An Intelligent Agents

Role of Intelligent Agents in AI Applications

Intelligent agents play a pivotal role in AI **by performing tasks autonomously in various environments**. Their roles include:

1.Decision-Making

- Analyze input from sensors to choose actions that optimize performance.
- Example: Self-driving cars deciding when to stop or accelerate.

2.Automation

- Automate repetitive or complex tasks, saving time and effort.
- Example: Virtual assistants managing schedules or answering queries.

3.Adaptation and Learning

- Learn from interactions and improve over time to handle dynamic environments.
- Example: Chatbots adapting to user preferences.

4.Interaction with the Environment

- Actively sense and modify their surroundings to achieve goals.
- Example: Robots in manufacturing assembling components.

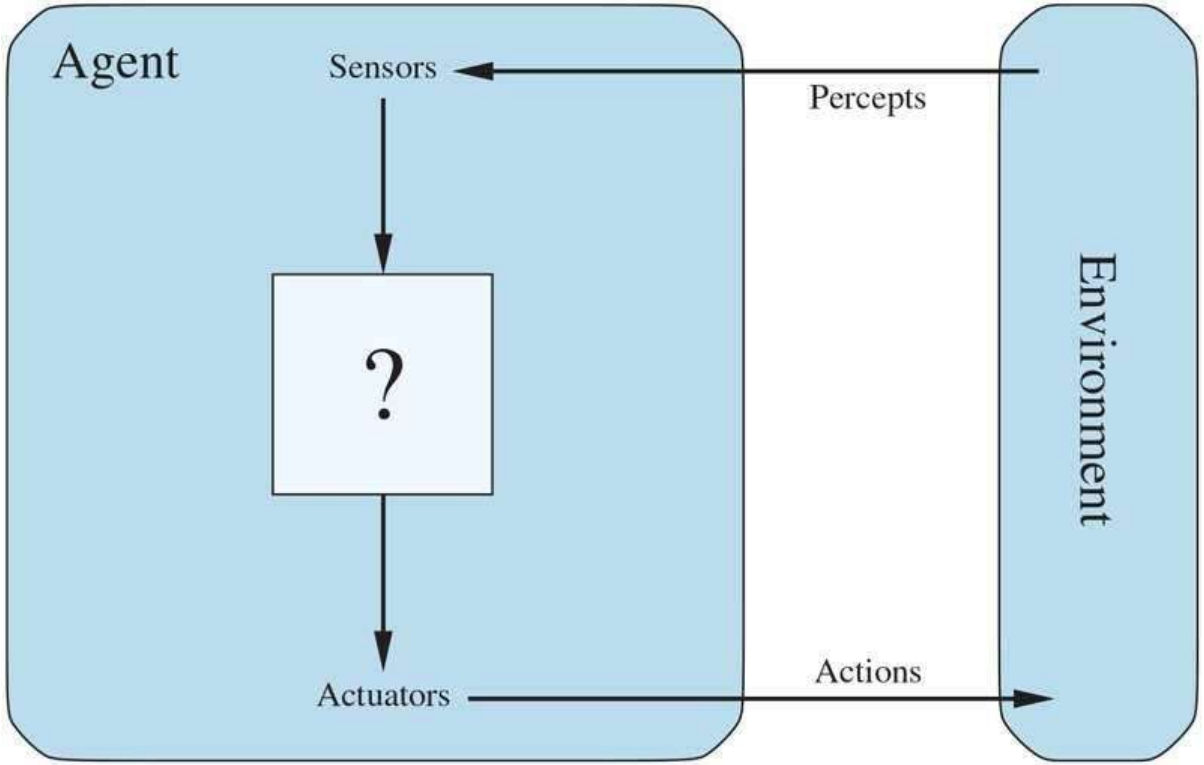
5.Problem Solving

- Solve complex problems by breaking them into simpler tasks.
- Example: Medical diagnosis systems suggesting treatments based on symptoms.

Intelligent agents enable AI systems to **function efficiently,** making them integral to diverse applications like healthcare, transportation, and robotics.

AI Agent and environments

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



The **agent function** for an agent specifies the action taken by the agent in response to any percept sequence.

Examples of agents

Example1:

A robotic agent might have cameras and infrared range finders

for sensors and various motors for actuators.

Example 2:

A software agent receives file contents, network packets, and human input (keyboard/mouse/touchscreen/voice) as sensory inputs and acts on the environment by writing files, sending network packets, and displaying information or generating sounds.

We use the term **percept** to refer to the content an agent's sensors are perceiving. An agent's **percept sequence** is **the complete history of everything the agent has ever perceived.**

Agent function – describes behavior of an agent. Agent program-implementation.

1.AI Agent

An AI agent is an entity that perceives its environment using **sensors** and acts upon it using **actuators** to achieve specific goals.

Example: A robot vacuum that senses dirt and moves/take action to clean it.

2. Sensors : Devices or mechanisms through which the agent perceives its environment.

- **Example**: A camera, microphone, or temperature sensor.

- 3. **Actuators:** Components that allow the agent to perform actions in the environment.
- **Example**: Robotic arms, wheels, or speakers.
- 4. **Task Environment:** The external setting or scenario in which the agent operates and interacts.
- **Example**: A chessboard for a chess-playing agent.

5. Percept Sequence : The complete history of all percepts (inputs) the agent has received.

- **Example**: A weather monitoring system's percept sequence could include past temperatures, humidity, and wind speed readings.

6. **Agent Program :** The implementation (software) that processes inputs (percepts) and determines the agent's actions.

- **Example**: The code inside a self-driving car deciding how to navigate.

7. **Agent Function**

A mathematical mapping from the percept sequence to actions that an agent performs.

Example: f(percept)→actionf(percept) \ rightarrow actionf(percept)→action, such as mapping "red light" to "stop.“ f(red light) -> stop.

Note: The agent function is an abstract mathematical description; the agent program is a concrete implementation, running within some physical system.

8.Performance Measure

A criterion for evaluating how well an agent achieves its goals.

Example: For a delivery robot, performance could be measured by the speed and accuracy of deliveries.

- These components collectively define how an AI intelligent agent is designed.

- **Examples of Intelligent Agents in AI**

1.Self-Driving Car

- **Sensors**: Cameras, LIDAR, GPS
- **Actuators**: Steering, brakes, throttle, signals.
- **Task Environment**: Roads, traffic, pedestrians.
- **Agent Function**: Maps sensor data to actions like turning, braking, or accelerating to navigate safely.

2. Virtual Assistant (e.g., Siri, Alexa)

Sensors: Microphone, text input.

- **Actuators**: Text-to-speech, display responses.
- **Task Environment**: User queries and the internet.
- **Agent Function**: Maps user input to responses or actions (e.g., playing music, setting reminders).

3.Autonomous Drone

- **Sensors**: Cameras, GPS, gyroscope
- **Actuators**: Propellers, camera gimbal.
- **Task Environment**: Airspace, delivery routes.
- **Agent Function**: Maps flight path data to adjust altitude, speed, and direction.

4.Roomba Vacuum Cleaner

- **Sensors**: Infrared, bump sensors, cameras.
- **Actuators**: Wheels, brushes, suction motor.
- **Task Environment**: Household floors, furniture.
- **Agent Function**: Maps dirt detection and obstacle data to movement and cleaning actions.

5.Spam Filter

- **Sensors**: Incoming email text.
- **Actuators**: Classifying emails as "spam" or "not spam."
- **Task Environment**: Email inbox.
- **Agent Function**: Maps email content and patterns to filter decisions.

2.2 Good Behavior: The Concept of Rationality

- A **rational agent** is one that does the right thing. Obviously, doing the right thing is better than doing the wrong thing, but what does it mean to do the right thing?

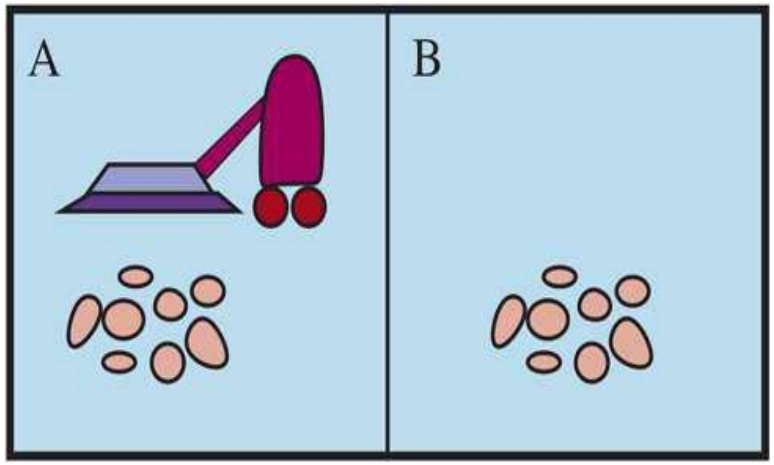
Notion of "right thing" --- Consequentialism – depends on agents behavior

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.

Vacuum –cleaning agent



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
⋮	⋮

A vacuum-cleaner world with just two locations. Each location can be clean or dirty, and the agent can move left or right and can clean the square that it occupies. Different versions of the vacuum world allow for different rules about what the agent can perceive, whether its actions always succeed, and so on.

- 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.

- **Omniscient Agent**
- **Definition**: An idealized agent that knows the actual outcomes of all possible actions in advance, including the future states of the environment.
- **Characteristics**:
  - Not practical in real-world scenarios as perfect knowledge is unattainable.
  - Serves as a theoretical benchmark for rationality.
- **Example**: A chess AI that knows all potential moves and their outcomes but is impossible in practice due to computational limits.

Omniscience ,learning and autonomy

- **Autonomous Agent**
- **Definition**: An agent that operates independently, relying on its own percepts and experiences rather than being controlled by external inputs or instructions.
- **Characteristics**:
  - Uses its sensors to perceive the environment.
  - Makes decisions and takes actions based on its internal state and learning.
- **Example**: A self-driving car that navigates without human intervention by analyzing road conditions and traffic.

Ideal Design of an AI Agent

- An ideal AI agent should balance the following principles:

1.Partial Omniscience

1. While full omniscience is impractical, the agent should strive for **informed decision-making** by gathering and processing as much relevant data as possible.

2.Autonomy

1. The agent should be **highly autonomous**, capable of operating independently and making decisions based on its own perceptions and internal models without constant external input.

3.Learning Capability

1. It should **learn as it perceives**, adapting to new situations and improving performance over time through reinforcement, supervised, or unsupervised learning.

4.Information Gathering

1. The agent must actively **explore and gather information** to reduce uncertainty about its environment and enhance its decision-making ability.

How an Intelligent Vacuum Cleaner Works in a 100-Square Room

1. **Environment**:
    1. A 10x10 grid representing a 100-square room.
    2. Each square can either be dirty or clean.
2. **Sensors**:
    1. Detects dirt in the current tile.
    2. Identifies room boundaries (to avoid moving outside).
3. **Actuators**:
    1. **Actions**: Move **right**, **left**, or **suck** dirt.
4. **Agent Program**:
    1. **Perception**: Uses sensors to determine if the tile is dirty or clean.
    2. **Decision Rules**:
        1. **If** the tile is dirty, **suck** the dirt.
        2. **If** the tile is clean, move to the next unclean tile (right or left).
    3. Avoids revisiting already cleaned tiles to maximize performance.

5. Performance Measure:

1. Maximizes the number of cleaned tiles.
2. Minimizes redundant movements and energy usage.

- **Example Workflow**

1.Starts at (1,1) on the grid.

2.Senses dirt. If present, sucks it up; otherwise, moves right.

3.Continues until all tiles are clean, avoiding cleaned squares and grid boundaries.

- This intelligent agent ensures efficient cleaning while optimizing time and energy.

- **Key Example**
- A self-driving car:
- Uses sensors (LIDAR, cameras) to gather data (partial omniscience).
- Operates autonomously without human intervention.
- Learns from past trips to optimize routes and handle new obstacles.
- Continuously collects environmental data for informed real-time decisions.
- This balance makes the AI agent rational, efficient, and adaptive to dynamic environments.

2.3 The Nature of Environments

In designing an agent, the first step must always be to specify the task environment as fully as possible.

A **task environment** specification includes the performance measure, the external environment, the actuators, and the sensors.

- **Dimensions of Task environment:**

Task environments vary along several significant dimensions.

They can be fully or partially observable, single-agent or multiagent, deterministic or nondeterministic, episodic or sequential, static or dynamic, discrete or continuous, and known or unknown.

1.Fully Observable vs. Partially Observable: The fully observable agent has access to complete information about the environment at all times.

- **Example**: Chess, where the board state is fully visible.

- **Partially Observable**: The agent only has limited or incomplete information.
  - **Example**: Poker, where opponents' cards are hidden.

2.Single-Agent vs. Multiagent: Only one agent operates in the environment.

- **Example**: A maze-solving robot.

- **Multiagent**: Multiple agents interact, potentially competing or cooperating.
  - **Example**: A multiplayer video game.

3.Deterministic vs. Nondeterministic

Deterministic: The outcome of actions is predictable and depends solely on the current state and action.

- **Example**: Solving a mathematical puzzle.

- **Nondeterministic**: The outcome may involve randomness or uncertainty.
  - **Example**: Weather prediction.

4.Episodic vs. Sequential

Episodic: Each decision is independent, and previous actions don't affect future actions.

Example: Image classification tasks, spam detection.

Sequential: Current decisions impact future outcomes.

Example: Chess game, self-driving cars.

5.Static vs. Dynamic

Static: The environment does not change while the agent is deciding.

- **Example**: A crossword puzzle.

- **Dynamic**: The environment changes over time, even while the agent is thinking.
  - **Example**: Real-time traffic navigation.

6.Discrete vs. Continuous

Discrete: The environment has a finite set of states or actions.

countable states, step-by step decision making.

- **Example**: (discrete moves) - chess game with defined position and moves.

- **Continuous**: The environment allows infinite states or actions.

- Require handling infinite possibilities, require ongoing/real time adjustments or changes.
  - **Example**: A robot arm's movement in 3D space.
    Self-driving cars navigating in real-world traffic.

7.Known vs. Unknown

Known: The agent knows the rules or model of the environment.

Example: Solving a Rubik's cube.

Unknown: The agent must learn the rules or model through exploration.

Example: Learning to play an unfamiliar video game.

- These dimensions help in understanding and designing AI agents suited to specific environments.
- In cases where the performance measure is unknown or hard to specify correctly, there is a significant risk of the agent optimizing the wrong objective.
- In such cases the agent design should reflect uncertainty about the true objective.

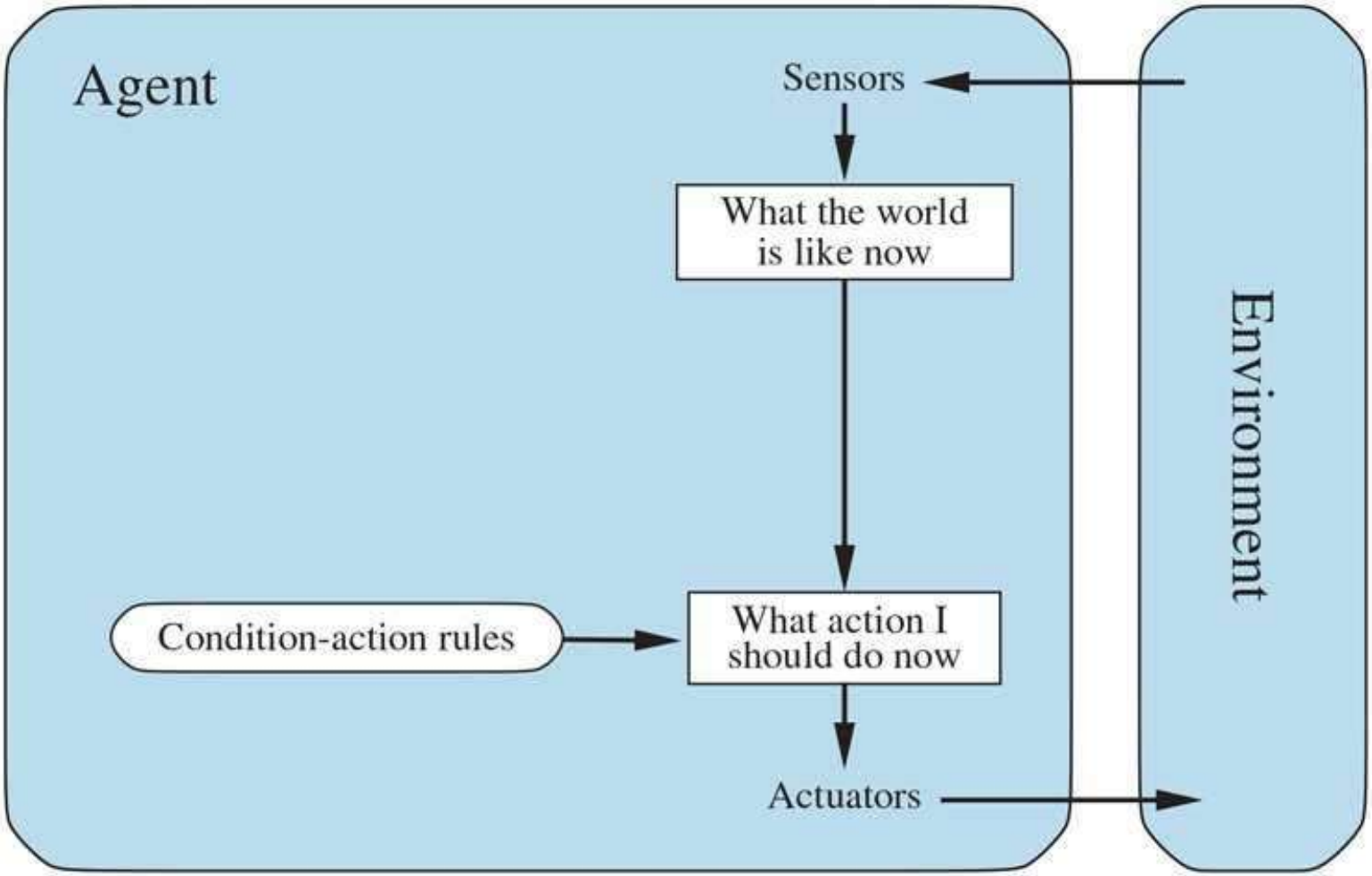
2.4 The Structure of Agent programs

- The **agent program** implements the agent function.
- There exists a variety of basic agent program designs reflecting the kind of information made explicit and used in the decision process.
- The designs vary in efficiency, compactness, and flexibility.
- The appropriate design of the agent program depends on the nature of the environment.
- Four basic kinds of agent programs that embodying the principles underlying almost all intelligent systems:
- **Simple reflex agents** respond directly to percepts, whereas **model-based reflex agents** maintain internal state to track aspects of the world that are not evident in the current percept.
- **Goal-based agents** act to achieve their goals, and
- **utility-based agents** try to maximize their own expected "happiness."

Types of AI Agent Programs with Examples

1. **Simple Reflex Agents**
    1. **How it Works**:
        1. Acts based on the current percept using condition-action rules (e.g., "IF condition THEN action").
        2. No memory of past states.
    2. **Example**:
        1. **Thermostat**: A thermostat turns on heating when the temperature drops below a certain level ("IF temperature < 20°C THEN turn on heater").
        2. **Light Sensor**: A streetlight turns on when it detects low light.
    3. **Strengths**: Fast and simple.
    4. **Limitations**: Cannot adapt to dynamic or unpredictable environments.

Simple reflex agents



Simple reflex agents

```
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
```

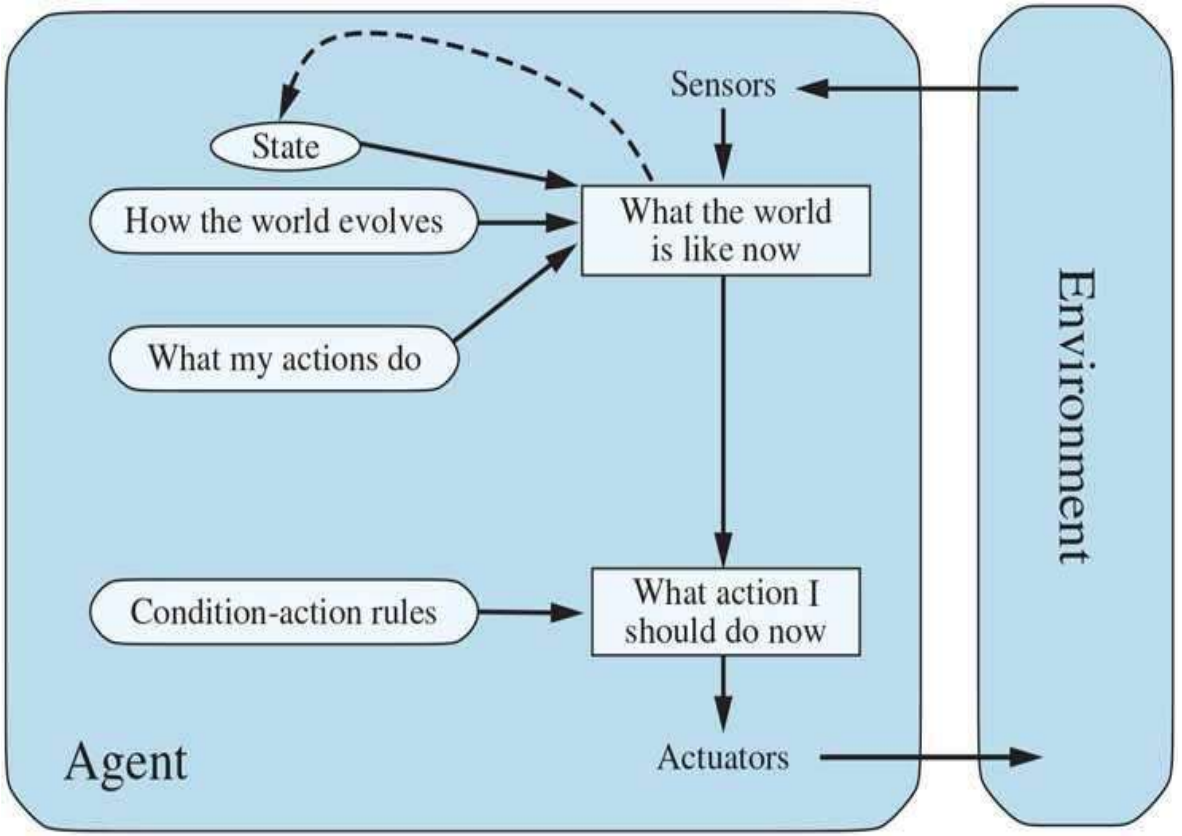
A simple reflex agent. It acts according to a rule whose condition matches the current state, as defined by the percept.

```
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
```

2.Model-Based Reflex Agents

- **How it Works**:
  - Maintains an internal model of the environment to consider past states along with current perceptions.
  - Uses this model to update its understanding of the world.
- **Example**:
  - **Robot Vacuum (e.g., Roomba)**: Keeps track of cleaned and uncleaned areas while navigating a room, ensuring it doesn't repeatedly clean the same spot.
  - **Autonomous Drone**: Remembers previous obstacles in its flight path to avoid them in future decisions.
- **Strengths**: Handles dynamic environments better than simple reflex agents.
- **Limitations**: Limited by the accuracy of its model and does not consider long-term goals.

Model-based reflex agents



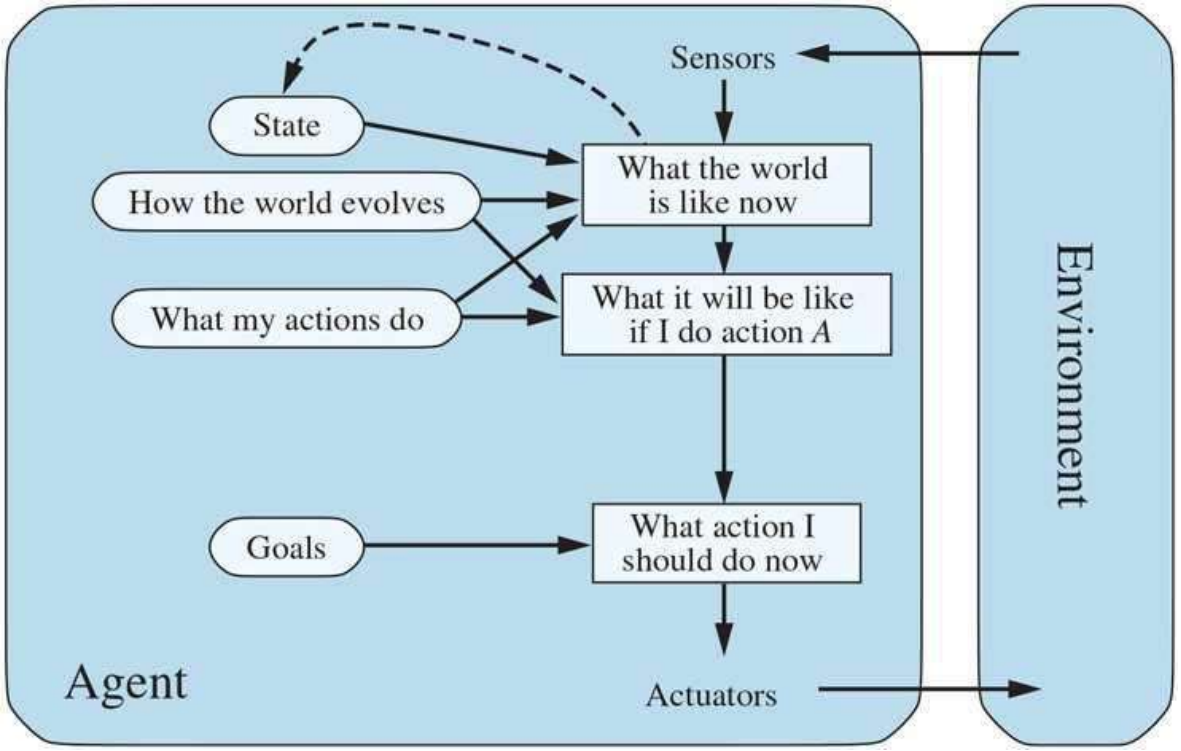
```
function MODEL-BASED-REFLEX-AGENT(percept) returns an action
  persistent: state, the agent's current conception of the world state
              transition_model, a description of how the next state depends on
                    the current state and action
              sensor_model, a description of how the current world state is reflected
                    in the agent's percepts
              rules, a set of condition–action rules
              action, the most recent action, initially none

  state ← UPDATE-STATE(state, action, percept, transition_model, sensor_model)
  rule ← RULE-MATCH(state, rules)
  action ← rule.ACTION
  return action
```

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

Gold-based agents

- **How it Works**:
  - Acts to achieve a specific goal by planning and executing actions.
  - Evaluates whether each action leads closer to the goal.
- **Example**:
  - **GPS Navigation System**: Plans the shortest or fastest route from Point A to Point B, considering traffic and road conditions.
  - **Chess-playing AI**: Chooses moves to achieve the goal of checkmating the opponent.
- **Strengths**: More flexible and adaptive, can plan sequences of actions.
- **Limitations**: Requires more computational resources and struggles with conflicting goals.

Gold – based agents



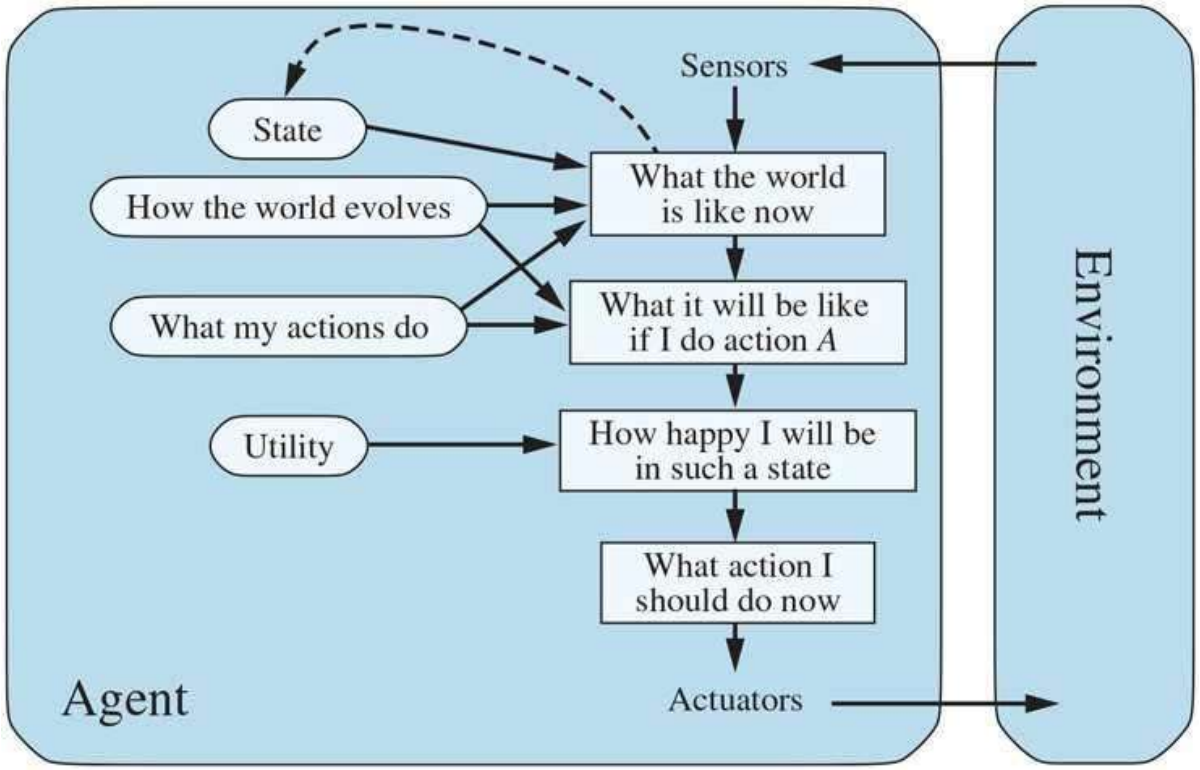
A model-based, goal-based agent. It keeps track of the world state as well as a set of goals it is trying to achieve, and chooses an action that will (eventually) lead to the achievement of its goals.

Utility-Based Agents

- **How it Works**:
  - Makes /optimizes decisions by maximizing a utility function, which quantifies the desirability of outcomes.
  - Balances multiple objectives or trade-offs (e.g., safety, speed, cost).
- **Example**:
  - **Self-Driving Cars**: Balances safety, fuel efficiency, and travel time to decide the best path.
  - **Recommendation Systems**: Suggests products on e-commerce platforms by weighing user preferences, product ratings, and availability.
- **Strengths**: Handles complex scenarios with competing priorities.
- **Limitations**: Computationally intensive and depends on a well-defined utility function.

Utility-based Agents

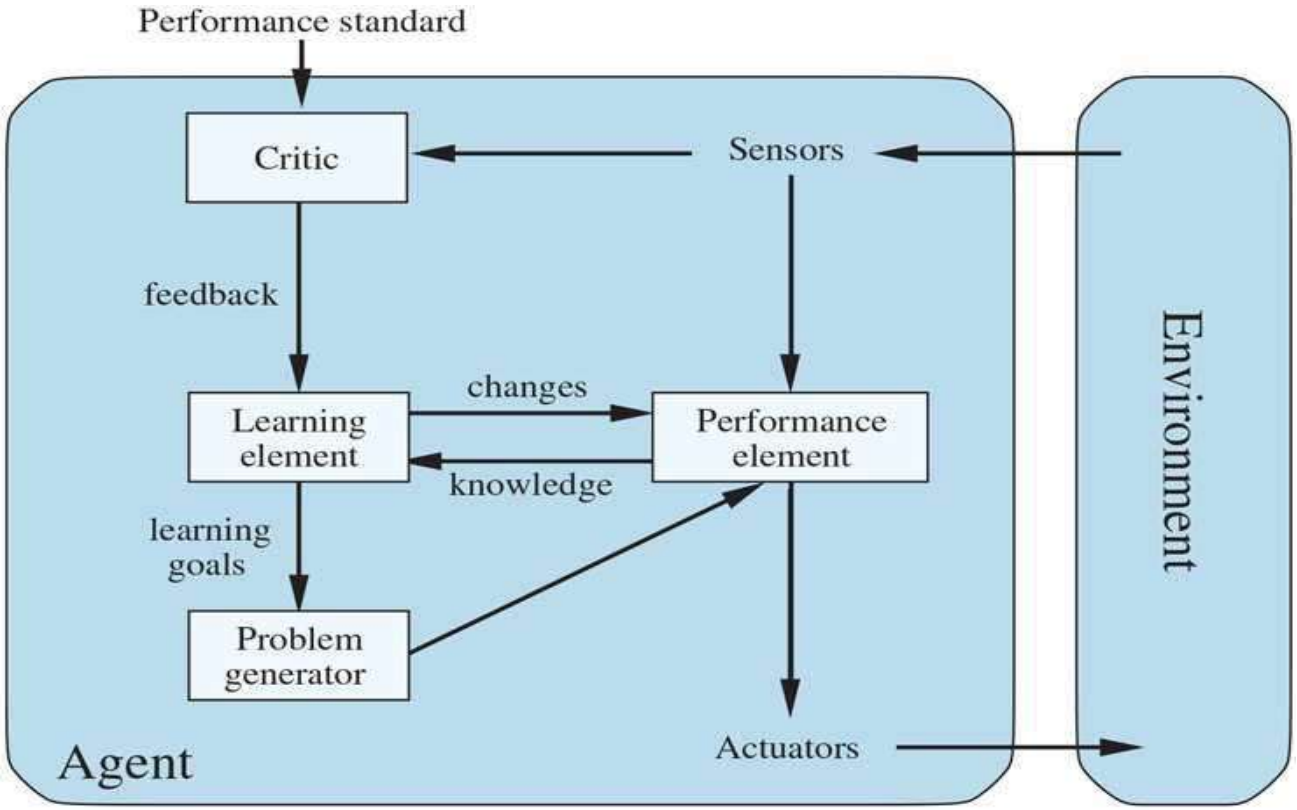
- A model-based, utility-based agent. It uses a model of the world, along with a utility function that measures its preferences among states of the world.
- Then it chooses the action that leads to the best expected utility, where expected utility is computed by averaging over all possible outcome states, weighted by the probability of the outcome.



Type	Description	Key Features	Examples
1. Simple Reflex Agents	Act based on **current percepts** only, using condition-action rules (e.g., "IF condition THEN action").	- No memory. - Reactive behavior.	**Thermosta** **Traffic light**
2. Model-Based Reflex Agents	Maintain an **internal model** of the environment to track unobservable states and improve decisions.	- Incorporates memory. - Tracks environment changes.	**Robot Vacu** and avoids
3. Goal-Based Agents	Make decisions to achieve specific **goals**, considering future actions and planning accordingly.	- Goal-oriented. - Plans actions.	**GPS Naviga** destination **Chess AI**: P
4. Utility-Based Agents	Optimize actions by maximizing a **utility function** that measures the desirability of different outcomes.	- Handles trade-offs. - Considers multiple factors.	**Self-Drivin** speed. **E-commerc** products ba rating, avail

Learning Agents

- Any type of agent (model-based, goal-based, utility-based, etc.) can be built as a learning agent (or not).
- Learning allows the agent to operate in initially unknown environments and to become more competent than its initial knowledge.
- A learning agent can be divided into four conceptual components. The most important distinction is between the **learning element**, which is responsible for making improvements, and the **performance element**, which is responsible for selecting external actions.
- The performance element is the entire agent, it takes in percepts and decides on actions. The learning element uses feedback from the **critic** on how the agent is doing and determines how the performance element should be modified to do better in the future.
- Program generator – suggests actions that will lead to new and informative experiences.



A general learning agent. The "performance element" box represents what we have previously considered to be the whole agent program. Now, the "learning element" box gets to modify that program to improve its performance.

Role of Learning Agents in AI Agent Programs

Learning agents enhance the capabilities of AI programs by enabling them to:

1.Improve Performance: Learn from past experiences to perform tasks more efficiently.

- **Example**: A chatbot improving responses over time based on user interactions.

2.Adapt to Changes: Handle dynamic environments and adapt to new situations.

- **Example**: Self-driving cars learning new traffic patterns or rules.

3.Optimize Decisions: Refine decision-making by minimizing errors and maximizing outcomes.

- **Example**: Recommendation systems improving accuracy based on user feedback.

4.Reduce Manual Intervention: Automate updates and improvements, reducing the need for human reprogramming.

- **Example**: Fraud detection systems updating rules based on evolving fraudulent behaviors.

Learning agents make AI systems more flexible, autonomous, and capable of handling complexity.

AI Agent program design for a Product Recommendation System ensures continuous improvement and accurate, user-centric product recommendations.

1.Learning Element

- **Role**: Improves recommendations by analyzing user behavior and preferences over time.
- **Example**: Machine learning algorithms that update based on customer interactions, like click-through rates or purchase history.

2.Performance Element

- **Role**: Delivers real-time product recommendations based on the current user's profile.
- **Example**: Suggesting "You may also like" items when browsing products.

3.Critic

- **Role**: Evaluates the effectiveness of recommendations by analyzing metrics like user engagement and conversion rates.
- **Example**: Monitoring the percentage of users who clicked or purchased recommended items.

4.Program Generator

- **Role**: Generates or adapts the recommendation model based on feedback and evolving user data.
- **Example**: Retraining the recommendation engine periodically using updated datasets.

5.Sensors

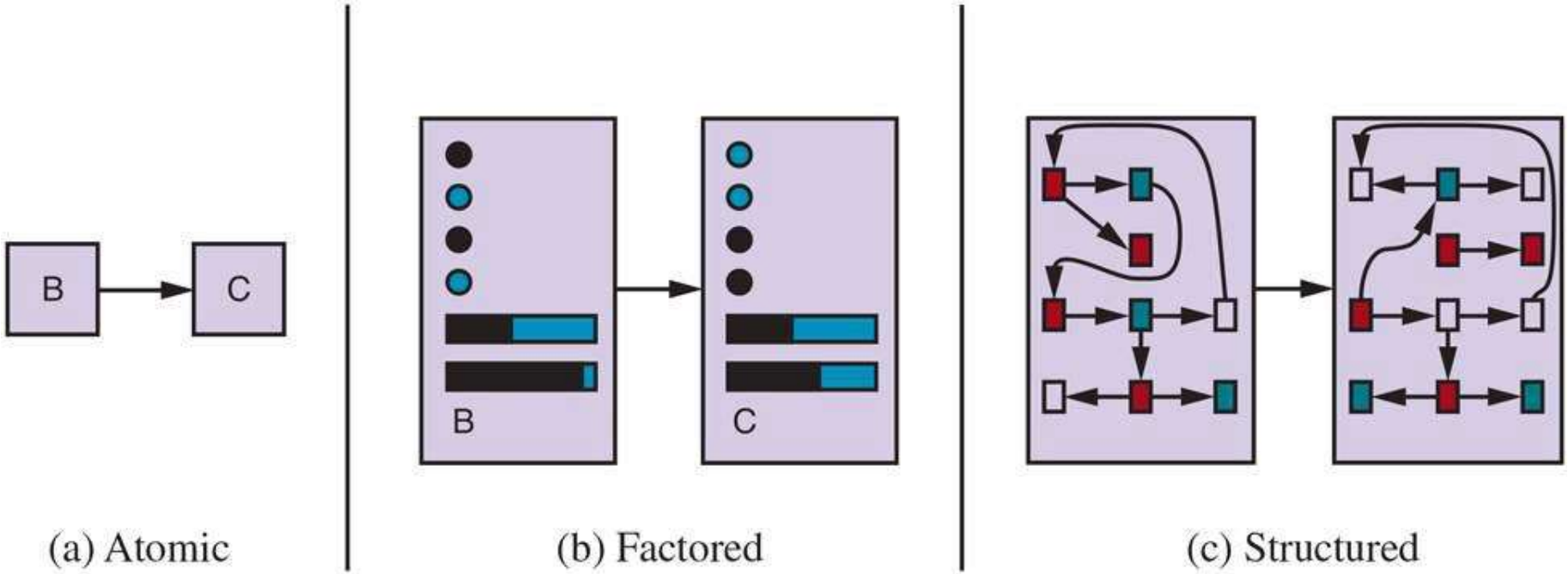
- **Role**: Collect user data, such as browsing patterns, purchase history, and search queries.
- **Example**: Tracking clicks, time spent on product pages, and cart additions.

6.Actuators

- **Role**: Display personalized product recommendations to users.
- **Example**: Updating the homepage with tailored suggestions or sending personalized emails.

How are the agent program work

- There exist various ways that the components can represent the environment that the agent inhabits.
- Roughly speaking, we can place the representations along an axis of increasing complexity and expressive power—atomic, factored, and structured.



Three ways to represent states and the transitions between them.

(a) Atomic representation: a state (such as B or C) is a black box with no internal structure;

The standard algorithms underlying search and game-playing, hidden Markov models, and Markov decision processes all work with atomic representations.

(b) Factored representation: a state consists of a vector of attribute values; values can be Boolean, real-valued, or one of a fixed set of symbols.

Many important areas of AI are based on factored representations,

including constraint satisfaction algorithms , propositional logic,

planning, Bayesian networks, and various machine learning algorithms.

(c) Structured representation: a state includes objects, each of which may have attributes of its own as well as relationships to other objects. First order probability models, things expressed as objects and their relationships in natural language models.