency may vary due to the stochastic nature of dirt distribution. The outcomes of cleaning actions may have some randomness.

Properties of task environments

- **Scenario: Robot Vacuum in a House**
- **Discrete vs. Continuous:**
- **Discrete:**
  - The robot moves from one grid cell to another within each room. Its cleaning actions are limited to discrete movements and operations.
- **Continuous:**
  - The robot's sensors and movements operate in a continuous space, allowing it to smoothly navigate around obstacles without being constrained to discrete steps.

Properties of task environments

- **Scenario: Robot Vacuum in a House**
- **Episodic vs. Sequential:**
- **Episodic:**
  - Each room-cleaning task can be considered as an episode with a clear beginning (entering the room) and end (finishing cleaning). The outcome of cleaning one room doesn't affect the cleaning of other rooms.
- **Sequential:**
  - The robot's actions, such as deciding where to move next based on its cleaning progress, are sequential. The consequences of its cleaning actions persist, and the state of the environment depends on its history of movements and observations.

Properties of task environments

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

Properties of task environments

Agent Type	Performance Measure	Environment	Actuators	Sensors
Medical diagnosis system	Healthy patient, minimize costs, lawsuits	Patient, hospital, staff	Display 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 categorization of scene	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	Maximize purity, yield, safety	Refinery, operators	Valves, pumps, heaters, displays	Temperature, pressure, chemical sensors
Interactive English tutor	Maximize student's score on test	Set of students, testing agency	Display exercises, suggestions, corrections	Keyboard entry

Properties of task environments

PEAS Descriptors for automated Taxi Driver	
Performance Measure	Safety, time, legal drive, comfort
Environment	Roads, other cars, pedestrians, road signs
Actuators	Steering, accelerator, brake, signal, horn
Sensors	Camera, sonar, GPS, Speedometer, odometer, accelerometer, engine sensors, keyboard

PEAS for vacuum cleaner	
Performance Measure	cleanness, efficiency: distance traveled to clean, battery life, security
Environment	room, table, wood floor, carpet, different obstacles
Actuators	wheels, different brushes, vacuum extractor
Sensors	camera, dirt detection sensor, cliff sensor, bump sensors, infrared wall sensors

Types of Agents

1. Simple reflex agent
2. Model based agent
3. Goal based agent
4. Utility based agent
5. Learning agent

Types of Agents

1. Simple reflex agent

Simple reflex agent is said to be the simplest kind of agent.

These agents select an action based on the current percept ignoring the rest of the percept history.

These percept to action mapping which is known as *condition-action rules* (so-called *situation–action rules, productions, or if–then rules)* in the simple reflex agent.

It can be represented as follows:

if *{set of percepts}* **then** *{set of actions}*

Types of Agents

Limitations

- **Sensors:** Collect information about the current state of the environment.
- **Rules/Conditions:** Define a set of rules or conditions based on the sensor input to make decisions.
- **Actuators:** Execute actions based on the decision made by the rules.

Types of Agents

- **Example:**
- Let's consider a simple reflex agent for a thermostat that controls the heating system in a room. The goal is to maintain the room temperature at a comfortable level (e.g., 20 degrees Celsius). The agent will have a sensor to measure the current room temperature and an actuator to control the heating system.

Types of Agents

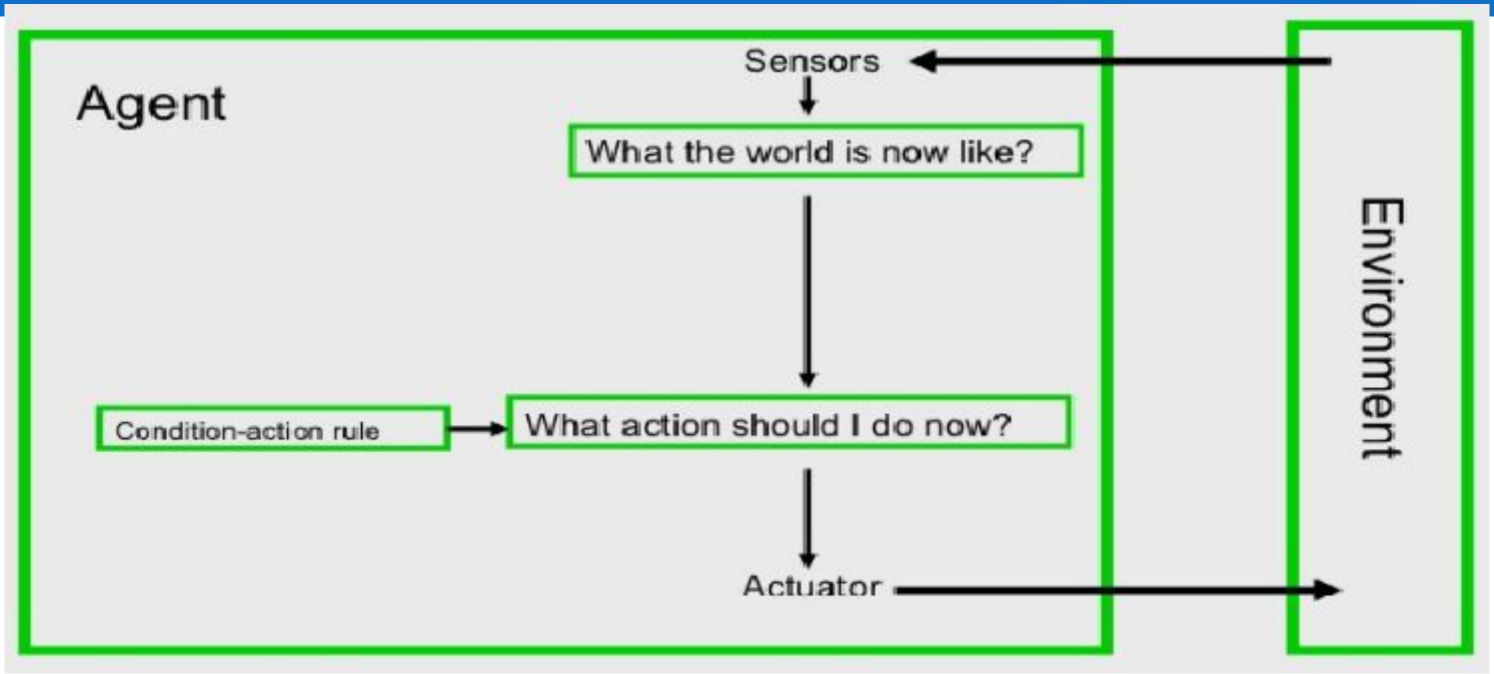

```
# Simple Reflex Agent Program for a Thermostat

def simple_reflex_agent(current_temperature):
    # Rules/Conditions
    if current_temperature < 20:
        return "Turn on heating"
    elif current_temperature > 20:
        return "Turn off heating"
    else:
        return "Maintain current state"
```

Types of Agents

- # Example Usage
- current_room_temperature = 18 # Assume the current room temperature is 18 degrees Celsius
- action = simple_reflex_agent(current_room_temperature)

- # Actuators
- if action == "Turn on heating":
-     print("Heating system turned on.")
- elif action == "Turn off heating":
-     print("Heating system turned off.")
- else:
-     print("Room temperature is at a comfortable level.")

Types of Agents



Types of Agents

Limitations

- Intelligence level in these agents is very limited.
- It works only in a fully observable environment.
- It does not hold any knowledge or information of nonperceptual parts of state.
- Because of the static knowledge based; it's usually too big to generate and store.
- If any change in the environment happens, the collection of the rules are required to be updated.

Types of Agents

2. Model based agent

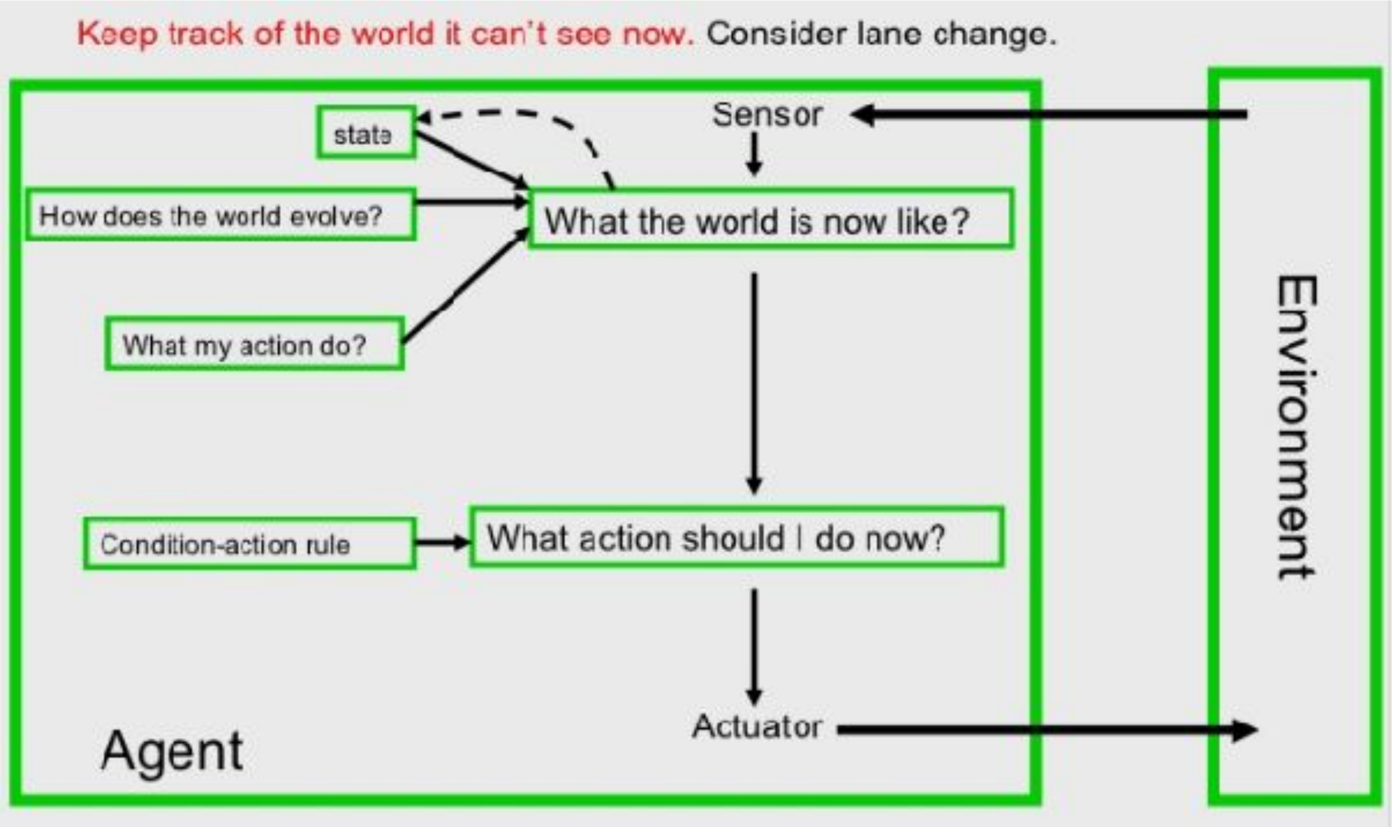
- A model-based agent is an artificial intelligence agent that maintains an internal model or representation of the world. Unlike a simple reflex agent, a model-based agent considers not only the current percept but also incorporates past percepts and actions to build a model of how the world behaves. This internal model helps the agent make more informed decisions by anticipating the consequences of its actions.

Types of Agents

2. Model based agent

- **Model-Based Agent Program:**
- **Percept History:** Maintain a history of past percepts and actions.
- **Update Model:** Use the percept history to update the internal model of the world.
- **Decision Making:** Make decisions based on the updated model.
- **Actuators:** Execute actions based on the decision made by the updated model.

Types of Agents



Types of Agents

2. Model based agent

- **Example:**
- Consider a simple model-based agent for a vacuum cleaner. The goal is to clean a room, and the agent needs to decide whether to move left, move right, or clean based on its internal model of the room's state.

Types of Agents

2. Model based agent

- **class VacuumCleanerAgent:**
-     def __init__(self):
-         self.internal_model = {'A': 'Dirty', 'B': 'Dirty'} # Initial model of room state
-         self.percept_history = []
-     **def update_model(self, percept, action):**
-         # Update internal model based on the percept and action
-         self.percept_history.append((percept, action))

Types of Agents

2. Model based agent

- # Update the model based on the observed state
- location, status = percept
- self.internal_model[location] = status
- def decide_action(self):
- # Decision making based on the internal model
- dirty_locations = [loc for loc, status in self.internal_model.items() if status == 'Dirty']

Types of Agents

2. Model based agent

- if dirty_locations:
- # If there are dirty locations, choose one to clean
- return 'Clean', dirty_locations[0]
- else:
- # If the room is clean, move to the next available location
- last_location, _ = self.percept_history[-1] if self.percept_history else ('A', 'Clean')
- return 'Move', 'B' if last_location == 'A' else 'A'

Types of Agents

2. Model based agent

- # Example
- vacuum_agent = VacuumCleanerAgent()
- # Initial percept
- initial_percept = ('A', 'Dirty')
- # Agent makes a decision and takes an action
- action, location = vacuum_agent.decide_action()
- print(f"Action: {action}, Location: {location}")
- # Update internal model based on the percept and action
- vacuum_agent.update_model(initial_percept, action)
- # Display the updated internal model
- print("Internal Model:", vacuum_agent.internal_model)

Types of Agents

3. Goal based

- A goal-based agent is an artificial intelligence agent that operates by considering its goals and making decisions to achieve those goals. Unlike simple reflex agents and model-based agents, goal-based agents focus on achieving specific objectives rather than just reacting to the current state of the environment.

Types of Agents

3. Goal based

- **Goal-Based Agent Program:**
- **Goals:** Define the goals or objectives the agent wants to achieve.
- **Action Planning:** Develop a plan or sequence of actions to reach the goals.
- **Decision Making:** Make decisions based on the current state and the plan to move closer to the goals.
- **Actuators:** Execute actions based on the decision made by the goal-based planning.

Types of Agents

3. Goal based

- consider a simple goal-based agent for a delivery robot. The goal is to deliver a package from point A to point B. The agent will have to navigate through a grid-like environment to reach its destination.

Types of Agents

3. Goal based

```
# Goal-Based Agent Program for a Delivery Robot
class DeliveryRobotAgent:
    def __init__(self, current_location, destination):
        self.current_location = current_location
        self.destination = destination
    def define_goals(self):
        # Define the goal of reaching the destination
        return f"Reach {self.destination}"
```

Types of Agents

3. Goal based

- def plan_action_sequence(self):
- # Create a plan to move from the current location to the destination
- # For simplicity, let's assume a grid-like environment
- actions = []

Types of Agents

3. Goal based

- while self.current_location != self.destination:
- if self.current_location[0] < self.destination[0]:
- actions.append('Move right')
- self.current_location = (self.current_location[0] + 1, self.current_location[1])
- elif self.current_location[0] > self.destination[0]:
- actions.append('Move left')
- self.current_location = (self.current_location[0] - 1, self.current_location[1])

Types of Agents

3. Goal based

- if self.current_location[1] < self.destination[1]:
- actions.append('Move up')
- self.current_location = (self.current_location[0], self.current_location[1] + 1)
- elif self.current_location[1] > self.destination[1]:
- actions.append('Move down')
- self.current_location = (self.current_location[0], self.current_location[1] - 1)
- return actions

Types of Agents

3. Goal based

```
def execute_action(self, action):
        # Execute the planned action
        print(f"Executing: {action}")
# Example Usage
robot_agent = DeliveryRobotAgent(current_location=(0, 0), destination=(3, 2))
# Define the goal
goal = robot_agent.define_goals()
# Plan the action sequence
action_sequence = robot_agent.plan_action_sequence()
# Execute each action in the sequence
for action in action_sequence:
    robot_agent.execute_action(action)
```

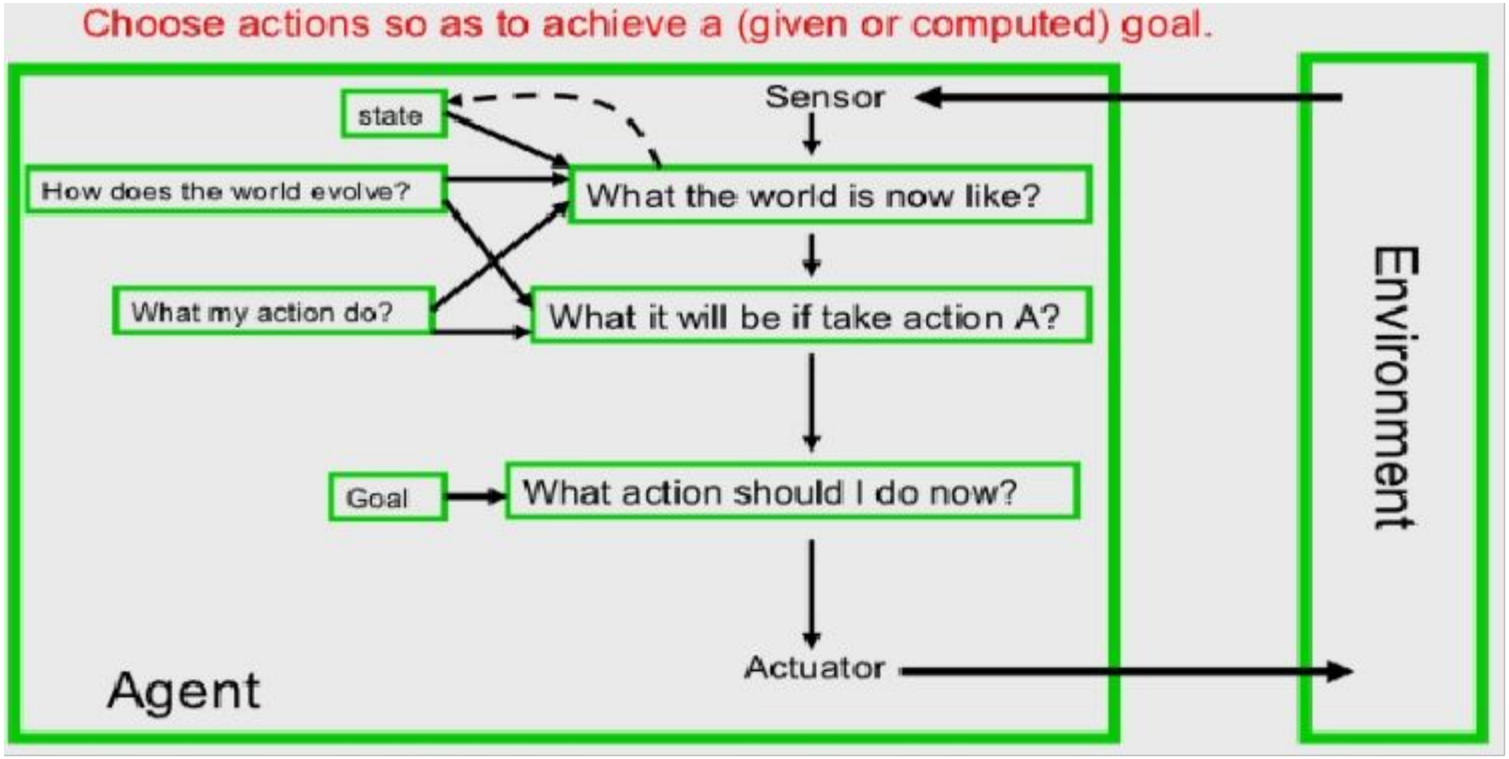
Types of Agents

3. Goal based

- In this example, the 'DeliveryRobotAgent' defines a goal of reaching a destination. It then plans a sequence of actions to move from the current location to the destination using simple grid-based movements (move left/right/up/down). The agent executes each action in the sequence until it reaches the goal.

- This showcases the basic structure of a goal-based agent, where the agent plans and executes actions to achieve a predefined goal. In more complex scenarios, the planning process might involve more sophisticated algorithms, and the goals could be dynamically adjusted based on the environment's changing conditions.

Types of Agents



Types of Agents

4. Utility based agent

- A utility-based agent is an artificial intelligence agent that makes decisions by considering the utility or desirability of different actions. The agent evaluates the outcomes of various actions and selects the one with the highest expected utility. Utility is a measure of the desirability or satisfaction associated with achieving a particular state or outcome.

Types of Agents

4. Utility based agent

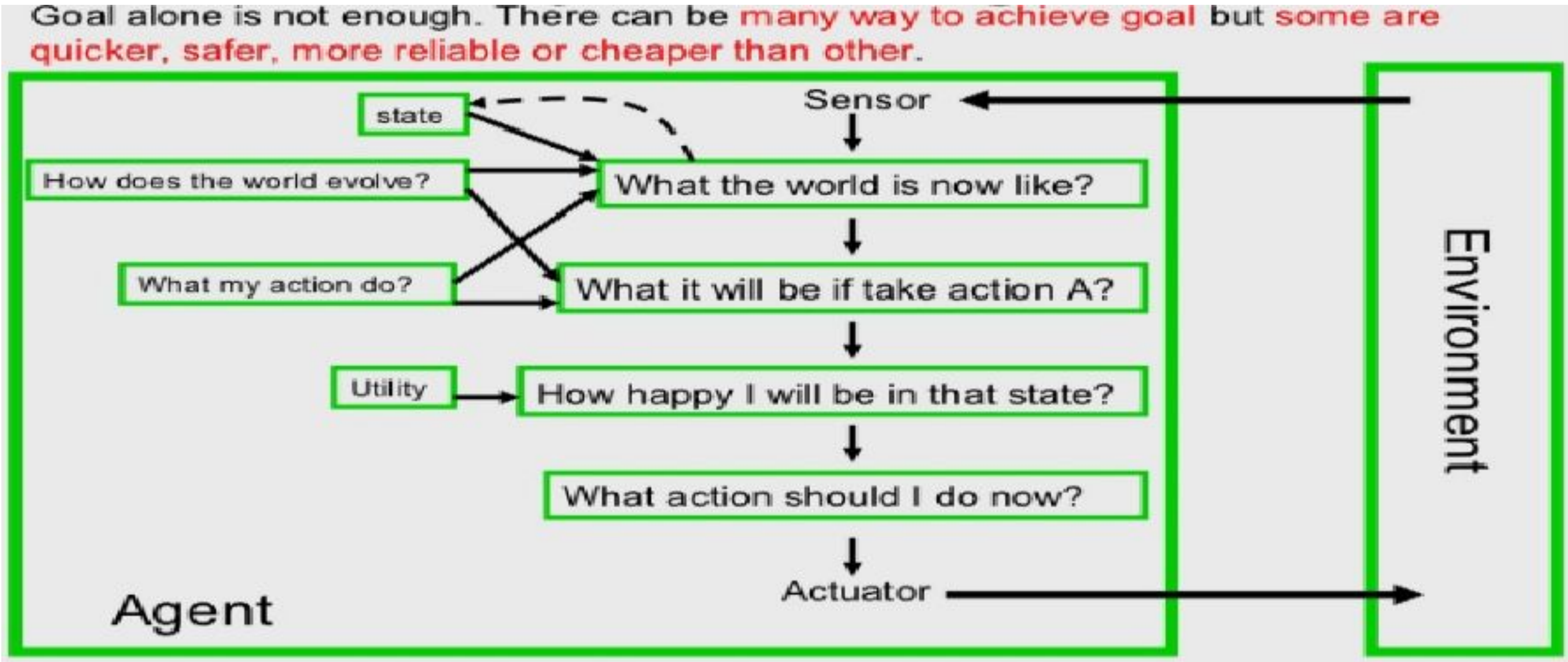
- **Utility-Based Agent Program:**
- **Define Utility Function:** Specify a utility function that assigns a numerica value to each possible state or outcome, indicating its desirability.
- **Evaluate Actions:** For each possible action, estimate the expected utility by considering the potential outcomes and their associated utilities.
- **Decision Making:** Select the action with the highest expected utility.
- **Actuators:** Execute the chosen action.
- **Example:**
- Consider a simple utility-based agent for a restaurant recommendation system. The goal is to recommend a restaurant to a user based on their preferences for cuisine, price, and distance.

Types of Agents

4. Utility based agent

- **Utility-Based Agent Program:**
- **Define Utility Function:** Specify a utility function that assigns a numerica value to each possible state or outcome, indicating its desirability.
- **Evaluate Actions:** For each possible action, estimate the expected utility by considering the potential outcomes and their associated utilities.
- **Decision Making:** Select the action with the highest expected utility.
- **Actuators:** Execute the chosen action.

Types of Agents



Types of Agents

5. Learning agent

- By actively exploring and experimenting with their environment, the most powerful agents are able to learn.
- A learning agent can be further divided into the four conceptual components