**Artificial Intelligence**

**Notes by MOHAMMAD EMRAN AHMED EMON**

**BSc in SOFTWARE ENGINEERING,**

**SHAHJALAL UNIVERSITY OF SCIENCE AND TECHNOLOGY, SYLHET.**

**Registration No: 2020831040**

**Session: 2020-21**

**Organizing Secretary, SWE Society**

**Former Assistant Publication Secretary, SPORTS SUST**

**Former Executive Member, SWE Society**

**Start: 23/10/2024**

**End: 26/10/2024**

**Q: What is Artificial Intelligence?**

AI is type of computer science that is focuses on creating machine and program that can think and learn like human

Also, AI is simulation of human intelligences processes by machines, especially computes system. These processes include learning, reasoning and self-correction.

In simpler terms, AI is about teaching computers to do tasks that normally require human intelligence

**Example:** Let's take the spam email example and break it down more:

1. You provide thousands of spam emails (the data) to the AI.
2. AI looks for patterns in those emails, like common words or formats.
3. Then, when a new email arrives, AI checks if it matches those patterns.
4. If the new email has lots of "spam-like" patterns, the AI decides it’s spam and moves it to the spam folder.

In this example, the AI learned from the data and used its algorithm to make a decision.

Like Siri/Google Assistant (understand voice and respond), Self-driving cars (can ‘see’ the road and make driving decisions.

AI helps you in-

1. understand,
2. solve problems, and
3. make decisions.

**Q: How does AI work?**

AI works by learning from data. For example, if you show an AI thousands of pictures of cats, it can learn what a cat looks like. Later, when you show it a new picture, it can tell whether there is a cat in it or not.

This learning process is like how humans learn. If you saw thousands of cats, you would easily recognize a cat later. AI uses **algorithms** (sets of rules) and **data** (information) to learn in a similar way.

**Q: How AI Makes Decisions:**

AI doesn’t just memorize things. It learns rules from the data, called algorithms. Algorithms are like step-by-step instructions AI follows to make decisions.

For instance:

If AI knows that spam emails often use certain words (like “free” or “urgent”), it will look for those words in new emails to decide if they are spam.

(ie.) AI first LEARNS from data, then use that data to THINK. So, if anyone ask you what is the THINKING mechanism of AI? First AI LEARNS anything from the data, then it uses those data to THINK.

**Q: What is Intelligent Agent?**

An Intelligent Agent is a program or system that can i) perceive its environment, ii) make decisions, and iii) take actions to achieve specific goals. It's designed to work autonomously, meaning it can act on its own without needing constant human supervision.

**Q: What is Perception in Agent?**

[ When we say an intelligent agent "perceives its environment", we mean that it gathers information about the world around it, using sensors or data inputs. This is how the agent understands what's happening so it can make decisions. In simple terms, Perception is like "seeing" or "sensing" what's going on around the agent.

Examples of how an agent perceives its environment:

1. A robot: Uses cameras (to "see") and sensors (to "feel") its surroundings. For example, a vacuum robot uses sensors to detect walls or obstacles so it can avoid them.
2. A self-driving car: Uses cameras, radar, and LIDAR to detect other cars, people, or traffic signs on the road.
3. An email filter: "Perceives" emails by looking at the content of incoming messages. It analyses the text to decide if the email is spam or important.
4. Virtual assistant: Listens to your voice commands. It "hears" what you say (perceives the sound) and then figures out how to respond.

In summary, perception is how the agent gathers information from its environment, which is the first step before it makes decisions or takes actions. The environment could be physical (like objects or sounds) or virtual (like emails or data)]

**Core Components of an Intelligent Agent:**

**1.Perception:**

The agent collects information from its surroundings using sensors or input data.

Example:

A robot might use cameras and sensors to understand its environment. An email filter (which is also an agent "sees" emails as incoming data.

**2.Decision-Making:**

Based on the information it perceives, the agent decides what action to take. It might follow rules, use patterns, or adapt over time. (Follow Algorithms)

Example:

If it detects an email marked as "urgent," it might decide to notify you immediately.

**3.Action:**

Once a decision is made, the agent performs an action. It uses *actuators* or executes commands to affect the environment.

Example: If the intelligent agent is a robot, it might move around objects. If it's a virtual agent, it might sort emails or turn on lights.

**4.Autonomy:**

The agent works on its own, without needing step-by-step instructions from a human. It continuously operates and learns, improving its performance over time.

Example: A self-driving car constantly perceives its environment, makes decisions about speed or direction, and takes actions like steering or braking—all on its own.

**Q: Could an intelligent agent living on your home computer manage your email, coordinate your work and social activities, help plan your vacations…… even watch your house while you take those well-planned vacations?**

Yes, an intelligent agent on your home computer could potentially do all those things! Let’s break it down step by step in the easiest way possible. It's like having a smart assistant living on your computer.

**How an Intelligent Agent Could Help You:**

**Manage Your Email:**

The intelligent agent can be trained to sort your emails, filter out spam, and even respond to certain messages.

For example, it could automatically put important work emails in a special folder and junk emails in the spam folder. It can even draft replies for you based on previous responses you've made.

**Coordinate Your Work and Social Activities:**

The agent could manage your calendar by keeping track of meetings, deadlines, and social events.

It could send you reminders, schedule meetings for you, and even avoid scheduling work events during your social activities. Example: “Hey, you have a friend’s birthday on Friday evening, so I won’t schedule any work meetings then.”

**Help Plan Your Vacations:**

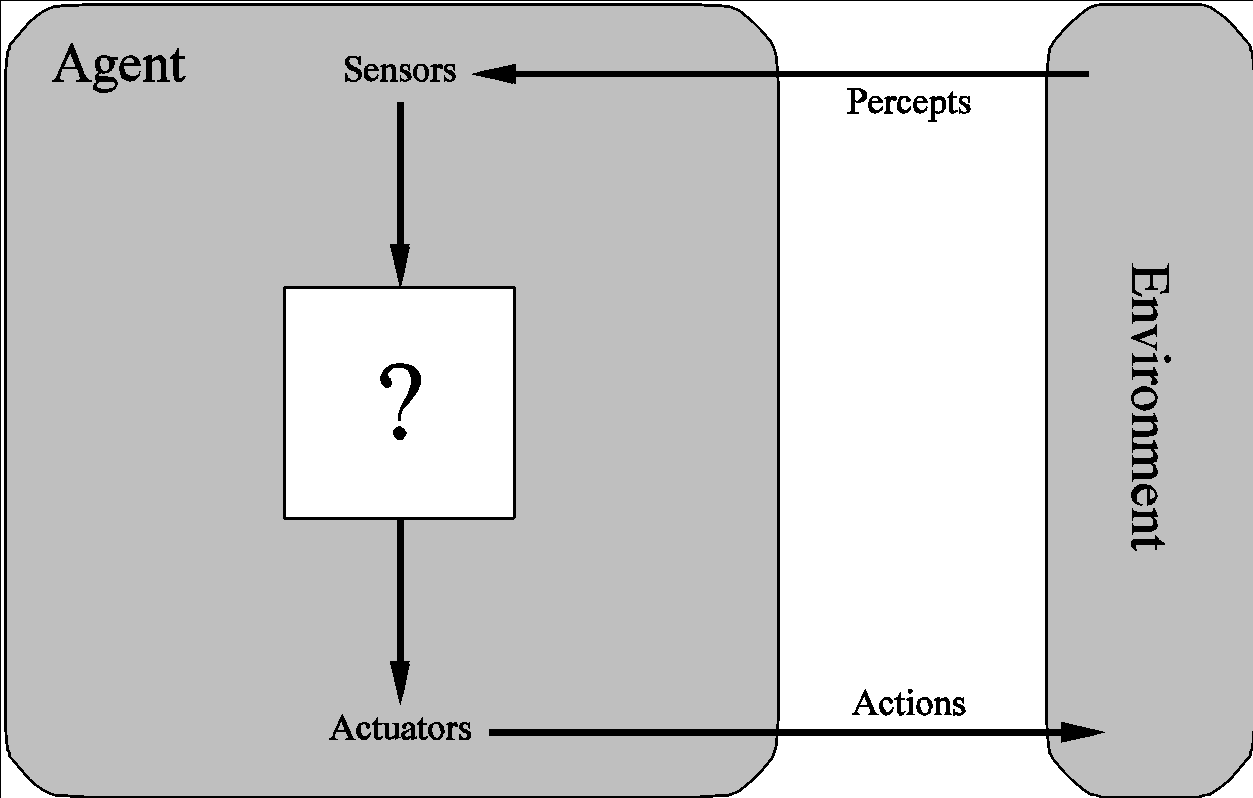
The intelligent agent could research flights, hotels, and activities based on your preferences. It can help find the best deals and even create a detailed itinerary for you.

Example: If it knows you like beach vacations, it might suggest places like Hawaii or Bali and even book a room at a hotel near the beach.

**Watch Your House While You’re on Vacation:**

It could monitor your smart home devices like security cameras, locks, and lights while you're away. If something unusual happens (like someone trying to break in), it can alert you or even call the police. It could also turn your lights on and off to make it look like someone is home.

**Q: How do you design an intelligent agent?**



**Properties of Agent**

1. Autonomy
2. Perception
3. Learning(optional)
4. Decision
5. Action
6. Reactivity
7. Rationality- The ability to make decisions based on achieving a goal or objective. An intelligent agent chooses the best possible action based on the information it has, with the goal of maximizing its performance or achieving its tasks.
8. Communication(optional)- can communicate with other agents or humans.

**Q: What do you mean by sensors/percepts and actuators/actions?**

Example-

* Humans
  + Sensors: Eyes (vision), ears (hearing), skin (touch), tongue (gustation), nose (olfaction), neuromuscular system (proprioception)
  + Percepts:
    - At the lowest level – electrical signals from these sensors
    - After preprocessing – objects in the visual field (location, textures, colors, …), auditory streams (pitch, loudness, direction), …
  + Actuators: limbs, digits, eyes, tongue, …
  + Actions: lift a finger, turn left, walk, run, carry an object

**Automated taxi driving system**

**Percepts**: Video, sonar, speedometer, odometer, engine sensors, keyboard input, microphone, GPS, …

**Actions**: Steer, accelerate, brake, horn, speak/display, …

**Goals (Rationality)**: Maintain safety, reach destination, maximize profits (fuel, tire wear), obey laws, provide passenger comfort, …

**Environment**: Urban streets, freeways, traffic, pedestrians, weather, customers, …

Let’s summarize-

**Autonomy**: Drives without human assistance.

**Perception**: Uses sensors to detect the road, traffic, and obstacles.

**Action**: Makes decisions about driving actions like turning, braking, and accelerating.

**Rationality (Goals)**: Chooses the safest and most efficient route to the destination.

**Learning**: May improve driving decisions over time through experience.

**Proactiveness**: Anticipates traffic and route conditions to make proactive decisions.

**Reactivity**: Quickly responds to unexpected events, such as a pedestrian crossing or a car cutting in.

**Communication**: Communicates with other vehicles or systems, and possibly with passengers

**Q: What is a Rational Agent?**

A rational agent is something (a machine or a system) that makes decisions or takes actions based on the information it receives and tries to **maximize its performance**.

It uses the data it gathers and the knowledge it already has to make the best decisions.

**How Does it Work?**

The agent gets inputs (called percepts) from its environment. For example, an automated taxi might see that (sensor camera) a traffic light is red.

Based on this input, the agent uses its knowledge to decide what to do (based on algorithm set). For example, it knows that red lights mean "stop," so it stops.

**Q: What is Rationality?**

Rationality means that the agent should gather information if it doesn’t know something. It shouldn’t ignore information ("rational ignorance"). For example, if the taxi doesn’t know a road’s speed limit, it should find out from traffic signs or maps, not just guess.

Performance Measure: To know how well the agent is doing its task, we need a way to measure its performance. This could be how fast the taxi reaches its destination or how safely it drives.

Types of performance measures: false alarm (false positive) and false dismissal (false negative) rates, speed, resources required (Fuels, Energy etc), effect on environment, etc.

**What is Rational Ignorance?**

["Rational ignorance" refers to the idea that it is sometimes logical to ignore certain information because the cost of acquiring that information (in terms of time, effort, or resources) might be higher than the benefit gained from knowing it]

\*\*\*However, in the context of rational agents, rational ignorance means that the agent should not choose to ignore information. Instead, it should always gather relevant information to make the best decisions possible. If the agent doesn’t know something important (like the speed limit in a certain area for an autonomous car), it shouldn’t ignore that fact. Instead, it should find a way to gather that information (like using traffic signs or maps).

Imagine you’re driving a car (like a rational agent). If you don't know what the road sign ahead says, rational ignorance would be just ignoring the sign and continuing to drive without finding out what it means. This is bad for a rational agent. Instead, you should slow down, read the sign, and then make a decision based on that new information.

In short: a rational agent should never ignore useful information if it will help make better decisions.

**Q: What is Autonomy?**

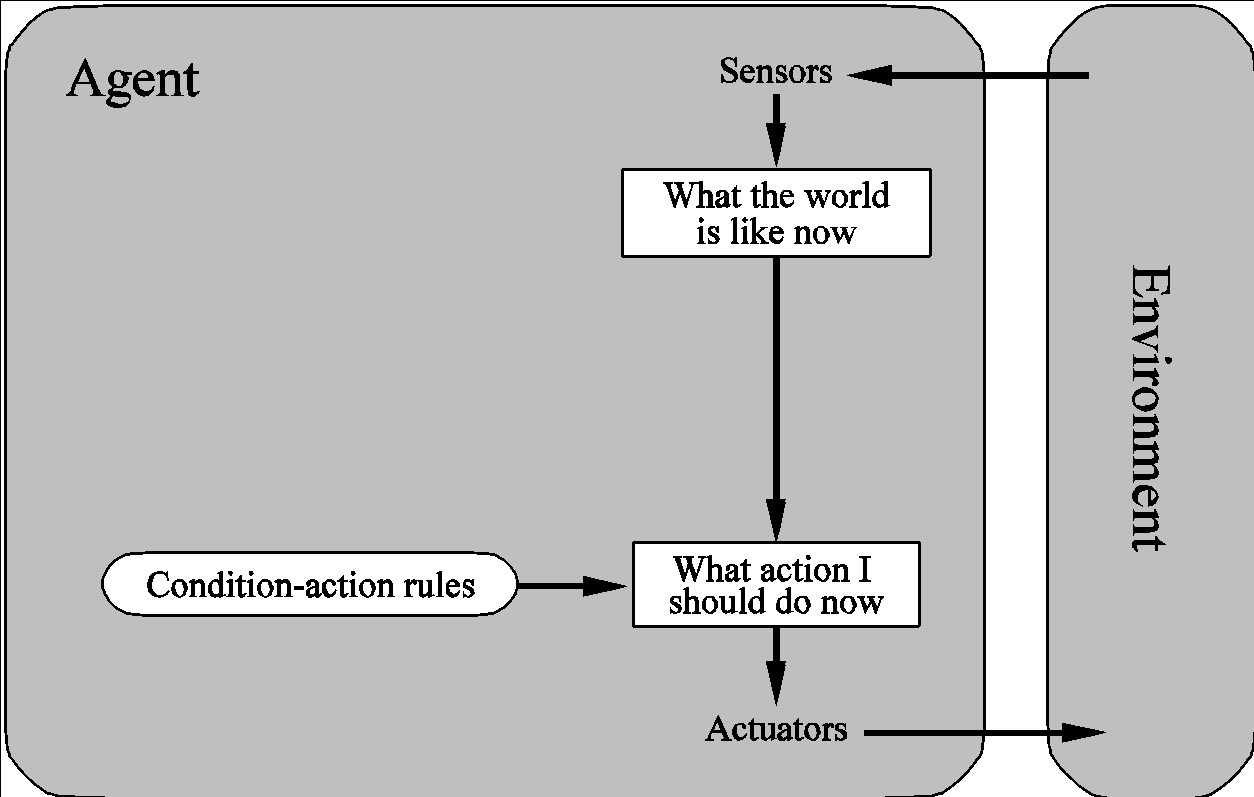
In Artificial Intelligence (AI), autonomy refers to the ability of an intelligent agent or system to make decisions and act independently without needing constant human intervention or control. In simpler terms, an autonomous AI can perceive its environment, make choices, and take actions based on those choices to achieve a goal, all on its own.

**Types of AGENTS**

**1.Table-driven agents:**

A table-driven agent stores all possible conditions and corresponding actions in a lookup table. It simply refers to the table when it senses the environment and looks up what action to take for each condition.

use a percept sequence/action table in memory to find the next action. They are implemented by a (large) **lookup table**



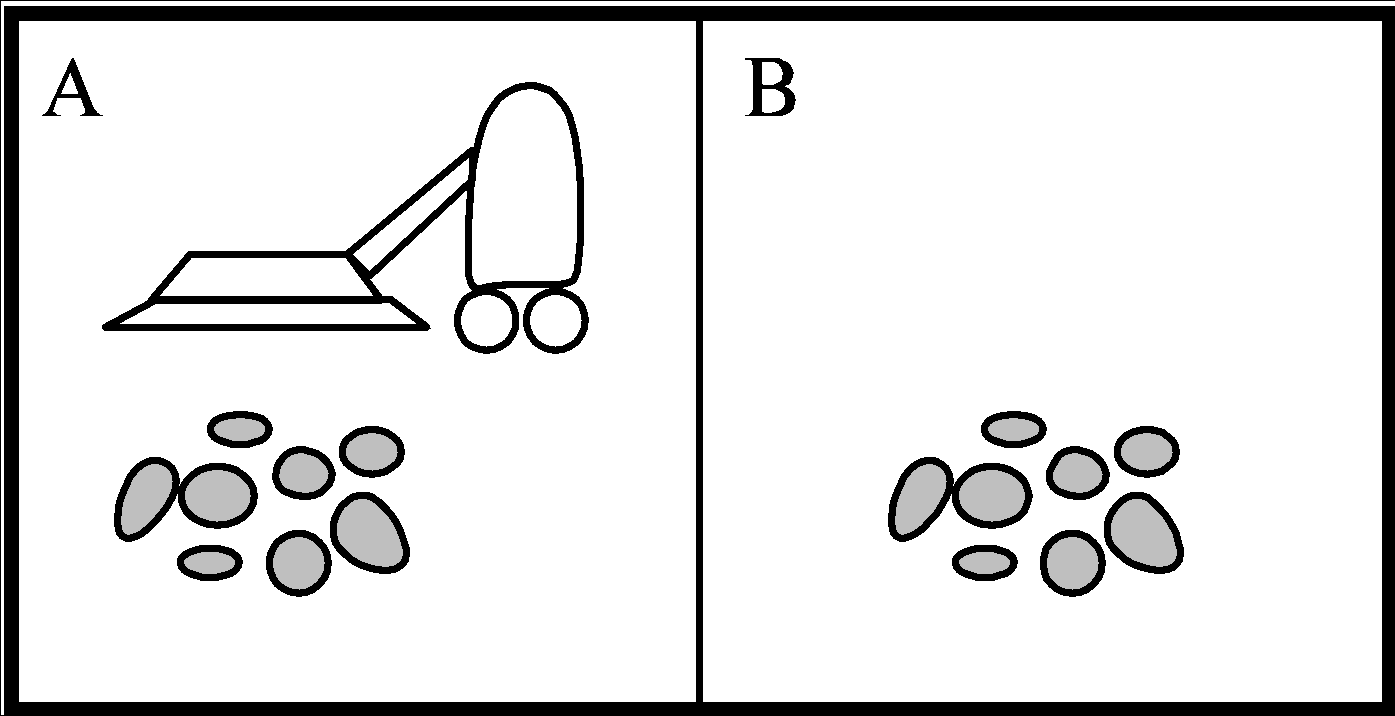
* **Problems** 
  + Too big to generate and to store (Chess has about 10120 states, for example)
  + No knowledge of non-perceptual parts of the current state
  + Not adaptive to changes in the environment; requires entire table to be updated if changes occur
  + Looping: Can’t make actions conditional on previous actions/states

**How it works**: The agent senses the environment, finds the current condition in its table, and performs the corresponding action.

**Limitations**: It becomes impractical as the number of possible conditions increases because the table would become too large.

**Real-Life Example**: A simple chess-playing AI that has a predefined table of moves for every board configuration could be a table-driven agent. However, such systems are rare because it’s \*nearly impossible to store all chess board configurations.

**Example: Vacuum Cleaner**



**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, Dirty] Suck

**2.Simple reflex Agents**

* + are based on condition-action rules, implemented with an appropriate production system. They are stateless devices which do not have memory of past world states.

**Concept**: A simple reflex agent acts only on the current situation, ignoring the history or consequences of its actions. It makes decisions based on a set of condition-action rules (if-then rules) and reacts accordingly.

* **Problems** 
  + Still usually too big to generate and to store
  + Still no knowledge of non-perceptual parts of state
  + Still not adaptive to changes in the environment; requires collection of rules to be updated if changes occur
  + Still can’t make actions conditional on previous state

**How it works**: The agent senses the environment and acts directly based on predefined rules without considering future outcomes or maintaining internal states.

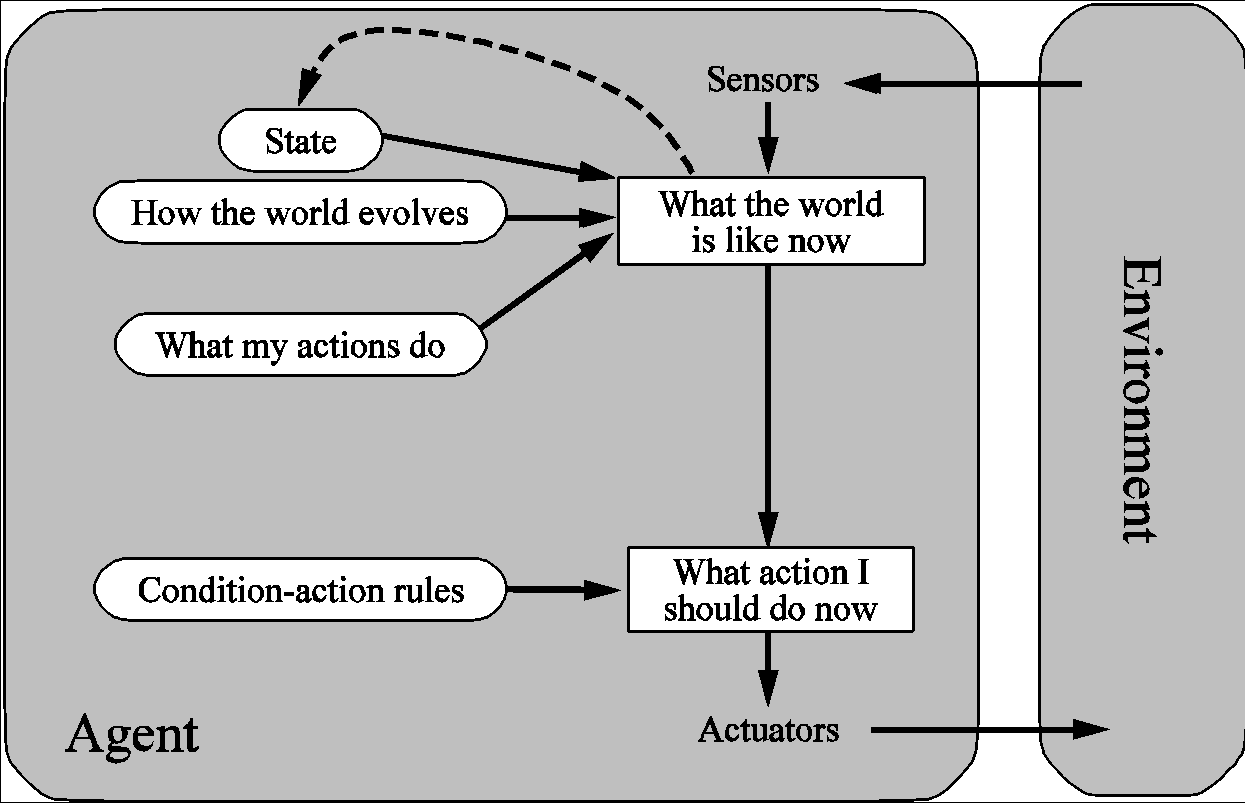
**Limitations**: These agents do not learn or adapt to changes in the environment and work only in fully observable environments.

**Real-Life Example**: A thermostat is an example of a simple reflex agent. It checks the current temperature and either turns the heating or cooling system on or off *based on preset temperature thresholds.*

**3.Model-Based Agents**

**Concept**: A model-based agent keeps track of the world by maintaining an internal model (a representation of the world). It uses this model to make better decisions, especially in environments that aren’t fully observable.

* Encode “internal state” of the world to remember the past as contained in earlier percepts.
* Needed because sensors do not usually give the entire state of the world at each input, so perception of the environment is captured over time. “State” is used to encode different "world states" that generate the same immediate percept.
* Requires ability to represent change in the world; one possibility is to represent just the latest state, but then can’t reason about hypothetical courses of action



**How it works**: The agent maintains an internal state that represents what it knows about the world. It updates this state based on what it senses and the actions it takes (evolves). This internal model helps the agent predict how its actions will affect the future.

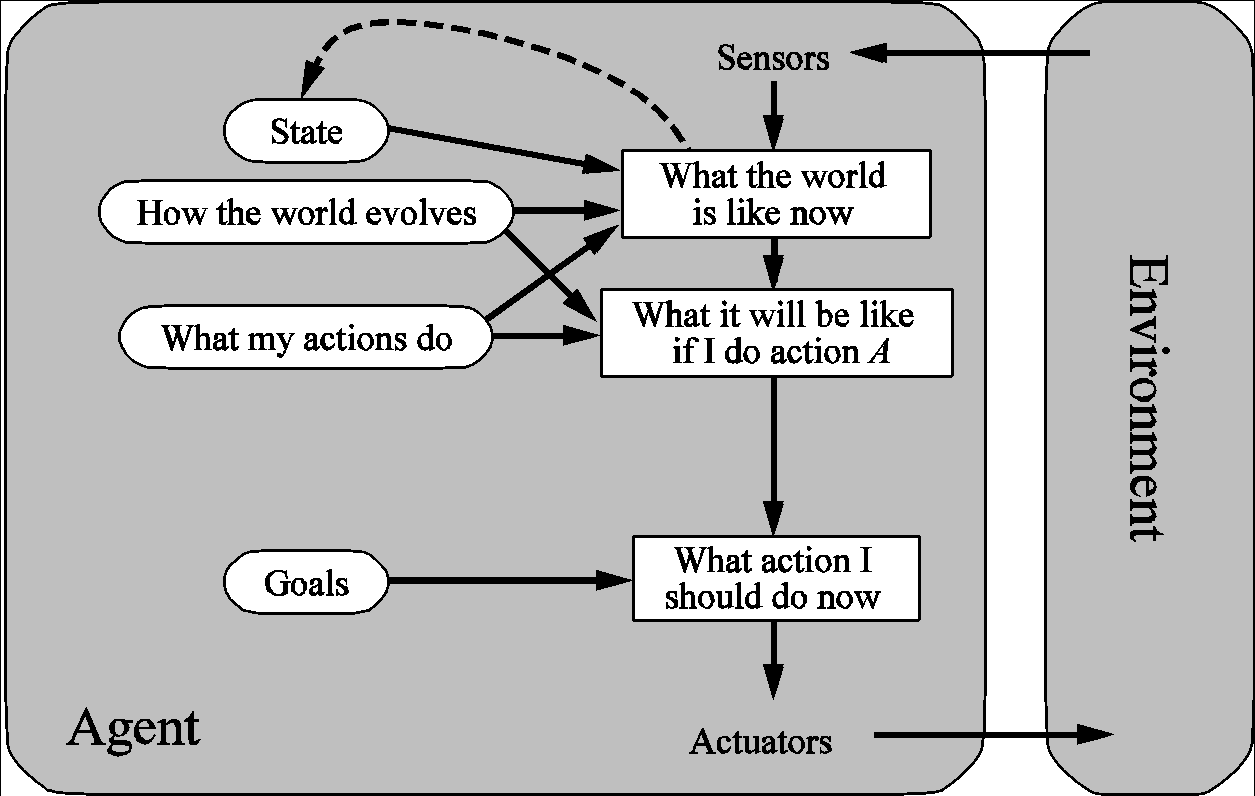
**Real-Life Example**: A *self-driving car* is a model-based agent. It maintains an internal map of the surrounding environment, including other vehicles, pedestrians, and obstacles, and uses this to navigate and make decisions.

**4.Goal Based Agents**

* + are agents that, in addition to state information, have goal information that describes desirable situations. Agents of this kind take future events into consideration.

**Concept**: A goal-based agent doesn’t just react to its environment; it also tries to achieve specific goals. It decides what actions to take based on which ones will move it closer to achieving its goal.

* Choose actions to achieve a (given or computed) goal.
* A goal is a description of a desirable situation.
* Keeping track of the current state is often not enough − need to add goals to decide which situations are good
* **Deliberative** instead of **reactive**.
* May have to consider long sequences of possible actions before deciding if goal is achieved – involves consideration of the future, *“what will happen if I do...?”*



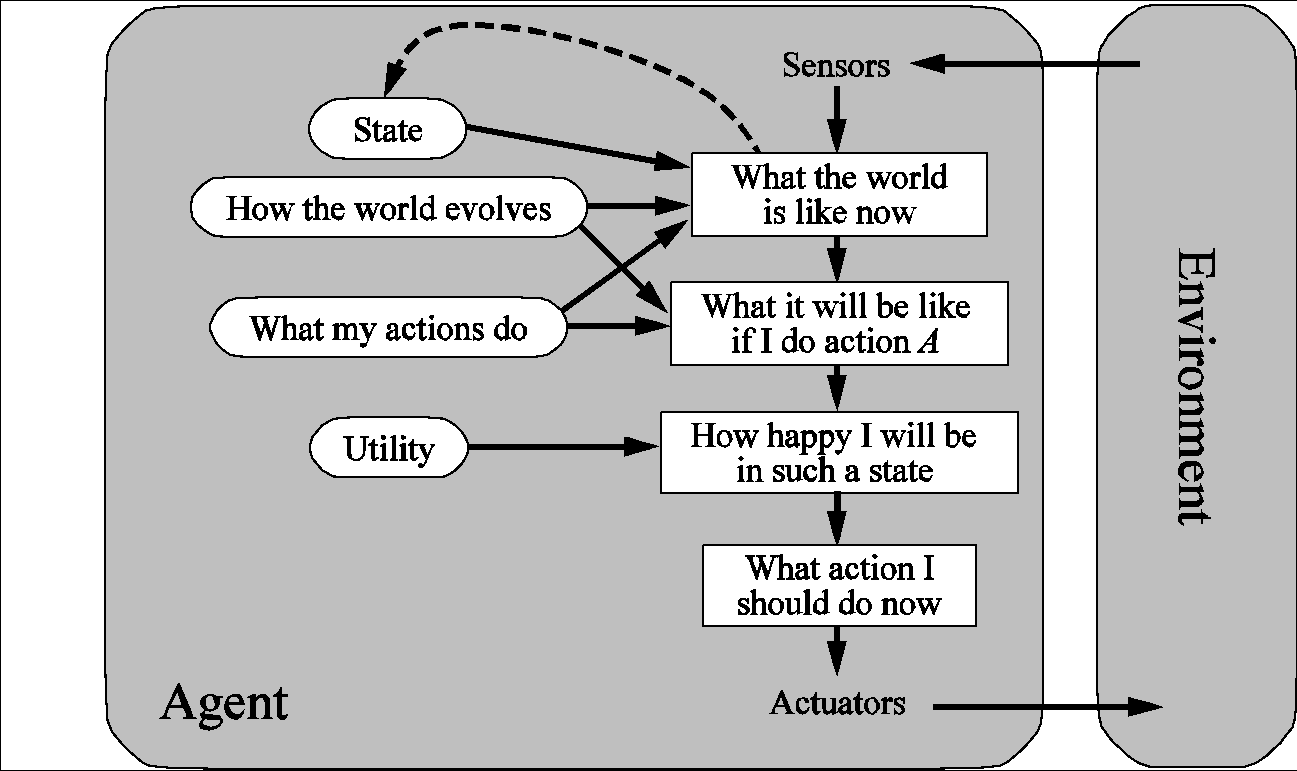
**How it works**: The agent looks ahead and evaluates different sequences of actions, selecting the one that helps it achieve its goal most efficiently.

**Real-Life Example**: Google Maps or any GPS navigation system can be considered a goal-based agent. It helps you reach a specific destination (goal) by suggesting the best route based on current conditions.

**5.Utility-based Agents**

**Concept**: A utility-based agent is similar to a goal-based agent but with an added layer of evaluating how desirable different outcomes are. It not only aims to achieve goals but also aims to maximize the overall quality of those goals, balancing trade-offs.

* When there are multiple possible alternatives, how to decide which one is best?
* A goal specifies a crude distinction between a happy and unhappy state, but often need a more general performance measure that describes “degree of happiness.”
* Utility function **U: State → Reals** indicating a measure of success or happiness when at a given state.
* Allows decisions comparing choice between conflicting goals, and choice between likelihood of success and importance of goal (if achievement is uncertain).

****

**How it works**: This agent evaluates different actions based on a utility function that measures the expected "happiness" or "success" of each possible outcome. It aims to maximize utility over time.

**Real-Life Example**: A ride-hailing service like Uber uses utility-based agents. It tries to maximize profits (utility) by balancing driver availability, rider demand, and pricing while ensuring both riders and drivers are satisfied.

**Summary Table of Agents**

A screenshot of a computer

Description automatically generated

**Properties of Environments**

**1. Observable vs. Partially Observable**

* **Fully Observable**: The agent has complete access to all information about the environment.
* **Partially Observable**: The agent has limited information and may need to infer missing details.
* **Fully observable/Partially observable.**
  + If an agent’s sensors give it access to the complete state of the environment needed to choose an action, the environment is **fully observable**.
  + Such environments are convenient, since the agent is freed from the task of keeping track of the changes in the environment.

**2. Deterministic vs. Stochastic**

* **Deterministic**: The next state of the environment is entirely determined by the current state and the agent’s action.
* **Stochastic**: There is some randomness in how actions affect the environment
* **Deterministic/Stochastic**.
  + An environment is **deterministic** if the next state of the environment is completely determined by the current state of the environment and the action of the agent; in a **stochastic** environment, there are multiple, unpredictable outcomes
  + In a fully observable, deterministic environment, the agent need not deal with uncertainty.

**3. Episodic vs. Sequential**

* **Episodic**: Each action is independent of previous actions.
* **Sequential**: Actions build on each other and affect future decisions
* **Episodic/Sequential**.
  + An **episodic** environment means that subsequent episodes do not depend on what actions occurred in previous episodes.
  + In a **sequential** environment, the agent engages in a series of connected episodes.
  + Such environments do not require the agent to plan ahead.

**4. Static vs. Dynamic**

* **Static**: The environment doesn’t change while the agent is thinking.
* **Dynamic**: The environment changes independently of the agent’s actions.
* **Static/Dynamic.** 
  + A **static** environment does not change while the agent is thinking.
  + The passage of time as an agent deliberates is irrelevant.
  + The agent doesn’t need to observe the world during deliberation.

**5. Discrete vs. Continuous**

* **Discrete**: The environment has a finite set of distinct states.
* **Continuous**: The environment has a range of states, often requiring more precise measurements.
* **Discrete/Continuous.**
  + If the number of distinct percepts and actions is limited, the environment is **discrete**, otherwise it is **continuous**.

**6. Single vs. Multi Agents**

* **Single-Agent**: Only one agent operates in the environment, and it does not need to consider the actions of other agents. (Crossword puzzles, image analysis, and medical diagnosis involve only one agent solving the task)
* **Multi-Agent**: Multiple agents operate in the environment, and they may interact or compete. This adds complexity, as agents might need to consider the actions and potential moves of others.

(In poker, multiple players (agents) interact and try to win. Similarly, in a taxi-driving environment, multiple taxis or vehicles (agents) might need to consider each other’s movements on the road)

* **Single agent/multi-agent.**
  + If the environment contains other intelligent agents, the agent needs to be concerned about strategic, game-theoretic aspects of the environment (for either cooperative *or* competitive agents)
  + Most engineering environments don’t have multi-agent properties, whereas most social and economic systems get their complexity from the interactions of (more or less) rational agents.

**Examples of Properties in Specific Scenarios:**

1. **Crossword Puzzle**:
   * **Observable**: Fully observable—all clues and positions are visible.
   * **Deterministic**: Deterministic—the placement of each word has a known effect.
   * **Sequential**: Sequential— choices impact other spaces in the puzzle.
   * **Static**: Static—the puzzle doesn’t change while solving it.
   * **Discrete**: Discrete—words and letters are placed in fixed squares.
   * **Single Agents**:
2. **Poker Card**:
   * **Observable**: Partially observable—only your cards and common cards are visible; other players’ hands are hidden.
   * **Stochastic**: Stochastic— randomness in card dealing.
   * **Sequential**: Sequential— each round depends on previous bets and actions.
   * **Dynamic**: Dynamic— actions of players continuously affect the game.
   * **Discrete**: Discrete— limited set of actions and outcomes.
   * **Multi Agents**:
3. **Taxi Driving**:
   * **Observable**: Partially observable—may have sensors but can’t see all at once.
   * **Stochastic**: Stochastic—traffic, pedestrians, and unpredictable conditions introduce randomness.
   * **Sequential**: Sequential— decisions (e.g., turns, stops) build on each other.
   * **Dynamic**: Dynamic—traffic conditions and environment change over time.
   * **Continuous**: Continuous— movement through roads and positioning.
   * **Multi Agents**:
4. **Image Analysis**:
   * **Observable**: Fully observable— image is entirely visible.
   * **Deterministic**: Deterministic— analyzing the image produces consistent results.
   * **Episodic**: Episodic— each image is analyzed independently.
   * **Static**: Static—the image doesn’t change during analysis.
   * **Continuous**: Continuous—requires detailed processing of visual data.
   * **Single Agents**:
5. **Medical Diagnosis**:
   * **Observable**: Partially observable— some information might be hidden or uncertain.
   * **Stochastic**: Stochastic— diseases and symptoms can vary unpredictably.
   * **Sequential**: Sequential— tests and symptoms build to form a diagnosis.
   * **Dynamic**: Dynamic— patient health can change over time.
   * **Discrete**: Discrete— diagnosis has distinct outcomes (disease classifications).
   * **Single Agents**:

A table with black text

Description automatically generated

**PART-2 (Search Techniques)**

**Q: What is Searching in AI?**

In Artificial Intelligence (AI), searching refers to the process of finding a solution or answer to a problem by exploring different possibilities. Think of it as a way of solving a puzzle by checking all the possible moves you can make until you find the correct one.

For example:

If you are trying to solve a maze, searching means exploring different paths (right, left, straight) until you find the exit.

If you're looking for information in a large database, AI uses searching to find the exact data you're asking for.

**Q: Why Searching in AI?**

Searching is **important in AI** because: **PDE**

1. **Problem Solving**: Many problems in AI (like finding the best route for a self-driving car, winning a game of chess, or diagnosing a disease) can be framed as search problems. AI explores various possible solutions to find the best one.
2. **Decision Making**: AI systems need to make decisions, like which move to make in a game or what recommendation to give to a user. Searching helps in exploring different options and picking the best one.
3. **Efficiency**: Searching helps AI to efficiently find the most suitable solution or answer. Instead of randomly trying everything, smart search algorithms can narrow down possibilities, making the process faster.

Examples of Searching in AI:

**Navigation**: A GPS system uses search to find the shortest route from your home to a destination.

**Game Playing**: A chess-playing AI uses search to figure out the best moves by exploring possible future moves.

**Robotics**: A robot might search for the best way to move around obstacles and reach its target.

In summary, searching in AI is like exploring different choices or options to solve a problem, make a decision, or find something specific. It is one of the key techniques AI uses to solve a wide range of tasks.

**Tic-Tac-Toe and AI Search**

Tic-tac-toe is a simple 3x3 grid game where two players (one as "X" and the other as "O") take turns placing their marks. The goal is to get three of your marks in a row (horizontally, vertically, or diagonally) while trying to block the opponent from doing the same.

**Algorithm of Tic-Tac-Toe:**

1. Initialize a 3x3 grid (empty).
2. Players take turns to place their marks in an empty cell.
3. After every move, check if:
4. Any player has three marks in a row (win condition).
5. If no cells are left and there’s no winner, it’s a tie.
6. The game ends if a player wins or there’s a tie.

**\*AI Search and Tic-Tac-Toe:**

AI uses search algorithms to play the game optimally by exploring all possible moves to find the best one.

One common approach is: ***Minimax Algorithm***:

This is a type of AI search that looks at every possible move the player can make, then calculates the possible responses of the opponent. *The goal is to minimize the possible loss in the worst-case scenario and maximize the chance of winning*.

1. Maximizing player (X) tries to win.
2. Minimizing player (O) tries to block or minimize the maximizing player’s score.

By looking ahead at possible moves, the AI picks the *most optimal one*, considering how the opponent might respond.

In short, AI uses search to explore all possible board states and decides the best move in Tic-Tac-Toe.

**Q: What are the types of Search Techniques in AI?**

In Artificial Intelligence (AI), search techniques are used to solve problems by exploring various possibilities until the best or optimal solution is found.

(e.g., finding the best move in a game, solving a puzzle, or finding the shortest route).

**Classifications**

Search techniques in AI can be broadly classified into two main categories:

Uninformed Search (Blind Search)

Informed Search (Heuristic Search)

A diagram of a search engine

Description automatically generated

**1. Uninformed Search (Blind Search)**

Uninformed search techniques *do not have any additional information* about the states beyond the problem definition. They explore the search space \*without guidance on the most efficient path.

**Types of Uninformed Search Techniques: BFS and DFS**

**Breadth-First Search (BFS):**

How it works: Explores all nodes at the current level before moving to the next level.

Example: Imagine you're searching for a friend's house. You explore all streets (options) nearby before moving to farther streets.

Advantages: It finds the shortest path to the solution (if all costs are equal).

Disadvantages: Requires a lot of memory and time for large problems.

**Depth-First Search (DFS):**

How it works: Explores as far down one branch as possible before backtracking and exploring other branches.

Example: If you go through each street fully (deep into a street) before moving to the next.

Advantages: Uses less memory compared to BFS.

Disadvantages: It can get stuck going down an infinite path if no limit is placed. (but when the limit is placed, this search will be called as **DLS**)

**Uniform-Cost Search (UCS):**

How it works: It expands the node with the lowest cost first, i.e., it *finds the least expensive path* in terms of cost.

Example: If you are choosing paths based on cost, such as fuel efficiency or distance.

Advantages: Always finds the least-cost solution.

Disadvantages: Can be slow for large problems with many paths.

**Depth-Limited Search (DLS):**

How it works: It *limits the depth* to which the search can go. \*If the solution is beyond this depth, it will not find it.

Example: Exploring only a certain number of streets deep into a neighbourhood.

Advantages: Avoids infinite loops.

Disadvantages: If the solution is beyond the set limit, it will be missed.

**\*Iterative Deepening Search (IDS):**

How it works: Combines the benefits of BFS and DFS by *performing DFS with increasing depth* limits until the solution is found.

Example: First exploring within one block, then two, then three, and so on until you find the house.

Advantages: Finds the solution efficiently without large memory requirements.

Disadvantages: Repeated exploration of the same nodes can cause inefficiency in terms of time.

**Q: What is heuristic?**

In the simplest terms, heuristic refers to a practical method or a "rule of thumb" that helps solve problems more quickly byusingeducated guesses or strategies, *instead of checking every possible solution*.In AI, a heuristic is a guiding function that helps an algorithm make decisions more efficiently. It provides an estimate or guess about how close you are to a solution or goal, helping the AI system make better decisions without having to explore every possible option.

**Q: What is heuristic function?**

A heuristic function is a function *f(n)* that gives an estimation on the “cost” of getting from node *n* to the goal state – so that the node with the least cost among all possible choices can be selected for expansion first.

Three approaches to defining *f*:

* 1. *f* measures the value of the current state (its “goodness”)
  2. *f* measures the estimated cost of getting to the goal from the current state:
     + 1. *f(n)* = *h(n)* where *h(n)* = an estimate of the cost to get from *n* to a goal
  3. *f* measures the estimated cost of getting to the goal state from the *current state* and the cost of the existing path to it. Often, in this case, we decompose *f*:
     + 1. *f(n)* = *g(n)* + *h(n)* where *g(n)* = the cost to get to *n* (from initial state)

**2. Informed Search (Heuristic Search)**

Informed search techniques *use additional information (heuristics)* to guide the search toward the solution more efficiently. A heuristic is a rule of thumb that helps estimate the best direction to move based on the available data.

**Types of Informed Search Techniques:**

**Greedy Best-First Search:**

**How it works**: It expands the node *that appears to be closest to the goal* according to a heuristic function (e.g., straight-line distance).

* + f(n) = h(n), choose the one that is nearest to the final state among all possible choices

**Example**: If you're trying to reach a destination, you choose the road that looks like it will get you closest to your goal.

**Advantages**: Faster than uninformed searches as it uses a heuristic.

**Disadvantages**: May not find the optimal solution because it only considers the heuristic and ignores actual path costs.

**A Search\*:**

How it works: Combines both the actual cost to reach a node and the heuristic cost to estimate the total cost. It selects nodes based on the formula:

* + f(n) = g(n) + h(n)
  + Best first search using an “admissible” heuristic function f that considers the current cost g
  + Always returns the optimal solution path

where g(n) is the cost to reach the node, and h(n) is the heuristic cost (estimated cost to reach the goal).

**Example**: Similar to Greedy Best-First Search, but it also considers the actual distance covered so far, ensuring the path found is the shortest.

**Advantages**: Guaranteed to find the optimal path if the heuristic is admissible (**never overestimates**).

**Disadvantages**: Requires significant memory as it stores all possible nodes.

\*\*\*UCS[g(n)] is- i.optimal, ii.complete but iii.not efficient. On the other hand, Greedy[h(n)] is i.efficient but ii.not optimal and iii. not complete. For these limitations, A\* was developed; which is- i.optimal, ii.complete, and iii.efficient.

**Hill Climbing Search:**

**How it works**: Moves step by step towards a solution by selecting the best immediate option (local maximum) based on the heuristic function.

Example: Like climbing a hill by always taking a step that goes higher. You stop when you can't go higher.

Advantages: Simple and memory efficient.

Disadvantages: Can get stuck at a local maximum (a peak that’s lower than the global peak), failing to find the optimal solution.

**Genetic Algorithms:**

**How it works:** Uses principles of natural selection and genetics (mutation, crossover) to search for solutions. It evolves a population of solutions over time.

**Example:** Generating random paths to a goal, crossing the best ones, and mutating parts of the solution until a good path is found.

**Advantages:** Can find solutions to very complex problems that other methods might struggle with.

**Disadvantages:** Computationally expensive and requires tuning for optimal performance.

**3. Other Advanced Search Techniques:**

**Beam Search:** Like greedy best-first search but only keeps a fixed number of nodes at each level to explore.

**Bidirectional Search:** Simultaneously searches forward from the start and backward from the goal, meeting in the middle.

**Q: What are the differences between A\* and IDA\* search?**

Imagine you want to explore a large map to find the shortest path from your workplace to home.

A\*: You keep track of all possible routes in memory and explore the shortest one, but your memory might fill up if the map is too large. (not Rational)

IDA\*: You explore one route at a time with a cost limit, increasing that limit in stages. You might take longer, but you don't run out of memory. (Rational)

A\* is faster but memory hungry. IDA\* is slower but uses far less memory.

A screenshot of a computer

Description automatically generated

**Questions Solving**

A paper with text and images

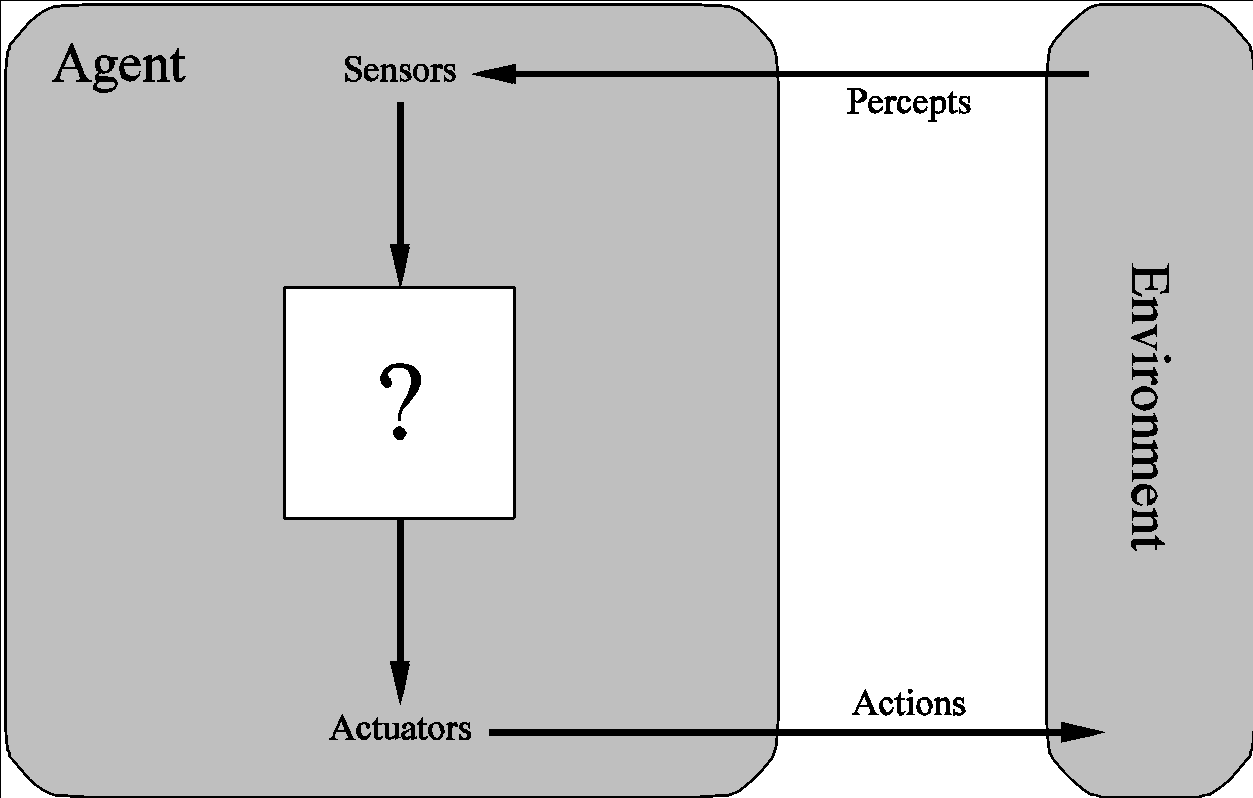
Description automatically generated

**1.How do you design an intelligent agent? Write the properties**

Ans:

**What is Intelligent Agent?**

An Intelligent Agent is a program or system that can i) perceive its environment, ii) make decisions, and iii) take actions to achieve specific goals. It's designed to work autonomously, meaning it can act on its own without needing constant human supervision.



**Properties of Agent**

1. **Perception**
2. Learning(optional)
3. **Decision**
4. **Action**
5. Autonomy
6. Reactivity
7. Rationality- The ability to make decisions based on achieving a goal or objective. An intelligent agent chooses the best possible action based on the information it has, with the goal of maximizing its performance or achieving its tasks.
8. Communication(optional)- can communicate with other agents or humans.

**2. What is your understanding of ‘rational ignorance’? Provide an example.**

**Answer:**

**What is Rational Ignorance?**

["Rational ignorance" refers to the idea that it is sometimes logical to ignore certain information because the cost of acquiring that information (in terms of time, effort, or resources) might be higher than the benefit gained from knowing it.]

In the context of rational agents, rational ignorance means that the agent should not choose to ignore information. Instead, it should always gather relevant information to make the best decisions possible. If the agent doesn’t know something important (like the speed limit in a certain area for an autonomous car), it shouldn’t ignore that fact. Instead, it should find a way to gather that information (like using traffic signs or maps).

Example:

Imagine you’re driving a car (like a rational agent). If you don't know what the road sign ahead says, rational ignorance would be just ignoring the sign and continuing to drive without finding out what it means. This is bad for a rational agent. Instead, you should slow down, read the sign, and then make a decision based on that new information.

In short: a rational agent should never ignore useful information if it will help make better decisions.

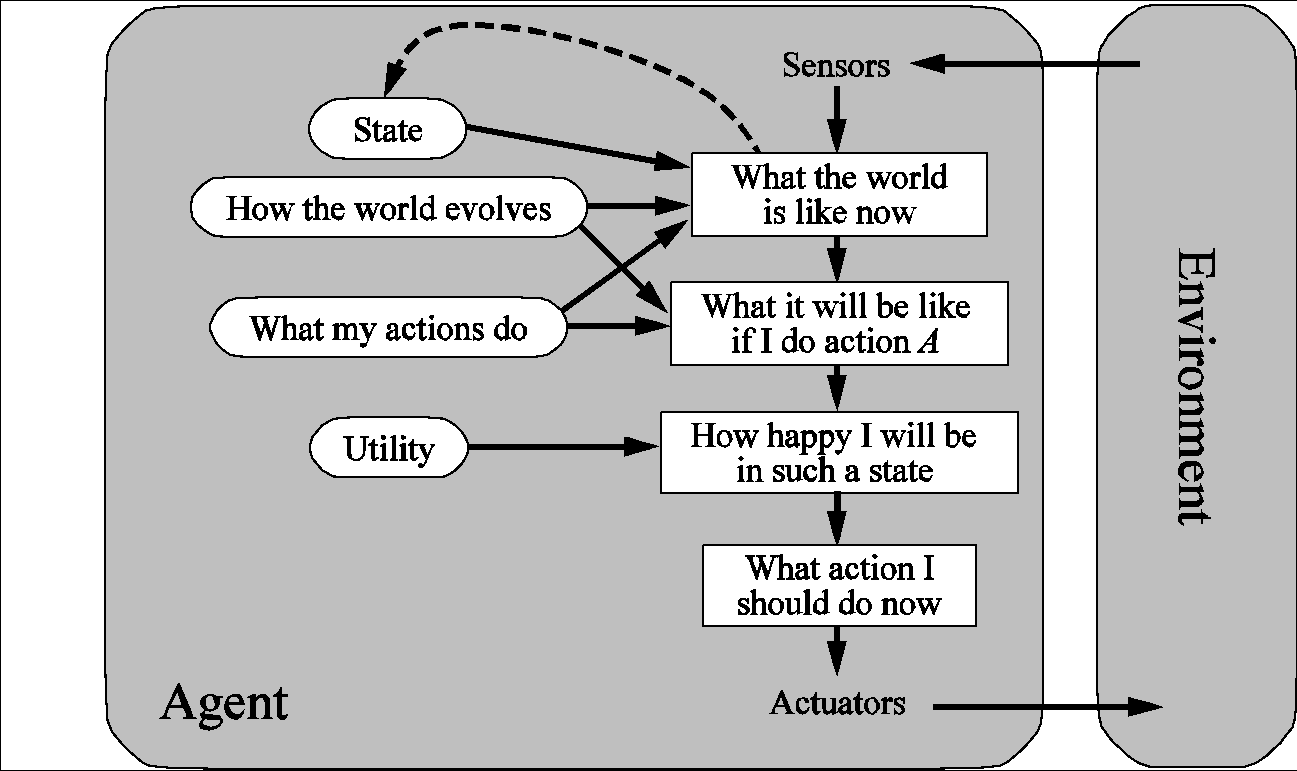
**3.Describe the architecture of utility-based agents.**

**Answer:**

**Utility-based Agents**

**Concept**: A utility-based agent is similar to a goal-based agent but it is not only aims to achieve goals but also aims to maximize the overall quality of those goals, balancing trade-offs.

* When there are multiple possible alternatives, how to decide which one is best?
* A goal specifies a crude distinction between a happy and unhappy state, but often need a more general performance measure that describes “degree of happiness.”
* Utility function **U: State → Reals** indicating a measure of success or happiness when at a given state.
* Allows decisions comparing choice between conflicting goals, and choice between likelihood of success and importance of goal (if achievement is uncertain).

****

**How it works**: This agent evaluates different actions based on a utility function(**U: State → Reals)** that measures the expected "happiness" or "success" of each possible outcome. It aims to maximize utility over time.

**Real-Life** **Example**: A ride-hailing service like Uber uses utility-based agents. It tries to maximize profits (utility) by balancing driver availability, rider demand, and pricing while ensuring both riders and drivers are satisfied.

**4.Write down the properties of the environment and describe them for Crossword, POKER, Taxi driving, Image analysis, and medical diagnosis.**

**Answer:**

**Properties of Environments**

**1. Observable vs. Partially Observable**

* **Fully Observable**: The agent has complete access to all information about the environment.
* **Partially Observable**: The agent has limited information and may need to infer missing details.
* **Fully observable/Partially observable.**
  + If an agent’s sensors give it access to the complete state of the environment needed to choose an action, the environment is **fully observable**.
  + Such environments are convenient, since the agent is freed from the task of keeping track of the changes in the environment.

**2. Deterministic vs. Stochastic**

* **Deterministic**: The next state of the environment is entirely determined by the current state and the agent’s action.
* **Stochastic**: There is some randomness in how actions affect the environment
* **Deterministic/Stochastic**.
  + An environment is **deterministic** if the next state of the environment is completely determined by the current state of the environment and the action of the agent; in a **stochastic** environment, there are multiple, unpredictable outcomes
  + In a fully observable, deterministic environment, the agent need not deal with uncertainty.

**3. Episodic vs. Sequential**

* **Episodic**: Each action is independent of previous actions.
* **Sequential**: Actions build on each other and affect future decisions
* **Episodic/Sequential**.
  + An **episodic** environment means that subsequent episodes do not depend on what actions occurred in previous episodes.
  + In a **sequential** environment, the agent engages in a series of connected episodes.
  + Such environments do not require the agent to plan ahead.

**4. Static vs. Dynamic**

* **Static**: The environment doesn’t change while the agent is thinking.
* **Dynamic**: The environment changes independently of the agent’s actions.
* **Static/Dynamic.** 
  + A **static** environment does not change while the agent is thinking.
  + The passage of time as an agent deliberates is irrelevant.
  + The agent doesn’t need to observe the world during deliberation.

**5. Discrete vs. Continuous**

* **Discrete**: The environment has a finite set of distinct states.
* **Continuous**: The environment has a range of states, often requiring more precise measurements.
* **Discrete/Continuous.**
  + If the number of distinct percepts and actions is limited, the environment is **discrete**, otherwise it is **continuous**.

**6. Single vs. Multi Agents**

* **Single-Agent**: Only one agent operates in the environment, and it does not need to consider the actions of other agents. (Crossword puzzles, image analysis, and medical diagnosis involve only one agent solving the task)
* **Multi-Agent**: Multiple agents operate in the environment, and they may interact or compete. This adds complexity, as agents might need to consider the actions and potential moves of others.

(In poker, multiple players (agents) interact and try to win. Similarly, in a taxi-driving environment, multiple taxis or vehicles (agents) might need to consider each other’s movements on the road)

* **Single agent/multi-agent.**
  + If the environment contains other intelligent agents, the agent needs to be concerned about strategic, game-theoretic aspects of the environment (for either cooperative *or* competitive agents)
  + Most engineering environments don’t have multi-agent properties, whereas most social and economic systems get their complexity from the interactions of (more or less) rational agents.

**Examples of Properties in Specific Scenarios:**

1. **Crossword Puzzle**:
   * **Observable**: Fully observable—all clues and positions are visible.
   * **Deterministic**: Deterministic—the placement of each word has a known effect.
   * **Sequential**: Sequential— choices impact other spaces in the puzzle.
   * **Static**: Static—the puzzle doesn’t change while solving it.
   * **Discrete**: Discrete—words and letters are placed in fixed squares.
   * **Single Agents**:
2. **Poker Card**:
   * **Observable**: Partially observable—only your cards and common cards are visible; other players’ hands are hidden.
   * **Stochastic**: Stochastic— randomness in card dealing.
   * **Sequential**: Sequential— each round depends on previous bets and actions.
   * **Dynamic**: Dynamic— actions of players continuously affect the game.
   * **Discrete**: Discrete— limited set of actions and outcomes.
   * **Multi Agents**:
3. **Taxi Driving**:
   * **Observable**: Partially observable—may have sensors but can’t see all at once.
   * **Stochastic**: Stochastic—traffic, pedestrians, and unpredictable conditions introduce randomness.
   * **Sequential**: Sequential— decisions (e.g., turns, stops) build on each other.
   * **Dynamic**: Dynamic—traffic conditions and environment change over time.
   * **Continuous**: Continuous— movement through roads and positioning.
   * **Multi Agents**:
4. **Image Analysis**:
   * **Observable**: Fully observable— image is entirely visible.
   * **Deterministic**: Deterministic— analyzing the image produces consistent results.
   * **Episodic**: Episodic— each image is analyzed independently.
   * **Static**: Static—the image doesn’t change during analysis.
   * **Continuous**: Continuous—requires detailed processing of visual data.
   * **Single Agents**:
5. **Medical Diagnosis**:
   * **Observable**: Partially observable— some information might be hidden or uncertain.
   * **Stochastic**: Stochastic— diseases and symptoms can vary unpredictably.
   * **Sequential**: Sequential— tests and symptoms build to form a diagnosis.
   * **Dynamic**: Dynamic— patient health can change over time.
   * **Discrete**: Discrete— diagnosis has distinct outcomes (disease classifications).
   * **Single Agents**:

A table with black text

Description automatically generated